IBM Migration Toolkit

User's Guide and Reference

Version 2.0.3.0

Note:

Before using this information and the product it supports, be sure to read the general information under “Notices” on page 471.

First edition (July 2007)

This edition applies to Version 2.0.3.0 and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Introduction 1
About this manual 1
The MTK team welcomes your comments 1
Supported source and target databases 1

Getting started with MTK 3
System requirements 3
  Hardware requirements 3
  Software requirements 3
Installing MTK. 4
  Installing on Windows 4
  Installing on UNIX and Linux 5
Starting MTK 6
  Starting on Windows 6
  Starting on UNIX or Linux 6
Setting application preferences 7
User preference options 7
Keyboard shortcuts. 8

Types of MTK user interfaces 9
Graphical user interface 9
Wizard 9
Command line 9

Migration projects 11
Project files 11
Project archives 12
Creating a project 12
Opening an existing project 13
Saving a project 13
Closing a project 14
Modifying project description 14
Deleting a project 14
  Deleting the currently open project. 15
  Deleting projects that are not currently open 15
Backing up a project 15
Restoring a project 15
Modifying user preferences 16

The MTK process 17
Connecting to the source database 17
  Connecting to Oracle 17
  Connecting to Informix Dynamic Server 18
  Connecting to MySQL 20
  Connecting to Microsoft SQL Server 20
  Connecting to Sybase 22
The GUI process 23
  Step 1: Specifying the source 23
  Step 2: Converting source metadata 28
  Step 3: Refining the metadata conversion 39
  Step 4: Generating scripts. 44
  Step 5: Deploy to the target database server 59
The command line process 73

Running MTK from the command line . . . 73
Configuration file . . . 75
Command line migration projects . . . 99
Command line examples . . . 100
The wizard process . . . 107
Step 1: Project Information . . . 108
Step 2: Source Database . . . 108
Step 3: Global Type Mapping . . . 108
Step 4: Data Movement . . . 109
Step 5: Target Database . . . 109
Step 6: Migration Summary . . . 110

Reports and logs . . . 111
Viewing migration reports . . . 111
Viewing the application log . . . 112
Viewing the changes report . . . 112

Frequently asked questions . . . 115

Introduction to the converter component . . . 127
How the converter works . . . 127
The source script file . . . 127
Script translation . . . 128
Restrictions for migrating scripts . . . 129
Object renaming . . . 129
Messages . . . 131

Informix Dynamic Server converter . . . 133
Converter behavior and limitations . . . 133
System catalogs . . . 133
Data types . . . 133
Operators . . . 135
Built-in functions . . . 137
Expressions . . . 139
Conditions . . . 140
SQL statements . . . 140
SPL statements . . . 146
Transaction statements . . . 149
Compatibility library (INFX functions) for translations to DB2 . . . 149

MySQL converter . . . 151
Converter behavior and limitations . . . 151
Data types . . . 151
Statements . . . 153
DB2 UDB supported features . . . 154
Informix Dynamic Server supported features . . . 156

Oracle converter . . . 159
Converter behavior and limitations . . . 159
Built-in-functions . . . 159
Conditions and predicates . . . 166
Cursors . . . 169
Data types . . . 171
Exceptions in PL/SQL . . . 182
Expressions . . . 185
Identifiers . . . 192

Naming convention . . . 193
Operators . . . 193
Packages . . . 197
Queries . . . 205
Statements . . . 206
Variable assignment . . . 236
Compatibility library (ORA functions) . . . 237
National language support . . . 237
List of functions that simulate Oracle built-in functions . . . 238
DB2 UDB supported features . . . 243
Informix Dynamic Server supported features . . . 253

SQL Server and Sybase converter . . . 263
Converter behavior and limitations . . . 263
Identifiers . . . 263
Data types . . . 264
System procedures . . . 267
Built-in functions . . . 268
Expressions and operators . . . 276
Queries . . . 279
Statements . . . 281
@@ERROR . . . 293
@@SQLSTATUS . . . 296
@@ROWCOUNT . . . 296
Transactions . . . 297
Cursors . . . 299
Triggers . . . 299
Temporary tables . . . 300
Table variables . . . 301
Stored procedures . . . 301
Functions . . . 302
Compatibility library (MSSQL and SYB functions) . . . 302

Informix Dynamic Server converter messages . . . 321
MTKI0000 . . . 321
MTKI0001 . . . 321
MTKI0002 . . . 321
MTKI0003 . . . 321
MTKI0004 . . . 321
MTKI0005 . . . 321
MTKI0006 . . . 322
MTKI0010 . . . 322
MTKI0011 . . . 322
MTKI0012 . . . 322
MTKI0013 . . . 322
MTKI0014 . . . 323
MTKI0015 . . . 323
MTKI0016 . . . 323
MTKI0017 . . . 323
MTKI0018 . . . 323
MTKI0019 . . . 323
MTKI0020 . . . 324
MTKI0021 . . . 324
MTKI0022 . . . 324
MTKI0023 . . . 324
MTKI0024 . . . 324

MTKI0025 325
MTKI0026 325
MTKI0027 325
MTKI0028 325
MTKI0029 325
MTKI0030 326
MTKI0031 326
MTKI0032 326
MTKI0033 326
MTKI0034 326
MTKI0035 326
MTKI0036 327
MTKI0037 327
MTKI0038 327
MTKI0039 327
MTKI0040 327
MTKI0041 328
MTKI0042 328
MTKI0043 328
MTKI0044 328
MTKI0045 328
MTKI0046 329
MTKI0047 329
MTKI0048 329
MTKI0049 329
MTKI0050 330
MTKI0051 330
MTKI0052 330
MTKI0053 330
MTKI0054 330
MTKI0055 331
MTKI0056 331
MTKI0057 331
MTKI0058 331
MTKI0059 331
MTKI0060 332
MTKI0061 332
MTKI0062 332
MTKI0063 332
MTKI0064 332
MTKI0065 333
MTKI0066 333
MTKI0067 333
MTKI0068 333
MTKI0069 333
MTKI0070 333
MTKI0071 334
MTKI0072 334
MTKI0073 334
MTKI0074 334
MTKI0075 334
MTKI0076 334
MTKI0077 335
MTKI0078 335
MTKI0079 335
MTKI0080 335

MTKI0081 335
MTKI0082 336
MTKI0083 336
MTKI0084 336
MTKI0085 336
MTKI0086 336
MTKI0087 337
MTKI0088 337
MTKI0089 337
MTKI0090 337
MTKI0091 337
MTKI0092 338
MTKI0093 338
MTKI0094 338
MTKI0095 338
MTKI0096 338
MTKI0097 339
MTKI0098 339
MTKI0099 339
MTKI0100 339
MTKI0101 340
MTKI0102 340
MTKI0103 340
MTKI0104 340
MTKI0105 340
MTKI0108 341
MTKI0109 341
MTKI0113 341
MTKI0114 341
MTKI0115 342
MTKI0119 342
MTKI0120 342
MTKI0121 342
MTKI0122 342
MTKI0123 343
MTKI0124 343
MTKI0125 343
MTKI0126 343
MTKI0127 343
MTKI0128 344
MTKI0129 344
MTKI0130 344
MTKI0131 344
MTKI0132 344
MTKI0133 345
MTKI0134 345
MTKI0135 345
MTKI0136 345
MTKI0137 346
MTKI0138 346
MTKI0139 346
MTKI0140 346
MTKI0141 347
MTKI0142 347
MTKI0143 347
MTKI0144 347

MTKI0153 347
MTKI0160 348
MTKI0161 348
MTKI0163 348
MTKI0164 348
MTKI0191 348
MTKI0195 349
MTKI0196 349
MTKI0212 349
MTKI0213 349
MTKI0214 349
MTKI0215 349
MTKI0216 349
MTKI0217 349
MTKI0218 350
MTKI0219 350
MTKI0220 350
MTKI0221 350
MTKI0226 350
MTKI0227 350
MTKI0244 350
MTKI0245 351
MTKI0246 351
MTKI0247 351

Oracle converter messages 353
MTKO0000 353
MTKO0001 353
MTKO0002 353
MTKO0003 353
MTKO0004 353
MTKO0005 353
MTKO0006 354
MTKO0010 354
MTKO0011 354
MTKO0012 354
MTKO0013 354
MTKO0014 355
MTKO0015 355
MTKO0016 355
MTKO0017 355
MTKO0018 355
MTKO0019 355
MTKO0020 356
MTKO0021 356
MTKO0022 356
MTKO0023 356
MTKO0024 357
MTKO0025 357
MTKO0026 357
MTKO0027 357
MTKO0028 357
MTKO0029 358
MTKO0030 358
MTKO0031 358
MTKO0032 358

MTKO0033 358
MTKO0034 358
MTKO0035 359
MTKO0036 359
MTKO0037 359
MTKO0038 359
MTKO0039 359
MTKO0040 360
MTKO0041 360
MTKO0042 360
MTKO0043 360
MTKO0044 360
MTKO0045 361
MTKO0046 361
MTKO0047 361
MTKO0048 362
MTKO0049 362
MTKO0050 362
MTKO0051 362
MTKO0052 363
MTKO0053 363
MTKO0054 363
MTKO0055 363
MTKO0056 363
MTKO0057 364
MTKO0058 364
MTKO0059 364
MTKO0060 364
MTKO0061 364
MTKO0062 365
MTKO0063 365
MTKO0064 365
MTKO0065 365
MTKO0066 365
MTKO0067 366
MTKO0068 366
MTKO0069 366
MTKO0070 366
MTKO0071 366
MTKO0072 367
MTKO0073 367
MTKO0074 367
MTKO0075 367
MTKO0076 367
MTKO0077 368
MTKO0078 368
MTKO0079 368
MTKO0080 368
MTKO0081 368
MTKO0082 369
MTKO0083 369
MTKO0084 369
MTKO0085 369
MTKO0086 369
MTKO0087 370
MTKO0088 370

MTKO0089 370
MTKO0090 370
MTKO0091 370
MTKO0092 371
MTKO0093 371
MTKO0094 371
MTKO0095 371
MTKO0096 371
MTKO0097 372
MTKO0098 372
MTKO0099 372
MTKO0100 373
MTKO0101 373
MTKO0102 373
MTKO0103 373
MTKO0104 373
MTKO0105 374
MTKO0106 374
MTKO0107 374
MTKO0108 374
MTKO0109 375
MTKO0110 375
MTKO0111 375
MTKO0112 375
MTKO0113 376
MTKO0114 376
MTKO0115 376
MTKO0116 376
MTKO0117 376
MTKO0118 377
MTKO0120 377
MTKO0121 377
MTKO0129 377
MTKO0135 377
MTKO0136 378
MTKO0137 378
MTKO0138 378
MTKO0139 378
MTKO0140 379
MTKO0141 379
MTKO0143 379
MTKO0145 379
MTKO0191 379
MTKO0194 380
MTKO0211 380
MTKO0212 380
MTKO0213 380
MTKO0214 380
MTKO0215 380
MTKO0216 380
MTKO0217 381
MTKO0218 381
MTKO0219 381
MTKO0220 381
MTKO0221 381
MTKO0225 381

MTKO0226 . . . 381
MTKO0227 . . . 381
MTKO0244 . . . 382
MTKO0245 . . . 382
MTKO0246 . . . 382
MTKO0247 . . . 382
MTKO0251 . . . 382
MTKO0252 . . . 383
MTKO0253 . . . 383
MTKO0254 . . . 383
MTKO0255 . . . 383
MTKO0256 . . . 383
MTKO0257 . . . 383
MTKO0258 . . . 384
MTKO0259 . . . 384
MTKO0260 . . . 384
MTKO0261 . . . 384
MTKO0262 . . . 384
MTKO0263 . . . 385
MTKO0264 . . . 385
MTKO0266 . . . 385
MTKO0267 . . . 385
MTKO0269 . . . 385
MTKO0270 . . . 385
MTKO0272 . . . 386
MTKO0273 . . . 386
MTKO0274 . . . 386
MTKO0275 . . . 386
MTKO0276 . . . 386
MTKO0277 . . . 387
MTKO0278 . . . 387
MTKO0279 . . . 387
MTKO0280 . . . 387
MTKO0281 . . . 387
MTKO0293 . . . 388
MTKO0294 . . . 388
MTKO0295 . . . 388

SQL Server and Sybase converter messages . . . 389
MTKS0000 . . . 389
MTKS0001 . . . 389
MTKS0002 . . . 389
MTKS0003 . . . 389
MTKS0004 . . . 389
MTKS0005 . . . 389
MTKS0006 . . . 390
MTKS0010 . . . 390
MTKS0011 . . . 390
MTKS0012 . . . 390
MTKS0013 . . . 390
MTKS0014 . . . 391
MTKS0015 . . . 391
MTKS0016 . . . 391
MTKS0017 . . . 391
MTKS0018 . . . 391
MTKS0019 . . . 391

MTKS0020 392
MTKS0021 392
MTKS0022 392
MTKS0023 392
MTKS0024 393
MTKS0025 393
MTKS0026 393
MTKS0027 393
MTKS0028 393
MTKS0029 394
MTKS0030 394
MTKS0031 394
MTKS0032 394
MTKS0033 394
MTKS0034 394
MTKS0035 395
MTKS0036 395
MTKS0037 395
MTKS0038 395
MTKS0039 395
MTKS0040 396
MTKS0041 396
MTKS0042 396
MTKS0043 396
MTKS0044 396
MTKS0045 397
MTKS0046 397
MTKS0047 397
MTKS0048 397
MTKS0049 398
MTKS0050 398
MTKS0051 398
MTKS0052 398
MTKS0053 399
MTKS0054 399
MTKS0055 399
MTKS0056 399
MTKS0057 399
MTKS0058 399
MTKS0059 400
MTKS0060 400
MTKS0061 400
MTKS0062 400
MTKS0063 400
MTKS0064 401
MTKS0065 401
MTKS0066 401
MTKS0067 401
MTKS0068 401
MTKS0069 402
MTKS0070 402
MTKS0071 402
MTKS0072 402
MTKS0073 402
MTKS0074 402
MTKS0075 403

MTKS0076 403
MTKS0077 403
MTKS0078 403
MTKS0079 403
MTKS0080 404
MTKS0081 404
MTKS0082 404
MTKS0083 404
MTKS0084 404
MTKS0085 404
MTKS0086 405
MTKS0087 405
MTKS0088 405
MTKS0089 405
MTKS0090 405
MTKS0091 405
MTKS0092 406
MTKS0093 406
MTKS0094 406
MTKS0095 406
MTKS0096 406
MTKS0097 407
MTKS0098 407
MTKS0099 408
MTKS0100 408
MTKS0101 408
MTKS0102 408
MTKS0103 408
MTKS0104 408
MTKS0105 409
MTKS0106 409
MTKS0107 409
MTKS0108 409
MTKS0109 410
MTKS0110 410
MTKS0111 410
MTKS0112 410
MTKS0113 411
MTKS0114 411
MTKS0115 411
MTKS0116 411
MTKS0117 412
MTKS0118 412
MTKS0120 412
MTKS0121 412
MTKS0122 412
MTKS0129 413
MTKS0130 413
MTKS0137 413
MTKS0138 413
MTKS0139 413
MTKS0143 414
MTKS0146 414
MTKS0147 414
MTKS0148 414
MTKS0149 414

MTKS0150 415
MTKS0151 415
MTKS0152 415
MTKS0153 415
MTKS0154 415
MTKS0155 415
MTKS0156 415
MTKS0157 415
MTKS0158 416
MTKS0159 416
MTKS0160 416
MTKS0161 416
MTKS0162 416
MTKS0163 417
MTKS0164 417
MTKS0165 417
MTKS0166 417
MTKS0167 417
MTKS0168 417
MTKS0169 418
MTKS0170 418
MTKS0171 418
MTKS0172 418
MTKS0173 418
MTKS0174 418
MTKS0175 418
MTKS0176 418
MTKS0177 419
MTKS0178 419
MTKS0179 419
MTKS0180 419
MTKS0181 419
MTKS0182 419
MTKS0183 420
MTKS0184 420
MTKS0185 420
MTKS0186 420
MTKS0187 421
MTKS0188 421
MTKS0189 421
MTKS0190 421
MTKS0191 422
MTKS0192 422
MTKS0193 422
MTKS0197 422
MTKS0198 422
MTKS0199 422
MTKS0200 423
MTKS0201 423
MTKS0202 423
MTKS0203 424
MTKS0204 424
MTKS0205 424
MTKS0206 424
MTKS0207 425
MTKS0208 425

MTKS0209 425
MTKS0210 425
MTKS0211 425
MTKS0212 426
MTKS0213 426
MTKS0214 426
MTKS0215 426
MTKS0216 426
MTKS0217 426
MTKS0218 426
MTKS0219 426
MTKS0220 426
MTKS0221 427
MTKS0222 427
MTKS0223 427
MTKS0224 427
MTKS0225 427
MTKS0226 427
MTKS0227 428
MTKS0228 428
MTKS0229 428
MTKS0230 428
MTKS0231 428
MTKS0232 428
MTKS0233 429
MTKS0234 429
MTKS0235 429
MTKS0236 429
MTKS0237 429
MTKS0238 429
MTKS0239 430
MTKS0240 430
MTKS0241 430
MTKS0242 431
MTKS0243 431
MTKS0244 431
MTKS0245 431
MTKS0246 432
MTKS0247 432
MTKS0248 432
MTKS0249 432
MTKS0250 432
MTKS0255 433
MTKS0256 433
MTKS0261 433
MTKS0262 433
MTKS0265 433
MTKS0266 433
MTKS0267 434
MTKS0268 434
MTKS0269 434
MTKS0270 434
MTKS0271 434
MTKS0282 435

MySQL converter messages 437

MTKM0000 437
MTKM0001 437
MTKM0002 437
MTKM0003 437
MTKM0004 437
MTKM0005 437
MTKM0006 438
MTKM0010 438
MTKM0011 438
MTKM0012 438
MTKM0013 438
MTKM0014 438
MTKM0015 439
MTKM0016 439
MTKM0017 439
MTKM0018 439
MTKM0019 439
MTKM0020 440
MTKM0021 440
MTKM0022 440
MTKM0023 440
MTKM0024 440
MTKM0025 441
MTKM0026 441
MTKM0027 441
MTKM0028 441
MTKM0029 441
MTKM0030 442
MTKM0031 442
MTKM0032 442
MTKM0033 442
MTKM0034 442
MTKM0035 442
MTKM0036 443
MTKM0037 443
MTKM0038 443
MTKM0039 443
MTKM0040 443
MTKM0041 444
MTKM0042 444
MTKM0043 444
MTKM0044 444
MTKM0045 445
MTKM0046 445
MTKM0047 445
MTKM0048 445
MTKM0049 445
MTKM0050 446
MTKM0051 446
MTKM0052 446
MTKM0053 447
MTKM0054 447
MTKM0056 447
MTKM0057 447
MTKM0058 447
MTKM0059 448

MTKM0060 448
MTKM0061 448
MTKM0062 448
MTKM0063 448
MTKM0064 449
MTKM0065 449
MTKM0066 449
MTKM0067 449
MTKM0068 449
MTKM0069 450
MTKM0070 450
MTKM0071 450
MTKM0072 450
MTKM0073 450
MTKM0074 451
MTKM0075 451
MTKM0076 451
MTKM0077 451
MTKM0078 451
MTKM0079 451
MTKM0080 452
MTKM0081 452
MTKM0082 452
MTKM0083 452
MTKM0084 452
MTKM0085 453
MTKM0086 453
MTKM0087 453
MTKM0088 453
MTKM0089 453
MTKM0090 454
MTKM0091 454
MTKM0092 454
MTKM0093 454
MTKM0094 454
MTKM0095 455
MTKM0096 455
MTKM0097 455
MTKM0098 456
MTKM0099 456
MTKM0100 456
MTKM0101 456
MTKM0102 456
MTKM0103 457
MTKM0104 457
MTKM0105 457
MTKM0106 457
MTKM0107 457
MTKM0108 458
MTKM0109 458
MTKM0110 458
MTKM0111 459
MTKM0112 459
MTKM0113 459
MTKM0114 459
MTKM0115 459

MTKM0116 460
MTKM0117 460
MTKM0118 460
MTKM0120 460
MTKM0121 460
MTKM0129 461
MTKM0135 461
MTKM0136 461
MTKM0137 461
MTKM0138 462
MTKM0139 462
MTKM0140 462
MTKM0141 462
MTKM0143 462
MTKM0145 463
MTKM0191 463
MTKM0194 463
MTKM0211 463
MTKM0212 463
MTKM0213 464
MTKM0214 464
MTKM0215 464
MTKM0216 464
MTKM0217 464
MTKM0218 464
MTKM0219 464
MTKM0220 464
MTKM0221 464
MTKM0225 464
MTKM0226 465
MTKM0227 465
MTKM0244 465
MTKM0245 465
MTKM0246 465
MTKM0247 466
MTKM0251 466
MTKM0252 466
MTKM0253 466
MTKM0254 466
MTKM0255 466
MTKM0256 467
MTKM0257 467
MTKM0258 467
MTKM0259 467
MTKM0283 467
MTKM0284 467
MTKM0285 468
MTKM0286 468
MTKM0287 468
MTKM0288 468
MTKM0289 468
MTKM0290 469
MTKM0291 469
MTKM0292 469

Notices 471

Trademarks. 473

Index 475

Introduction

This introduction provides an overview of the information in this manual.

About this manual

This manual provides guidance information that you can use to perform a variety of tasks with IBM Migration Toolkit (MTK).

The MTK team welcomes your comments

We want to know about any bugs, suggested improvements, corrections, or clarifications to the IBM Migration Toolkit and IBM Migration Toolkit User's Guide and Reference.

To contact the MTK team, send your comments to us at the following email address: mtk@us.ibm.com.

Supported source and target databases

The source and target databases combinations that MTK supports are listed here.

		Source DBMS				
		Oracle Versions 8i and 9i	Sybase Adaptive Server Enterprise Versions 11, 12, 12.5 and 15	Microsoft® SQL Server Versions 7, 2000 and 2005	Sybase SQL Anywhere Version 9	MySQL Versions 4 and 5
Target DBMS	DB2® UDB on Linux®, Unix, and Windows® - Versions 8.1, 8.2, and DB2 9	√	√	√	√	√
	DB2 UDB for iSeries™ V5R2,V5R3 and V5R4 (System i5™)	√	√	√		
	DB2 UDB for z/OS® Version 8	√				
	Informix® Dynamic Server Version 10	√			√	√

Note: The MTK command line interface does not support migrations to DB2 Viper II or Informix Dynamic Server.

Getting started with MTK

To get started, become familiar with the system requirements and install, configure, and start MTK.

System requirements

To use MTK, you need the required software and hardware.

Hardware requirements

The hardware required to use MTK is described in this topic.

Disk space:

- 50 MB for installation.
- 5 MB per project.
- Additional space for the project script and data files.

Memory:
1 GB or more (increase for large SQL files).

Software requirements

The software required to use MTK is described in this topic.

General requirements

- **Operating system:**
  - Windows XP, Windows 2000
  - AIX 5L™ 5.2
  - Linux RHEL 3
  - Solaris 2.9/9
  - HP-UX B.11.11
- **Java™ Runtime Environment:** You must have Java Runtime Environment 1.4.2 installed and accessible through the PATH environment variable.
- **JDBC or ODBC driver (for source database connections):** For more information, see "Connecting to the source database" on page 17.

Windows-specific requirements

- If you are deploying SQL stored procedures to DB2 UDB Version 8.1 or earlier, the following software requirement applies:
  - Microsoft Visual C++ version 5 or later

  DB2 UDB Version 8.2 does not require a compiler.
- If you are migrating to DB2 for z/OS, the following software requirements apply:
  - On the Windows client:
    - DB2 for z/OS Version 8.1 Fix pack 4 or later.

      If you have fix pack 4, you must update the DB2Build.bat file. For Windows clients the file to update is:
      ```
      ...sqllib/bin/db2build.bat
      ```
      Add the following line:
      ```
      set CPATH=%CPATH%;%DB2PATH%\tools\sqlassist2.jar
      ```

    - Do not use DB2 V9 client because deployment of stored procedures and Java UDFs will not work.
  - On DB2 for z/OS:
    - z/OS Version 8, configured with support for Java UDF and Stored Procedures
    - Work Load Manager
    - Resource Recovery Services
    - FTP server

UNIX and Linux-specific requirements

- On Linux, increase the message queue number to at least 128;

  ```
  sysctl -w kernel.msgmni=128
  ```
- To view HTML reports, include the browser directory in the $PATH variable. If the browser cannot be found, MTK will launch an internal Java Web browser, which can display HTML files, but does not handle frames or format the tables well.
- If you are extracting from a data source using ODBC or Java, configure the client connection.

Related tasks

"Importing source files" on page 24
You can specify that an external file be translated if the file contains valid source SQL.

"Extracting objects from a source database" on page 24
You can specify that an extracted object be translated. The objects are extracted to a file that MTK can translate.

"Deleting a source file" on page 27
Files can be deleted from the list on the Specify source page.

"Specifying the order of the source files" on page 27
During the conversion process, source files are converted in the order they are listed. The MTK parser only recognizes objects that have been defined. Therefore, when converting multiple files, you must put them in the correct order.

Installing MTK

You can install MTK on Windows, UNIX®, or Linux.

Prerequisites:

- Uninstall earlier releases of MTK before installing the latest release.
- All versions of MTK are intercompatible.
- To simplify the deployment step, run MTK while logged-in with a system user ID that has system administration authority for the target Informix Dynamic Server or DB2 UDB database (for example in DB2 UDB, the user ID used to install DB2 UDB). Use caution when you are logged in as the database administrator.

Installing on Windows

MTK can be installed on the Windows operating system.

To install on Windows:

1. Download and unzip the MTK file from the MTK download site. The zip file contains the MTKSilentInstall.iss file, the readme.txt file, and the mtk.exe file.

2. Navigate to the directory where you unzipped the files and double-click the `mtk.exe` file to start the installation and follow the instructions.
3. Verify that you can access Java and that it is the correct version:

   ```
   java -version
   ```

   Related tasks

   "Starting on Windows" on page 6
   You can start the IBM® Migration Toolkit (MTK) on the Microsoft Windows operating system.

Installing on Windows using silent install

You can install MTK on Windows using the silent install feature.

To perform a silent install:

1. Download and unzip the MTK file from the MTK download site. The zip file contains the MTKSilentInstall.iss file, the readme.txt file, and the mtk.exe file.
2. From a command prompt, navigate to the directory where you unzipped the files and issue the following command:

   ```
   mtk.exe -s -f1"path/MTKSilentInstall.iss" -f2"log_path\setup.log"
   ```

   Where:
   - *path* indicates your local directory containing the MTKSilentInstall.iss file.
   - *log_path* indicates the location where you want your log file stored.

   Important: Do not to add a space between the `f1` option and the location of the silent install .iss file, and do not add a space between the `-f2` and the location of the setup.log file.
3. Open your setup.log file and check whether the ResultCode is set to `0`, indicating a successful install. For example:

   ```
   [InstallShield Silent]
   Version=v7.00
   File=Log File
   [ResponseResult]
   ResultCode=0
   [Application]
   Name=IBM Migration Toolkit 2.0
   Version=2.0
   Company=IBM
   Lang=0009
   ```

Installing on UNIX and Linux

MTK can be installed on UNIX and Linux operating systems.

Prerequisites: Ensure that you have the necessary hardware and software requirements, which are listed in "System requirements" on page 3.

To install on UNIX or Linux:

1. Log in with the user ID you want to install MTK with. Do not install MTK as root. Do not use the root user ID unless you are deploying to DB2 UDB.

   If you are deploying to a DB2 UDB database:
   - Install MTK using a user ID in the db2admin group.
   - Verify that the DB2 UDB INSTHOME environment variable is set to your DB2 UDB instance directory and that it is properly exported when you start a new environment. For example, in Korn shell type:

     ```
     export $INSTHOME
     ```

- MTK uses the Korn shell for deployment. If your environment has the Korn shell installed in a location other than the default location, make a symbolic link to it from **/usr/bin/ksh**, which is where MTK expects it to be located:

  ```
  ln -s `which ksh` /usr/bin/ksh
  ```

Important: Do not attempt to install and run MTK in a shared environment (for example, **/usr/local/mtk**). If multiple users will run MTK on the same system, they should install and run their own copies, using projects and files local to their home directory. Sharing projects and logs can result in conflicts and overwritten files.

2. Create a new directory and download the IBM Migration Toolkit into the newly created directory.
3. Untar and extract the `mtk.tar.gz` file into the MTK directory that you specify by issuing the following command:

   ```
   tar -xzf mtk.tar.gz
   ```
4. Verify that you can access Java and that it is the correct version:

   ```
   java -version
   ```

Related tasks

"Starting on UNIX or Linux"
You can start the IBM Migration Toolkit (MTK) on the UNIX or Linux operating system.

Starting MTK

The process for starting the IBM Migration Toolkit (MTK) varies depending on the type of operating system that you are using.

Starting on Windows

You can start the IBM Migration Toolkit (MTK) on the Microsoft Windows operating system.

To start MTK on the Microsoft Windows operating system:

- From the Start menu, select **Programs** → **IBM Migration Toolkit** and choose either **Toolkit** or **Wizard**.
- From a command prompt, navigate to the directory where MTK is installed and type `MTKMain.bat` and press **Enter**.

Related tasks

"Installing on Windows" on page 4
MTK can be installed on the Windows operating system.

Starting on UNIX or Linux

You can start the IBM Migration Toolkit (MTK) on the UNIX or Linux operating system.

To start MTK on the UNIX or Linux operating system:

Navigate to the directory where MTK is installed and type `MTKMain.sh` and press the **Enter** key.

Related tasks

"Installing on UNIX and Linux" on page 5
MTK can be installed on UNIX and Linux operating systems.

Setting application preferences

You can change the behavior of some aspects of the IBM Migration Toolkit by changing the application preferences.

To set preferences:

1. Select **Application** → **User Preferences**.
2. Modify the preferences as desired.
3. Click **OK**.

Related tasks

"Importing source files" on page 24
You can specify that an external file be translated if the file contains valid source SQL.

"Closing a project" on page 14
You can close a project at any time.

User preference options

The IBM Migration Toolkit (MTK) provides a number of customizable user preference options.

The following table describes the MTK user preference options.

Option	Description
Open the launchpad at startup	Select this option to have the launchpad open on startup rather than the MTK application. The launchpad helps introduce the application to new users and offers the following options: • View a quick product overview • Quickly convert a database using the wizard • Launch the Migration Toolkit product • Exit
Save desktop setting on close	Select this option to have the size and position of the windows saved on exit.
Enable pop-up help (takes effect at restart of application)	Select this option to enable extended pop-up window help for interface controls.
Save current project on close	Select this option to save all information associated with the currently open project upon exiting the application.
Reload previous project on open	Select this option to automatically open the previously opened project when the application is started.
Preset editor	Select this option to use one of the following editors: • Refine Editor • JLpex • Notepad
External editor	Select this option to specify the editor to use when opening text files from the within MTK.

Option	Description
System look-and-feel	Select this option to use the file selection window that is provided with the system, when you select files from within MTK.
Java look-and-feel (allows selection of multiple files)	Select this option to use the file selection window provided with the Java runtime environment whenever you select files from within MTK. This particular file selection window allows you to select multiple files.
Automatically go to the Refine page after conversions	Choosing this option enables MTK to move directly to the Refine page after converting your data on the Convert page.
Data Extraction: use ″fast extract″ feature	Select this option for faster data extractions, which are achieved by using more advanced queries.

Keyboard shortcuts

Action	Control
To move forward between fields and controls	Tab
To move backward between fields and controls	Shift-Tab
To activate a button, indicate the completion of entering text in a single-line field, or activate an item in a list	Enter
To toggle a check box or radio button as checked or unchecked	Spacebar
To view a combo box menu	Alt-Down
To retract a combo box menu	Esc Alt-Up
To select items in a combo box	• Down, Up (after menu is expanded) • Type the initial character of the desired item
To select items in a list, spinner list, or menu	Down, Up
To select all items in a list	Ctrl-A
To select, clear additional items in a list	Ctrl-Spacebar
To select, clear a range of items in a list	Shift-Spacebar
To move to the adjacent cell in a table	Right, Left, Down, Up, and Tab, Shift-Tab, Enter, Shift-Enter respectively
To Navigate in and out of a tree	Tab, Shift-Tab
To move to the adjacent node in a tree	Up, Down
To expand and collapse a node in a tree	Right, Left
To navigate out of a menu	Esc
To show a menu	Enter, Spacebar, Alt-Char accelerator key (if defined)
To show a popup menu	Shift-F10
To activate help	F1

Types of MTK user interfaces

The IBM Migration Toolkit (MTK) is available from three types of user interfaces.

MTK can be used from either of the following user interfaces:
- “Graphical user interface”
- “Wizard”
- “Command line”

Graphical user interface

You can invoke MTK using the graphical user interface (GUI).

The GUI interface offers the MTK migration functionality using a Java interface. It is customizable and easy to use, especially for those unfamiliar with the command line interface or database migrations in general. The process for using the MTK GUI is documented in “The GUI process” on page 23.

Related concepts

“The GUI process” on page 23
The MTK graphical user interface contains five tabs, which each represent a step in the migration process. You can migrate to either IBM DB2 or IBM Informix Dynamic Server. For a list of supported source and target databases, see the MTK readme file.

Wizard

You can invoke MTK using the wizard.

Related tasks

“The wizard process” on page 107
The MTK wizard offers a streamlined version for use in simple database migrations.

Command line

You can invoke MTK using the command line.

Related concepts

“The command line process” on page 73
The command line interface offers a way to operate MTK from the command line using a configuration file and arguments.

Migration projects

The IBM Migration Toolkit (MTK) enables you to manage your migration through ″projects,″ which enables you to contain all of the information associated with a migration. Project files are organized in a project directory.

Each project directory contains the defining project file, and as you progress through the migration, the directory contains imported and extracted source SQL files, converted SQL files, data files (in the DataOutScripts subdirectory), and deployment scripts.

Projects help organize separate migrations. They can also be used to separate a migration into parts that require different settings.

Related tasks

"Creating a project" on page 12
When you start MTK for the first time, you are prompted to create a project. You can also create a new project at any time.

"Opening an existing project" on page 13
You can open an existing project when you start the IBM Migration Toolkit or when another project is open. If a project is already open, MTK automatically saves and closes that project.

"Closing a project" on page 14
You can close a project at any time.

"Saving a project" on page 13
Projects are automatically saved periodically during the migration and when you exit the application. You can also save a project at any time.

"Importing source files" on page 24
You can specify that an external file be translated if the file contains valid source SQL.

"Modifying project description" on page 14
You can modify the description of an open project.

"Deleting a project" on page 14
You can drop either the currently opened project or another project. When you drop a project, the project directory and all its contents are deleted, including the imported source files, converted source, logs, and reports.

"Backing up a project" on page 15
You can back up a project directory into another directory or a zip file. It is recommended that you back up your projects on a regular basis.

"Restoring a project" on page 15
If a project is lost or becomes corrupted, you can rebuild it from a backup copy of the project. Therefore, it is recommended that you back up projects regularly.

Project files

You can manage migration projects the same way you manage any directory on the file system, all of the information for a particular migration project is contained within its own project directory. The default location for projects is in the projects subdirectory in the directory in which MTK is installed. However, you can save project directories anywhere on your local or network file system.

MTK automatically saves projects to ensure consistency among the project files and the project description (for example, when you open another project while the current project is active). Projects are also automatically saved when you exit MTK, provided that the corresponding fields in the Application preferences window are selected.

Related tasks

"Saving a project" on page 13
Projects are automatically saved periodically during the migration and when you exit the application. You can also save a project at any time.

Project archives

It is recommended that you back up a project.

For example, if you changed your source files and also modified a previous database conversion, you might want to back up before converting the new source in case the new refinement technique does not produce the desired results. MTK provides project back up and restore capabilities for this purpose.

Related tasks

"Backing up a project" on page 15
You can back up a project directory into another directory or a zip file. It is recommended that you back up your projects on a regular basis.

"Restoring a project" on page 15
If a project is lost or becomes corrupted, you can rebuild it from a backup copy of the project. Therefore, it is recommended that you back up projects regularly.

Creating a project

When you start MTK for the first time, you are prompted to create a project. You can also create a new project at any time.

To create a project:

1. Select **Project** → **New**.
2. From the New Project window, type a name for the new project in the **Project Name** field.
3. Optional: To change the directory in which you want the project saved, type a new path in the **Project Path** field. The default location for your projects is `install_dir\projects`, where *install_dir* is your local MTK installation directory. For example, `C:\MTK\149\projects`.
4. Optional: Type descriptive information about the project that you are creating in the **Project description** field.
5. Select the source database version from the **Source database** list.
6. Select the target database from the **Target database** list.
7. Click **OK**.

Related concepts

"Migration projects" on page 11
The IBM Migration Toolkit (MTK) enables you to manage your migration through ″projects,″ which enables you to contain all of the information associated with a migration. Project files are organized in a project directory.

Opening an existing project

You can open an existing project when you start the IBM Migration Toolkit or when another project is open. If a project is already open, MTK automatically saves and closes that project.

To open a project:

1. Open the Open Project window:
   - When MTK starts, click **Open a project** from the Project Management window.
   - Select **Project** → **Open**.
2. From the Open project window, specify a project file name:
   - Select a project file name from the list of names shown.
   - To browse the file system for the file, click ... .
   - Type a name directly into the **Existing file** field.
3. Optional: Type information about the project that you are creating into the **Description** field.
4. Click **OK**.

   Related concepts

   "Migration projects" on page 11
   The IBM Migration Toolkit (MTK) enables you to manage your migration through ″projects,″ which enables you to contain all of the information associated with a migration. Project files are organized in a project directory.

   Related tasks

   "Setting application preferences" on page 7
   You can change the behavior of some aspects of the IBM Migration Toolkit by changing the application preferences.

Saving a project

Projects are automatically saved periodically during the migration and when you exit the application. You can also save a project at any time.

To save a project:

Select **Project** → **Save**.

Related concepts

"Migration projects" on page 11
The IBM Migration Toolkit (MTK) enables you to manage your migration through ″projects,″ which enables you to contain all of the information associated with a migration. Project files are organized in a project directory.

"Project files" on page 11
You can manage migration projects the same way you manage any directory on the file system, all of the information for a particular migration project is contained within its own project directory. The default location for projects is in the projects subdirectory in the directory in which MTK is installed. However, you can save project directories anywhere on your local or network file system.

Closing a project

You can close a project at any time.

To close a project:

Select **Project** → **Close**.

When you close a project, all the information associated with it is saved in the project folder.

Related concepts

"Migration projects" on page 11
The IBM Migration Toolkit (MTK) enables you to manage your migration through ″projects,″ which enables you to contain all of the information associated with a migration. Project files are organized in a project directory.

Related tasks

"Setting application preferences" on page 7
You can change the behavior of some aspects of the IBM Migration Toolkit by changing the application preferences.

Modifying project description

You can modify the description of an open project.

To modify the project description:

1. From within the main MTK window, click **Project** → **Modify**.
2. From the Modify project window, type any new information into the **Project description** field.
3. Click **OK**.

Related concepts

"Migration projects" on page 11
The IBM Migration Toolkit (MTK) enables you to manage your migration through ″projects,″ which enables you to contain all of the information associated with a migration. Project files are organized in a project directory.

Deleting a project

You can drop either the currently opened project or another project. When you drop a project, the project directory and all its contents are deleted, including the imported source files, converted source, logs, and reports.

Important: You cannot undo a project deletion. Therefore, do not drop a project unless you are absolutely certain that you do not need it.

Related concepts

"Migration projects" on page 11
The IBM Migration Toolkit (MTK) enables you to manage your migration through ″projects,″ which enables you to contain all of the information associated with a migration. Project files are organized in a project directory.

Deleting the currently open project

You can delete a project that you currently have open.

To delete the currently opened project:

Select **Project** → **Drop** → **current project**.

Deleting projects that are not currently open

You can delete a project that is not currently open.

To delete a project that is not currently open:

1. Select **Project** → **Drop** → **other projects**.
2. From the Drop project window, select a recently opened project or click ... to find one on the file system. Once specified, the project is displayed in the field to the left of the ... button.
3. Click **Drop**.

Backing up a project

You can back up a project directory into another directory or a zip file. It is recommended that you back up your projects on a regular basis.

To back up a project:

1. Open the project that you want to back up.
2. Select **Project** → **Backup**.
3. From the Project Backup window, type a unique name for the backup in the **Name of your backup field**.
4. Optional: If you have extracted data for the project, click **Back up data** to include the data in the archive.
5. Optional: If the project is relatively small and contains no extracted data files, you can clear **Zip archive** to archive with no compression.
6. Click **OK**.

The files associated with the project are backed up in the project directory, into a new subdirectory with the specified backup name. The original files remain at their initial location.

Related concepts

"Migration projects" on page 11
The IBM Migration Toolkit (MTK) enables you to manage your migration through ″projects,″ which enables you to contain all of the information associated with a migration. Project files are organized in a project directory.

"Project archives" on page 12
It is recommended that you back up a project.

Restoring a project

If a project is lost or becomes corrupted, you can rebuild it from a backup copy of the project. Therefore, it is recommended that you back up projects regularly.

Important: During restoration, all information in the currently open project is overwritten by the archive. Close all open projects before restoring.

To restore a project:

1. Open the project to be restored.
2. Click **Project** → **Restore**.
3. In the window that opens, type or select the name of the backup file to be restored.
4. Click **OK**.

   Related concepts

   “Migration projects” on page 11
   The IBM Migration Toolkit (MTK) enables you to manage your migration through ″projects,″ which enables you to contain all of the information associated with a migration. Project files are organized in a project directory.

   “Project archives” on page 12
   It is recommended that you back up a project.

Modifying user preferences

You can set your MTK user preferences.

To set the user preferences:

1. Select **Applications** -> **User Preferences**.
2. Choose your preferences. For a complete list of user preference options, see “User preference options” on page 7.
3. Click **OK**.

The MTK process

The MTK process varies depending on which MTK user interface you choose.

Connecting to the source database

Before you extract data, you need to connect to the source database.

If you are using JDBC or ODBC to connect to the source database, you should consider the following:

- The ODBC and JDBC drivers treat VARCHAR data differently during transfer. Because ODBC trims any trailing spaces, fields of only spaces are transferred as empty strings. Therefore, for MTK data movement involving VARCHAR, use the JDBC driver when all of the following are true:
  - VARCHAR data is to be transferred.
  - Data ends with spaces.
  - Trailing spaces are significant and should not be trimmed.
- If you connect to a database using JDBC and then want to reconnect to another database using ODBC, you must first restart MTK.

Connecting to Oracle

You can connect to an Oracle source database using either a JDBC or ODBC connection.

- Depending on the target database, the following types of connections are supported:

Target database	JDBC	ODBC
DB2	Yes	Yes
Informix Dynamic Server	Yes	No

- Here are the driver details:

Driver	Details
ojdbc14.jar	A JDBC driver available for download from http://www.oracle.com/technology/software/tech/java/sqlj_jdbc/index.html.
ODBC driver	An ODBC driver available for download from http://www.oracle.com/technology/software/tech/windows/odbc/index.html.

To connect to an Oracle database using JDBC:

Prerequisite: The client must be configured and ojdbc14.jar must be in the system CLASSPATH variable, accessible by MTK.

1. In the **Service/ODBC DSN Alias** field, specify the Oracle service name that you want to establish. The client connection information must already be defined. Refer to the client connection configuration instructions for information on how to create a service name.
2. Specify the Native driver information:

   a. Select **Use native JDBC driver** to use the JDBC driver provided by Oracle. If the option is unavailable, then the necessary driver was not found in classpath. Ensure that the `ojdbc14.jar` file is located in the classpath.
   b. In the **Host IP address** field, specify the DNS name or IP address of the host server to connect to.
   c. In the **Port number** field, specify the server TCP/IP port through which you want to connect to the source database.
3. Specify the Database User information:
   a. Select **Remember password** if you want the user ID and password to be retained for future connections.
   b. In the **User ID** field, specify the user ID for the connection.
   c. In the **Password** field, specify the password for the connection.
4. Click **OK**.

To connect to an Oracle database using ODBC:

Prerequisite: Create an ODBC DSN per the instructions from your ODBC vendor.
1. In the **Service/ODBC DSN Alias** field, specify the ODBC DSN given to the connection that you want to establish. The client connection information must already be defined. Refer to the client connection configuration instructions for information on how to create a service name.
2. Specify the Database User information:
   a. Select **Remember password** if you want the user ID and password to be retained for future connections.
   b. In the **User ID** field, specify the user ID for the connection.
   c. In the **Password** field, specify the password for the connection.
3. Click **OK**.

   Related tasks

   "Extracting objects from a source database" on page 24
   You can specify that an extracted object be translated. The objects are extracted to a file that MTK can translate.

Connecting to Informix Dynamic Server

You can connect to an Informix Dynamic Server source database using either a JDBC or ODBC connection.
- Here are the ways to connect to source database Informix Dynamic Server:

Name	Details
Informix JDBC driver	The Informix JDBC Type 4 driver is included with the distribution of MTK and no extra configuration on the MTK system is required.
DB-Access	To use DB-Access, the Informix environment must be set before you start MTK.
ODBC driver	The ODBC driver is included with IBM Informix Connect or IBM Informix Client SDK (CSDK).

To connect to an Informix Dynamic Server database using JDBC:
1. Specify the Native driver information:

   a. Select **Use native JDBC driver** to use the JDBC driver provided by Informix Dynamic Server. If the option is unavailable, then the necessary driver was not found in the class path. Ensure that the `ifxjdbc.jar` file is located in the classpath.
   b. In the **Host IP address** field, specify the DNS name or IP address of the host server to connect to.
   c. In the **Port number** field, specify the server TCP/IP port through which you want to connect to the source database.
   d. In the **Server name** field, specify the name of the source database or instance. The server name is case-sensitive.
   e. In the **Client Locale** field, specify the locale of the client that is accessing the database. If you leave this field blank, the IBM Informix JDBC driver default value for CLIENT_LOCALE is used. The locale is added to the JDBC URL as the CLIENT_LOCALE connection property.
   f. In the **Database Locale** field, specify the locale of the database. The JDBC Driver uses this variable to perform code-set conversion. If this field is left blank, the JDBC driver default value for DB_LOCALE is used. The locale is added to JDBC URL as the DB_LOCALE connection property.
   g. **Optional:** In the **Advanced properties** field, specify connection properties supported by JDBC driver. If you specify properties, you must provide the values in a semicolon-separated list of key value pairs. For example, specify:

      ```
      INFORMIXCONTIME=5;JDBCTEMP=/tmp
      ```
2. In the **Database name** field, specify the alias of the Informix Dynamic Server database connection that you want to establish. The database name is case sensitive. The client connection information must already be defined. Refer to the client connection configuration instructions for information on how to create an alias.
3. Specify the Database User information:
   a. Select **Remember password** if you want the user ID and password to be retained for future connections.
   b. In the **User ID** field, specify the user ID for the connection.
   c. In the **Password** field, specify the password for the connection.
4. Click **OK**.

To connect to an Informix Dynamic Server database using ODBC:

Prerequisite: The ODBC DSN must be configured as prescribed by the client software for the data source.

1. In the **ODBC DSN Alias** field, specify the alias given to the connection that you want to establish. The client connection information must already be defined. Refer to the client connection configuration instructions for information on how to create an alias.
2. Specify the Database User information:
   a. Select **Remember password** if you want the user ID and password to be retained for future connections.
   b. In the **User ID** field, specify the user ID for the connection.
   c. In the **Password** field, specify the password for the connection.
3. Click **OK**.

   Related tasks

"Extracting objects from a source database" on page 24
You can specify that an extracted object be translated. The objects are extracted to a file that MTK can translate.

Connecting to MySQL

You can connect to a MySQL source database using a JDBC connection.

For connecting to MySQL, download the JDBC Driver for MySQL (Connector/J) for your version of MySQL:

Driver	Details
MySQL version 4: JDBC driver (Connector/J 3.1), 3.1.14	Available for download from http://www.mysql.com/products/connector/.
MySQL version 5: JDBC driver (Connector/J 5.0), 5.0.5	Available for download from http://www.mysql.com/products/connector/.

To connect to a MySQL database using JDBC:

Prerequisite: The client must be configured and its JAR file must be in the system CLASSPATH variable, accessible by MTK.

1. Specify the Native driver information:
   a. Select **Use native JDBC driver** to use the JDBC driver provided by MySQL. If the option is unavailable, then the necessary driver was not found in classpath. Ensure that the correct JDBC jar file is located in the CLASSPATH variable.
   b. In the **Host IP address** field, specify the DNS name or IP address of the host server to connect to.
   c. In the **Port number** field, specify the server TCP/IP port through which you want to connect to the source database.
2. In the **Database name** field, specify the alias of the MySQL database connection that you want to establish. The database name is case sensitive. The client connection information must already be defined. Refer to the client connection configuration instructions for information on how to create an alias.
3. Specify the Database User information:
   a. Select **Remember password** if you want the user ID and password to be retained for future connections.
   b. In the **User ID** field, specify the user ID for the connection.
   c. In the **Password** field, specify the password for the connection.
4. Click **OK**.

   Related tasks

   "Extracting objects from a source database" on page 24
   You can specify that an extracted object be translated. The objects are extracted to a file that MTK can translate.

Connecting to Microsoft SQL Server

You can connect to a Microsoft SQL Server source database using a JDBC or ODBC connection.

- Here are the driver details:

Driver	Details
SQL Server 2000 Driver for JDBC	A JDBC Type 4 driver available for download from http://www.microsoft.com/downloads/.
Microsoft SQL Server 2005 JDBC Driver 1.0	A JDBC Type 4 driver available for download from http://www.microsoft.com/downloads/.
ODBC driver	Included in your Microsoft SQL Server installation.

To connect to Microsoft SQL Server database using JDBC:

Prerequisite: The client must be configured and its JAR file must be in the system class path, accessible by MTK.

1. Specify the Native driver information:
   a. Select **Use native JDBC driver** to use the JDBC driver provided by Microsoft SQL Server or Sybase. If the option is unavailable, then the necessary driver was not found in classpath. Ensure that the correct JDBC jar files for Microsoft SQL Server or Sybase file are located in the classpath.
   b. In the **Host IP address** field, specify the DNS name or IP address of the host server to connect to.
   c. In the **Port number** field, specify the server TCP/IP port through which you want to connect to the source database.
   d. Type the connection string as prescribed by the JDBC client.
2. Specify the Database User information:
   a. Select **Remember password** if you want the user ID and password to be retained for future connections.
   b. In the **User ID** field, specify the user ID for the connection.
   c. In the **Password** field, specify the password for the connection.
3. Click **OK**.

To connect to Microsoft SQL Server database using ODBC:

Prerequisite: The ODBC DSN must be configured as prescribed by the client software for the data source.

1. In the **JDBC/ODBC DSN Alias** field, specify the alias given to the connection that you want to establish. The client connection information must already be defined. Refer to the client connection configuration instructions for information on how to create an alias.
2. Specify the Database User information:
   a. Select **Remember password** if you want the user ID and password to be retained for future connections.
   b. In the **User ID** field, specify the user ID for the connection.
   c. In the **Password** field, specify the password for the connection.
3. Click **OK**.

   Related tasks

   "Extracting objects from a source database" on page 24
   You can specify that an extracted object be translated. The objects are extracted to a file that MTK can translate.

Connecting to Sybase

You can connect to a Sybase source database using a JDBC or ODBC connection.

- Here are the driver details:

Driver	Details
Sybase jConnect for JDBC 6.0 or later	Available for download from http://www.sybase.com/products/allproductsa-z/softwaredeveloperkit/jconnect.
ODBC driver	Included in your Sybase server installation.

To connect to Sybase database using JDBC:

Prerequisite: You must have jConnect for JDBC 6.0 installed and jconn3.jar accessible through the system class path.

1. Specify the Native driver information:
   a. Select **Use native JDBC driver** to use the JDBC driver provided by or Sybase. If the option is unavailable, then the necessary driver was not found in classpath. Ensure that the correct JDBC jar files for Sybase file are located in the class path variable.
   b. In the **Host IP address** field, specify the DNS name or IP address of the host server to connect to.
   c. In the **Port number** field, specify the server TCP/IP port through which you want to connect to the source database.
   d. Type the connection string as prescribed by the JDBC client.
2. Specify the Database User information:
   a. Select **Remember password** if you want the user ID and password to be retained for future connections.
   b. In the **User ID** field, specify the user ID for the connection.
   c. In the **Password** field, specify the password for the connection.
3. Click **OK**.

To connect to Sybase database using ODBC:

Prerequisite: The ODBC DSN must be configured as prescribed by the client software for the data source.

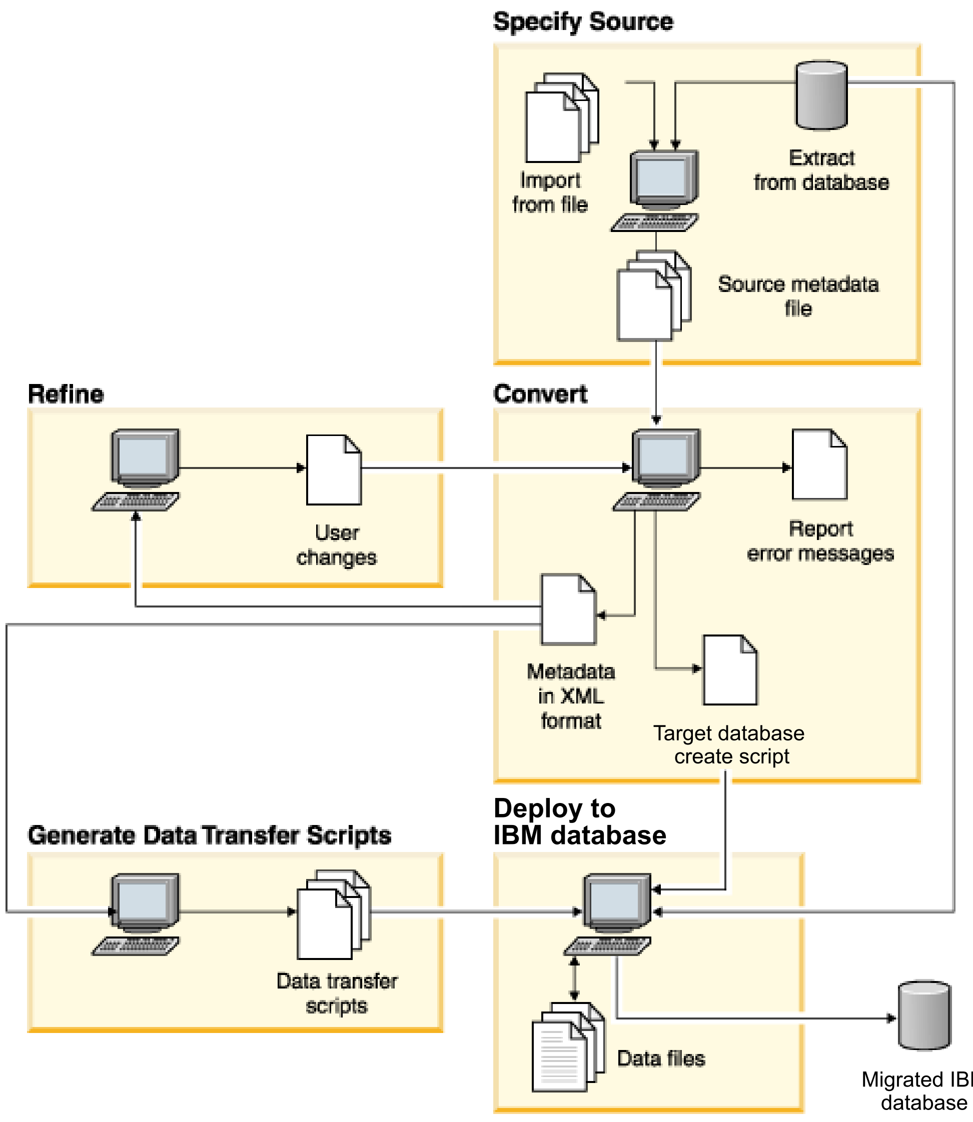
1. In the **JDBC/ODBC DSN Alias** field, specify the alias given to the connection that you want to establish. The client connection information must already be defined. Refer to the client connection configuration instructions for information on how to create an alias.
2. Specify the Database User information:
   a. Select **Remember password** if you want the user ID and password to be retained for future connections.
   b. In the **User ID** field, specify the user ID for the connection.
   c. In the **Password** field, specify the password for the connection.
3. Click **OK**.

   Related tasks

   "Extracting objects from a source database" on page 24
   You can specify that an extracted object be translated. The objects are extracted to a file that MTK can translate.

The GUI process

The MTK graphical user interface contains five tabs, which each represent a step in the migration process. You can migrate to either IBM DB2 or IBM Informix Dynamic Server. For a list of supported source and target databases, see the MTK readme file.



Related concepts

"Graphical user interface" on page 9
You can invoke MTK using the graphical user interface (GUI).

Step 1: Specifying the source

After a migration project is created or opened, you can start the migration process. Your first step is use the Specify Source page to obtain the source files to be converted to the SQL of the target database server.

You can extract from a source database through a database server connection or you can import metadata from a source file. Multiple source files can be specified and converted.

MTK supports source SQL and data represented with double-byte character sets (DBCS), with the following restrictions:

- Conversion of functions that manipulate characters might be incomplete and could require manual intervention. For example, DB2 UDB built-in string functions do not handle DBCS character counts in the same manner as the source DBMS. Refer to the DB2 UDB documentation for details.
- Currency symbols, decimal separators, and month and weekday names are not localized.

Importing source files

You can specify that an external file be translated if the file contains valid source SQL.

Files containing metadata extracted by using MTK are always valid. Some other extraction tools also work, such as the one used with Sybase Central.

To import a source file:

1. From the **Specify Source** page, click **Import**.
2. From the Import file window, select the desired source file and click **Import file**. The imported file is copied into the project directory and listed at the right on the Specify Source page.
   - If the application preferences have been set to use the Java look-and-feel file selection window, you can select more than one file to be imported at the same.
   - A file manually moved into the project directory must still be imported so that the contents of the file can be loaded into the MTK object tree.
   - The file name can contain only alphanumeric and underscore (_) characters. If the file name contains any other characters, a new file name is created with the underscore replacing the invalid characters. For example, ″$srce name″ is imported as_srce_name.

Related tasks

"Setting application preferences" on page 7
You can change the behavior of some aspects of the IBM Migration Toolkit by changing the application preferences.

Related reference

"Software requirements" on page 3
The software required to use MTK is described in this topic.

Extracting objects from a source database

You can specify that an extracted object be translated. The objects are extracted to a file that MTK can translate.

Prerequisites:

- Connect to the source database.
- **For source database Informix Dynamic Server**:
  - The database instance must be configured in advance to run Java user-defined routines (UDR) because MTK deploys Java UDRs to Informix Dynamic Server during migration.

  - Some problems can occur when extracting from Informix Dynamic Server, Version 7.
- **For source database Oracle**:
  - Before extracting objects from an Oracle database, run statistics on the following database system tables: SYS.DEPENDENCY$, SYS.OBJ$, and SYS.USER$. You can use the DBMS_STAT package or DBA Studio to run statistics. If you do not run statistics, the extraction process is much slower.
  - When extracting objects and data from an Oracle database the following user permissions are required:

Objects	**Permissions**
Procedures, Functions, Packages	CREATE ANY PROCEDURE
Triggers	CREATE ANY TRIGGER
Tables (data extraction)	SELECT ANY TABLE

- **For source database Sybase**:
  - The client code does not support the LC_ALL system variable. If the variable has been defined, remove it before running the Sybase client.
  - If you receive error messages 010MX, 010SJ, or 010SK after connecting to Sybase, disregard these messages. MTK does not depend on the Sybase server stored procedures for its extractions.
  - Space and performance management are not translated. Use the DB2 UDB facilities to adapt the Sybase configuration to the new DB2 UDB environment.

To extract objects from the source database:

1. From the **Specify Source** page, click **Extract**. If a database connection does not exist for this project, the Connect to Database window will open. Follow the step-by-step instructions for connecting to your source database: “Connecting to the source database” on page 17.
2. From the Extract window, select which objects you want to extract from the database.
3. Type a name for the extraction in the **File name** field. This name will be used to identify the files that are associated with the extraction. The file name can contain only alphanumeric and underscore (_) characters.
4. Optional: Select other options:

Option	**Description**
Create one file per stored procedure	This option lists each stored procedure as a separate file in the project subdirectory. If **Include other needed objects** is also specified, the necessary tables, views, and data types will be placed in the root extraction file. Procedure specifications will be listed above the place from which they are called.

Option	Description
Include other needed objects	This option specifies whether all object dependencies should be included in the extraction. For example, if I select procedure *p* for extraction and it references table *t*, table *t* will also be extracted. It is possible that some required objects might not be included even though this control is selected. For example, system tables are never extracted. In some cases, source catalog tables are not always accurately maintained by the source database system. **Note:** This option is only applicable for procedures and triggers. It is not applicable for other database objects such as tables or views. In the case of tables, other tables that are connected by referential integrity are not extracted. In the case of views, the table on which the view is based is not extracted. This option is designed to allow you to target specific objects, for example to test migration scenarios. If you are migrating a large database with many objects, you will most likely want to separate the migration into manageable files, converting the tables first, followed by triggers, procedures, and other objects. Unless you are aware of every reference to each object, do not use the **Include other needed objects** option in a full migration. If you do, the same object will likely be redefined many times, in which case MTK will throw an error during conversion each time it encounters a duplicate definition.
Make context file	Select this option to have any other needed objects to be put into a file with a .context extension. The context file is put at the top of the list in the window on the extractor panel because these objects are required by statements in the .src file.
Connect to database	Used for multiple extractions. Select this option if you want to connect to a different database server while in the Extract window. The Connect to Database window opens, where you can specify a different alias, user ID, and password for the new connection. The IBM Informix JDBC driver does not support shared memory connections. If an Informix Dynamic Server instance uses shared memory connections, you must create an alias that uses TLI or socket protocol for MTK to use.

Option	Description
Refresh available objects	Updates the list of available objects from the current source database. You must refresh the objects: • To update any changes that have occurred to the database after you initially connected • After making a new database connection
Sybase **Set quoted_identifier on**	Select this option if any of the selected objects include spaces in the names, or if the objects were created in a database session with QUOTED_IDENTIFIER ON. This option should be used only to extract individual objects that require it.

5. Click **Extract**.

   Related tasks

   "Connecting to the source database" on page 17
   Before you extract data, you need to connect to the source database.

   Related reference

   "Software requirements" on page 3
   The software required to use MTK is described in this topic.

Deleting a source file

Files can be deleted from the list on the Specify source page.

If the deleted file was previously used to convert to a target database server, the conversions associated with the file will be deleted along with the source file (deleted files include files used in context). For example, if the file basetables_defs.src is used in context for three separate conversions (employees, sales, and admin), all of the files associated with the three conversions will be deleted when the basetables_defs.src file is deleted.

To delete a source file:

1. From the Specify source page, select a source file from the file list on the right.
2. Click **Delete**.

   Related reference

   "Software requirements" on page 3
   The software required to use MTK is described in this topic.

Specifying the order of the source files

During the conversion process, source files are converted in the order they are listed. The MTK parser only recognizes objects that have been defined. Therefore, when converting multiple files, you must put them in the correct order.

There are two options you can choose from to arrange your source files:

- ***To sort files by name or file extension:***
  1. From the Specify Source page, click **Sort By Name** or **Sort By Extension**.
- ***To arrange source files individually:***
  1. From the Convert page, you can select individual files and click **Move File Up** or **Move File Down** to arrange them.

  Related reference

"Software requirements" on page 3
The software required to use MTK is described in this topic.

Step 2: Converting source metadata

The Convert step converts source metadata to target database metadata. You can specify various options that affect the converted output before you convert source SQL into DB2 UDB or Informix Dynamic Server SQL.

Prerequisites

- At least one source file must be specified in the left column on the Convert page.
- In conversions from Microsoft SQL Server, Sybase, and Sybase SQL Anywhere:
  - If you are reconverting existing projects and you have previously changed the mapping for CHAR and NCHAR, ensure that you explicitly map them before reconverting. You must do this because these mappings have changed.
- In conversions from Oracle:
  - Package bodies are translated, however, procedure and function specifications are not. Procedures and functions cannot be referenced until they have been defined.
- In conversions from MySQL:
  - MySQL allows NULL to be inserted in NOT NULL AUTO_INCREMENT columns, however, MTK does not support this.
  - BLOB and CLOB data types are incorrectly deployed in conversions to DB2.
  - BLOB and TEXT data movement is not supported in conversions to DB2 and Informix Dynamic Server.

To convert source metadata:

1. From the **Convert** page, the source file specified in "Step 1: Specifying the source" on page 23, is listed on the left. If you imported more than one source file, all of the files are listed.
2. Optional: Specify the order of the source files.
3. Click the file you want to convert (Ctrl-click to select multiple files).
4. Optional: Map data types.
5. Optional: Modify the options on this page:

Option	Explanation
Prefix for generated files	Type the prefix that you want to use for the converted file names. The default value is the name of the first source file selected.

Option	Explanation

Source date format	Select or type the format that you want the date constants to be in when they are converted. The format must match that used in the source.
	
	IDS The value specified depends upon the DBDATE environment variable. If DBDATE is not specified for the source database, the default date format for Informix Dynamic Server is `MDY4/`.
	
	`Form:   Example:`
	`MDY4/   12/03/2004`
	`DMY2-   03-12-2004`
	`MDY4    12/03/2004`
	`Y2DM.   04.03.12`
	`MDY20   120304`
	`Y4MD    2004/12/03`

Source data language	Select the source data language from the list of available languages.
Sybase **Target variable prefix**	This option is only available for conversions from Sybase or SQL Server. If a source variable begins with a prefix of @, the prefix must be changed to an acceptable prefix for the target database. The default prefix is v_. For example, if you do not change the prefix, `@obj` becomes `v_obj` after conversion to DB2 UDB. If you want to chose your own prefix, type the prefix into the **Target variable prefix** field. For example, type `my` into the field. This results in `@obj` becoming `myobj` after conversion to DB2 UDB.

Option	Explanation
Default source schema	Specify the object name qualifier that you want to be used as the default target database schema. (If a source database is extracted, the list is populated with the name qualifiers that are available in the source file.) The name you choose identifies those objects that you want to belong to the default target database schema, and will therefore have no schema name assigned. If you choose *from_first_object*, the name qualifier of the first object encountered in the source file is used as the default. Objects that do not have a qualifying name are given one of the following default schema names: • IDS informix • Sybase SQL Server dbo • Oracle dba You can force every object to be given a schema name by selecting *specify_schema_for_all_objects* or by entering an unused qualifier as the **Default source schema**. You can force a particular schema name for a set of objects by including a CONNECT SCHEMA_NAME statement in the source file. All objects that follow the connect statement are assigned the specified schema name. See the examples shown after these steps. When the default schema entry field in the convert step is set to ″from_first_object,″ the translator uses the schema from the first object as the schema to be replaced by the IBM target user ID that was used in deployment. This means that even if this schema for the first object is refined later in Step 3, it does not take effect because the IBM target user ID is used during deployment and verification steps.
Set DELIMIDENT	Select this option to indicate that the source SQL contains object names that use delimited identifiers (case-sensitive names within quotation marks, which can contain white space and other special characters). If you are converting from Informix Dynamic Server, this setting must match the setting of the Informix DELIMIDENT environment variable setting for the source SQL.

Option	Explanation
Sybase **Ignore case in user-defined identifiers**	This option is available only for conversions from Sybase or Microsoft SQL Server. A Sybase or SQL Server database can be defined as case sensitive; for example, table abc can be different from table Abc. In DB2 UDB, identifiers are not case sensitive unless they are quoted, so abc is the same as Abc, but ″abc″ is different from ″Abc″. Select this option to specify the source database as not being case sensitive (default for SQL Server). With this selection, both abc and Abc are converted to the same identifier. If you do not select this option, during conversion MTK will name abc and Abc distinctly, for example as ABC and ABC1.
Input file contains DBCS characters (incompatible with UTF-8)	Select this option if the object names contain DBCS characters.
View source	Click to display the source SQL file in an external file editor (defined in the Preference window). MTK does not modify the source file during conversion, but you can make changes using the editor and reconvert.
View output	Click to display the output SQL file in an external file editor (defined in the Preferences window). However, you can take advantage of many more features if you use the Refine page to view the conversion results.

6. Optional: “Referring to previously converted metadata” on page 34.
7. Optional: Click **Advanced options** to modify the advanced options:

Advanced Option	Explanation
Copy inter-statement source comments to the translated code	Select this option to have summary source text related to each statement within a stored procedure copied as comments to the translated code. Normally, the source text corresponding to each statement that is internal to a stored procedure is printed prior to the procedure’s translation.
Copy source (as comments) to translated code	Select this option to copy the original source as comments to statements that have been converted. Inter-statement comments are also copied.
Copy source (as comments) to output file if error occurs for a statement	During conversion, MTK copies source statements into the target SQL file. Select this option to copy only those source statements that apply to the message displayed.

Advanced Option	Explanation
Sybase SQL Server **Add source directives to translated output**	This option is available only for conversions from Sybase or Microsoft SQL Server. Select this option to automatically include source directives listed as comments before each converted statement. Do not select this option if you are concerned with the output file size.
Sybase ASA Oracle **Pad strings with spaces during comparison**	This option is available only for conversions from Sybase SQL Anywhere or Oracle. Select this option if you want string comparisons to evaluate to TRUE regardless of any extra spaces before or after the string.
Insert DROP statement before CREATE TABLE, VIEW and INDEX	Select this option to generate an appropriate DROP statement before each corresponding CREATE statement (INDEX, TABLE, or VIEW).
Sybase SQL Server **Convert sp_PrimaryKey and sp_ForeignKey to ALTER TABLE**	This option is available only for conversions from Sybase or Microsoft SQL Server. Select this option to convert these system procedures into ALTER TABLE statements to actually define the keys. Because these system procedures are frequently used for documentation purposes only, the default is that they not be converted to ALTER TABLE statements.
Insert DROP statement before translated CREATE PROCEDURE statements	In DB2 UDB, creating a procedure will fail if the procedure already exists. Select this option to drop the existing procedure before the new procedure is created. If the generated script is run multiple times, it is safe to drop the procedure before creating it. However, the DB2 UDB DROP PROCEDURE statement will generate an error if the procedure does not exist.
Copy full source for procedures before their translation	Select this option to copy the SQL for each source procedure as a comment before its associated translated SQL.
Copy source separately for statements in stored procedures	Select this option to copy statements of source procedures as comments between statements in the resulting translated SQL.
Sybase ASE SQL Server **Scope of temp table is global**	This option is available only for conversions from Sybase Adaptive Server Enterprise (ASE) or Microsoft SQL Server. Select this option to instruct the translator to first process all of the global temporary tables declared in the input script and treat them as global. This means references are resolved regardless of where they occur. However, if the script declares two global temporary tables that have the same name, the converter considers them duplicates and issues an error message.

8. Click **Convert**.

The result of each conversion is a DB2 UDB file (.db2) or an Informix Dynamic Server file (.ids) and a report file (.rpt), each containing the prefix specified in the **Prefix for generated files** field. The DB2 UDB or Informix Dynamic Server files contain the metadata created during the conversion, normally preceded by the source metadata as comments. The report file contains a list of errors identified during the conversion. You can view these files on the Refine page, which opens immediately after conversion.

The following examples show the resulting schema names in the generated output file.

- In this example, the source database is named alpha, the first object definition in the source file is CREATE TABLE john.r, followed by CREATE TABLE mary.s and CREATE TABLE t. The resulting names in the DB2 UDB file are as follows:

  IDS

  ```
  Default source schema     john.r     mary.s             t
  ---------------------     ------     ------    ----------
  from_first_object              r     mary.s    informix.t
  alpha:mary                john.r          s    informix.t
  unusedname                john.r     mary.s    informix.t
  ```

  Sybase

  ```
  Default source schema     john.r     mary.s             t
  ---------------------     ------     ------    ----------
  from_first_object              r     mary.s         dbo.t
  alpha.mary                john.r          s         dbo.t
  unusedname                john.r     mary.s         dbo.t
  ```

- In this example, the source database is named alpha, the first object definition in the source file is CREATE TABLE r, followed by CREATE TABLE mary.s and CREATE TABLE john.t. The resulting names in the DB2 UDB file are as follows:

  IDS

  ```
  Default source schema              r     mary.s        john.t
  ---------------------     ----------     ------    ----------
  from_first_object                  r     mary.s        john.t
  alpha:mary.t              informix.r          s        john.t
  unusedname                informix.r     mary.s        john.t
  ```

  Sybase

  ```
  Default source schema              r     mary.s        john.t
  ---------------------     ----------     ------    ----------
  from_first_object                  r     mary.s        john.t
  alpha.mary.t                   dbo.r          s        john.t
  unusedname                     dbo.r     mary.s        john.t
  ```

Related concepts

"How the converter works" on page 127
The converter is written in Java and uses ANTLR (ANother Tool for Language Recognition) as its parsing engine. It operates much like a compiler for a conventional programming language. It takes as input a sequence of source SQL scripts and generates a corresponding target SQL script as output.

Related reference

"The TABLESPACE clause" on page 215
When evaluating a CREATE TABLE statement, the translator will ignore any physical and table properties specified, without issuing a warning message.

"CREATE TABLE" on page 212
In a few situations, differences can occur in translations from Oracle to DB2 UDB and IBM Informix Dynamic Server.

Working on a project collaboratively

MTK is not designed for concurrent use on the same system. Use only one instance of MTK on the same project on the same system at the same time, or files and logs will become corrupt.

If multiple people are to work on the project, you can work as follows:

1. Divide the source into multiple files with one common context file. In this way the convert-refine process can be performed by multiple people on separate systems.
2. Consolidate the refined scripts onto a central system from which you can deploy them.

Reloading a converted file

You can select a previously converted file that is contained in the current project and restore the IBM Migration Toolkit to the point in the process where this file was last converted.

You can continue processing this file from this conversion point without having to perform an actual conversion.

To reload a converted file:

1. From the **Convert** page, in the Previous Conversions field, select the previously converted source file.
2. Click **Reload**.

Deleting a previous conversion

If you have performed many conversions, conversion files can clutter your project file. You can delete previous conversions.

Deleting a conversion removes it from the Previous conversions list and, except for any extracted data, erases all the files associated with the conversion.

To delete a previous conversion:

1. From the **Convert** page, in the **Previous Conversions** field, select the previously converted source file.
2. Click **Delete**.

Referring to previously converted metadata

If you are converting metadata that refers to other metadata that has already been converted, you still must include all of the files in the conversion. However, your translated data might be in a state where you would not want to translate it again. You can specify that the IBM Migration Toolkit use certain metadata *in-context*.

For example, if you want to convert triggers and views that refer to a table that has already been converted, you can specify the source file that contains the particular table definition to be used only in context, rather than being converted again.

Restrictions:

- Ensure the objects are only specified once. If you do not do this, an error will occur. For example, you might have a table specified in both an in-context file and as a supporting object in a procedure

source file that you extracted. To avoid this potential problem, give your source files meaningful names, particularly when you choose to include other needed objects when extracting from the source database.
- Metadata that is intended for use in-context must be specified at the file level. Plan your metadata accordingly.

1. From the **Convert** page, click **Set Context**.
2. Move the desired files to the Selected context files column, and click **OK**. The in-context files must be parsed before the files to be converted, so ensure they are at the top of the list by using **Move up** and **Move down** as necessary.

During deployment, the program attempts to compare all objects, including those found in the in-context files. The report shows an error for each object unless the object does, in fact, exist in the target database. This is as designed, since you will most likely use in-context objects only after they have already been deployed to the target database.

Related concepts

"How the converter works" on page 127
The converter is written in Java and uses ANTLR (ANother Tool for Language Recognition) as its parsing engine. It operates much like a compiler for a conventional programming language. It takes as input a sequence of source SQL scripts and generates a corresponding target SQL script as output.

Mapping data types

The IBM Migration Toolkit automatically maps source data types to target database data types. These default mappings are listed in a table provided in the Global Type Mapping window, where some of the mappings can be changed. If you are certain that the default mapping for a particular data type does not suit your migration, you can choose to change the mapping to something more appropriate.

Recommendation: If you are migrating from Oracle to Informix Dynamic Server and have columns that store very large amounts of binary or text data in each field, map these columns to BLOB or CLOB columns. Mapping columns with very large data fields to TEXT or BYTE columns might cause MTK to run out of memory. In addition, if you are mapping source data types to Informix Dynamic Server large object data types, check to be sure you created spaces for the large objects in Dynamic Server. You must do this before you can deploy data.

To change the default mapping for a data type:

1. From the **Convert** page, click **Global Type Mapping**. A window with a three-column table opens. The **Source type** and **Target type** columns provide data type mapping information.

   For example, the FLOAT data type in Sybase and SQL Server maps to the DOUBLE data type in DB2 UDB.

   If a **Target type** field contains an edit (pencil) icon, you can edit it.
2. In the **Target type** column, click the field that you want to edit.
3. Change the values as appropriate. Click **Apply**.
4. Sybase If you have already converted at least once, you can click **Restore previous mapping** to restore data type mappings to the mapping relationship used in the last conversion.

5. Click **Close**.

Example: Click the DECIMAL Target type on the SMALLMONEY row. A Data Type Editor window opens containing four fields: **SQL Type**, **Length**, **Precision**, and **Scale**. Click **SQL Type** to determine if an alternate name can be specified. The other fields can be changed if they contain a value. In this case, **Precision** and **Scale** can be changed.

Maintaining object dependencies

In general, MTK keeps track of object dependencies; however, statements that update object definitions cannot be guaranteed to translate correctly.

ALTER TABLE statements can be used as long as all operations on the referenced table occur after the ALTER TABLE statement in the script file. DROP statements are translated as is. However, they do not cancel any previous definition of the specified object in the converter; therefore, any subsequent CREATE statement definitions of an object of the same name will be considered as a duplicate definition of the object.

You can overcome this restriction by changing the source code before conversion, using one of the following methods:

Method 1
: If the second CREATE statement defines the same object as the first statement, remove the second statement and its preceding DROP statement, if one exists.

Method 2
: If the second CREATE statement defines a new object, use a different name for the object in the second definition and replace all references to the second object with the new name.

Method 3
: Break the original source into three files (or groups of files). Translate the first two groups together. Then translate the third group using the first group in context.

The groups are:

1. Source code without definitions or references to the redefined object.
2. Source code with the first definition of the object and any references to it.
3. Source code with the second definition of the object and any references to it.

The second and third methods might not work well if the object is referenced in a procedure that can be called when different definitions of the object are active. In this case you might need to make an additional copy of the procedure for each definition of the object that is active during a call to it, renaming the procedures and the calls appropriately, and placing each procedure definition after the corresponding definition of the object.

Example

In the following Sybase source code, the insert in procedure *p1* occurs twice, once inserting the string ’Jan 01 2000’ into the table #temp1 and next inserting the date value represented by ″Jan 01 2000″ into the table #temp1. For the second insert to work correctly in DB2 UDB, the string date value must be converted to the appropriate format for DB2 UDB.

```
create table #temp1 (x char(12))
go
create procedure p1 as
insert into #temp1 values ('Jan 01 2000')
go
exec p1
go
drop table #temp1
go
create table #temp1(x datetime)
go
exec p1
```

The source code should be rewritten as follows before converting. Notice the procedure is duplicated with a different name and the insert statements are translated without error.

```
create table #temp1 (x char(12))
go
create procedure p1 as
insert into #temp1 values ('Jan 01 2000')
go
exec p1
go
drop table #temp2
go
create table #temp2(x datetime)
go
create procedure p2 as
insert into #temp2 values ('Jan 01 2000')
go
exec p2

   ----- is translated
to -----

DECLARE GLOBAL TEMPORARY TABLE SESSION."#temp1"
(
  x CHAR(12) NOT NULL
) WITH REPLACE ON COMMIT PRESERVE ROWS NOT LOGGED!

CREATE PROCEDURE p1()
DYNAMIC RESULT SETS 1
LANGUAGE SQL
BEGIN
  DECLARE SQLCODE INT;
  DECLARE l_sqlcode INT DEFAULT 0;
  DECLARE CONTINUE HANDLER FOR SQLEXCEPTION,SQLWARNING,NOT FOUND
    SET l_sqlcode = SQLCODE;

  INSERT INTO SESSION."#temp1"
  VALUES ('Jan 01 2000');
  COMMIT;
END!

CALL p1()!

DROP TABLE SESSION."#temp2"!

DECLARE GLOBAL TEMPORARY TABLE SESSION."#temp2"
(
  x TIMESTAMP NOT NULL
) WITH REPLACE ON COMMIT PRESERVE ROWS NOT LOGGED!
```

```
CREATE PROCEDURE p2()
DYNAMIC RESULT SETS 1
LANGUAGE SQL
BEGIN
  DECLARE SQLCODE INT;
  DECLARE l_sqlcode INT DEFAULT 0;
  DECLARE CONTINUE HANDLER FOR SQLEXCEPTION,SQLWARNING,NOT FOUND
    SET l_sqlcode = SQLCODE;

  INSERT INTO SESSION."#temp2"
  VALUES ('2000-01-01-00.00.00');
  COMMIT;
END!

CALL p2()!
```

Related concepts

"Restrictions for migrating scripts" on page 129
If you are importing scripts that have been created by the source DBMS command interpreter, some restrictions apply.

Data format recommendations

When converting a database with DBCS data, you must give careful consideration to functions that handle character data. Where Sybase or SQL Server count characters, DB2 UDB counts bytes in functions such as left() or length().

Month names, decimal separators, and currency symbols are not localized.

Removing conversion actions

The changes report provides an interface that you can use to view what changes will be applied during the next conversion. If you do not want certain changes to take place, you can remove them using the report.

To remove a conversion action:

1. Select **Tools** → **Changes Report** to view the report. Entries that have a **Y** in the Next column are to be applied during the next conversion.
2. Select the entry that you want to remove and click **Delete** or to delete all entries click **Delete All**.
3. Click **Yes** to accept the deletion.
4. A warning message is issued instructing you to run "Step 2: Converting source metadata" on page 28 again, click **OK**.
5. Click **Close**.

The changes you have removed from the list will not take place during the next conversion.

Related tasks

"Viewing migration reports" on page 111
You can verify the migration as it progresses using the reports that are provided.

"Viewing the application log" on page 112
The application log contains a detailed record of the events that have occurred during the migration process for the current project.

"Viewing the changes report" on page 112
The Changes report records changes made to the metadata during the convert-refine process. Changes include Global Type Mapping changes that occur during the Convert step and edit changes that occurred during the Refine step.

Step 3: Refining the metadata conversion

The Refine step gives you the opportunity to view the results of the conversion and to make changes.

The recommended strategy for addressing messages for a clean deployment is to first alter the source SQL and re-convert. When you can no longer address any problems by changing the source, alter the final output before deploying it to the target database.

To refine the conversion:

1. Optional: From the **Refine** page, change the names of objects using the tools provided. MTK keeps track of the name mapping each time that you re-convert. For more information, see “Changing an object name” on page 41.
2. Optional: From the **Refine** page, edit the body of procedures, functions, and triggers using the tools provided. For more information, see “Editing a procedure, function, trigger, or view” on page 42.

   Important: You can also edit the body of procedures, functions, and triggers in the original source. However, do not use both methods because mixing the source editing methods can produce unpredictable results. If you need to edit the source for other reasons, you should also edit procedures, functions, and triggers in the original source.
3. To apply the changes, from the **Convert** page, click **Convert**. Upon re-conversion, the translator merges the changes with the original extracted source metadata to produce updated target database and XML metadata. The original metadata is not changed (unless you edited it directly).

   Repeat the refine and convert processes to achieve as clean a result as possible.
4. When you have finished making all possible changes to the source metadata, you can modify the resulting target database SQL file as necessary for a successful deployment. Be sure to make a backup copy first.

   Important: Do not return to the Convert step after making any manual SQL changes, because conversion of the source metadata replaces the existing file, destroying any manual changes.

Tools other than those on the refine page exist to help you while you refine the conversion. They are the SQL Translator, Log, and Reports.

After you have tuned the source to your satisfaction, you can either go to the Generate Data Transfer Scripts page to prepare the scripts for data transfer or go to the Deploy page to deploy the metadata.

Related tasks

“Working on a project collaboratively” on page 34
MTK is not designed for concurrent use on the same system. Use only one instance of MTK on the same project on the same system at the same time, or files and logs will become corrupt.

“Testing SQL translations” on page 43
While you are refining a conversion, you can use the SQL Translator to test the translation of individual statements, a series of statements, or a procedure.

“Removing conversion actions” on page 38
The changes report provides an interface that you can use to view what

changes will be applied during the next conversion. If you do not want certain changes to take place, you can remove them using the report.

Analysis of conversion results

After conversion, the IBM Migration Toolkit opens the Refine page, where you can evaluate and fix any problems.

The Refine page provides various options to view your converted data. The view types are discussed in the following table:

Option	Description
source_database	Where ***source_database*** is the source database and ***target_database*** is the target database. For example, **MySQL** and **DB2**. The ***source_database*** and ***target_database*** tabs display the database object hierarchy. If you select an object in either tab, you can modify the object on the right. You can change the names of the database, tables, views, columns, indexes, functions, and procedures. If the object is a trigger, procedure, or function, you can also edit the contents. For more information, see “Changing an object name” on page 41. The changes made here do not modify the source, they inform MTK to apply the changes to the output. See “Step 3: Refining the metadata conversion” on page 39 for information on editing the source.
target_database	
Messages	This Messages view displays the messages, sorted by message category and number. If you select the root node of the Message hierarchy, the right panel displays the messages, grouped by message number in decreasing order of importance. You can use this top-level list to display all occurrences of a particular message in the tree for high-level analysis. Click **Expand message in tree**. To see where in the file the message occurs, select a filename under a particular message number in the hierarchy. The source code is displayed on the right, highlighted where the message occurs. The full text of the message and access to message help is displayed above the file window. Both the source and target DB2 UDB files cannot be edited from this view; they are for reference only. To make changes, you can click **Go to source object** to open the source database view.

Option	Description
Report	The Report view displays the messages, sorted by message category and number. If you select the root node of the Report hierarchy, the right panel displays the messages, grouped by message number in decreasing order of importance. You can use this top-level list to filter messages from the tree for easier analysis. **Click Show message in tree** or **Hide message in tree** to toggle the filter. To see where in the file the message occurs, select a message number under a particular filename in the hierarchy. The source code is displayed on the right, highlighted where the message occurs. The full text of the message and access to message help is displayed above the file window. Both the source and target files cannot be edited from this view; they are for reference only. To make changes, you can click **Go to source object** to open the source database view.

HTML reports of the error messages are also available. Select **Tools** → **Migration Report** to access these reports.

Changing an object name

If an edit icon (📝) exists in the target column, you can change the name of the object.

The names of columns, tables, views, indexes, functions and procedures can be changed. Trigger and foreign key names cannot be changed. Changes made in the table are global and only take effect when you re-convert the source metadata.

1. From the **Refine** page, select an object in the source or target view.
2. Click the edit icon (📝) in the Target column. The Edit Object Name window opens.
3. In the **Name** field, type the new name.
4. Click **Apply**. The new name appears in the table, but will not take effect until the next time you convert your data.

After conversion, you can verify the name change by clicking the **Target** page on the Refine page and expanding the tree to find the renamed object.

Related tasks

"Step 2: Converting source metadata" on page 28
The Convert step converts source metadata to target database metadata. You can specify various options that affect the converted output before you convert source SQL into DB2 UDB or Informix Dynamic Server SQL.

Related reference

"Identifiers" on page 192
Oracle SQL, DB2 UDB SQL, and IBM Informix Dynamic Server SQL have different length specifications for identifiers. The converter adjusts identifiers as necessary to meet the length limitations.

"Identifiers" on page 263
The converter adjusts identifiers as necessary to compensate for any length limitations.

"System catalogs" on page 133

"CREATE INDEX" on page 285
Sybase and SQL Server allow index names to be the same as long as they belong to different tables. DB2 UDB requires that index names are unique across the schema, and Informix Dynamic Server requires that indexes are unique for a particular user. The generated index names are changed as necessary.

Editing a procedure, function, trigger, or view

You can edit source procedures, triggers, and views from the MTK Refine page.

Important: Care must be taken in renaming object or changing types in the body. These changes are local only. If you change a name in the body, ensure that you also change the name using the facilities provided on the Refine page so that all references to the object are changed accordingly. Likewise, use the data type mapping table on the Convert page to change data types.

1. Select a procedure, function, trigger, or view from the source or target view of the Refine page.
2. Click **Edit Source**. The body is opened in the editor specified in the Preferences window.
3. Save your editing changes and close the editor.
4. Click **Refresh Changes** to view the changes in MTK. The new changes are viewable here, but will not take effect until you re-convert.

Related tasks

"Step 2: Converting source metadata" on page 28
The Convert step converts source metadata to target database metadata. You can specify various options that affect the converted output before you convert source SQL into DB2 UDB or Informix Dynamic Server SQL.

Removing conversion actions

The changes report provides an interface that you can use to view what changes will be applied during the next conversion. If you do not want certain changes to take place, you can remove them using the report.

To remove a conversion action:

1. Select **Tools** → **Changes Report** to view the report. Entries that have a **Y** in the Next column are to be applied during the next conversion.
2. Select the entry that you want to remove and click **Delete** or to delete all entries click **Delete All**.
3. Click **Yes** to accept the deletion.
4. A warning message is issued instructing you to run "Step 2: Converting source metadata" on page 28 again, click **OK**.
5. Click **Close**.

The changes you have removed from the list will not take place during the next conversion.

Related tasks

"Viewing migration reports" on page 111
You can verify the migration as it progresses using the reports that are provided.

"Viewing the application log" on page 112
The application log contains a detailed record of the events that have occurred during the migration process for the current project.

"Viewing the changes report" on page 112
The Changes report records changes made to the metadata during the convert-refine process. Changes include Global Type Mapping changes that occur during the Convert step and edit changes that occurred during the Refine step.

Testing SQL translations

While you are refining a conversion, you can use the SQL Translator to test the translation of individual statements, a series of statements, or a procedure.

Restriction: DBCS (double-byte character set) characters cannot be used in the SQL Translator window.

The source SQL must be either self-contained or dependent upon metadata in the currently active conversion.

To test an SQL translation:

1. Select **Tools** → **SQL Translator**. The SQL Translator window opens.
2. Type or paste your test source SQL statements into the top frame.
3. Select the object dependencies for the statements.

Option	Description

None	All objects referenced are contained in the sample code. For example:
	`create table t1 (a int, y char(4));`
	`insert into t1 (1,'abc');`
	`select * from t1 where a=1;`
	
	Some statements have no dependencies. For example, the following is a valid test of the conversion of the Oracle sysdate function:
	`select sysdate from dual;`

Context files	Any object referenced is contained in either the sample code or in any file that is specified as a context file in the active conversion. Ensure that the sample code does not conflict with the SQL in the context files.
All files	Any object referenced is contained in either the sample code or in any of the files in the active conversion. Ensure that the sample code does not conflict with the SQL in any of the other files.

4. Click the **Convert** button.

The resulting SQL output and any messages are displayed in the bottom frame of the window.

Step 4: Generating scripts

In this step, you set any data transfer options and generate both the deployment and data-transfer scripts. You also specify where the files generated by data extraction are placed. Even if you are not transferring data, you must complete this step to obtain the deployment scripts.

Important: The data scripts are written specifically for the target database that is deployed in the next step. Do not attempt to load the data into a database created by other means. Also be sure that you are completely satisfied with the conversion results before you use any data-transfer scripts.

Generating scripts for DB2 UDB

You can generate transfer scripts for DB2 UDB.

To generate data transfer scripts:

1. Click the **Generate Data Transfer Scripts** page. The Generate Data Transfer Scripts page opens, with the source files and the target files listed.
2. Select the DB2 data loading options:

Option	Description
use IMPORT utility (logging and triggers active)	Select this option to use the IMPORT utility. This utility keeps logging and triggers active. It inserts data into a database table from an external file, inserting one row at a time. While data is imported, this table remains accessible to other applications. All applicable constraints and triggers remain in effect and are activated in the usual way as rows are inserted. The IMPORT utility can be used in different modes which you can specify in 3 on page 45. If you use the IMPORT utility, you are provided with safeguards for protecting data against unauthorized access and modification. These safeguards are called privilege and authority. See the IMPORT command information in the DB2 UDB documentation. The IMPORT modes use these safeguards: • To insert rows into a table, INSERT privilege on the table is required. • To replace the contents of a table, CONTROL privilege on the table is required. • To create a table in a database, CREATETAB authority on the database is required.

Option	Description
use LOAD utility (logging and triggers deactivated)	Select this option to use the LOAD utility. This utility deactivates logging and triggers while loading. It is a higher performing alternative to the IMPORT utility. However, its use requires a more detailed understanding of DB2 UDB. See LOAD command information in the DB2 UDB documentation. This utility can only be invoked by a user with SYSADM or DBADM authority. The LOAD utility constructs page images containing many rows and inserts them into the database one page at a time. It requires exclusive access to the table space being loaded. During the loading of data, the table space that contains the table is not accessible to other applications. Applicable constraints and triggers are deactivated for the duration of the load. Applicable check constraints and foreign key constraints are enforced at the end of the load process. Business rules that are implemented by triggers are not guaranteed to be enforced after the load completes.

3. Select the DB2 LOAD/IMPORT mode options:

Option	Description
INSERT: new rows will be added, existing rows will be unchanged	Select this option to have the LOAD utility add new data without affecting the existing data.
REPLACE: new rows will replace existing ones	Select this option to have the LOAD utility delete all existing data and replace it with the data being loaded. For example, if a table in the target database contains data that needs to be replaced before new data is added, the REPLACE option should be selected.

If you plan to modify the load or import options, you should have an understanding of the load and import options in the target database. For more information on the LOAD and IMPORT commands, refer to the *DB2 UDB Command Reference*. For more information on DB2 UDB data movement and other administrative tasks, see the *DB2 UDB Data Movement Utilities Guide and Reference* and the *DB2 UDB Administrative Guide*.

4. Select the File format options:

Option	Description
DEL: delimited ASCII format	Select this option to separate the data fields by a delimiter. Available for both IMPORT and LOAD utilities. Specifies that the data be represented in delimited text files, which contain streams of data values, ordered by row and then by column. The values are separated by column delimiters (a comma is the default), and rows are separated by new line characters. Character strings are enclosed in string delimiters (a double-quote is the default). NULL values are represented by nothing between the column delimiters for a particular column. If delimited ASCII is selected, the IBM Migration Toolkit automatically sets some advanced options: **char delimiter first** The character delimiter is given the highest priority, so that a special character between two character delimiters will be read as just another character. **Column delimiter** Set to a comma (,). **Character string delimiter** Set to a double-quote (″).
ASC: non-delimited ASCII format	Select this option to separate the data fields by character length. Available for both IMPORT and LOAD utilities. Specifies that the data be represented in non-delimited text files, which have the columns of data in fixed positions. No delimiters are needed. Nulls are identified by a table of null value indicators at the end of the row. If non-delimited ASCII is chosen, the **record length** advanced option is set by the IBM Migration Toolkit. Instead of new line characters marking the end of each record, the length of the data is used to set the number of characters read for each row.

Option	Description
Create file as UTF-8	Select this option if the data is encoded in UTF-8 (8-bit UCS/Unicode Transformation Format). If selected, the load and import statements will use codepage 1208 during deployment. If you are deploying to an existing database, ensure that it has been created in UTF-8.

5. Specify the **Directory for data extraction** or accept the default directory DataOutScripts, located in the project directory. If you do not want to extract the data to this system, for example if you do not have enough disk space, you can run the scripts from another system. For information on moving data manually, see “Manually transferring data to DB2 using files” on page 68.
6. Optional: Click **Advanced options** to select more options and click **OK**.

   The description of each advanced option includes the combination of import or load utilities and ASCII formats that it applies to. In the table below, the following abbreviated notation is used to discuss these options:

I	IMPORT
L	LOAD
D	DEL
A	ASC

   For example, **Column delimiter** (I L D) states that this advanced option applies to the combinations of IMPORT, DEL, and LOAD.

Option	Description
Column delimiter (I L D)	Type the character to be treated as the column delimiter. The default is the comma.
Character string delimiter (I L D)	Type the character to be treated as the character string delimiter. The default is the double-quote (″).
implied decimal (I L D A)	Select to have the location of an implied decimal point be determined by the column definition. If not selected, the decimal point is assumed to be at the end of the value. For example, if selected, the value 12345 is loaded into a DECIMAL(8,2) column as 123.45; otherwise, the value is loaded as 12345.00.
no eof character (I L D A)	If you select this option, the optional end-of-file character, `1A`, is not recognized as the end of file. Processing continues as if the end-of-file character were a normal character.
use defaults (I L D A)	If you select this option, default values are loaded into a source column containing no data for one or more row instances. If not selected, a NULL is loaded into a source column containing no data for a row instance (a nullable column). Either of the utilities will reject the row if the column is not nullable.

Option	Description
char delimiter first (I L D)	If you select this option, the specified character string delimiter has highest priority. The check box is selected by default. When selected, character strings can contain new line characters, which will not be interpreted as row delimiters. If not selected, any new line character in a character string will start a new row. If a row insertion fails, the reason is usually because the string's columns do not match the table definition.
blank for decimal plus (I L D)	Select this option for positive decimal values to be prefixed with a blank space instead of a plus (+).
keep blanks (I L D)	Select this option to preserve the leading and trailing blanks in each field of type CHAR, VARCHAR, LONG VARCHAR, or CLOB. If not selected, leading and trailing blanks not within character delimiters are removed and NULL is inserted for blank fields.
no double delimiters (I L D)	Select this option to suppress recognition of double character delimiters for fields of type CHAR, VARCHAR, LONG VARCHAR, or CLOB (except when lobsinfile is specified). Any pair of character delimiters found between the enclosing character delimiters is imported or loaded into the database as a single character.
compound (I D A)	Specifies the number of inserts the IMPORT utility includes in each non-atomic compound statement. The value can range from 1 to 100. (See the **commit count** option for more details).
commit count (I D A)	Specify the number of records to be imported by the IMPORT utility before a COMMIT. The DB2 UDB transaction log must be large enough to accommodate the total number of rows imported prior to a COMMIT statement; otherwise, the import operation will fail. Therefore, a reasonably sized entry can avoid a transaction log failure.
messages file (I L D A)	To override the default standard output, you can type the full path of the file to be used to record error and warning messages during the load operation. If the complete path to the file is not specified, the load or import utility uses the current directory and the default drive as the destination. If the name of a file that already exists is specified, the utility appends the information.
restart count (I D A)	You can specify a number of records for the IMPORT utility to skip when reading the data file. The utility will read values starting one row after the number of records you specify.

Option	Description
record length (I L A)	This is a required default selection. The new line character is not used to indicate the end of the row. Instead, the IBM Migration Toolkit sets the number of characters to be read for each row to the length of the data. This option prevents new line characters from being recognized and responded to within character strings. However, it does make the data file harder for you to read.
no check lengths (I L A)	If you click this check box, the Import utility attempts to import each row, even if the source data has a column definition that exceeds the size of the target table column. Such rows can be successfully imported if code page conversion causes the source data to shrink. For example, 4-byte EUC data in the source could shrink to 2-byte DBCS data in the target, requiring half the space. This option is particularly useful if you know that the source data will fit all cases despite mismatched column definitions.
strip blanks (I L A)	If you select this option, any trailing blank spaces are truncated when loading data into a variable-length field. If not selected, blank spaces are retained. Use this option if the table definition has VARCHAR columns and trailing blanks are not significant in any column.
strip nulls (I L A)	If you click this check box, any trailing NULLs (0x00 characters) are truncated when loading data into a variable-length field. If not selected, NULLs are retained. This condition cannot be specified with **strip blanks**. The only place the data file might have trailing NULLs is in a BINARY column if the source database added them as the data was being exported.
binary numerics (L A)	If you select this option, numeric data (with the exception of DECIMAL) must be in binary form and not in character representation.
any order (L D A)	Select if the preservation of the order of the source data is not important. This can significantly improve performance on SMP systems. This is applicable only when **CPU parallelism** is not `1` and if **save count** is > `0`, because crash recovery after a consistency point requires that data be loaded in sequence.

Option	Description
CPU parallelism (L D A)	Type the number of processes or threads that the LOAD utility generates for parsing, converting, and formatting records when building table objects. This parameter is designed to exploit intra-partition parallelism. It is particularly useful when loading presorted data, because the record order in the source data is preserved. If the value of this parameter is `0` or is not specified, the load utility uses an intelligent default value (usually based on the number of CPUs available) at run time.
fast parse (L D A)	If you select this option, there is reduced syntax checking on user-supplied column values, resulting in enhanced performance. Tables loaded under this option are guaranteed to be architecturally correct, and the utility is guaranteed to perform sufficient data checking to prevent a segmentation violation or trap. Data that is in the correct form will be loaded correctly. For example, assume a value of `123qwr4` is encountered as a field entry for an integer column in a non-delimited ASCII file. The LOAD utility usually flags this as a syntax error, because the value does not represent a valid number. With **fast parse**, a syntax error is not detected, and an arbitrary number is loaded into the integer field. Be sure to use this option with clean data only. Performance improvements using this option with ASCII data can be quite substantial.
no header (L D A)	If you select this option, the header verification code is skipped.
no row warnings (L D A)	If you select this option, all warnings about rejected rows are suppressed.
code page (L D A)	Type the code page of the data in the input data set. During the load operation, character data (and numeric data specified in characters) are converted from this code page to the target database code page.
dump file (L D A)	Type the fully qualified name of an exception file to which rejected rows are written (according to the server node). A maximum of 32 KB of data is written per record.
page free space (L D A)	Type the percentage of each data page that is to be left as free space.

Option	Description
total free space (L D A)	Type the percentage, *x*, of the total pages in the table that is to be appended to the end of the table as free space. For example, if *x* is 20%, and the table has 100 data pages, 20 additional empty pages will be appended. The total number of data pages for the table will be 120.
save count (L D A)	Type the value, *n*, that is converted to a page count, and rounded up to intervals of the extent size. This entry stipulates that the LOAD utility is to establish consistency points after every *n* rows. Because a message is issued at each consistency point, you should choose this option if the load operation will be monitored using the LOAD QUERY command. If the value of *n* is not sufficiently high, the synchronization of activities performed at each consistency point will impact performance. The default value is zero, meaning that no consistency points will be established, unless necessary.
row count (L D A)	Specifies the number of physical records to be loaded from the file.
warning count (L D A)	Type the value, *n*, to stop the load operation after *n* warnings. Set this parameter if no warnings are expected, but you want verification that the correct file and table are being used. If the load operation is stopped because the threshold of warnings is encountered, another load operation can be started in RESTART mode. The load operation will automatically continue from the last consistency point. Alternatively, another load operation can be initiated in REPLACE mode, starting at the beginning of the input file. If **warning count** is not given a value or if *n* is zero, the load operation continues regardless of the number of warnings issued.
tempfiles path (L D A)	Type the name of the path to be used when creating temporary files during a load operation. The name should be fully qualified according to the server node.

Option	Description
for exception (L D A)	Type a name for an exception table, into which rows in error are copied during the load operation. For example, the LOAD utility stores copies of rows that violate unique index rules. The exception table is a user-created table that reflects the definition of the table being loaded. The table also includes columns for containing the time of the violation (timestamp) and the names of the constraints violated. After the load operation completes, information in the exception table can be used to correct data found to be in error. The corrected data can then be inserted into the table. An exception table must not contain an identity column or any other type of generated column. If an identity column is present in the primary table, the corresponding column in the exception table only contains the column's type, length, and nullability attributes. The utility will not check for constraints or foreign key violations other than violations of uniqueness. An exception table should be used when loading data with a unique index and the possibility of duplicate records exists. If an exception table is not specified and duplicate records are found, the load operation continues with only a warning message issued about the deleted duplicate records. The records themselves are not logged.
data buffer (L D A)	Type the number of 4 KB pages (regardless of the degree of parallelism) to be used as buffered space for transferring data within the utility. If the value specified is less than the algorithmic minimum, the minimum required resource is used, and no warning is returned.
disk parallelism (L D A)	Type the number of processes or threads that the LOAD utility will generate for writing data to the table space containers. If a value is not specified, the utility selects a default based on the number of table space containers and the characteristics of the table.

hold quiesce (L D A)	If you select this option, the utility should leave the table in an exclusive quiescent state after the load operation. To remove the table spaces from this state of quiescence, you can issue the following statement:
	`db2 quiesce tablespaces for`
	` table (tablename) reset`

Option	Description
nonrecoverable (L D A)	If you select this option, the table effected during a load transaction will be marked as nonrecoverable. This table will not be recoverable by a subsequent roll forward action. The ROLLFORWARD utility will skip the transaction results and will mark this table, into which data was being loaded, as ″invalid″. The utility will also ignore any subsequent transactions against the table. After the load operation and subsequent roll forward operations are completed, such a table can only be dropped or restored from a backup (whole database or table spaces) taken after a commit point. With this option, table spaces are not put in a backup pending state following the load operation, and a copy of the loaded data does not need to be made during the load operation.
without prompting (L D A)	If you select this option, the IBM Migration Toolkit operates as if the list of data files contains all the files that are to be loaded. The devices or directories listed are sufficient for the entire load operation. If a continuation input file is not found, or if the copy targets are filled before the load operation finishes, the load operation fails and the table remains in a load-pending state.
Copy (L D A)	If you click YES for **Copy** you must set the following conditions: If the database is configured for forward recovery, an extra copy of all the data that is loaded is needed so committed changes that took place after the last backup can be applied. The LOAD utility will create the extra copy if the name of the file or device (where the copy is to be created) is specified with the Yes option. The selection of NO for **Copy** is the default. If Copy No is chosen and the database is configured for forward recovery, LOAD will leave the table space containing the loaded table in the backup pending state.

Option	Description
Indexing mode (L D A)	You must choose from the five index options to determine whether the LOAD utility is to rebuild the indexes or to extend them incrementally. The default option, **No index option**, specifies that indexes are not dealt with. The remaining four index options are: **Auto select** The load utility will automatically decide between the **Rebuild** or **Incremental** option. **Rebuild** All indexes will be rebuilt. The utility must have sufficient resources to sort all index key parts for both old and appended table data. **Incremental** Indexes will be extended with new data. This approach consumes index free space. It only requires enough sort space to append index keys for the inserted records. The **Incremental** option is only supported in cases where the index object is valid and accessible at the start of a load operation. (For example, the **Incremental** option is not valid immediately following a load operation in which the **Deferred** option was specified.) If the **Incremental** option is specified, but not supported due to the state of the index, a warning is returned, and the load operation continues in Rebuild mode. Similarly, if a load restart operation begins in the load build phase, the Incremental mode is not supported. Incremental indexing is not supported when all of the following conditions are true: • The LOAD utility with the COPY YES option is specified (logretain or userexit is enabled). • The table resides in a DMS table space. • The index object resides in a table space that is shared by other table objects belonging to the table being loaded. (To bypass this restriction, it is recommended that indexes be placed in a separate table space.)

Option	Description
Indexing mode (L D A) (continued)	**Deferred** The LOAD utility does not perform index creation if this option is selected. Indexes will be marked as needing a refresh. The first access to such indexes that is unrelated to a load operation might force a rebuild (for more information, see the *DB2 UDB Administration Guide*), or indexes might be rebuilt when the database is restarted. This approach requires enough sort space for all key parts for the largest index. The total time subsequently taken for index construction is longer than that required in rebuild mode. Therefore, when performing multiple load operations with deferred indexing, it is advisable (from a performance viewpoint) to let the last load operation in the sequence perform an index rebuild, rather than allow indexes to be rebuilt at first non-load access. Deferred indexing is only supported for tables with non-unique indexes, so that duplicate keys inserted during the load phase are not persistent after the load operation.
Statistics (L D A)	This option is supported only if the LOAD utility uses the REPLACE mode option. You have two options: **NO** If you click this check box, statistics are not collected, and the statistics in the catalogs are not altered. (This is the default choice.) **YES** If you click this check box, statistics are collected for the table and for any existing indexes. Enter one of the following options into the accompanying field to determine the level of statistics collected: • WITH DISTRIBUTION: Distribution statistics are collected. • INDEXES ALL: Both table and index statistics are collected. • DETAILED: Extended index statistics are collected.

Additional options: Some options are supported that are not shown in the Advanced Options window. They are provided by MTK when appropriate. For example, lobsinfile and the lobpath are generated by MTK when needed to put LOB data into separate files in a subdirectory named for the table to which those LOB columns belong. The LOB data files will be named according to the column with a sequence number. The data file for the table will contain file names for the LOB data files instead of the actual data. LOB data is written as a stream of bytes and not converted to readable characters.

Unsupported options: Some options are not supported. For example, alternative formats for date and time values are not supported because date and time values are extracted into a standard DB2 UDB format. Datalink specification options are not offered. generatedoverride, generatedmissing, generatedignore, identityoverride, identitymissing, and identityignore are not offered.

Example: choosing a file option: The following cases illustrate situations where you must choose one file option over the another:

- BINARY and VARBINARY data types: A byte in these two data types can be assigned the code for a column delimiter (a comma) or a row delimiter (a new line character). If you are using the delimited ASCII option, the IMPORT or LOAD utility misinterprets the byte as a delimiter and does not import or load properly. In this case, the non-delimited option avoids the problem.

  However, non-delimited ASCII is not an acceptable format. You can specify delimited ASCII and change the **column delimiter** advanced option to a character that does not appear in the binary data. You should keep the **char delimiter first** advanced option set. This ensures that the new line character will not be treated as end-of-record when it appears within a character string.
- VARCHAR data type: If a source table has several columns of type VARCHAR, a larger target data file with non-delimited ASCII then with delimited ASCII is produced. If you need to use non-delimited ASCII (for example, you have BINARY columns), then select the advanced option, **strip blanks**. This option stops the padding of VARCHAR columns in the DB2 UDB table. Blanks are truncated instead of extending to the column’s maximum length.

- CHAR and VARCHAR data types: If you have these data types containing new line characters, then either file option is acceptable.

7. Click **Create script** to generate the transfer scripts.

The resulting files for Table 1, Table 2, and Table 3 are shown in the tables below.

Table 1. Resulting files for Microsoft Windows

File description	Purpose
_data.bat	Contains statements to extract data from the source database
_db2.bat	Contains statements to load data into DB2 UDB

Table 2. Resulting files for AS/400®

File description	Purpose
data.qsh	Contains statements to extract data from the source database
db2.qsh	Contains statements to load data into DB2 UDB
.qry	Contains examples of statements generated for selecting and converting the data from the source database
.fsf	Defines the fields used with CPYFRMIMPF, which is used to import the data into DB2 UDB

Table 3. Resulting files for UNIX

File description	Purpose
_data.sh	Contains statements to extract data from the source database
_db2.sh	Contains statements to load data into DB2 UDB

Generating scripts for DB2 for z/OS

You can generate transfer scripts for DB2 for z/OS.

Restriction

Data can be loaded into DB2 only when the data is available on zSeries®. Unless requested, a data file is not created on the Windows client. The name of the DB2 schema and the DB2 tables are split in tokens with a maximum of eight characters. If the table name contains a non-alpha numeric character, that character is replaced by a hyphen.

To generate data transfer scripts:

1. Click the **Generate Data Transfer Scripts** page. The Generate Data Transfer Scripts page opens, with the source files and the target files listed.
2. Specify the Location of extracted data:

a. Select the method for transferring the data:

Option	Description
Store data on zSeries	Select this option to extract the data directly to the zSeries system using FTP. The necessary FTP information is requested during data deployment.
Store data on local system	Select this option to extract the data to the local system first. During deployment, the data from the local files are loaded to the zSeries system after you find a way for these files to be transferred to the zSeries system.
Store data on both zSeries and local system	Select this option to extract the data to the local system and to the zSeries system using FTP.

b. In the **zSeries root DSN for storing extracted data** field, specify the root name to assign to the data files when deployed to the zSeries system.

c. In the **Location of extracted data field**, specify the local directory where you want the data to be extracted.

3. Click **Create script** to generate the transfer scripts.

Generating scripts for Informix Dynamic Server

You can generate transfer scripts for Informix Dynamic Server.

Prerequisites:

- You can choose between JDBC and DB-Access for data deployment. If you plan to use DB-Access to deploy the data, make sure that the INFORMIXDIR environment variable is set and that DB-Access is in the PATH before you start MTK. For more information, see the *IBM Informix Dynamic Server Administrator's Guide*.
- Make sure that the IBM Informix JDBC driver is present in the MTK product bundle. Database object deployment always occurs through JDBC. The choice between DB-Access and JDBC is only for data deployment.
- You must set the data transfer options in this step. When Informix Dynamic Server is the target, deployment occurs only by using the MTK user interface; you cannot deploy to Informix Dynamic Server using a data-transfer script.

To generate data transfer scripts:

1. Click the **Generate Data Transfer Scripts** page. The Generate Data Transfer Scripts page opens, with the source files and the target files listed.
2. Specify the Data Loading Options:

Option	Description
Use JDBC	Select this option to use JDBC to deploy the data.
Use dbaccess	Select this option to use DB-Access to deploy the data.

3. Optional: Click **Advanced Options** and specify the Data Extraction Options:

a. Specify the Data Extraction Options:

Option	Description
Column delimiter	The default column delimiter is |. If your data contains the | character, you can change the delimiter to another character so that deployment occurs successfully. This field is applicable for deployment through both JDBC and DB-Access.

b. Click **OK**.

4. Specify the **Directory for data extraction** or accept the default directory DataOutScripts, located in the project directory. If you do not want to extract the data to this system, for example if you do not have enough disk space, you can run the scripts from another system.
5. Click **Create script** to generate the transfer scripts.

Step 5: Deploy to the target database server

After you generate the scripts in step 4, you can deploy the converted objects and data to the target database.

Prerequisites

- Generate the scripts used for deployment.
- If the script file contains error messages, be sure that you have reviewed the messages and understand the differences between the original code and its conversion.
- The data extracted from the source database is specific to the target database that is deployed in this step. Do not attempt to load the data into a database that already contains tables and views that have been created by other means.
- Ensure that you have not modified the conversion since the last time you extracted data. You should extract and deploy the data only after the final conversion.
- If, during data extraction, a table contains millions of rows, ensure that the file system can accommodate the size of the largest object in the table. On UNIX file systems you can modify the file size limit by using the `ulimit` command.
- If you choose to deploy to a system that is remote to the system on which MTK is running, then you cannot choose to create a new database during deployment. The database must first be created on the remote system and registered in the local catalog. For more information, see the “Cataloging a database” topic in the DB2 Information Center (http://publib.boulder.ibm.com/infocenter/db2luw/v8/index.jsp) or *IBM DB2 Universal Database Administration Guide: Implementation*.
- Data cannot be loaded into LOB columns during remote deployment. The LOBPATH parameter in the LOAD or IMPORT command must refer to a directory on the target database server. You can load data into LOB columns by moving the generated scripts to the target machine and running them locally on the desired target server.

MTK can deploy the database to a local or remote system.

You can deploy the converted objects and data to the target database at the same time or separately. For example, you could load the metadata during the day along with some sample data to test your procedures. Then, you could load the data at night when the database has been tested and when network usage is low.

When MTK deploys data, it extracts the data onto the system running MTK before loading the data into the database.

Deploying to DB2 UDB

You can deploy the converted objects to DB2 UDB.

Prerequisites

- If you want to manually create a DB2 UDB database to migrate to (instead of letting MTK create it during deployment), then ensure you use the following DB2 UDB commands:

  ```
  update database manager configuration using keepdari no db2stop
  force db2start
  ```

  Run these commands each time that you manually create a new database instance.
- To deploy DB2 UDB SQL stored procedures to a DB2 UDB for Windows system, some Visual C++ environment variables must be altered on the target system. Upon installation of Visual C++, the variables are set as user variables. DB2 UDB requires that these be system variables. Change the variables in one of two ways:
  - Using the System control panel, redefine the VCV5_DRIVE variable as a system variable, and move the relevant portion of the user include, lib, and path variables to the system variable of the same name. Create the system variable if necessary.
  - Uncomment the VCV5_DRIVE, include, lib, and path settings in the following batch file, save the changes, and run it.

    ```
    %DB2PATH%\function\routine\src_path.bat
    ```
- The ODBC and JDBC drivers treat VARCHAR data differently during data movement. ODBC trims any trailing spaces. Consider this behavior with fields of only spaces; they are transferred as empty strings. Therefore, for MTK data movement, be sure that you use the JDBC driver when all of the following is true:
  - VARCHAR data is to be transferred.
  - The data ends with spaces.
  - The trailing spaces are significant and not be trimmed.
- During deployment, the connection to DB2 UDB uses a Java native driver, not ODBC. If you encounter problems when connecting remotely (such as a ″no suitable SQL driver″ error), be sure that the following directory and files are in the Java class path:

  ```
  %DB2PATH%/java/db2java.zip;
  %DB2PATH%/java/runtime.jar;
  %DB2PATH%/java/sqlj.zip;
  %DB2PATH%/bin
  ```
- If the MTK system does not have enough disk space, you have the following other data deployment options:
  - Copy the IBM Migration Toolkit and the project directory to the system running the target database and then deploy the converted objects and data to the database locally.
  - Manually transfer the data using scripts and data files.
  - Manually transfer the data using scripts and named pipes.
- For conversions from Microsoft SQL Server, Sybase, and Sybase SQL Anywhere:

  - Space and performance management are not translated. Use the DB2 UDB facilities to adapt the Sybase configuration to the new DB2 UDB environment.
  - Top level IF statements are not translated when they have two branches (when there are both IF and ELSE clauses).
- In conversions from MySQL:
  - BLOB and TEXT data movement is not supported.

To deploy the database to the target database server DB2 UDB:

1. Start the target database server and if deploying to a remote system, start the target database server client (for example, the DB2 UDB client).
2. In the IBM Migration Toolkit, click the **Deploy to Target** page.
3. If more than one conversion exists for this project, you can select the files to deploy from the **Conversion name** list.
   - If you have converted everything into one SQL file, select this conversion.
   - If you have converted tables and procedures separately, select first the conversion that contains the base objects, such as the tables, views, and indexes.

   As you select a conversion, its file and deployment scripts are listed on the left.
4. Type a new or existing target database name or alias in the **DB2 database name** field. The name cannot exceed eight characters and must begin with an alphabetic character, $ or @.
5. Select either **Use a local database** or **Use a remote database**. If you select **Use a local database**, you can choose **(Re)create** to recreate a new local database. If you do not select **(Re)create**, the objects and data will be added to the existing database.

   When a database is created, a bufferpool with pages of size 32 KB and three table spaces of the same size are created. These provide enough space for the deployment of tables with any row length. Before launching the database into production, perform some performance tuning and adjust the size of the bufferpool as necessary.
6. Optional: Click **Advanced options**. Available options are as follows:

Table 4. DB2 advanced options

Field	Explanation
Territory	Specifies the national language territory setting of the target database. If an existing target database is used, the territory must match that of the database.
Code set	Specifies the national language code set setting of the target database. If an existing target database is used, the code set must match that of the database.
Use default collating sequence	Specifies that the character strings are sorted according to the default target database sequence.
Use system collating sequence	Specifies that character strings are sorted according to the code set specified in the target database.

Table 4. DB2 advanced options (continued)

Field	Explanation
Use identity collating sequence	Specifies that character strings are sorted according to their hexadecimal value.

7. Click OK to return to the **Deploy to Target** page.
8. Specify a target database **User ID** and **Password** or select **Use your system current user ID and password** to use the user ID and password of the target system.

   The system user ID is always used to create a database, regardless of whether you choose to specify a user ID for deployment. If you choose to specify a user ID, that ID will be used for all other deployment actions.

   To deploy objects and data to a previously created database, the user ID must have been granted database administrative authority (DBADM in DB2 UDB).

   When you are deploying a database from multiple database schemas, be sure to specify a user ID that does not cause conflicts between object names. If a default object name is the same as an object name under a particular schema and if that schema name matches the specified user ID, then an attempt will be made to deploy the object under the same name. This will cause an error.

   For example, you want to deploy both TABLE1 and BATCH.TABLE1. If you specify BATCH as the user ID, then the default TABLE1 will become BATCH.TABLE1 and thereby conflict with the TABLE1 of the BATCH schema.
9. To create the metadata objects in the database, select **Launch script_name in the database** and click **Deploy**. When complete, a report is displayed. Use the report to verify that each of the table objects exists in DB2 UDB before deploying the data.
10. To use MTK to extract and load data, clear the option in step 9, select **Extract and store data** and **Load data to the target database using scripts**. The data extracted from the source database is typed specifically for the target database that is deployed in this step. Do not attempt to load the data into a database created by other means. Also ensure that you have not modified the conversion since the last time you have extracted data. You should extract and deploy the data only after your final conversion.
11. **Optional:** To change your source database information, click **Change source database**. For more information, see “Extracting objects from a source database” on page 24.
12. Click **Deploy**.

During deployment of the database, the following events take place:

- If **Launch script_name in the database** is selected:
  1. If (Re)create database is selected, the database is created. (Any database with the same name is dropped first.)
  2. The script named Deploy* runs.
     a. A connection is made to the target database.
     b. Parameters needed to set up the database are loaded into the target database.
     c. The MTK UDFs are created in the target database.
     d. The target SQL script runs, creating the metadata objects.
- If **Extract and store data on this system** is selected, the data is extracted from the source database and loaded onto the system where MTK is installed.

- If **Load data to the target database using scripts** is selected, the source data is loaded into the target database.
- The integrity of the database is checked.
- The deployment process is verified.
- The verification report is displayed.

Deploying to DB2 for z/OS

You can deploy the converted objects to DB2 for z/OS.

Prerequisites

- Include the following jar files in the user or system environment (in the sqllib\java directory):
  - db2java.zip
  - db2jcc.zip
  - sqlj.zip
- The DB2 variable DB2PATH cannot contain spaces. You can use the DB2SET command to verify the current setting. If the DB2PATH variable contains spaces, for example `C:\PROGRAM FILES\IBM\SQLLIB`, you can type `DB2SET DB2PATH=C:\PROGRA~1\IBM\SQLLIB` to fix. You can then use the `DIR /X` command to show the short name for the folder.
- The DB2 for z/OS database must be properly cataloged with the client computer.
- The Java UDF and stored SQL procedure support must be properly installed.
- An FTP server must be running on the zSeries server.

Restrictions and limitations

- Populated tables that contain foreign keys are not verified correctly. This happens because the underlying tablespace of each table is in a CHECK-pending state. After completing data deployment, run the CHECK DATA utility to complete the necessary check.
- A table extraction file size is limited to the z/OS FTP file size limit.
- When a conversion contains several tables, the same file allocation size is used for all the tables and for FTP and the auxiliary files used by the LOAD utility.
- The verification HTML page generated at the end of a deployment might contain incomplete or incorrect results.
- Triggers, functions, and procedures are properly converted, but not completely deployed.
- Data movement is not supported for binary columns, BLOBs, and CLOBs.
- The LOAD utility is always run in replace mode.
- Function calls in conditions of branching statements are not translated.
- MTK cannot produce a procedure call in a dynamic compound statement.

To deploy the database to the target database server DB2 for z/OS:

1. Start the target database server and if deploying to a remote system, start the target database server client (for example, the DB2 for z/OS client).
2. In the IBM Migration Toolkit, click the **Deploy to Target** page.
3. If more than one conversion exists for this project, you can select the files to deploy from the **Conversion name** list.

- If you have converted everything into one SQL file, select this conversion.
- If you have converted tables and procedures separately, select first the conversion that contains the base objects, such as the tables, views, and indexes.

As you select a conversion, its file and deployment scripts are listed on the left.

4. Specify the database connectivity information:
   a. In the **Local Name of z/OS Database** field, specify the database name that you want to connect to. The database name is the alias provided when cataloguing the database in the local client. The name cannot exceed eight characters and must begin with an alphabetic character, $ or @.
   b. In the **User ID** and **Password** fields, specify the password and user ID for the database that you want to connect to.

      The system user ID is always used to create a database, regardless of whether you choose to specify a user ID for deployment. If you choose to specify a user ID, that ID will be used for all other deployment actions.

      The user ID and password are those required to access the database. The user ID must have enough authority to create the objects.

      To deploy objects and data to a previously created database, the user ID must have been granted database administrative authority (DBADM).

      When you are deploying a database from multiple database schemas, be sure to specify a user ID that does not cause conflicts between object names. If a default object name is the same as an object name under a particular schema and if that schema name matches the specified user ID, then an attempt will be made to deploy the object under the same name. This will cause an error.

      For example, you want to deploy both TABLE1 and BATCH.TABLE1. If you specify BATCH as the user ID, then the default TABLE1 will become BATCH.TABLE1 and will conflict with the TABLE1 of the BATCH schema.
   c. Select **Drop and Recreate Objects** if you want to drop and recreate the database that you specified in **Local Name of z/OS Database**.
5. Specify the workload manager information:
   a. In the **WLM for UDFs** field, specify the name of the Java workload manager. The default value is WLMENVJ. See the *DB2 UDB Application Programming Guide and Reference for Java* for details.
   b. In the **WLM for Stored Procedures** field, specify the name of the workload manager for SQL stored procedures. See the *DB2 UDB Application Programming and SQL Reference* for details.
6. Specify the information for FTP:
   a. In the **FTP Host Name** field, specify the zSeries host name to connect to when transferring data using FTP.
   b. In the **FTP Userid** field, specify the zSeries system user ID to connect with when transferring data using FTP.
   c. In the **FTP User Password** field, specify the password associated with the FTP user ID.
   d. In the **Volume for FTP Data** field, specify the volume on the zSeries system where the data files will be written.
   e. In the **Allocation (cycls)** field, specify the allocation for each data file in cylinders on the target system. Each table is written to its own file, allocated using this size. If a table is larger than 2GB, several files will be created for the same table, each with a maximum size of 2GB.

f. **Optional:** Click **Advanced Allocations for Load** to specify custom allocations. The Dataset Allocation table opens and displays the current allocation values. Use the allocation table to change the assigned volume and amount of cylinders to allocate for the auxiliary data sets. The table displays four columns:

Table Specifies each table name.

DSN Specifies the data set names for each table.

Volume
Specifies the volume on which to write the data. The field is editable, enabling you to assign a different volume.

Allocation
Specifies the number of cylinders to allocate to the particular data set. The field is editable, enabling you to allocate a different number.

Click **OK** to return to the **Deploy to Target** page.

7. To create the metadata objects in the database, select **Launch script_name in the database** and click **Deploy**. When complete, a report is displayed. Use the report to verify that each of the table objects exists in DB2 UDB before deploying the data.
8. To use MTK to extract and load data, clear the option in step 7, select **Extract and store data** and **Load data to the target database using scripts**. The data extracted from the source database is typed specifically for the target database that is deployed in this step. Do not attempt to load the data into a database created by other means. Also ensure that you have not modified the conversion since the last time you have extracted data. You should extract and deploy the data only after your final conversion.
9. **Optional:** To change your source database information, click **Change source database**. For more information, see "Extracting objects from a source database" on page 24.
10. Click **Deploy**.

Deploying to Informix Dynamic Server

You can deploy the converted objects to Informix Dynamic Server.

Prerequisites

- You can use the IFX_EXTEND_ROLE configuration parameter to restrict the ability of users to register external routines. The default value of the IFX_EXTEND_ROLE configuration parameter is 1 (on). See Informix Dynamic Server documentation for more information.
- In migrations from MySQL to Informix Dynamic Server, the deployment fails if there is a mismatched data type for a referenced SERIAL column. All auto-generated columns are mapped to the SERIAL data type in Informix Dynamic Server. If there is a foreign key relationship to the generated column, the data types in the parent and child will be different.
- Informix Dynamic Server does not raise the the NO_DATA_FOUND exception when the table is empty.
- Manual data movement is not supported. Data can only be extracted and deployed using the MTK user interface.
- When Informix Dynamic Server UDF's are deployed, MTK does not add the full three part name to the routine's EXTERNAL NAME. This can create a problem with user access. For example, if a UDF is owned by ″informix″ and a user other than ″informix″ attempts to access this UDF,

Informix Dynamic Server will not allow the user access. To solve this problem, update the EXTERNAL NAME in your MTKORAINFXT.udf file with the three-part name. This means that all references of:

```
EXTERNAL NAME 'oraInfxtUDFs:com.ibm.mtk.udf.infxt.
oracle.OraInfxtUDFs.<routine name>
```

are changed to:

```
EXTERNAL NAME 'informix.oraInfxtUDFs:com.ibm.mtk.udf.infxt.
oracle.OraInfxtUDFs.<routine name>
```

To give users access, you can use `GRANT USAGE ON LANGUAGE JAVA TO PUBLIC;`.

- MTK UDFs must be deployed to Informix Dynamic Server using the informix administrator account.
- In conversions from MySQL:
  - BLOB and TEXT data movement is not supported.

To deploy the database to the target database server Informix Dynamic Server:

1. Start the target database server and if deploying to a remote system, start the target database server client (for example, the Informix Dynamic Server client).
2. In the IBM Migration Toolkit, click the **Deploy to Target** page.
3. If more than one conversion exists for this project, you can select the files to deploy from the **Conversion name** list.
   - If you have converted everything into one SQL file, select this conversion.
   - If you have converted tables and procedures separately, select first the conversion that contains the base objects, such as the tables, views, and indexes.

   As you select a conversion, its file and deployment scripts are listed on the left.
4. Specify the Informix Dynamic Server **Host Name**, **Port**, **Server Name**, and **Database Name**. The database name cannot exceed eight characters and must begin with an alphabetic character, $ or @.
5. Optional: To change your advanced options, click **Advanced options**. The options you see and can use depend on the target database. Available options are as follows:

Table 5. Informix Dynamic Server advanced options

Field	Explanation
DB_Locale	Specifies the database locale, which Informix Dynamic Server uses to process locale-sensitive data. If you do not set DB_LOCALE on the client computer, client applications assume that the database locale has the value of the CLIENT_LOCALE environment variable. The client application, however, does not send this default value to the database server when it requests a connection.

Table 5. Informix Dynamic Server advanced options (continued)

Field	Explanation
Client_Locale	Specifies the client locale, which the client application uses in read and write operations, in end-user formats, and for processing ESQL statements. If you do not set CLIENT_LOCALE, the client application uses the default locale, U.S. English, as the client locale. If you do not set DB_LOCALE on the client computer, client applications assume that the database locale has the value of the CLIENT_LOCALE environment variable. The client application, however, does not send this default value to the database server when it requests a connection.
DB Space	Specifies the name of dbspace in which MTK will create the new Informix Dynamic Server database. MTK does not create the dbspace; the dbspace must be created in advance.
Log Mode ANSI	Indicates that the logging mode of the target database is ANSI-compliant.
Deploy MTK UDFs	Deploys user-defined functions to the target database.

Click **OK** to return to the **Deploy to Target** page.

6. If you choose to deploy to a local database, select **(Re)create** unless you want to add the objects and data to what already exists on the database.

   When a database is created, a bufferpool with pages of size 32 KB and three table spaces of the same size are created. These provide enough space for the deployment of tables with any row length. Before launching the database into production, perform some performance tuning and adjust the size of the bufferpool as necessary.

7. Specify the target database **User ID** and **Password** or select **Use your system current user ID and password** to use the user ID and password of the target system.

   The system user ID is always used to create a database, regardless of whether you choose to specify a user ID for deployment. If you choose to specify a user ID, that ID will be used for all other deployment actions.

   - If you select **Use your system current user ID and password**, make sure that MTK is running on a host that is trusted by the Dynamic Server instance.
   - If you are using DB-Access for data movement, you will be prompted for a password. However, if you select **Use your system current user ID and password** and you made sure that MTK is running on a host that is trusted by an Informix Dynamic Server instance, DB-Access will not prompt for a password.

   To deploy objects and data to a previously created database, the user ID must have been granted database administrative authority.

   When you are deploying a database from multiple database schemas, be sure to specify a user ID that does not cause conflicts between object names. If a default object name is the same as an object name under a particular schema and if that schema name matches the specified user ID, then an attempt will be made to deploy the object under the same name. This will cause an error.

For example, you want to deploy both TABLE1 and BATCH.TABLE1. If you specify BATCH as the user ID, then the default TABLE1 will become BATCH.TABLE1 and thereby conflict with the TABLE1 of the BATCH schema.

8. To create the metadata objects in the database, select **Launch script_name in the database** and click **Deploy**. When complete, a report is displayed. Use the report to verify that each of the table objects exists in Informix Dynamic Server before deploying the data.
9. To use MTK to extract and load data, clear the option in step 9, select **Extract and store data** and **Load data to the target database using scripts**, and click **Deploy**. The data extracted from the source database is typed specifically for the target database that is deployed in this step. Do not attempt to load the data into a database created by other means. Also ensure that you have not modified the conversion since the last time you have extracted data. You should extract and deploy the data only after your final conversion.
10. **Optional:** To change your source database information, click **Change source database**. For more information, see "Extracting objects from a source database" on page 24.

During deployment of the database, the following events take place:

- If **Launch script_name in the database** is selected:
  1. If (Re)create database is selected, the database is created. (Any database with the same name is dropped first.)
  2. The script named Deploy* runs.
     a. A connection is made to the target database.
     b. Parameters needed to set up the database are loaded into the target database.
     c. The MTK UDFs are created in the target database.
     d. The target SQL script runs, creating the metadata objects.
- If **Extract and store data on this system** is selected, the data is extracted from the source database and loaded onto the system where MTK is installed.
- If **Load data to the target database using scripts** is selected, the source data is loaded into the target database.
- The integrity of the database is checked.
- The deployment process is verified.
- The verification report is displayed.

Manually transferring data to DB2 using files

You can migrate source data to a DB2 database manually without using the MTK interface.

If you are migrating to DB2 and if you are running MTK on a system that can connect to both the source and target databases and has enough storage to extract the data, then you can use MTK to automatically extract and migrate the data to the target database server at the same time that MTK deploys the database. However, if you do not have those resources available on one machine, you might decide to convert and refine the metadata in a convenient location, such as on your desktop workstation, and later go to the server system to migrate the large amount of data.

Tip: If you have previously deployed the database and are now choosing to migrate data using the same system as MTK, there is no need to manually move the data. Use MTK to generate the data transfer scripts, and on the Deploy page, make the following selections:

- Clear **Launch *project_name* in the database**.
- Select **Extract and store source data on this system**.
- Select **Load data to target database using generated scripts**.

The procedure that follows explains how to migrate source data to a DB2 database without using the MTK user interface. In this procedure, you manually deploy data by moving scripts to the target server system. MTK also supports manually deploying data to a DB2 Linux, UNIX, or Windows server from a remote computer. See information following the procedure below for additional instructions when manually deploying data to a remote computer.

Prerequisites

- You have completed your final conversion of the source files and have deployed the converted objects to a target database.
- The system where you are migrating the data has the required version of the JDK installed.
- The system where you are migrating the data has a database connection setup for both the source DBMS and target database.

  For a Java native connection:

  – IDS The file ifxjdbc.jar is already provided in the MTK directory. No configuration is necessary for a Java native connection to Informix Dynamic Server.

  – Sybase Install JConnect or ensure jconn2.jar is available in the class path to connect to Sybase.

  – Oracle Ensure $ORAHOME/jdbc/lib/classes111.zip is available in the class path to connect to Oracle.

Restriction

Though it is possible to extract the data onto one system and copy it to another system for deployment, it is not recommended. Data files can be easily corrupted when moved across different systems, especially when they contain BLOBs.

Procedure

1. On the Generate Data Transfer Scripts page, in the **Directory for data extraction** field, type the desired directory of the system from where you will run the scripts. For example, if you are running MTK on a Windows system, but want to download the data to a UNIX system in the /Users/debra/datamigrate/out/ directory, enter the directory as shown.
2. Click **Create scripts**.
3. Create a project directory and data output directory on the system on which you will be migrating the data. For example:
   ```
   C:\> telnet db2aix12
   [debra@db2aix12:~/]> mkdir datamigrate
   [debra@db2aix12:~/]> mkdir datamigrate/out
   ```
4. Copy the following files, which are found in the project directory, to the system on which you will be migrating the data:
   - Verify_*projectname*.out
   - mtk.jar

- cwm.jar
- xmistore.jar
- ifxjdbc.jar

If moving to a Windows system, copy these additional files:

- DataMove_*projectname*_db2.bat (found in the DataOutScripts subdirectory)
- DataMove_*projectname*_data.bat

If moving to a UNIX system, copy these additional files:

- DataMove_*projectname*_db2.sh (found in the DataOutScripts subdirectory)
- DataMove_*projectname*_data.sh

For example:

```
C:\mtk\projects\SalesDB\>
   jar cvf SalesDB.jar ..\..\mtk.jar ..\..\cwm.jar \
..\..\xmistore.jar DataOutScripts\DavaMove_SalesDB_db2.sh
   DataMove_SalesDB_data.sh
C:\mtk\projects\SalesDB\> ftp db2aix12
ftp> bin
ftp> cd datamigrate
ftp> put SalesDB.jar
ftp> bye
C:\mtk\projects\SalesDB\> telnet db2aix12
[debra@db2aix12:~/]> cd datamigrate
[debra@db2aix12:~/datamigrate/]> jar xvf SalesDB.jar
```

5. Edit the data extraction script (*_data*), supplying the correct values for the following variables:

   SRC_USERNAME
   User ID used to connect to the source database for extraction.

   SRC_PASSWORD
   Password for the SRC_USERNAME.

   SRC_DATABASE
   Source database connection information. It can be the JDBC or ODBC data source name, or the combination of the host IP address and port number needed for a native driver connection. If using a native driver connection, ensure the necessary JAR file is in the classpath.

   SRC_NATIVE
   Set to TRUE if using a Java native driver to connect to the source database.

   USERDIR
   The directory containing the MTK jar files (most likely the current directory).

   PROJPATH
   The directory containing the Verify_*.out file (most likely the current directory).

6. Run the data extraction script (On UNIX systems, ensure the file has execute permission).

```
[debra@db2aix12:~/datamigrate/]> chmod 755 DataMove_SalesDB_data.sh
[debra@db2aix12:~/datamigrate/]> ./DataMove_SalesDB_data.sh >log 2>&1
```

7. Edit the data deployment script (*_db2*), supplying the correct values for the following variables:

   DBNAME
   Name of the target database.

USERNAME
: User ID used to connect to the target database. Leave blank to use the system user ID.

PASSWORD
: Password for the username. Leave blank to use the password of the system user ID.

USERDIR
: The directory containing the MTK jar files (most likely the current directory).

PROJPATH
: The directory containing the Verify_*.out file (most likely the current directory).

PROJDATA
: The directory where the extracted data is located.

8. After the data extraction script has completed extracting the data, run the data deployment script
   - On UNIX systems, ensure the file has execute permission:
     ```
     [debra@db2aix12:~/datamigrate/]> chmod 755 DataMove_SalesDB_db2.sh
     [debra@db2aix12:~/datamigrate/]> ./DataMove_SalesDB_db2.sh >>log 2>&1
     ```
   - On Windows systems, ensure you are working in a DB2 UDB Command environment:
     ```
     C:\mtk\> db2cmd
     C:\mtk\> projects\SalesDB\DataMove_SalesDB_db2.bat
     ```

Deploying data from a remote computer:

The previous procedure explains how to manually deploy data by moving scripts to the target server. This procedure contains information on manually deploying data to a DB2 Linux, UNIX, or Windows server from a remote computer.

Although it is possible to extract the data onto one system and copy it to another system for deployment, it is not recommended. Data files can be easily corrupted when moved across different systems, especially when they contain BLOBs.

Prerequisites

- You must first complete your final conversion of the source files and you have deployed the converted objects to a target database.
- The system from which the scripts will be called must have the required version of the JDK.
- The system from which the scripts will be called must have the appropriate DB2 LUW client installed and must have connectivity to the target server.
- The remote DB2 server must be cataloged locally.

Procedure

Follow the steps in the procedure above, except:

- In step 3, the directories you create do not need to be on the system hosting the target server.
- In step 4, copy the specified to the system to the system containing the directories you created in step 3.
- In step 7, when you edit the data deployment script (*_db2*), also supply the correct values for the following variable:

REMOTE
In the line containing the DB2 LOAD command, set **%REMOTE%** to `CLIENT`.

Manually transferring data using named pipes

If you have a large amount of data and prefer not to use an intermediary machine to transfer the data from the source to the target database, then you can use UNIX named pipes to handle the transfer.

Prerequisites

- You have completed your final conversion of the source files and have deployed the converted tables to the target database.
- The system on which you are executing the pipes has the required version of the JDK installed.
- The system on which you are executing the pipes has a database connection setup for both the source DBMS and DB2 UDB. The file ifxjdbc.jar contains the classes necessary for a Java native connection to Informix Dynamic Server.

  If using a native driver:
  - install JConnect or ensure jconn2.jar is available in the classpath.
  - ensure $ORAHOME/jdbc/lib/classes111.zip is available in the classpath.

Restrictions

- The resulting shell script is written for a UNIX system using the Korn shell.
- If you are deploying to a DB2 UDB database, the database must be DB2 UDB for Multiplatform.

Procedure

1. On the **Generate Data Transfer Scripts** page, in the **Directory for data extraction** field, type the desired UNIX directory of the system from where you will be running the scripts.

   For example, if you are running MTK on a Windows system, but want to run the scripts on a UNIX system in the /Users/debra/datamigrate/ directory, enter the directory as shown.
2. Click **Create scripts**.
3. Copy the following files, which are found in the project directory, to the system on which you will run the scripts. Ensure you copy them to the directory specified in step 1:
   - Verify_*projectname*.out
   - mtk.jar
   - cwm.jar
   - xmistore.jar
   - ifxjdbc.jar
   - DataMove_*projectname*_pipe.sh

   For example:

```
C:\mtk\> jar cvf SalesDB.jar mtk.jar cwm.jar mtkInfx.jar mtkOra.jar \
xmistore.jar projects\SalesDB\DataMove_SalesDB_pipe.sh
C:\mtk\> ftp db2aix12
ftp> bin
ftp> cd datamigrate
ftp> put SalesDB.jar
ftp> bye
```

```
C:\mtk\> telnet db2aix12
[debra@db2aix12:~/]> cd datamigrate
[debra@db2aix12:~/datamigrate/]> jar xvf SalesDB.jar
```

4. Edit the data transfer script (*_pipe*), supplying the correct values for the following variables:

   SRC_USERNAME
   User ID used to connect to the source database for extraction.

   SRC_PASSWORD
   Password for the SRC_USERNAME.

   DBNAME
   Name of the target database.

   USERNAME
   User ID used for connecting to the target database. Leave blank to use the DB2 UDB system user ID.

   PASSWORD
   Password for the username. Leave blank to use the password of the system user ID.

5. Run the script.

   Important: If you are migrating to DB2 UDB, the two ends of the named pipe wait for each other. A java program reads the source database and the DB2 UDB loader consumes the data. If one of the processes fails to start (for example, if incorrect connect information is provided), then the other process will remain active and must be terminated manually.

   ```
   [debra@db2aix12:~/datamigrate/]> chmod 755 DataMove_SalesDB_pipe.sh
   [debra@db2aix12:~/datamigrate/]> ./DataMove_SalesDB_data.sh >log 2>&1
   ```

The command line process

The command line interface offers a way to operate MTK from the command line using a configuration file and arguments.

The command line arguments tell MTK what migration steps to perform while the configuration file contains the options needed by MTK to perform the specified steps. Each of the command line arguments correspond to one of the steps described in “The GUI process” on page 23.

Running MTK from the command line

You can run MTK from the command line.

Running on Microsoft Windows

You can run MTK from the command line using the Microsoft Windows operating system.

Prerequisites: A valid configuration file.

From a command prompt, issue the following command:

```
MTKmain.bat –CONFIG configfile.xml argument
```

where:

configfile.xml
The name of your configuration file.

argument
A valid set of arguments.

Related concepts

"Arguments" on page 75
The MTK command line interface uses arguments to invoke MTK. The arguments tell MTK what actions to perform.

Related reference

"Configuration file" on page 75
The command line interface uses the information specified in the configuration file to perform each migration step.

"CONVERSIONS" on page 78
The CONVERSIONS element contains the information needed to perform the convert, generate data transfer scripts, and deployment.

"PROJECT" on page 96
All of the information that describes a project is contained the PROJECT element. Every configuration file must contain exactly one PROJECT element.

"SPECIFY_SOURCE" on page 98
The SPECIFY_SOURCE element contains information that specifies the source SQL file information for importing and extracting source objects..

Related information

"Elements" on page 77
The configuration file contains elements along with element attributes.

Running on UNIX and Linux

You can run MTK from the command line using the UNIX or Linux operating system.

Prerequisites: A valid configuration file.

From a command prompt, issue the following command:

```
MTKmain.sh –CONFIG configfile.xml argument
```

where:

configfile.xml
The name of your configuration file.

argument
A valid set of arguments.

Related concepts

"Arguments" on page 75
The MTK command line interface uses arguments to invoke MTK. The arguments tell MTK what actions to perform.

Related reference

"Configuration file" on page 75
The command line interface uses the information specified in the configuration file to perform each migration step.

"CONVERSIONS" on page 78
The CONVERSIONS element contains the information needed to perform the convert, generate data transfer scripts, and deployment.

"PROJECT" on page 96
All of the information that describes a project is contained the PROJECT element. Every configuration file must contain exactly one PROJECT element.

"SPECIFY_SOURCE" on page 98
The SPECIFY_SOURCE element contains information that specifies the source SQL file information for importing and extracting source objects..

Related information

"Elements" on page 77
The configuration file contains elements along with element attributes.

Configuration file

The command line interface uses the information specified in the configuration file to perform each migration step.

The configuration file contains all of the migration details. It is an XML file that adheres to a strict document structure which is defined in the mtk.dtd file included in your MTK installation directory (for example, `c:/mtk/mtk.dtd`). A sample configuration file, named config.xml, is also included in your MTK installation directory (for example, `c:/mtk/config.xml`).

The basic structure of the configuration file is:

```
<MTK>
  <PROJECT...>
    <SPECIFY_SOURCE></SPECIFY_SOURCE>
    <CONVERSIONS></CONVERSIONS>
  </PROJECT...>
</MTK>
```

Related tasks

"Running on Microsoft Windows" on page 73
You can run MTK from the command line using the Microsoft Windows operating system.

"Running on UNIX and Linux" on page 74
You can run MTK from the command line using the UNIX or Linux operating system.

Related reference

"CONVERSIONS" on page 78
The CONVERSIONS element contains the information needed to perform the convert, generate data transfer scripts, and deployment.

"PROJECT" on page 96
All of the information that describes a project is contained the PROJECT element. Every configuration file must contain exactly one PROJECT element.

"SPECIFY_SOURCE" on page 98
The SPECIFY_SOURCE element contains information that specifies the source SQL file information for importing and extracting source objects..

Related information

"Elements" on page 77
The configuration file contains elements along with element attributes.

Arguments

The MTK command line interface uses arguments to invoke MTK. The arguments tell MTK what actions to perform.

Each of the command line arguments correspond to one of the steps described in "The GUI process" on page 23. Most of the command-line arguments consist of just the argument without any value assigned, except the source and target database access credentials.

Restrictions:

- Each argument with MTKMain.bat or MTKMain.sh must be separated from the next with at least one blank space character. For example:
  - In Microsoft Windows:

    ```
    MTKMain.bat –CONFIG config.xml -EXTRACT
    ```
  - In UNIX or Linux:

    ```
    MTKMain.sh –CONFIG config.xml -EXTRACT
    ```
- Each argument, that includes data (for example, -CONFIG, -SRCUSR, -SRCPWD, -TRGTUSR, and -TRGTPWD), must precede its corresponding data value.
- The order in which the arguments occur is not important.
- A valid set of arguments must be supplied. If MTK encounters an invalid argument, a warning or error is issued, inline help is displayed, and the argument is ignored. For example:

  ```
  INFO    MTKCLI  Using MTK from the command line...
  ERROR   MTKCLI  Illegal option found usage: MTKMain.bat
  [ -config [CONFIGFILE] ] [ -import ] [ -extract ] [ -convert ]
  [ -genscript ] [ -deploy ] [ -all ]
  [ -trgtusr [USERNAME] ] [ -trgtpwd [PASSWORD] ]
  [ -srcusr [USERNAME] ] [ -srcpwd [PASSWORD] ]
  INFO    MTKCLI  Exiting...
  ```
- The values of the element and attributes are not case sensitive with the following exceptions: TRGTUSR, TRGPWD, SRCUSR, SRCPWD, NAME, and DIRECTORY.

The following table shows the supported command line arguments:

Argument	Description
-CONFIG	This argument specifies the configuration file to use. It must be followed by the configuration file's name, including the full path, enclosed in double quotes. No part of the file name or path can include the double quote character.
-IMPORT	This argument instructs MTK to import files.
-EXTRACT	This argument instructs MTK to extract objects from source databases.
-CONVERT	This argument instructs MTK to perform the conversion.
-GENSCRIPT	This argument instructs MTK to generate the data transfer scripts.
-DEPLOY	This argument instructs MTK to deploy the database.
-ALL	This argument instructs MTK to: • Perform all specify source operations • Convert the data • Generate the data transfer scripts • Deploy to the target database For example: `MTKMain.bat –CONFIG config.xml -ALL`

Argument	Description
-SRCUSR	This argument specifies the user ID to use to connect to the source database. This argument must be followed by [*username*], where *username* is the user ID for the source database. For example: `-SRCUSR [username]`.
-SRCPWD	This argument specifies the password to use to connect to the source database. This argument must be followed by [*password*], where *password* is the password for the source database. For example: `-SRCPWD  [password]`.
-TRGTUSR	This argument specifies the user name to use to connect to the target database. This argument must be followed by [*username*], where *username* is the user name for the target database. For example: `-TRGTUSR [username]`.
-TRGTPWD	This argument specifies the password to use to connect to the target database. This argument must be followed by [*password*], where *password* is the password for the target database. For example: `-TRGTPWD password`.

Related tasks

"Running on Microsoft Windows" on page 73
You can run MTK from the command line using the Microsoft Windows operating system.

"Running on UNIX and Linux" on page 74
You can run MTK from the command line using the UNIX or Linux operating system.

Related reference

"CONVERSIONS" on page 78
The CONVERSIONS element contains the information needed to perform the convert, generate data transfer scripts, and deployment.

"PROJECT" on page 96
All of the information that describes a project is contained the PROJECT element. Every configuration file must contain exactly one PROJECT element.

"SPECIFY_SOURCE" on page 98
The SPECIFY_SOURCE element contains information that specifies the source SQL file information for importing and extracting source objects..

Related information

"Elements"
The configuration file contains elements along with element attributes.

Elements

The configuration file contains elements along with element attributes.

Related concepts

"Arguments" on page 75
The MTK command line interface uses arguments to invoke MTK. The arguments tell MTK what actions to perform.

Related tasks

"Running on Microsoft Windows" on page 73
You can run MTK from the command line using the Microsoft Windows operating system.

"Running on UNIX and Linux" on page 74
You can run MTK from the command line using the UNIX or Linux operating system.

"Example: end to end migration" on page 106
In this example you are shown and end-to-end migration using the MTK command line interface.

Related reference

"Configuration file" on page 75
The command line interface uses the information specified in the configuration file to perform each migration step.

CONVERSION:

The CONVERSION element represents a set of data that MTK will apply the given convert, generate data transfer scripts, and deployment operations. There can be zero or more CONVERSION elements per configuration file.

Each CONVERSION element must be unique within the same configuration file. Uniqueness is determined by the value of the CONVERTFILE attributes in the configuration file and the state of the project. Failure to meet this criteria will result into MTK issuing a message.

The CONVERSION element can contain the following elements and attributes:

Important: There is both a CONVERSION and CONVERSIONS element.

Name	Required	Multiple occurrences allowed	Description
CONVERT	N	N	This element contains information about the source SQL file to convert and the settings to apply.
GENERATE_DATA_ TRANSFER_SCRIPTS	N	N	This element contains settings for the generate data transfer scripts step.
DEPLOY_TO_TARGET	N	N	This element contains settings for the deployment step.

Related tasks

"Step 2: Converting source metadata" on page 28
The Convert step converts source metadata to target database metadata. You can specify various options that affect the converted output before you convert source SQL into DB2 UDB or Informix Dynamic Server SQL.

"Example: convert SQL from a source database" on page 102
In this example you are shown how to convert source SQL using the MTK command line interface.

CONVERSIONS:

The CONVERSIONS element contains the information needed to perform the convert, generate data transfer scripts, and deployment.

It is comprised of no more than one GLOBAL_SETTINGS element and one or more unique CONVERSION elements. The CONVERSIONS element can contain the following elements and attributes:

Important: There is both a CONVERSIONS and CONVERSION element.

Name	Required	Multiple occurrences allowed	Description
GLOBAL_SETTINGS	N	N	This elements contains settings that are applicable to all conversions.
CONVERSION	N	Y	This element contains information about the conversion, data transfer script generation, and deployment of a single source SQL file.

Related concepts

"Arguments" on page 75
The MTK command line interface uses arguments to invoke MTK. The arguments tell MTK what actions to perform.

Related tasks

"Step 2: Converting source metadata" on page 28
The Convert step converts source metadata to target database metadata. You can specify various options that affect the converted output before you convert source SQL into DB2 UDB or Informix Dynamic Server SQL.

"Running on Microsoft Windows" on page 73
You can run MTK from the command line using the Microsoft Windows operating system.

"Running on UNIX and Linux" on page 74
You can run MTK from the command line using the UNIX or Linux operating system.

"Example: convert SQL from a source database" on page 102
In this example you are shown how to convert source SQL using the MTK command line interface.

Related reference

"Configuration file" on page 75
The command line interface uses the information specified in the configuration file to perform each migration step.

CONVERT:

The CONVERT element contains various options available for the converted output.

Important: The CONVERT and the GLOBALCONVERT elements contain most of the same elements and attributes, however, the purpose of the GLOBALCONVERT element is to specify the available CONVERT options globally so that you don't have to repeat them.

The CONVERT element can contain the following elements and attributes:

Name	Required	Multiple occurrences allowed	Description
ADDROWID	N	N	This attribute specifies whether or not to have a ROWID column added to the end of each table to simulate the Oracle ROWID pseudo-column feature. This column is an identity (an integer). During data movement, old values are not preserved and new ones are generated for each row. Allowable values are Y or N. The default value is N.
ADDSRCDIRECTIVES	N	N	This attribute specifies whether or not the source directives should be copied as comments in the output SQL file. Allowable values are Y or N. The default value is N.
CONVERSIONNAME	N	N	This attribute contains the user-specified name for the conversion. This becomes the name of the target SQL file. If the CONVERSIONS element contains the CONVERT element this is optional, otherwise it is required.
CPYFULLSRCSP	N	N	This attribute specifies whether or not to copy statements of source procedures as comments between statements in the resulting translated SQL. Allowable values are Y or N. The default value is Y.
CPYSRCASCOMMENTS	N	N	This attribute specifies whether or not to copy the original source as comments to statements being converted. Inter-statement comments are also copied. Allowable values are Y or N. The default is Y.
CPYSRCCOMMENTS	N	N	This attribute specifies whether or not to have summary source text that is related to each statement within a stored procedure, copied as comments to the translated code. Allowable values are Y or N. The default value is Y.
CPYSRCSTMTINSP	N	N	This attribute specifies whether or not to copy the SQL for each source procedure as a comment before its associated translated SQL. Allowable values are Y or N. The default value is Y.

Name	Required	Multiple occurrences allowed	Description
DEFAULTSCHEMA	N	N	This attribute specifies the schema name that is used as the default for the converted target SQL. If the value is FROM_FIRST_OBJECT, the N schema qualifier is used for all objects that are in the same schema as the first object that is encountered. The default value is FROM_FIRST_OBJECT.
DELIMIDENT	N	N	This attribute, which is only available for source database Informix Dynamic Server, specifies whether or not the source SQL object names use delimident identifiers (case sensitive within quotation marks; can contain white space and other special characters). This setting must match the setting of the Informix DELIMIDENT environment variable setting for the source SQL. The allowable values are: • **Oracle:** AMERICAN, SIMPLIFIED_CHINEESE, FRENCH, GERMAN, DANISH, SPANISH, ITALIAN, DUTCH, NORWEGIAN, PORTUGUESE, FINNISH, SWEDISH, HUNGARIAN, ROMANIAN, CROATIAN, SLOVENIAN, GREEK, RUSSIAN, TURKISH, ENGLISH, ESTONIAN, LATVIAN, LUTHUANIAN, BRAZILLIAN_PRTUGUESE. • **Informix Dynamic Server, Sybase SQL Anywhere, Sybase Adaptive Server Enterprise, and SQL Server:** ENGLISH. The default values are: • **Oracle:** AMERICAN. • **Informix Dynamic Server, Sybase SQL Anywhere, Sybase Adaptive Server Enterprise, and SQL Server:** ENGLISH.
GLOBALSCOPETMP TABLES	N	N	This attribute, which is only available for source databases Sybase Adaptive Server Enterprise (ASE) and Microsoft SQL Server, specifies whether or not to instruct the translator to first process all of the global temporary tables declared in the input script and treat them as global. Allowable values are Y or N. The default value is N.

Name	Required	Multiple occurrences allowed	Description
IFERRCPYSRCAS COMMENTS	N	N	During conversion, MTK copies source statements into the target SQL file. This attribute specifies whether or not to copy only the source statements that apply to the message displayed. Allowable values are Y or N. The default value is N.
IGNORECASEINUSR DEFID	Y	N	This attribute, which is only available for source databases Sybase SQL Anywhere, Sybase Adaptive Server Enterprise, and SQL Server, indicates if the source objects names are case sensitive. Allowable values are Y or N. The default values are: • Sybase SQL Anywhere and SQL Server: Y • Sybase Adaptive Server Enterprise: N
INSRTDROPDDLSTMT	N	N	This attribute specifies whether or not to generate an appropriate DROP statement before each corresponding CREATE statement (INDEX, TABLE, or VIEW). The allowable values are Y or N. The default value is N.
INSRTDROPDMLSTMT	N	N	This attribute, which is only available for target database DB2 UDB, specifies whether or not to drop the existing procedure before the new procedure is created. Creating a procedure will fail if the procedure already exists. Allowable values are Y or N. The default value is Y.
PADSTRINGSFOR COMPR	N	N	This attribute, which is only available for source databases Sybase SQL Anywhere or Oracle, indicates whether or not you want string comparisons to evaluate to TRUE regardless of any extra spaces before or after the string. Allowable values are Y or N. The default value is Y.
SRCDATEFORMAT	N	N	This attribute specifies the format of the date literals in the source file. For information on the source date format, see “Step 2: Converting source metadata” on page 28. The default values are: • **Oracle:** DD-MON-RR • **Informix Dynamic Server:** MDY4 • **Sybase SQL Anywhere, Sybase Adaptive Server Enterprise, and SQL Server:** mdy

Name	Required	Multiple occurrences allowed	Description
SRCDATELANGUAGE	N	N	This attribute specifies the language of the date literals in the source file. The available values are: • **Oracle:** AMERICAN, SIMPLIFIED_CHINEESE, FRENCH, GERMAN, DANISH, SPANISH, ITALIAN, DUTCH, NORWEGIAN, PORTUGUESE, FINNISH, SWEDISH, HUNGARIAN, ROMANIAN, CROATIAN, SLOVENIAN, GREEK, RUSSIAN, TURKISH, ENGLISH, ESTONIAN, LATVIAN, LUTHUANIAN, BRAZILLIAN_PRTUGUESE • **Informix Dynamic Server, Sybase SQL Anywhere, Sybase Adaptive Server Enterprise, and SQL Server:** ENGLISH The default values are: • **Oracle:** American • **Informix Dynamic Server, Sybase SQL Anywhere, Sybase Adaptive Server Enterprise, and SQL Server:** English
SRCSQLFILE	Y	N	This attribute specifies the source file name to convert. The file name can be specified with or without the file extension. The file name can contain only alphanumeric and underscore characters. The file name of the source SQL file must be a system-legal file name.
TRANSTABLESPACE	N	N	This attribute specifies whether or not you want to allow for the table properties to be added after translation. Allowable values are Y or N. If you select Y, the TABLESPACE clause is copied as-is and you will need manually edit it before deploying to DB2 UDB or Informix Dynamic Server. The default value is N.
TRANSKEYSTOAL TERTABLE	N	N	This attribute, which is only available for source databases Sybase or Microsoft SQL Server, specifies whether or not to convert the system procedures into ALTER TABLE statements to define the keys. Because these system procedures are frequently used for documentation purposes only, the default is that they are not converted to ALTER TABLE statements. Allowable values are Y or N. The default value is N.

Name	Required	Multiple occurrences allowed	Description
TRGTVARIABLEPREFIX	N	N	This attribute, which is only available for source databases Sybase SQL Anywhere, Sybase Adaptive Server Enterprise, and SQL Server, specifies the prefix for to be used for source variables. Use only alphanumeric characters. For example, if a source variable begins with a prefix of @, the prefix must be changed to an acceptable prefix. The default value is V_.

Related tasks

"Step 2: Converting source metadata" on page 28
The Convert step converts source metadata to target database metadata. You can specify various options that affect the converted output before you convert source SQL into DB2 UDB or Informix Dynamic Server SQL.

"Example: convert SQL from a source database" on page 102
In this example you are shown how to convert source SQL using the MTK command line interface.

Related reference

"GLOBALCONVERT" on page 89
The GLOBALCONVERT element contains settings that are applied to all CONVERT elements.

DATABASE:

The DATABASE element contains information about the objects to extract.

The DATABASE element can contain the following elements and attributes:

Name	Required	Multiple occurrences allowed	Description
ENTIREDATABASE	N	N	This attribute instructs MTK to extract all objects from this database
NAME	Y	N	This attribute specifies the database name.
SCHEMA	N	Y	This element contains information about objects to extract from a specific schema in the given database.

Related information

"Step 1: Specifying the source" on page 23
After a migration project is created or opened, you can start the migration process. Your first step is use the Specify Source page to obtain the source files to be converted to the SQL of the target database server.

DEPLOY_TO_TARGET:

The DEPLOY_TO_TARGET element contains target database connection information and information about what options to use during deployment.

The DEPLOY_TO_TARGET element can contain the following attributes:

Name	Required	Multiple occurrences allowed	Description
TRGTUSR	Y[1]	N	This attribute specifies the user ID to use to connect to the target database.
TRGTPWD	Y[2]	N	This attribute specifies the password to use to connect to the target database.
TRGTDBNAME	Y	N	This attribute specifies the target database name to connect to.
DBLOCATION	N	N	This attribute specifies if the target database is local or remote.
RECREATE	N	N	This attribute indicates whether or not to drop and recreate the database specified in TRGTDBNAME. Allowable values are Y or N. The default value is N.
USECURRENTSYS LOGIN	N	N	This attribute specifies whether to use system user id and password to connect. Allowable values are Y or N. The default value is N.
LAUNCHSCRIPT	N	N	This attribute indicates whether or not to run the target SQL file on the source database server. Allowable values are Y or N. The default value is Y.
EXTRACTDATATOSYS	N	N	This attribute indicates whether or not to extract the source data to a file on local machine. This attribute must be specified either before or in conjunction with a data deployment. Allowable values are Y or N. The default value is Y.
LOADDATATOTRGT	N	N	This attribute indicates whether or not to deploy the data to a target database. This attribute must be specified either after or in conjunction with a data deployment. Allowable values are Y or N. The default value is N.
TERRITORY	N	N	This attribute is only available in conversions to target database DB2 UDB. It specifies the national language territory settings of the target database. If an existing target database is used, the territory must match that of the database. The default value is DEFAULT.
CODESET	N	N	This attribute specifies the national language code set settings of the target database. Possible values are DEFAULT and UTF-8. The default value is DEFAULT.

Name	Required	Multiple occurrences allowed	Description
COLLATINGSEQ	N	N	This attribute specifies how a character string is sorted. Allowable values are DEFAULT, SYSTEM, and IDENTITY. When the value is DEFAULT, character strings are sorted according to the default target database sequence. When value is SYSTEM, character strings are sorted according to the code set specified in the target database. The default value is DEFAULT.
DEPLOYMTKUDF	N	N	This attribute indicates whether or not to deploy a UDF to replace the standard Java UDF that is provided with MTK. Allowable values are Y or N. The default value is Y.
RUNSTATONCATLG	N	N	This attribute indicates whether or not to run RUNSTATS on the target database catalog after deploying a script. Allowable values are Y or N. The default value is Y.
RUNSTATONLOAD IMPORT	N	N	This attribute indicates whether or not to run RUNSTATS after a LOAD or IMPORT. Allowable values are Y or N. The default value is Y.

Notes:

1. This attribute is required if it has not already been provided using the -TRGTUSR argument and if there has not been a previous extraction.
2. This attribute is required if it has been provided using the -TRGTPWD argument and if there has not been a previous extraction.

Related tasks

"Step 5: Deploy to the target database server" on page 59
After you generate the scripts in step 4, you can deploy the converted objects and data to the target database.

"Example: deploying converted objects" on page 104
In this example you are shown how to deploy converted objects using the MTK command line interface.

"Example: deploying converted objects with advanced options" on page 105
In this example you are shown how to deploy converted objects with advanced attributes using the MTK command line interface.

EXTRACT:

The EXTRACT element contains information that specifies what objects to extract as well as the connection information needed to access the source database.

It provides equivalent functionality to the MTK GUI operations up to and including the click of the **Extract** button in the Specify Source window.

Each EXTRACT element must be unique within the same configuration file. Uniqueness is determined by the value of the EXTRACTTOFILE attributes, the

IMPORTFILE attributes, and the project state. The name of each file already existing in the project, to be imported, or to be extracted into must be unique. Failure to do this will result in MTK issuing a message and existing source SQL files will be replaced in order of execution. For example, if the user initially imported a source SQL file named “source1.src” and subsequently extracts to the same file, the content of the file will be the result of the last operation, in this case is the result of the extraction.

The EXTRACT element can contain the following elements and attributes:

Name	Required	Multiple occurrences allowed	Description
DATABASE	N	Y	This element contains information about objects to extract from a specific database.
SRCUSR	Y[1]	N	This attribute specifies the user ID to use for the connection.
SRCPWD	Y[2]	N	This attribute specifies the password for the connection.
EXTRACTTOFILE	Y	N	This attribute specifies the name that is given to the set of extracted objects in the extraction block. This name must contain only alphanumeric and underscore characters. The suffix of the file name is SRC. The file name must be a system-legal file name.
CREATEONEFILE PERSTOREDPROC	N	N	This attribute specifies whether or not to create one file for each stored procedure that is extracted. Allowable values are Y or N.
CREATEONEFILE PERTRIGGER	N	N	This attribute specifies whether or not to create one file for each trigger that is extracted. Allowable values are Y or N.
EXTRACTGRANT STATEMENT	N	N	This attribute specifies whether or not to extract grant statements. Allowable values are Y or N.
SETQUOTED IDENTIFIERON	N	N	This attribute, which applies only to extractions from SQL Server, specifies whether or not to set the quoted identifier on. Allowable values are Y or N.
INCLUDENEEDED OBJECTSIN	N	N	This attributes specifies what database objects to include. Allowable values are main, content, and n. The default value is main.
ALLOBJECTS	N	N	This attribute specifies whether or not to extract all objects from the given source database. Allowable values are Y or N.
ODBC_CONNECTION	N	N	This element contains information about how to connect to the source database using ODBC.

Name	Required	Multiple occurrences allowed	Description
JDBC_CONNECTION	N	N	This element contains information about how to connect to the source database using JDBC.

Notes:

1. Required if not provided via -SRCUSR argument and if there has not been a previous extraction.
2. Required if provided via -SRCPWD argument and if there has not been a previous extraction.

Related tasks

"Example: extracting objects from a source database" on page 101
In this example you are shown how to extract objects from a source database using the MTK command line interface.

Related information

"Step 1: Specifying the source" on page 23
After a migration project is created or opened, you can start the migration process. Your first step is use the Specify Source page to obtain the source files to be converted to the SQL of the target database server.

GENERATE_DATA_TRANSFER_SCRIPTS:

The GENERATE_DATA_TRANSFER_SCRIPTS element is where you specify any data transfer script attributes.

The GENERATE_DATA_TRANSFER_SCRIPTS element can contain the following elements and attributes:

Name	Required	Multiple occurrences allowed	Description
LOADINGOPTIONS	N	N	This attribute specifies which utility to use to deploy data. Allowable values are load and import. The default value is load.
LOADIMPORTMODE	N	N	This attribute, which is only available for DB2 as a target database, specifies which method to use to deploy data. Allowable values are INSERT and REPLACE. The default value is INSERT.
UTF8	N	N	This attribute, which is only available for DB2 as a target database, specifies if the data file should be in UTF8 format or not. Allowable values are Y or N. The default value is N.
DIRECTORYFOR DATAEXTRACTION	N	N	This attribute specifies the directory into which the data is extracted. The default value is `<project root="">\DataOutScripts</project>`.

Name	Required	Multiple occurrences allowed	Description
FILEFORMAT	N	N	This attribute, which is only available for DB2 as a target database specifies what delimiter to use for the data extraction file. Allowable values are DEL and ASC. The default value is ASC.

Related tasks

"Step 4: Generating scripts" on page 44
In this step, you set any data transfer options and generate both the deployment and data-transfer scripts. You also specify where the files generated by data extraction are placed. Even if you are not transferring data, you must complete this step to obtain the deployment scripts.

"Example: generating data transfer scripts" on page 103
In this example you are shown how to generate data transfer scripts using the MTK command line interface.

GLOBAL_SETTINGS:

The GLOBAL_SETTINGS element contains settings that are applied to all CONVERSION elements. However, all of these settings can be overridden within the CONVERSION element.

The GLOBAL_SETTINGS element can contain the following elements and attributes:

Name	Required	Multiple occurrences allowed	Description
GLOBALCONVERT	N	N	This element contains settings for conversion that apply to all CONVERSION elements, unless overridden.
GENERATE_DATA _TRANSFER_SCRIPTS	N	N	This element contains settings for generating data transfer scripts.
DEPLOY_TO_TARGET	N	N	This element contains settings for deployment.

Related tasks

"Step 2: Converting source metadata" on page 28
The Convert step converts source metadata to target database metadata. You can specify various options that affect the converted output before you convert source SQL into DB2 UDB or Informix Dynamic Server SQL.

GLOBALCONVERT:

The GLOBALCONVERT element contains settings that are applied to all CONVERT elements.

Important: The CONVERT and the GLOBALCONVERT elements contain most of the same elements and attributes, however, the purpose of the

GLOBALCONVERT element is to specify the available CONVERT options globally so that you don't have to repeat them.

The GLOBALCONVERT element can contain the following elements and attributes:

Name	Required	Multiple occurrences allowed	Description
ADDROWID	N	N	This attribute specifies whether or not to have a ROWID column added to the end of each table to simulate the Oracle ROWID pseudo-column feature. This column is an identity (an integer). During data movement, old values are not preserved and new ones are generated for each row. Allowable values are Y or N. The default value is N.
ADDSRCDIRECTIVES	N	N	This attribute specifies whether or not the source directives should be copied as comments in the output SQL file. Allowable values are Y or N. The default value is N.
CPYFULLSRCSP	N	N	This attribute specifies whether or not to copy statements of source procedures as comments between statements in the resulting translated SQL. Allowable values are Y or N. The default value is Y.
CPYSRCASCOMMENTS	N	N	This attribute specifies whether or not to copy the original source as comments to statements being converted. Inter-statement comments are also copied. Allowable values are Y or N. The default is Y.
CPYSRCCOMMENTS	N	N	This attribute specifies whether or not to have summary source text that is related to each statement within a stored procedure, copied as comments to the translated code. Allowable values are Y or N. The default value is Y.
CPYSRCSTMTINSP	N	N	This attribute specifies whether or not to copy the SQL for each source procedure as a comment before its associated translated SQL. Allowable values are Y or N. The default value is Y.
DEFAULTSCHEMA	N	N	This attribute specifies the schema name that is used as the default for the converted target SQL. If the value is FROM_FIRST_OBJECT, the N schema qualifier is used for all objects that are in the same schema as the first object that is encountered. The default value is FROM_FIRST_OBJECT.

Name	Required	Multiple occurrences allowed	Description
DELIMIDENT	N	N	This attribute, which is only available for source database Informix Dynamic Server, specifies whether or not the source SQL object names use delimident identifiers (case sensitive within quotation marks; can contain white space and other special characters). This setting must match the setting of the Informix DELIMIDENT environment variable setting for the source SQL. The allowable values are: • **Oracle:** AMERICAN, SIMPLIFIED_CHINEESE, FRENCH, GERMAN, DANISH, SPANISH, ITALIAN, DUTCH, NORWEGIAN, PORTUGUESE, FINNISH, SWEDISH, HUNGARIAN, ROMANIAN, CROATIAN, SLOVENIAN, GREEK, RUSSIAN, TURKISH, ENGLISH, ESTONIAN, LATVIAN, LUTHUANIAN, BRAZILLIAN_PRTUGUESE. • **Informix Dynamic Server, Sybase SQL Anywhere, Sybase Adaptive Server Enterprise, and SQL Server:** ENGLISH. The default values are: • **Oracle:** AMERICAN. • **Informix Dynamic Server, Sybase SQL Anywhere, Sybase Adaptive Server Enterprise, and SQL Server:** ENGLISH.
GLOBALSCOPETMP TABLES	N	N	This attribute, which is only available for source databases Sybase Adaptive Server Enterprise (ASE) and Microsoft SQL Server, specifies whether or not to instruct the translator to first process all of the global temporary tables declared in the input script and treat them as global. Allowable values are Y or N. The default value is N.
IFERRCPYSRCAS COMMENTS	N	N	During conversion, MTK copies source statements into the target SQL file. This attribute specifies whether or not to copy only the source statements that apply to the message displayed. Allowable values are Y or N. The default value is N.

Name	Required	Multiple occurrences allowed	Description
IGNORECASEINUSR DEFID	Y	N	This attribute, which is only available for source databases Sybase SQL Anywhere, Sybase Adaptive Server Enterprise, and SQL Server, indicates if the source objects names are case sensitive. Allowable values are Y or N. The default values are: • Sybase SQL Anywhere and SQL Server: Y • Sybase Adaptive Server Enterprise: N
INSRTDROPDDLSTMT	N	N	This attribute specifies whether or not to generate an appropriate DROP statement before each corresponding CREATE statement (INDEX, TABLE, or VIEW). The allowable values are Y or N. The default value is N.
INSRTDROPDMLSTMT	N	N	This attribute, which is only available for target database DB2 UDB, specifies whether or not to drop the existing procedure before the new procedure is created. Creating a procedure will fail if the procedure already exists. Allowable values are Y or N. The default value is Y.
PADSTRINGSFOR COMPR	N	N	This attribute, which is only available for source databases Sybase SQL Anywhere or Oracle, indicates whether or not you want string comparisons to evaluate to TRUE regardless of any extra spaces before or after the string. Allowable values are Y or N. The default value is Y.
SRCDATEFORMAT	N	N	This attribute specifies the format of the date literals in the source file. For information on the source date format, see "Step 2: Converting source metadata" on page 28. The default values are: • **Oracle:** DD-MON-RR • **Informix Dynamic Server:** MDY4 • **Sybase SQL Anywhere, Sybase Adaptive Server Enterprise, and SQL Server:** mdy

Name	Required	Multiple occurrences allowed	Description
SRCDATELANGUAGE	N	N	This attribute specifies the language of the date literals in the source file. The available values are: • **Oracle:** AMERICAN, SIMPLIFIED_CHINEESE, FRENCH, GERMAN, DANISH, SPANISH, ITALIAN, DUTCH, NORWEGIAN, PORTUGUESE, FINNISH, SWEDISH, HUNGARIAN, ROMANIAN, CROATIAN, SLOVENIAN, GREEK, RUSSIAN, TURKISH, ENGLISH, ESTONIAN, LATVIAN, LUTHUANIAN, BRAZILLIAN_PRTUGUESE • **Informix Dynamic Server, Sybase SQL Anywhere, Sybase Adaptive Server Enterprise, and SQL Server:** ENGLISH The default values are: • **Oracle:** American • **Informix Dynamic Server, Sybase SQL Anywhere, Sybase Adaptive Server Enterprise, and SQL Server:** English
TRANSTABLESPACE	N	N	This attribute specifies whether or not you want to allow for the table properties to be added after translation. Allowable values are Y or N. If you select Y, the TABLESPACE clause is copied as-is and you will need manually edit it before deploying to DB2 UDB or Informix Dynamic Server. The default value is N.
TRANSKEYSTOAL TERTABLE	N	N	This attribute, which is only available for source databases Sybase or Microsoft SQL Server, specifies whether or not to convert the system procedures into ALTER TABLE statements to define the keys. Because these system procedures are frequently used for documentation purposes only, the default is that they are not converted to ALTER TABLE statements. Allowable values are Y or N. The default value is N.

Name	Required	Multiple occurrences allowed	Description
TRGTVARIABLEPREFIX	N	N	This attribute, which is only available for source databases Sybase SQL Anywhere, Sybase Adaptive Server Enterprise, and SQL Server, specifies the prefix for to be used for source variables. Use only alphanumeric characters. For example, if a source variable begins with a prefix of @, the prefix must be changed to an acceptable prefix. The default value is V_.

Related tasks

"Step 2: Converting source metadata" on page 28
The Convert step converts source metadata to target database metadata. You can specify various options that affect the converted output before you convert source SQL into DB2 UDB or Informix Dynamic Server SQL.

Related reference

"CONVERT" on page 79
The CONVERT element contains various options available for the converted output.

IMPORT:

The IMPORT element contains information that specifies what objects to import for data migration.

There can be more than one occurrence of the IMPORT element to import multiple source files. The IMPORT element contains the local file path to import. For example:

```
<IMPORT>c:\file1.src</IMPORT>
```

Related tasks

"Example: importing source SQL files" on page 100
In this example you are shown how to import source SQL files using the MTK command line interface.

JDBC_CONNECTION:

The JDBC_CONNECTION element contains information for use when connecting to the source database using JDBC.

The JDBC_CONNECTION element can contain the following elements and attributes:

Name	Required	Multiple occurrences allowed	Description
IP	Y	N	This attribute specifies the IP (internet protocol) address of the source database.
PORT	Y	N	This attribute specifies the port number of the source database.

Name	Required	Multiple occurrences allowed	Description
SERVICE	Y	N	This attribute, which is only available for source database Oracle, specifies the service name.
DATABASE	Y	N	This attribute, which is only available for source database Informix Dynamic Server, species the database name.
SERVER	Y	N	This attribute, which is only available for source database Informix Dynamic Server, specifies the server name.
DATABASE_LOCALE	N	N	This attribute, which is only available for source database Informix Dynamic Server, specifies the database locale.
CLIENT_LOCALE	N	N	This attribute, which is only available for source database Informix Dynamic Server, specifies the client locale.
ADVANCED_ PROPERTIES	N	N	This attribute, which is only available for source database Informix Dynamic Server JDBC, specifies the advanced properties.

Related tasks

"Connecting to the source database" on page 17
Before you extract data, you need to connect to the source database.

Related information

"Step 1: Specifying the source" on page 23
After a migration project is created or opened, you can start the migration process. Your first step is use the Specify Source page to obtain the source files to be converted to the SQL of the target database server.

ODBC_CONNECTION:

The ODBC_CONNECTION element contains information for use when connecting to the source database using ODBC.

The ODBC_CONNECTION element can contain the following attribute:

Name	Required	Multiple occurrences allowed	Description
DSN	Y	N	This attribute specifies the data source name of the source database.

Related tasks

"Connecting to the source database" on page 17
Before you extract data, you need to connect to the source database.

Related information

"Step 1: Specifying the source" on page 23
After a migration project is created or opened, you can start the migration process. Your first step is use the Specify Source page to obtain the source files to be converted to the SQL of the target database server.

PROJECT:

All of the information that describes a project is contained the PROJECT element. Every configuration file must contain exactly one PROJECT element.

The PROJECT element can contain the following elements and attributes:

Name	Required	Multiple occurrences allowed	Description
NAME	Y	N	This attribute specifies the name of the project. It can contain only alphanumeric and underscore characters. There is a character limit of 255.
CONVERSIONS	Y	N	This element contains information about conversions.
DIRECTORY	Y	N	This attribute specifies the directory for the project. The DIRECTORY attribute must conform with the file system convention.
SRCDBTYPE	Y[1]	N	This attribute specifies the type of database to migrate from. Allowable values are: • ids oracle • sql_server • sybase_asa • sybase_ase
TRGTDBTYPE	Y[1]	N	This attribute specifies the target database to migrate to. Allowable values are: • viper_2 • viper • db2_luw_8.2 • db2_luw_8.1 • db2_i_v5r4 • db2_i_v5r3 • db2_i_v5r2 • db2_z_8 • ids_10
DESCRIPTION	N	N	This attribute contains a description of the project.
SPECIFY_SOURCE	Y	N	This attribute contains information about files to import and extractions to perform.
Note: 1. These attributes are required only when creating a new project.			

Here is an example of the PROJECT element:

```
<MTK>
  <PROJECT NAME="" DIRECTORY=""
  DESCRIPTION=""
  SRCDBTYPE=""
  TRGTDBTYPE="">
  ...
  </PROJECT>
</MTK>
```

Related concepts

"Migration projects" on page 11
The IBM Migration Toolkit (MTK) enables you to manage your migration through ″projects,″ which enables you to contain all of the information associated with a migration. Project files are organized in a project directory.

"Arguments" on page 75
The MTK command line interface uses arguments to invoke MTK. The arguments tell MTK what actions to perform.

Related tasks

"Creating a project" on page 12
When you start MTK for the first time, you are prompted to create a project. You can also create a new project at any time.

"Opening an existing project" on page 13
You can open an existing project when you start the IBM Migration Toolkit or when another project is open. If a project is already open, MTK automatically saves and closes that project.

"Closing a project" on page 14
You can close a project at any time.

"Saving a project" on page 13
Projects are automatically saved periodically during the migration and when you exit the application. You can also save a project at any time.

"Importing source files" on page 24
You can specify that an external file be translated if the file contains valid source SQL.

"Modifying project description" on page 14
You can modify the description of an open project.

"Deleting a project" on page 14
You can drop either the currently opened project or another project. When you drop a project, the project directory and all its contents are deleted, including the imported source files, converted source, logs, and reports.

"Backing up a project" on page 15
You can back up a project directory into another directory or a zip file. It is recommended that you back up your projects on a regular basis.

"Restoring a project" on page 15
If a project is lost or becomes corrupted, you can rebuild it from a backup copy of the project. Therefore, it is recommended that you back up projects regularly.

"Running on Microsoft Windows" on page 73
You can run MTK from the command line using the Microsoft Windows operating system.

"Running on UNIX and Linux" on page 74
You can run MTK from the command line using the UNIX or Linux operating system.

Related reference

"Configuration file" on page 75
The command line interface uses the information specified in the configuration file to perform each migration step.

SCHEMA:

The SCHEMA element contains information that specifies the schema for the source database.

The SCHEMA element can contain the following elements and attributes:

Name	Required	Multiple occurrences allowed	Description
NAME	Y	N	This attribute specifies the schema name.
PACKAGE	N	Y	This element specifies the package to extract.
SEQUENCE	N	Y	This element specifies the sequence to extract.
TABLE	N	Y	This element specifies the name of the table to extract.
VIEW	N	Y	This element specifies the view to extract.
TRIGGER	N	Y	This element specifies trigger to extract.
PROCEDURE_ FUNCTION	N	Y	This element specifies the procedure or function to extract.

Related information

"Step 1: Specifying the source" on page 23
After a migration project is created or opened, you can start the migration process. Your first step is use the Specify Source page to obtain the source files to be converted to the SQL of the target database server.

SPECIFY_SOURCE:

The SPECIFY_SOURCE element contains information that specifies the source SQL file information for importing and extracting source objects..

It contains the IMPORT and EXTRACT elements.

The SPECIFY_SOURCE element can contain the following elements:

Attribute	Required	Multiple occurrences allowed	Description
IMPORT	N	Y	This element specifies a file on the local file system that contains source SQL to import into the project. This element should contain the full path to a file. For example: `C:\dir1\dir1\myFile.sql`, `/etc/mydb/schema1.sql`
EXTRACT	N	Y	This element specifies a file on the source database that contains source SQL to extract into the project.

Related concepts

"Arguments" on page 75
The MTK command line interface uses arguments to invoke MTK. The arguments tell MTK what actions to perform.

Related tasks

"Running on Microsoft Windows" on page 73
You can run MTK from the command line using the Microsoft Windows operating system.

"Running on UNIX and Linux" on page 74
You can run MTK from the command line using the UNIX or Linux operating system.

Related reference

"Configuration file" on page 75
The command line interface uses the information specified in the configuration file to perform each migration step.

Related information

"Step 1: Specifying the source" on page 23
After a migration project is created or opened, you can start the migration process. Your first step is use the Specify Source page to obtain the source files to be converted to the SQL of the target database server.

Command line migration projects

The MTK command line process enables you to manage your migration through projects.

Related concepts

"Migration projects" on page 11
The IBM Migration Toolkit (MTK) enables you to manage your migration through ″projects,″ which enables you to contain all of the information associated with a migration. Project files are organized in a project directory.

Create a project

With the MTK command line, projects are automatically created or opened when you specify your configuration file.

New project

If the project specified in the configuration file does not yet exist, MTK will create one when you specify your configuration file. All of the required settings for each migration step to be performed must be contained in the configuration file.

Existing project

If the project specified in the configuration file does exist, MTK will open the existing project. If you want to specify new source files and perform further migration steps on them, all of the required settings must be present in the configuration file. MTK never deletes an existing project when used through the command line.

If you want to work with a previously specified source file (in other words, a file that has been previously imported using the GUI or extracted using the command line), MTK will apply and prioritize settings. Settings from the CONVERSION element take the highest priority and override any settings found in the GLOBAL_SETTINGS element. The GLOBAL_SETTINGS element overrides any remaining settings found in the project. If the required settings are not specified by any of these information sources, MTK will log an error and not perform the desired operation.

Related concepts

"Migration projects" on page 11
The IBM Migration Toolkit (MTK) enables you to manage your migration through ″projects,″ which enables you to contain all of the information associated with a migration. Project files are organized in a project directory.

Save a project

When using MTK from the command line, project information is automatically saved when any step in the migration process completes.

You do not need to explicitly instruct MTK to save a project.

Related concepts

"Migration projects" on page 11
The IBM Migration Toolkit (MTK) enables you to manage your migration through ″projects,″ which enables you to contain all of the information associated with a migration. Project files are organized in a project directory.

Command line examples

The example procedures in this section show basic methods to invoke the command line. However, there are many other available attributes that are not shown in the these procedures.

Example: importing source SQL files

In this example you are shown how to import source SQL files using the MTK command line interface.

To import files, you must specify each file separately. There is no limit on the number of files that can be specified. The full path for each file must be specified.

To import a source SQL file:

Issue the following command:

```
MTKMain.bat -CONFIG config.xml -IMPORT
```

The contents of config.xml include:

```
<MTK>
  <PROJECT
      ...
      <SPECIFY_SOURCE>
            <IMPORT>c:\file1.src</IMPORT>
            <IMPORT>c:\file2.src</IMPORT>
      </SPECIFY_SOURCE>
      ...
  ></PROJECT>
</MTK>
```

Result:

- MTK imports file1.src and file2.src into the given project name.

Related reference

"IMPORT" on page 94
The IMPORT element contains information that specifies what objects to import for data migration.

Example: extracting objects from a source database

In this example you are shown how to extract objects from a source database using the MTK command line interface.

To extract objects from the source database:

Issue the following command:

```
MTKMain.bat –CONFIG config.xml -EXTRACT
```

The contents of config.xml include:

```
<MTK>
   <PROJECT>
      <SPECIFY_SOURCE>
            ...
            <EXTRACT
               EXTRACTTOFILE="extract1"
               SRCPWD="USECASE1"
               SRCUSR="USER1">
               <ODBC_CONNECTION DSN="SQLSERVER">
               </ODBC_CONNECTION>
               <DATABASE NAME="DBTEST1">
                    <SCHEMA NAME="HR">
                         <VIEW>SEEHREMPLOYEES1</VIEW>
                             <PROCEDURE_FUNCTION>INSERT_EMPLOYEE1
                             </PROCEDURE_FUNCTION>
                    </SCHEMA>
               </DATABASE>
            </EXTRACT>
            ...
            <EXTRACT
               EXTRACTTOFILE="extract2"
               SRCPWD="USECASE2"
               SRCUSR="USER2">
               <JDBC_CONNECTION IP="255.255.255.255" PORT="5125">
               </JDBC_CONNECTION>
               <DATABASE NAME="DBTEST2">
                    <SCHEMA NAME="OLLIE">
                         <TABLE>EMPLOYEE2</TABLE>
                    </SCHEMA>
               </DATABASE>
           </EXTRACT>
           ...
      </SPECIFY_SOURCE>
   </PROJECT>
</MTK>
```

Result:

1. MTK connects to the database at ODBC DSN "SQLSERVER" as user USER1 with password USECASE1.
2. MTK extracts both view SEEHREMPLOYEES1 and procedure INSERT_EMPLOYEE1 from the schema "HR" and database "DBTEST1" into file extract1.src. This file will be located in the project's root directory.
3. Then MTK connects to the SQL Server database at 255.255.255:5125 using JDBC as user USER2 with password USECASE2.
4. MTK extracts table EMPLOYEES2 from the schema "OLLIE" and database "DBTEST2" into file extract2.src.

   Related reference

"EXTRACT" on page 86
The EXTRACT element contains information that specifies what objects to extract as well as the connection information needed to access the source database.

Example: convert SQL from a source database

In this example you are shown how to convert source SQL using the MTK command line interface.

To perform a conversion, you must have already imported or extracted the source SQL files to convert.

To convert source SQL:

Issue the following command:

```
MTKMain.bat –CONFIG config.xml –CONVERT
```

The contents of config.xml include:

```
<MTK>
  <PROJECT...>
    ...
    <CONVERSIONS>
      <GLOBAL_SETTINGS>
            <GLOBALCONVERT
            SRCDATEFORMAT="dd-mon-rr"
            SRCDATELANGUAGE="ENGLISH"
            TRGTVARIABLEPREFIX="V_"
            DEFAULTSCHEMA="from_first_object"
            IGNORECASEINUSRDEFID="N">
            </GLOBALCONVERT>
      </GLOBAL_SETTINGS>
      <CONVERSION>
            <CONVERT SRCSQLFILE="file1.src"></CONVERT>
      </CONVERSION>
      <CONVERSION CONVERSIONNAME="myExtract1">
            <CONVERT SRCSQLFILE="extract2.src"
               IGNORECASEINUSRDEFID="Y"
            </CONVERT>
      </CONVERSION>
   </CONVERSIONS>
  </PROJECT>
</MTK>
```

The following actions take place:

1. The source SQL file file1.src is converted to target SQL file file1.db2 using the attributes specified in GLOBALCONVERT.
2. Report file file1.rpt is generated.
3. The source SQL file extract2.src is converted to target SQL file myExtract1.db2 using the attributes specified in GLOBALCONVERT and overriding the IGNORECASEINUSRDEFID attributes.
4. Report file meExtract1.rpt is generated.

Related reference

"CONVERSION" on page 78
The CONVERSION element represents a set of data that MTK will apply the given convert, generate data transfer scripts, and deployment operations. There can be zero or more CONVERSION elements per configuration file.

"CONVERSIONS" on page 78
The CONVERSIONS element contains the information needed to perform the convert, generate data transfer scripts, and deployment.

"CONVERT" on page 79
The CONVERT element contains various options available for the converted output.

Example: generating data transfer scripts

In this example you are shown how to generate data transfer scripts using the MTK command line interface.

To generate data transfer scripts:

Issue the following command:

```
MTKmain.bat –CONFIG config.xml –GENSCRIPT
```

The contents of the config.xml are:

```
<MTK>
  <PROJECT>
    <CONVERSIONS>
      <GLOBAL_SETTINGS>
            <GENERATE_DATA_TRANSFER_SCRIPTS
               LOADINGOPTIONS="LOAD"
               LOADIMPORTMODE="INSERT"
               UTF8="N"
               DIRECTORYFORDATAEXTRACTION="c:\"
               FILEFORMAT="DEL">
            </GENERATE_DATA_TRANSFER_SCRIPTS>
      </GLOBAL_SETTINGS>
      <CONVERSION CONVERSIONNAME="all_tables">
            <GENERATE_DATA_TRANSFER_SCRIPTS
               LOADINGOPTIONS="IMPORT">
            </GENERATE_DATA_TRANSFER_SCRIPTS>
     </CONVERSION>
     <CONVERSION CONVERSIONNAME="view123">
            <GENERATE_DATA_TRANSFER_SCRIPTS
               DIRECTORYFORDATAEXTRACTION="j:\">
            </GENERATE_DATA_TRANSFER_SCRIPTS>
            <DEPLOY_TO_TARGET></DEPLOY_TO_TARGET>
      </CONVERSION>
    </CONVERSIONS>
  </PROJECT>
</MTK>
```

The following actions take place:

1. Data transfer scripts are generated for conversion "all_tables" using the following options:
   - Data is deployed using the IMPORT utility.
   - A new row is inserted and existing rows are unchanged.
   - Data is represented in delimited text files. The delimiter is a comma.
   - Data is not enclosed in UTF-8 format.
   - Scripts are generated in root C drive.
2. Data transfer scripts are generated for conversion ″view123″ using the following options:
   - Data is deployed using the LOAD utility.
   - A new row is inserted and existing rows remain unchanged. c.
   - Data is represented in delimited text files. The delimiter is a comma.

- Data is not enclosed in UTF-8 format.
- Scripts are generated in root J drive.

Related reference

"GENERATE_DATA_TRANSFER_SCRIPTS" on page 88
The GENERATE_DATA_TRANSFER_SCRIPTS element is where you specify any data transfer script attributes.

Example: deploying converted objects

In this example you are shown how to deploy converted objects using the MTK command line interface.

To deploy converted objects:

Issue the following command:

```
MTKMain.bat -CONFIG config.xml -DEPLOY
```

The contents of config.xml include:

```
<MTK>
  <PROJECT>
    ...
    <CONVERSIONS>
      <GLOBAL_SETTINGS>
            <DEPLOY_TO_TARGET
               TRGTUSR="DEPLOY1"
               TRGTPWD="ADEPLOY1"
               TRGTDBNAME="DEPLOYDB"
               DBLOCATION="LOCAL"
               USECURRENTSYSLOGIN="N"
               LAUNCHINDB="Y">
            </DEPLOY_TO_TARGET>
      </GLOBAL_SETTINGS>
      <CONVERSION CONVERSIONNAME="myExtract1">
            <DEPLOY_TO_TARGET
               LAUNCHSCRIPT="y">
            </DEPLOY_TO_TARGET>
      </CONVERSION>
      <CONVERSION CONVERSIONNAME="file1">
            <DEPLOY_TO_TARGET
               TRGTUSR="DEPLOY2"
               TRGTPWD="ADEPLOY2"
               LAUNCHSCRIPT="y">
               EXTRACTDATATOSYS="n"
               LOADDATATOTRGT="n">
            ></DEPLOY_TO_TARGET>
      </CONVERSION>
    </CONVERSIONS>
  </PROJECT>
</MTK>
```

The following actions take place:

1. A connection is made to the target database using DEPLOY1 and ADEPLOY1 as the connection parameters (the user ID and password respectively).
2. The source database objects from project conversion myExtract1 are deployed to the local target database DELPLOYDB.
3. A connection is made to the target database using DEPLOY2 and ADEPLOY2 as the connection parameters (the user ID and password respectively).
4. The source database objects from project conversion file1 are deployed to the local target database DELPLOYDB.

Related reference

"DEPLOY_TO_TARGET" on page 84
The DEPLOY_TO_TARGET element contains target database connection information and information about what options to use during deployment.

Example: deploying converted objects with advanced options

In this example you are shown how to deploy converted objects with advanced attributes using the MTK command line interface.

To deploy converted objects with advanced options:

Issue the following command:

```
MTKMain.bat -CONFIG config.xml -DEPLOY
```

The contents of config.xml are:

```
<MTK>
  <PROJECT>
        ...
        <CONVERSIONS>
             <GLOBAL_SETTINGS>
                  <DEPLOY_TO_TARGET
                  TRGTUSR="DEPLOY1"
                  TRGTPWD="ADEPLOY1"
                  TRGTDBNAME="DEPLOYDB"
                  DBLOCATION="LOCAL"
                  USECURRENTSYSLOGIN="N"
                  LAUNCHINDB="Y"
                  EXTRACTDATATOSYS="Y"
                  LOADDATATOTRGTDB="Y"
                  RUNSTATONCATLG="Y"
                  RUNSTATONLOADIMPORT="Y"
                  DEPLOYMTKUDF="Y">
                  </DEPLOY_TO_TARGET>
             </GLOBAL_SETTINGS>
             <CONVERSION CONVERSIONNAME="myExtract1">
                  <DEPLOY_TO_TARGET
                  LAUNCHSCRIPT="N"
                  EXTRACTDATATOSYS="Y"
                  LOADDATATOTRGT="Y">
                  </DEPLOY_TO_TARGET>
             </CONVERSION>
             ...
             <CONVERSION CONVERSIONNAME="file1">
                 <DEPLOY_TO_TARGET
                 LAUNCHSCRIPT="N"
                 EXTRACTDATATOSYS="Y"
                 LOADDATATOTRGT="Y">
                 </DEPLOY_TO_TARGET>
             </CONVERSION>
             ...
       </CONVERSIONS>
 </PROJECT>
</MTK>
```

Result:

1. Data is extracted from the source database to the current system MTK is running by using the scripts generated from conversion myExtract1.
2. The data is now loaded to the target database using the connection parameter specified in the global settings to connect to the target database. 3.
3. Data is extracted from the source database to the current system MTK is running by using the scripts generated from conversion file1.

4. The data is now loaded to the target database using the current system user name and password to connect to the target database.

   Related reference

   "DEPLOY_TO_TARGET" on page 84
   The DEPLOY_TO_TARGET element contains target database connection information and information about what options to use during deployment.

Example: end to end migration

In this example you are shown and end-to-end migration using the MTK command line interface.

Migration can be performed in one execution of MTK from the command line interface. The configuration file must have all the necessary information MTK. The order the flags appear is irrelevant.

To perform an end-to-end migration:

Issue the following command:

```
MTKMain.bat –CONFIG config.xml –ALL
```

The contents of config.xml are:

```
<MTK>
     <PROJECT NAME="MTKCLI"
                 DIRECTORY="c:\mtk\projects"
                 DESCRIPTION="this is cli config file"
                 SRCDBTYPE="sql_server"
                 TRGTDBTYPE="db2_luw_9">
          <SPECIFY_SOURCE>
                 <IMPORT>c:\file1.src</IMPORT>
                 <EXTRACT
                      EXTRACTTOFILE="file2.src"
                      SRCPWD="pwd"
                      SRCUSR="name"
                      ALLOBJECTS="Y">
                      <ODBC_CONNECTION DSN="SQLSERVER">
                      </ODBC_CONNECTION>
                      <DATABASE NAME="MYDB">
                           <SCHEMA NAME="HR">
                                <TABLE>T1</TABLE>
                                <VIEW>V1</VIEW>
                                <PROCEDURE_FUNCTION>F1</PROCEDURE_FUNCTION>
                           </SCHEMA>
                      </DATABASE>
                 </EXTRACT>
          </SPECIFY_SOURCE>
          <CONVERSIONS>
               <GLOBAL_SETTINGS></GLOBAL_SETTINGS>
          <CONVERSION>
                   <CONVERT SRCSQLFILE="file1.src"></CONVERT>
               <GENERATE_DATA_TRANSFER_SCRIPTS></GENERATE_DATA_TRANSFER_SCRIPTS>
               <DEPLOY_TO_TARGET
                    CONVERSIONNAME="file1"
                    TRGTUSR="user1"
                    TRGTPWD="user1"
                    TRGTDBNAME="start"
                    DBLOCATION="local"
                    LAUNCHSCRIPT="y"
                    EXTRACTDATATOSYS="y"
                    LOADDATATOTRGTDB="y">
               </DEPLOY_TO_TARGET>
          </CONVERSION>
          <CONVERSION>
```

```
                    <CONVERT
                         SRCSQLFILE="file2.src"></CONVERT>
              <GENERATE_DATA_TRANSFER_SCRIPTS></GENERATE_DATA_TRANSFER_SCRIPTS>
              <DEPLOY_TO_TARGET
                    CONVERSIONNAME="file2"
                    TRGTUSR="user2"
                    TRGTPWD="user2"
                    TRGTDBNAME="stop"
                    DBLOCATION="local"
                    LAUNCHSCRIPT="y">
              </DEPLOY_TO_TARGET>
         </CONVERSION>
         </CONVERSIONS>
   </PROJECT>
</MTK>
```

Result:

1. Source SQL file file1.src is imported into the project.
2. Connection to the source database is made using the connection credentials below:
   - User name: name
   - Password: pwd
   - DSN: SQLSERVER
3. Table T1, view V1, and procedure P1 are extracted to source SQL file file2.src (all from the HR schema).
4. File file1.src is converted.
5. File file2.src is converted.
6. Data transfer scripts are generated for conversion file1.
7. Data transfer scripts are generated for conversion file2.
8. Conversion file1.src is deployed to the target database (objects are deployed and data is copied from the source database to the local system and then moved to the target database). This database is a local database. The connection parameters are:
   - User name: user1
   - Password: user1
   - Database name: start
9. Conversion file2 is deployed to the target database (i.e. objects are deployed). This database is a local database. The connection parameters are:
   - User name: user2
   - Password: user2
   - Database name: stop

Related information

"Elements" on page 77
The configuration file contains elements along with element attributes.

The wizard process

The MTK wizard offers a streamlined version for use in simple database migrations.

Related concepts

"Wizard" on page 9
You can invoke MTK using the wizard.

Step 1: Project Information

In this panel of the wizard you specify the necessary project information.

To specify the project information:

1. In the **Project Name** field, specify your project name. Your project name must consist only of letters, digits, or the underscore character. Spaces are not allowed.
2. Click on the **Source database type** field to select your source database type.
3. Click on the **Target database type** field to select your target database type.
4. Click **Next**.

Step 2: Source Database

In this panel of the wizard you specify the source database information.

You can select to either have the tool extract the information directly from a database server or you can have it import the metadata information from a file that contains the previously extracted metadata.

To specify the source database information:

1. Select either **Extract from a server...** or **Import from a file...**.
   - If you selected **Import from a file...**:

     From the Import file window, select the desired source file and click **Import file**.
   - If you selected **Extract from a server...**:

     If a database connection does not exist for this project, the Connect to Database window will open. Follow the step-by-step instructions for connecting to your source database: "Connecting to the source database" on page 17.
2. Click **Next**.

   Related tasks

   "Connecting to the source database" on page 17
   Before you extract data, you need to connect to the source database.

Step 3: Global Type Mapping

In this panel of the wizard you can verify or change the default global type mapping information.

A three-column table is shown in this panel of the wizard. The **Source type** and **Target type** columns provide data type mapping information. If a **Target type** field contains an edit (📝) icon, you can edit it.

To edit the default global type mapping information:

1. In the **Target type** column, click the field that you want to edit.
2. Change the values as appropriate. Click **Apply**.
3. Optional: For conversions from Sybase only, if you have already converted at least once, you can click **Restore previous mapping** to restore data type mappings to the mapping relationship used in the last conversion.
4. Click **Next**.

   Related tasks

   "Mapping data types" on page 35

Step 4: Data Movement

In this panel of the wizard you specify whether or not you want to generate the data transfer scripts for deployment.

To specify whether or not to generate data transfer scripts for deployment:

1. Select either **Do not prepare data for deployment** or **Generate the data movement scripts only, for later use**.
2. Click **Next**.

Step 5: Target Database

In this panel of the wizard you specify whether or not you want to deploy the database objects into the target database.

To specify whether or not you want to deploy the database objects:

1. Select either:
   - **Do not deploy**
   - **Deploy...**
     a. Type a new or existing target database name or alias in the **DB2 database name** field. The name cannot exceed eight characters and must begin with an alphabetic character, $ or @.
     b. Select either **Use a local database** or **Use a remote database**. If you select **Use a local database**, you can choose **(Re)create** to recreate a new local database. If you do not select **(Re)create**, the objects and data will be added to the existing database.

        When a database is created, a bufferpool with pages of size 32 KB and three table spaces of the same size are created. These provide enough space for the deployment of tables with any row length. Before launching the database into production, perform some performance tuning and adjust the size of the bufferpool as necessary.
     c. Optional: Click **Advanced options**. Available options are as follows:

Table 6. DB2 advanced options

Field	Explanation
Territory	Specifies the national language territory setting of the target database. If an existing target database is used, the territory must match that of the database.
Code set	Specifies the national language code set setting of the target database. If an existing target database is used, the code set must match that of the database.
Use default collating sequence	Specifies that the character strings are sorted according to the default target database sequence.
Use system collating sequence	Specifies that character strings are sorted according to the code set specified in the target database.
Use identity collating sequence	Specifies that character strings are sorted according to their hexadecimal value.

     d. Click **OK**.
2. Click **Next**.

Step 6: Migration Summary

In this panel of the wizard you are shown a summary of the migration.

The stage needs to be set just so.

Review the migration summary information and if you are satisfied click **Finish**. If you would like to make changes before converting, click **Back** and repeat the steps as necessary.

After you click Finish, the data is migrated and you are shown a processing status screen that displays a summary of the migration.

Reports and logs

Various logs and reports are generated throughout the migration process to aid you in understanding every aspect of the process.

Viewing migration reports

You can verify the migration as it progresses using the reports that are provided.

To view migration reports:

Select **Tools** → **Migration reports** to obtain a series of reports on your migrations. You can navigate to the reports for the current or previous migrations using the contents listing in the left frame of the browser window.

The types of reports are as follows:

Conversion summary report
The conversion summary report presents details on a particular conversion. You will most likely have many conversions during your migration of each database. You can view each of the conversion reports and compare the objects that were successfully converted as well as various errors reported each time.

Translation error message reports
The error message reports detail the specific error messages found during each translation. One report presents the messages by object and the other by message. These reports are useful when determining how to prioritize your work, for example if the same error exists for many objects it might be a good idea to solve that problem first. Or, when viewing the messages by category, you might choose to overlook the translator information messages until the omissions are taken care of.

Estimate of table size report
The table size estimates are presented in this report to help you determine what kind of table space adjustments you might have to make.

Large statement warnings report
During conversion some statements are too long to be properly handled. The large statement warnings report lists each of the statements that could not be converted.

Verification report and deployment log
The verification report and deployment log are generated to present any problems that might have occurred during deployment. If, during deployment, you specified in-context files that contain objects that have not yet been deployed into the target database, both the report and log will show that the in-context files failed to deploy. Objects in in-context files are not deployed. This behavior is as designed.

Related tasks

"Viewing the application log" on page 112
The application log contains a detailed record of the events that have occurred during the migration process for the current project.

"Viewing the changes report" on page 112
The Changes report records changes made to the metadata during the

convert-refine process. Changes include Global Type Mapping changes that occur during the Convert step and edit changes that occurred during the Refine step.

"Removing conversion actions" on page 38
The changes report provides an interface that you can use to view what changes will be applied during the next conversion. If you do not want certain changes to take place, you can remove them using the report.

Viewing the application log

The application log contains a detailed record of the events that have occurred during the migration process for the current project.

To view the application log:

1. Select **Tools** → **Log**. The log is displayed with key events denoted by icons. The log contains three types of messages: information (blue), Error (red), and trace (yellow) messages. To view trace messages, start MTK in debug mode:
   ```
   ./MTKMain -debug
   ```
2. To view the log file in a text editor, click **Edit log file**.

   Related tasks

   "Viewing migration reports" on page 111
   You can verify the migration as it progresses using the reports that are provided.

   "Viewing the changes report"
   The Changes report records changes made to the metadata during the convert-refine process. Changes include Global Type Mapping changes that occur during the Convert step and edit changes that occurred during the Refine step.

   "Removing conversion actions" on page 38
   The changes report provides an interface that you can use to view what changes will be applied during the next conversion. If you do not want certain changes to take place, you can remove them using the report.

Viewing the changes report

The Changes report records changes made to the metadata during the convert-refine process. Changes include Global Type Mapping changes that occur during the Convert step and edit changes that occurred during the Refine step.

Each entry in the report describes a change and indicates whether or not the change has gone through conversion. Changes are not applied until after the next conversion. If you continue to make changes while in the convert-refine process, previous changes are retained in the Changes report. The changes are identified by a Y in both the Last and Next columns. The changes will be applied each time you redo the convert step.

To view the Changes report

Select **Tools** → **Changes report** to view the report.

The columns of the report are as follows:

Column	Description
Last	Relates the change to the last conversion: • **Y:** The change was applied during the last conversion. • **N:** The change was not yet applied.
Next	Relates the change to the next conversion: • **Y:** The change will be applied at the next conversion. (This is normally the status directly after making the change. The change has not yet taken effect.) • **N:** The change was deleted after it was applied. (The actual deletion does not take place until after you reconvert.)
Type	Lists the type of change: • **Column Rename:** Indicates that a column has been renamed. • **Table Rename:** Indicates that a table has been renamed. • **Procedure Rename:** Indicates that a procedure name has been renamed. • **Edited content:** Indicates that a change has been made to a procedure or trigger body. • **Type Mapping:** Indicates that a global type mapping change has been made.
Changes	Provides detailed information about each change.

Related tasks

"Viewing migration reports" on page 111
You can verify the migration as it progresses using the reports that are provided.

"Viewing the application log" on page 112
The application log contains a detailed record of the events that have occurred during the migration process for the current project.

"Removing conversion actions" on page 38
The changes report provides an interface that you can use to view what changes will be applied during the next conversion. If you do not want certain changes to take place, you can remove them using the report.

Frequently asked questions

Here are some MTK frequently asked questions.

- “What is MTK?” on page 116
- “Why is the IBM Migration Toolkit Needed?” on page 116
- “What does MTK convert?” on page 116
- “What does MTK not convert?” on page 117
- “What is involved in a database migration?” on page 117
- “Installation directory name” on page 117
- “MTK won’t start at all” on page 117
- “Source metadata extraction” on page 117
- “Translation problems” on page 118
- “Refine problems” on page 118
- “Deployment to DB2” on page 118
- “What source databases are supported?” on page 118
- “On what platforms does MTK run?” on page 118
- “Migrating DB2 UDB V7.2 to DB2 UDB V8.1 on a Windows platform” on page 119
- “DB2 database connection problem” on page 119
- “DB2-specific questions” on page 119
- “Source database connectivity errors” on page 120
- “Out of memory errors” on page 120
- “Continuing out of memory errors” on page 120
- “What are the steps for conversion?” on page 120
- “In Sybase or SQLServer conversion, CREATE DEFAULT and sp_bindefault statements are not replaced by DB2 statements” on page 120
- “Why does MTK produce separate files for TEXT, IMAGE, CLOB, or BLOB data?” on page 120
- “Why isn’t the data moved?” on page 121
- “Common setup issues” on page 121
- “MTK fails to load LOB data into the DB2 table” on page 121
- “When I selected from my table I saw references (such as ″D161/C20_807113.out″) instead of the actual LOB data” on page 122
- “I want to distribute MTK UDF’s with my migrated application” on page 122
- “I have encountered ″transaction log full″ problem when deploying data to DB2 by selecting the ″import″ function” on page 122
- “I have some stored procedures in MSSQL with default values. It looks like this functionality is not supported in DB2 - is that correct?” on page 122
- “Problem with moving Unicode data from SQL Server to DB2” on page 122
- “Codepage issues into DB2” on page 122
- “I am getting this error when I try to run the SQL Translator: --* [200040] ″C:\MTK\projects\testsql\SQLTranslator.input″(24:1)-(24:14) Input Error: card_trans_vue is not a valid table, view, or table function name.” on page 123
- “Whenever I try to migrate the objects from Sybase to DB2, by default it is moving all of the UDT data types also. How can I avoid this?” on page 123
- “How do we specify a specific Tablespace name for DB2 tables?” on page 123

- "MTK creates a lot of functions with SYB, MS7, ORA, or INFX schema. How do I migrate those from one DB2 database to another? " on page 123
- "The installation of the JAR file completed successfully, but creation of each JAVA UDF fails with DB21034E" on page 124
- "I want to keep the original comments on the translation of my stored procedures" on page 124
- "I have my tables in one file and my procedures in another. When I translate the procedures with the tables file set as the context file, the translator says it cannot find the tables." on page 124
- "Why is a NULLABLE CHAR translated to NULLABLE VARCHAR? " on page 125
- "I cannot connect to the Oracle database" on page 125
- "I do not have an ODBC connection and the native driver option is grayed out" on page 125

What is MTK?

MTK is a utility to convert source DBMS constructs and data to those compatible with IBM DBMS:
- Originally developed for DB2 LUW
- Extended for DB2 on z/OS and iSeries and Informix Dynamic Server
- Supports wide range of source DBMS

MTK is a free downloadable utility:
- No part number
- Internally used by IBM migration specialists and externally used by business partners, ISVs, and more.

Why is the IBM Migration Toolkit Needed?

Because SQL is not all the same:
- Different standards levels
- Differences in ANSI compliance
- Differences in implementations:
  - Proprietary source extensions
  - Data types and type semantics
  - Syntax Procedural language
  - Built-in functions

What does MTK convert?

For supported source database platforms, MTK converts:
- DDL statements:
  - CREATE TABLE, CREATE INDEX, CREATE VIEW, and more.
  - ALTER TABLE and more.
  - Constraints
- SQL Statements:
  - SELECT, INSERT, UPDATE, DELETE, and more.
- Triggers
- Procedures

- Functions

What does MTK not convert?

- Applications
- Replication schemes
- OLAP specific features
- Catalog/system tables
- Statements specific to system administration
- Disk Partitioning Schemes

What is involved in a database migration?

1. Assess source database – size, complexity
2. Educate and train
3. Setup environment – develop, administer, test, production
4. Obtain source structure
5. Port source structure
6. Deploy objects to target
7. Migrate data
8. Verify target
9. Tune performance
10. Modify applications

Installation directory name

Here are some restrictions on installation directory name:

- MTK must not be installed into a directory with a space in the name. Everything will work, until deployment. During deployment, the DB2 commands to load the DB2 file into DB2 will fail because of the space in the file name.

MTK won't start at all

Java 1.4.2 or later must be installed and accessible through the PATH environment variable.

Source metadata extraction

The problem is probably with the connection to the source database. Either the ODBC or JDBC connection is not set up correctly, or the user ID does not have read access to the system catalog. See "DB2 database connection problem" on page 119 for details about the requirements for the JDBC connection.

Problems with this step are best diagnosed from the mtk.log which is found in the MTK installation directory. Since the mtk.log file is overwritten every time MTK starts, you need to:

1. Reproduce the problem.
2. Close MTK.
3. Send the mtk.log file to the MTK development team for further analysis.

Translation problems

Most of the common translation problems are caused by the use of context files, schema handling, and translation of statements that do not map easily to DB2.

The extractor puts statements in the file to set the database or schema environment for the objects in the file. Imported files often lack those USE, CONNECT, or SETUSER statements. The translator uses a default schema for the objects, but it does not always work for helping the translator resolve object names (especially if you are using a mixture of imported and extracted files), and the default schema rarely works for data extraction.

MTK users are sometimes confused about the default schema entry field. When it is set to ″from_first_object,″ the translator takes the schema for the first object as the schema to be replaced by the DB2 user ID used in deployment. When it is set to ″specify_schema_for_all_object,″ the translator keeps the schema on every object through deployment. The user can also type in a specific schema, which is then replaced by the IBM target server user ID used during deployment.

If the problem is not caused by user error or is an exception in the translator code, take the source and try to isolate the problem to the smallest test case possible. Provide this test case to the MTK development team.

Refine problems

When the default schema entry filed in the Convert step is set to ″from_first_object,″ the translator takes the schema for the first object as the schema to be replaced by the IBM target user ID used in deployment. Because of this, even if this schema for the first object is refined, it does not take effect as the IBM target user ID that is used during deployment and verification steps.

Deployment to DB2

Make sure you have your IBM target and java environment setup correctly.

To diagnose problems with data extraction, you want to look at mtk.log and the Verify_*.out file. To diagnose problems with deployment into DB2, you want to look at the Deploy_*.log files.

What source databases are supported?

- Sybase 11, 12, 12.5 SQL Server 7 and SQL Server 2000.
- SQLServer 6.5 has been reported to work, but is not tested.
- Oracle 7, 8, and 9.
- Informix 7.3 and Informix 9. Data movement from Informix 7.2 is limited by the lack of a TO_CHAR function to put dates into a format that DB2 consumes easily.

Generally, MTK does not process invalid source code successfully.

On what platforms does MTK run?

- Windows: Windows XP, Windows 2000.
- Unix: AIX 5L 5.2, Linux RHEL 3, and Solaris 2.9/9 platforms.
- Java on Solaris is hard to get installed and running. All Sun’s JREs come with a list of required solaris patches. http://sunsolve.sun.com/pub-cgi/

retrieve.pl?doctype=patch&doc=J2SE_Solaris_7_Recommended. The MTK readme has the recommended patches for Java on Solaris 7.

Migrating DB2 UDB V7.2 to DB2 UDB V8.1 on a Windows platform

The IBM Migration Toolkit is not designed to migrate from DB2 UDB to DB2 UDB. Please consider the following options to get proper help:

- Look at this IBM Redbooks™ publication which discusses moving data across the DB2 family: http://www.redbooks.ibm.com/abstracts/sg246905.html
- Look at this IBM Redbooks publication that highlights differences between DB2 systems: http://www.ibm.com/developerworks/db2/library/techarti
- Find help on the DB2 support website: http://www.ibm.com/db2/udb/support.html
- Contact your IBM representative and ask for support. If you have no IBM representative, send an email to mailto://db2mig@us.ibm.com.

DB2 database connection problem

Question

A customer received the following errors while trying to deploy their database from a local machine to a remote server, although they were able to connect to the database using the DB2 Client. They said it appeared that the message was looking for a DB2 UDB JDBC driver, but they were not sure where to check or what to do to correct the problem.

```
ERROR MTKDB2ConnectionUNO     DB2 UDB JDBC Driver not
found:COM.ibm.db2.jdbc.app.DB2Driver
```

```
ERROR DeploymentExtensionUNO  The DB2 Connection to the database
ASDTTST1 failed (rc=-6) with the following error message:No suitable
driver
```

Answer

MTK uses the JDBC driver COM.ibm.db2.jdbc.app.DB2Driver which is in db2java.zip. SQLLIB\java\db2java.zip is needed in the classpath.

Check your CLASSPATH by opening a window and typing `SET CLASSPATH`. The revealed classpath should include `D:\SQLLIB\java\db2java.zip;D:\SQLLIB\java\runtime.zip;D:\SQLLIB\java\sqlj.zip;D:\SQLLIB\bin;` with *`D:\SQLLIB`* replaced by the drive and path for the local installation of DB2.

The classpath should not include any *.dll files.

You can either change the classpath for the system or add the DB2 jar files to the classpath in MTKMain.bat file.

If you are running a local 32-bit application (such as MTK) in a 64-bit DB2 instance, LIBPATH, SHLIB_PATH, and LD_LIBRARY_PATH should contain sqllib/lib32 instead of sqllib/lib.

DB2-specific questions

- A request to size an xSeries® hardware configuration is not within the expertise or scope of MTK or the SMPO team. Techline can do this kind of thing. Check out the following website: http://www.ibm.com/support/americas/techline/sizing.html.
- A C++ compiler is required for stored procedures in DB2 before version 8.2. Alternatives are listed in the DB2 documentation.

Source database connectivity errors

Verify installation of the source database client.

Out of memory errors

The IBM Migration Toolkit runs in the available Java memory space. For large migrations that require more memory, you can allocate more memory space.

To allocate more memory space:

1. In the directory where the IBM Migration Toolkit is installed, locate and edit the launching file (MTK.bat, MTKMain, or MTKMainOra).
2. Add the -Xmx*NNN*m flag to the Java command, where *NNN* is the amount of memory, in megabytes, that you want to allocate. For example, to allocate 256 Mb of memory for migration, type the following command:

   ```
   java  -Xmx256m -classpath "..."
   ```

Do not change any of the other flags. The default size is 1/3 or 1/2 of the physical memory on the machine.

Continuing out of memory errors

The best workaround for this is to extract half of the objects into one file, then extract the other half of the objects into a second file. During conversion, you can select both files at the same time to convert all of the stored procedures. If you encounter memory restrictions during conversion, then you should convert by sections.

What are the steps for conversion?

See the MTK help documentation. From inside the MTK interface, go to **Help** → **Help Content** and “The GUI process” on page 23.

In Sybase or SQLServer conversion, CREATE DEFAULT and sp_bindefault statements are not replaced by DB2 statements

The CREATE DEFAULT statement is translated by adding a DEFAULT clause to the column named in the sp_bindefault statement. DB2 does not name defaults, so the default name is not migrated to DB2.

Why does MTK produce separate files for TEXT, IMAGE, CLOB, or BLOB data?

There are potential problems with putting LOB data inline if the LOB values are long. With ASCII format, each row has to have enough space for the longest LOB value.

With delimited format, binary data (BLOB) or strings of undefined content, as it is often the case with CLOB data, may cause problems. If the target is specified as DB2 Version 8, MTK will use LOB LOCATOR STRINGS, and put lob data for many rows into a single file.

Why isn't the data moved?

Assuming you have checked the ″Extract and store data on this system″ and ″Load data to target database using generated scripts″ boxes on the deployment page, the problem is probably that you are connected to Oracle with a userID that does not have SELECT permission on the tables you are trying to migrate. There will be error messages saying this in the log file (mtk.log in the installation directory).

MTK can extract the metadata because that only requires SELECT permission on the system catalog, however, extracting data requires SELECT permission on the specific table.

Common setup issues

Cannot connect to the source database
Refer to the MTK documentation for information on setup of source database clients: .

MTK on Unix does not support connections to source database via ODBC
Make sure you have the required JDBC clients on Unix.

Common usage issues

SQL 1042C error on DB2 on Linux when running MTK Java UDFs
Make sure your Java environment for Linux is correctly set. For more information, refer to DB2 Information center at http://publib.boulder.ibm.com/infocenter/db2luw/v8//index.jsp

DB2 UDB JDBC Driver not found
Make sure SQLLIB\java\db2java.zip is needed in the classpath.

MTK fails to load LOB data into the DB2 table

You have run into a DB2 limitation that afflicts remote deployment. Remote deployment does not work with LOB data, because DB2 requires a path on the server for the LOBSPATH parameter.

If the DB2 server can connect to the source database server, either MTK can be installed on the DB2 server to run the migration there, or the scripts that MTK creates for moving data manually can be executed on the DB2 server, which gets around the LOBSPATH problem. There are detailed instructions for doing this in the help, in the section titled .

If the DB2 server cannot connect to the source database server, things get more complicated:

1. You can use the Directory for data extraction field on the Generate Data Transfer Scripts page to put the extracted data into a directory on the DB2 server.
2. After deploying the objects and extracting the data on the Deploy to DB2 panel, you can move the DataMove_*_db2.bat or DataMove_*_db2.sh file to the DB2 server.
3. Edit DataMove_*_db2.bat or DataMove_*_db2.sh to correct the path.
4. Run DataMove_*_db2.bat or DataMove_*_db2.sh to load the data into DB2

When I selected from my table I saw references (such as ″D161/C20_807113.out″) instead of the actual LOB data

The LOAD statement used should have two parameters to instruct DB2 on where to get the LOB data. For example:

```
LOBS FROM C:\MTK\projects\SQL2000\DataOutScripts\dbo_Categories\ MODIFIED
BY  LOBSINFILE
```

If either is missing, then references are written instead of LOB data.

I want to distribute MTK UDF's with my migrated application

Given that IBM provides you the UDFs ″as-is″ without warranty or indemnification, you can redistribute them with your application.

I have encountered ″transaction log full″ problem when deploying data to DB2 by selecting the ″import″ function

To fix this:

1. In the Generate Data Transfer Script panel, select **Import** and then select the **Advanced** options button.
2. In the dialog box that appears, enter a number of rows in the commit count field. If the DB2 database is configured with ″Log retain for recovery status″ set to no (the default), the log space will be recovered at each commit.

I have some stored procedures in MSSQL with default values. It looks like this functionality is not supported in DB2 - is that correct?

MTK stripped off the default values without issuing a message. The default parameter values are not supported by DB2. However, the converter does not lose the default value when it removes it from the procedure statement, instead it adds the default value in the calling statement.

Problem with moving Unicode data from SQL Server to DB2

The load or import statement for a unicode file needs the ″no check lengths″ option selected on the advanced options page before creating the data transfer scripts. The codepage may also need to be set on that page.

Codepage issues into DB2

In the Advanced Options for Generate Data Transfer Scripts panel there is a field labeled ″code page.″ This field needs to be filled in with the appropriate value so that the generated LOAD command contains the ″codepage=xxx″ modifier that tells DB2 to interpret the special characters in the transfer file correctly.

See this article for more information: http://www.ibm.com/developerworks/db2/library/techarticle/0

I am getting this error when I try to run the SQL Translator: --* [200040] ″C:\MTK\projects\testsql\SQLTranslator.input″(24:1)-(24:14) Input Error: card_trans_vue is not a valid table, view, or table function name.

The SQL Translator does not have a definition of card_trans_vue available. You can include the definition of card_trans_vue in the script window. If you have run a script containing the definition of card_trans_vue through the Convert step, you can select “Use all files” in the drop-down list next to the Paste button. Then the SQL Translator will have the definition of card_trans_vue available and will not issue the error.

Whenever I try to migrate the objects from Sybase to DB2, by default it is moving all of the UDT data types also. How can I avoid this?

MTK always extracts the UDT data types. The only way to avoid migrating those to DB2 is to edit the source file after extraction and delete the unneeded UDT data types in the file before conversion.

How do we specify a specific Tablespace name for DB2 tables?

The MTK team recommends that customers use the DB2 Control Center Wizard to create their tablespaces.

MTK provides an initial CREATE TABLESPACE statement at the beginning of the DB2 file when you chooses that option in the converter advanced options. Providing this CREATE TABLESPACE statement causes the CREATE TABLE statements to keep the IN tablespace parameter and put the tables into tablespaces of the correct names.

You need to add index and long tablespace clauses to the DB2 file after the column list:

```
►►──IN──tablespace-name1──┬──────────────────────────────┬──┬─────────────────────────────┬──────►◄
                          └─INDEX IN──tablespace-name2─┘  └─LONG IN──tablespace-name3─┘
```

MTK creates a lot of functions with SYB, MS7, ORA, or INFX schema. How do I migrate those from one DB2 database to another?

The functions with the SYB schema are being created by mtksyb.udf during the deployment. If you look in the Deploy_conversion.bat file, you will see a block of lines like this (with *cmp* replaced by your database name):

```
@ECHO Installing JAVA UDFs file under name syb.cmp...
DB2 -v -td! -f "C:\MTK\mtksybdrop.udf" >null
DB2 CALL SQLJ.REMOVE_JAR('syb.cmp') >null
DEL null
DB2 -v CALL SQLJ.INSTALL_JAR('file:C:\MTK\sybUDFs.jar','syb.cmp') >>%UDFLOGFILE%
DB2 -v -td! -f "mtksyb.udf" >>%UDFLOGFILE%
@ECHO Creation of MTK UDFs done.
```

Here is a guide for repeating the deployment of the UDFs into another database:

- SYB from mtksyb.udf or mtksybISeries.udf
- MS7 from mtkms7.udf or mtkms7ISeries.udf

- ORA from mtkora8.udf or mtkora8ISeries.udf
- INFX from mtkinfx.udf or mtkinfxISeries.udf

The installation of the JAR file completed successfully, but creation of each JAVA UDF fails with DB21034E

The command was processed as an SQL statement because it was not a valid Command Line Processor command. During SQL processing it returned:

```
SQL20204N The user defined function or procedure "PROCID" was unable to map
to a single Java method.  LINE NUMBER=3.  SQLSTATE=46008
```

The JAR file is installed under a different name than is used in the external name of the procedure. In other words, `CALL SQLJ.INSTALL_JAR('file:/db2home/db2inst2/worksp/scripts/sybUDFs.jar','syb.cmp')`:

```
CREATE FUNCTION SYB.procid()
RETURNS INTEGER
EXTERNAL NAME 'syb.udfjar:com.ibm.db2.tools.mtksybudf.sybUDFs.procid'
LANGUAGE JAVA
PARAMETER STYLE DB2GENERAL
DETERMINISTIC
NOT FENCED
NULL CALL
NO SQL NO EXTERNAL ACTION
DBINFO
```

When MTK prepares to deploy the UDF file to DB2, it changes the instances of udfjar in the external name to match the DB2 database name.

I want to keep the original comments on the translation of my stored procedures

MTK only preserves all comments (both those created by MTK and original sp comments) or removes all comments. Most of the comments produced by MTK are the original source statements. Because translation to DB2 cannot be a one-for-one mapping of statements, there is no way to copy only the source comments into the DB2 with any assurance that the comments would be placed somewhere that makes sense. Therefore, MTK keeps source comments only with the original source statements.

I have my tables in one file and my procedures in another. When I translate the procedures with the tables file set as the context file, the translator says it cannot find the tables.

The translator maps each database or schema combination into a schema for DB2, since all the objects go into a single DB2 database:

- The objects in the context file all go to dbo schema.
- Objects with dbo schema in the next database go into dbo1 schema.
- If the procedures file contains use database or a setuser username with a different database or username than appears in the tables file, it tells the translator that the following objects are in a new database or schema combination, which will map to dbo1.

The referenced tables are not found in schema dbo1, only in schema dbo, so it is not recognized as the same name.

You can either remove the use or setuser statement from the procedures file, or add the same statement to the tables file to tell the translator to treat all of the objects as being in the same database or schema combination. The translator will then successfully find the tables.

Why is a NULLABLE CHAR translated to NULLABLE VARCHAR?

Except for the special case of character types of length one (CHAR(1)), nullable character types are converted to variable length character types. This is because Sybase the behavior of VARCHAR is the same as a nullable CHAR. Meaning extra space padding is truncated and NULL values are allowed. However, in DB2 even if a CHAR column is nullable, it remains fixed length for all values except NULL. Also, in DB2 the default is that columns are nullable rather than non-nullable as is the default for Sybase.

I cannot connect to the Oracle database

The machine running MTK needs either an ODBC connection to the Oracle server or an Oracle client and classes111.zip or classes12.zip in the classpath in order for MTK to be able to access the Oracle server. The Oracle native driver connection uses a service name and relies on the Oracle client to use the information in the tnsnames.ora file to translate that into the details of the connection string.

The entry in the tnsnames.ora file on that machine supplies the host and port information. For example:

```
oracle =
(DESCRIPTION =
(ADDRESS_LIST =
(ADDRESS = (PROTOCOL = TCP)(HOST = machine.at.ibm.com)(PORT = 1521))
)
(CONNECT_DATA =
(SERVICE_NAME = oracle)
)
)
```

I do not have an ODBC connection and the native driver option is grayed out

Oracle
: MTK needs either classes111.zip or classes12.zip to be found in the classpath.

MS SQLServer 2000
: MTK needs three MS jar files to be found in the classpath. The jar files needed are:

- msbase.jar
- mssqlserver.jar
- msutil.jar

You can either add them to the system class path, or edit MTKMain.bat to add them. It will be easier to add them to the classpath if you first copy them to your c:\MTK directory.

MS SQLServer 7
: There is no native driver option. There is a Microsoft document titled “HOW TO: Get Started with Microsoft JDBC” at .

Sybase

MTK needs jconn2.jar to be found in the classpath. This can be downloaded as the JConnect tool from Sybase.

Informix

MTK needs ifxjdbc.jar, which is included in the MTK installation and in the MTKMain.bat classpath.

Introduction to the converter component

The converter component of the IBM Migration Toolkit is used to convert source SQL scripts to DB2 UDB or Informix Dynamic Server SQL scripts. The converter is run during the Convert step of MTK.

These reference topics are not intended to be a comprehensive overview of the features of the source and target SQL languages. The information here lists any translation limitations and provides some tips when you need to provide your own modifications to the SQL.

How the converter works

The converter is written in Java and uses ANTLR (ANother Tool for Language Recognition) as its parsing engine. It operates much like a compiler for a conventional programming language. It takes as input a sequence of source SQL scripts and generates a corresponding target SQL script as output.

The converter also generates *metadata* information about each source object definition and corresponding generated target object definition. The metadata is encoded in the XML-based Metadata Interchange (XMI) format for easy reuse. The metadata information summarizes important properties about source objects (for example, names and types of table columns), and is used by MTK to generate an overview of source and target objects in the Refine step.

Related tasks

"Step 2: Converting source metadata" on page 28
The Convert step converts source metadata to target database metadata. You can specify various options that affect the converted output before you convert source SQL into DB2 UDB or Informix Dynamic Server SQL.

"Referring to previously converted metadata" on page 34
If you are converting metadata that refers to other metadata that has already been converted, you still must include all of the files in the conversion. However, your translated data might be in a state where you would not want to translate it again. You can specify that the IBM Migration Toolkit use certain metadata *in-context*.

The source script file

A source SQL script can contain a sequence of SQL statements and procedure language commands. The SQL statements are translated as they are encountered in the script. The order in which source objects are defined in scripts is critical for proper conversion. As is the case with the source DBMS language interpreter, the converter requires that an object be *defined* before it is used. This might seem obvious, but it's a principle that can be easily overlooked when using many files for the same conversion. For example, the converter cannot translate a query unless it first encounters a definition.

```
create table t ...
SELECT * from t
```

Therefore, if you import the DDL and procedures as separate files, the file containing the DDL must occur first in the list of files to be translated. The converter will attempt to recover if an object is used before being defined, but the translation of the statement that uses the object could be omitted or incomplete.

If the SQL source is written using double-byte characters, temporary UTF-8 files are used internally for processing.

Script translation

The converter attempts to ensure that the generated code matches the style of the source whenever possible. However, certain source statements necessarily undergo considerable transformation in the translation process. Consider, for example, the following T-SQL source fragment:

```
create default yes_no_default as "Y"
...
exec sp_addtype yes_no_class, "char(1)", "not null"
...
exec sp_bindefault yes_no_default, yes_no_class
...
create table test1 (c1 yes_no_class)
```

When migrating to DB2 UDB, the converter generates the following code:

```
CREATE DISTINCT TYPE yes_no_class AS CHAR(1) WITH COMPARISONS!
...
CREATE TABLE test1 (c1 yes_no_class DEFAULT 'Y' NOT NULL)!
```

Although the DB2 UDB code behaves identically to the T-SQL code, for two of the source statements no corresponding target statements are generated. Instead, the effects of the source CREATE DEFAULT and EXEC SP_BINDEFAULT statements have been incorporated into the DEFAULT clause of the definition of column c1 in the target CREATE TABLE statement. In this example, the converter generates DB2 UDB code that is simpler than the source; in other cases, for example, those with certain nonstandard T-SQL query forms, the resulting translation might be more complex than the source. During the translation process, the converter might perform other complex transformations, for example:

- Renaming objects or identifiers that do not conform to the target DBMS naming conventions
- Altering declared types of columns or variables to their closest target DBMS equivalent
- Generating cursors where no cursor was used in the source (to simulate certain complex queries)
- Inserting explicit type casting operations to simulate implicit casts in the source
- Generating explicit triggers to simulate implicit operations in the source
- Generating new variables to contain auxiliary information maintained implicitly in the source
- Moving definitions and statements to alternate locations in stored procedures
- Generating sub-queries where no sub-queries were used in the source
- Calling UDFs from a compatibility library of functions, which are provided with MTK to simulate source functions that have no equivalent in the target DBMS

Context-dependent analysis

For certain source constructs the converter analyzes all of its uses in the script to infer the most appropriate translation. An example is a Sybase stored procedure that contains a row count variable. This context dependency means that the same object or variable definition might be converted differently in scripts that use the object in different ways.

Restrictions for migrating scripts

If you are importing scripts that have been created by the source DBMS command interpreter, some restrictions apply.

Interpreter commands

Commands for the source DBMS interpreter are ignored by the converter, except for CONNECT and EXECUTE which contain information useful for the translation.

Statement terminators

Certain SQL statements must be properly terminated according to the source language (for example, a ″/″ terminator after an Oracle CREATE statement). These statements are translated as is, and any subsequent procedure language commands that would alter the statement before running it are ignored.

SQL Plus substitution variables

Substitution variables such as &1 or &var are not supported by the converter. Such variables will cause syntax errors in the converter even though they may be valid in the SQL Plus script. Replace such variables by their definitions by hand before converting the scripts that contain them.

Static DDL restrictions

The converter is designed primarily for migrating scripts that contain *static* object definitions (DDL statements). *Non-static* scripts that create new definitions for objects during the conversion process will not be converted properly. This restriction is necessary to ensure that when the converter converts an object definition, all uses of the object are converted consistently.

Statements that update an object definition typically cannot be used. Such statements might occur in the script after a statement where the object is used. A single database catalog is used for the entire conversion so it is not possible to change definitions in the catalog during the conversion. You can, however, modify the scripts before conversion.

Related tasks

"Maintaining object dependencies" on page 36
In general, MTK keeps track of object dependencies; however, statements that update object definitions cannot be guaranteed to translate correctly.

Related reference

"ALTER" on page 282
For DB2 UDB deployments, ALTER statements (on tables and views) are converted to DROP and CREATE statements. For Informix Dynamic Server deployments, ALTER statements are translated except as noted. If you are converting multiple script files, ensure there are no dependencies on the altered object.

Object renaming

Because of differences in rules for identifiers, some objects must be renamed during the translation. The converter generates new names in such a way that they do not conflict with any preexisting names.

Renaming rules

When converting to Informix Dynamic Server

When converting from Sybase ASA to Informix Dynamic Server, be aware that Sybase ASA allows some characters in its identifiers that Informix Dynamic Server does not allow. Therefore, MTK converts the characters '@', '#', and spaces into underscores.

In migrations from Oracle to Informix Dynamic Server, delimited identifiers are retained.

The maximum identifier length in Informix Dynamic Server is 128 bytes.

When converting to DB2 UDB

In migrations to DB2 UDB, MTK converts delimited identifiers into non-delimited identifiers.

DB2 UDB restricts the length of identifiers to 16 characters, whereas SQL Server and Sybase allows 128 characters. Therefore, the translator truncates names and appends a number to ensure uniqueness.

Quoted identifiers are recognized and translated accordingly.

Renamed identifiers are order-dependent

Because of character replacement and truncation, the renaming of identifiers is quite common during translation. The name generation process ensures that each identifier is unique; however, if the converter is used to convert two scripts that contain the same object definitions, but in different orders, then the resulting scripts might contain inconsistently renamed objects. Therefore, take care when converting files separately.

For example, one file, ordered as shown, would translate the two objects to the same names as the other file, which is ordered differently. If converted separately, the translations would be incorrect because the resulting names represent different objects.

```
File 1:
...verylooooooooooooongname...
...verylooooooooooooong_othername...
   ----- is translated
to -----
...VERYL00000000001...
...VERYL00000000002...
```

```
File 2:
...verylooooooooooooong_othername...
...verylooooooooooooongname...
   ----- is translated
to -----
...VERYL00000000001...
...VERYL00000000002...
```

Related reference

"Identifiers" on page 192
Oracle SQL, DB2 UDB SQL, and IBM Informix Dynamic Server SQL have different length specifications for identifiers. The converter adjusts identifiers as necessary to meet the length limitations.

"Identifiers" on page 263
The converter adjusts identifiers as necessary to compensate for any length limitations.

"System catalogs" on page 133

"CREATE INDEX" on page 285
Sybase and SQL Server allow index names to be the same as long as they belong to different tables. DB2 UDB requires that index names are unique across the schema, and Informix Dynamic Server requires that indexes are unique for a particular user. The generated index names are changed as necessary.

Messages

The converter provides information on the success of the translation with error messages, which are written to the report file (*.rpt) and to the translated SQL script immediately preceding the relevant statement.

By default, each translated statement in the resulting script is preceded by a comment containing its corresponding source SQL statement, including any source comment statements. For example, the following DB2 UDB SQL contains the source Oracle SQL commented out.

```
--|create table t
--|(x date default sysdate);

CREATE TABLE T(X TIMESTAMP DEFAULT CURRENT TIMESTAMP)!
```

The comments are provided to help you understand how the generated code relates to the source and to help in cases when manual refinement is necessary. If you don't want the comments copied over during the translation, you can disable this feature in the preferences window.

You can view all translation messages in the Refine page, which is automatically opened after the Convert step. All of the converter messages fall under one of the following categories:

Translator Information
Messages in this category provide you with information of possible interest.

Translation Warning
Messages in this category alert you of cases where changes were made during translation that you might want to review. The translation should be correct, however, and allow the affected object or statement to deploy to the target database without error.

Translation Error
Messages in this category alert you of translation errors; that is, cases where a direct translation was not possible, where the translation might produce undesired results, or where more information needs to be provided. Changes must be made to the affected object or statement for it to be successfully deployed.

Input Script Error
Messages in this category alert you of errors in the input script; that is, cases where the source code is invalid or is presented incorrectly. For example, a common error is when the source refers to objects that have not yet been specified, which usually occurs when CREATE statements occur after a calling procedure. As with errors, input errors must be corrected for the affected objects and statements to be deployed to.

Two translator information messages are always given during a conversion:

- MTK*x*0000: Displays the version number of the converter used.
- MTK*x*0308: Displays the success ratio as a percentage of statements successfully converted.

Informix Dynamic Server converter

The following sections detail the support that the converter provides for translations from Informix Dynamic Server to DB2 UDB.

Converter behavior and limitations

The topics in this section describe how some principal features of the Informix Dynamic Server SQL and SPL source language might be effected during a conversion.

These reference topics are not intended to be a comprehensive overview of the features of the source and target SQL languages. The information here lists any translation limitations and provides some tips when you need to provide your own modifications to the SQL.

System catalogs

Any references to the system catalog are not translated.

Related concepts

"Object renaming" on page 129
Because of differences in rules for identifiers, some objects must be renamed during the translation. The converter generates new names in such a way that they do not conflict with any preexisting names.

Related tasks

"Changing an object name" on page 41

If an edit icon ( ) exists in the target column, you can change the name of the object.

Related reference

DB2 UDB Version 8 - SQL Limits

DB2 UDB Version 8 - Identifiers

Data types

MTK converts Informix Dynamic Server data types are into DB2 UDB data types according to the mapping described in the following table:

Informix Dynamic Server Data Type	DB2 UDB Data Type
BOOLEAN	CHAR(1)
SMALLFLOAT REAL FLOAT(n) DOUBLE PRECISION	REAL REAL DOUBLE DOUBLE

Informix Dynamic Server Data Type	DB2 UDB Data Type
SMALLINT	SMALLINT
INTEGER	INTEGER
BIGINT	BIGINT
INT8	BIGINT
DEC	DECIMAL(16,0)
DECIMAL(p)	DECIMAL(p,0)
DECIMAL(p,s)	DECIMAL(p,s)
SERIAL	INTEGER GENERATED BY DEFAULT AS IDENTITY
SERIAL(n)	INTEGER GENERATED BY DEFAULT AS IDENTITY (START WITH n)
SERIAL8	BIGINT GENERATED BY DEFAULT AS IDENTITY
SERIAL8(n)	BIGINT GENERATED BY DEFAULT AS IDENTITY (START WITH n)
INT is a synonym for INTEGER	
NUMERIC is a synonym for DECIMAL	
MONEY	DECIMAL(16,2)
MONEY(p)	DECIMAL(p,2)
MONEY(p,s)	DECIMAL(p,s)
CHAR(1..254)	CHAR(n)
CHAR(255..max)	VARCHAR(m)
i5/OS®: CHAR(n)	CHAR(n)
VARCHAR(m,r)	VARCHAR(n)
VARVCHAR(m,r)	VARCHAR(m)
CLOB	CLOB(2G)
TEXT	
LVARCHAR	
BLOB	BLOB(2G)
BYTE	
NCHAR(1..127)	GRAPHIC(n)
NCHAR(128..max)	VARGRAPHIC(n)
iSeries: NCHAR(128..max)	GRAPHIC(n)
NVARCHAR(m,r)	VARGRAPHIC(m)
	iSeries: with CCSID of 13488
DATE	DATE[2]
DATETIME SECOND..HOUR	TIME
DATETIME DAY..YEAR	DATE
DATETIME	TIMESTAMP
DATETIME YEAR TO FRACTION (5)	TIMESTAMP[3]

Informix Dynamic Server Data Type	DB2 UDB Data Type
INTERVAL (year, month) INTERVAL (day, second) INTERVAL (day, fraction)	DECIMAL(15,0) number of months[4] DECIMAL(15,0) number of seconds[5] DECIMAL(20,5) number of seconds + number of microseconds[5]
SET (e) LIST (e) MULTISET (e)	(not implemented)
ROW, Named ROW, Unnamed	(not implemented)
Distinct Opaque	(not implemented)

Note:

1. Date literals are translated according to the rules specified in the DBDATE and GL_DATE environment variables that are specified in the source script or MTK.
2. Literals in this range are automatically converted to a number of months. For example, INTERVAL(1-2) YEAR TO MONTH is translated to 14 (1*12+2).
3. Because the DB2 TIME datatype does not support a fraction, MTK translates HOUR TO FRACTION(5) to TIMESTAMP and inserts the default date (1900-01-01) into TIMESTAMP.
4. Literals in this range are automatically converted to a number of seconds. For example, INTERVAL(01 01:00:50) DAY TO SECOND is translated to 90050 (1*24*60*60+1*60*60+0*60+50).
5. Literals in this range are translated accordingly. For example, INTERVAL(10 10:01:01.01) DAY(5) TO FRACTION(2) is translated to 900061.01000.

Related reference

DB2 UDB Version 8 - Data types

Operators

The following date arithmetic limitations occur in translations from Informix Dynamic Server to DB2 UDB:

- UNIT is not supported (for example, `today + 5 UNITS day`).
- The rounding performed during multiplication or division of an interval by a float can be unpredictable.
- DB2 UDB does not recognize a year or month of 0; therefore, a DATETIME literal can be generated differently than expected if the upper bound is a value between a day and a year.

The tables detail how + and - operators are treated in Informix Dynamic Server and DB2 UDB. Other operators are not yet documented.

Table 7. Addition operator. With the column data type as the first operand added to the row data type as the second operand, the data type at the intersection results.

+	FLOAT	INT	DATE	DATETIME	INTERVAL
FLOAT	FLOAT	FLOAT	DATE (float truncated to number of days)	(invalid)	INTERVAL (converted as invalid)
INT	FLOAT	INT	DATE (number of days as int)	(invalid)	(invalid)
DATE	DATE (float truncated to number of days)	DATE (number of days as int)	DATE (converted as invalid)	(invalid)	DATETIME YEAR TO DAY
DATETIME	(invalid)	(invalid)	(invalid)	(invalid)	DATETIME (when interval range is within datetime range)
INTERVAL	INTERVAL (converted as invalid)	(invalid)	DATE	DATETIME (when interval range be with datetime range)	INTERVAL (when within same range; resulting range is that of the first operand)

Table 8. Subtraction operator. With the column data type as the first operand subtracting the row data type as the second operand, the data type at the intersection results.

-	FLOAT	INT	DATE	DATETIME	INTERVAL
FLOAT	FLOAT	FLOAT	DATE (converted as invalid)	(invalid)	INTERVAL (converted as invalid)
INT	FLOAT	INT	DATE (converted as invalid)	(invalid)	(invalid)
DATE	DATE (float truncated to number of days)	DATE (number of days as int)	INT (number of days as int)	INTERVAL DAY TO DAY	DATETIME YEAR TO DAY
DATETIME	(invalid)	(invalid)	INTERVAL (result as datetime	INTERVAL (result is smallest range; missing fields completed with current time)[1]	DATETIME (when interval range is within datetime range)

Table 8. Subtraction operator (continued). With the column data type as the first operand subtracting the row data type as the second operand, the data type at the intersection results.

-	FLOAT	INT	DATE	DATETIME	INTERVAL
INTERVAL	INTERVAL (converted as invalid)	(invalid)	(invalid)	(invalid)	INTERVAL (when within same range; resulting range is that of the first operand)

Built-in functions

There are three possible scenarios when converting Informix Dynamic Server built-in functions to DB2 UDB SQL:

- An Informix Dynamic Server built-in function has an equivalent DB2 UDB function. In this case, function calls are mapped directly to DB2 UDB SQL.
- An Informix Dynamic Server built-in function does not have an equivalent DB2 UDB function, yet a similar DB2 UDB function is available, which has only minor differences in the output (such as NULL, empty strings, or maximums).
- An Informix Dynamic Server built-in function has no DB2 UDB equivalent. In many of these cases a DB2 UDB SQL or Java UDF is used to provide similar functionality. These functions have names of the form `INFX`.function.

The following table shows the mapping between the Informix Dynamic Server built-in functions and their DB2 UDB equivalent.

Informix Dynamic Server function	Type	DB2 UDB function
abs	int	abs
mod	int	mod
pow	int	power
root	int	sqrt (1 parameter only)
round	int	round
sqrt	int	sqrt
trunc	int	trunc infx.trunc
exp	float	exp
logn	float	ln
log10	float	log10
hex	int	hex
length	string	infx.length
char_int character_int	string	
octet_length	string	(unsupported)
char_length character_length	string	(unsupported)

initcap	string	infx.initcap
upper	string	upper
lower	string	lower
lpad	string	infx.lpad
rpad	string	infx.rpad
replace	string	infx.replace
substr	string	infx.substr
substring	string	infx.substring
to_char	conv	infx.to_char*
to_date	conv	infx.to_date*
date	string, int	(unsupported)
today	date	current_date
day	date	day
month	date	month
weekday	date	infx.weekday
year	date	year
mdy	date	infx.mdy
extend	date	infx.extend
cos	float	cos
sin	float	sin
tan	float	tan
asin	float	asin
acos	float	acos
atan	float	atan
atan2	float	atan2
trim	string	infx.ltrim infx.rtrim
count	aggr	count
avg	aggr	avg
max	aggr	max
min	aggr	min
sum	aggr	sum
range	aggr	(unsupported)
stdev	aggr	stdev
variance	aggr	variance
nvl	misc	coalesce
locopy	misc	(unsupported)
lotofile	misc	(unsupported)
filetoblob	misc	(unsupported)
filetoclob	misc	(unsupported)
ifx_allow_newline	misc	(unsupported)

ifx_replace_module	misc	(unsupported)
dbinfo	misc	(unsupported)
user	misc	user
dbservername	misc	current_server
sitename	misc	(unsupported)
decode	exp	case expression

* The format elements that are supported for to_date and to_char functions are: %b, %B, %d, %D, %h, %H, %I, %m, %M, %S, %y, %Y, %iy, and %iY. The followings format elements are not supported: %%, %a, %A, %c, %C, %e, %F, %F<n>, %p, %P, %r, %R, %T, %w, %x, and %X.

Related concepts

"Data format recommendations" on page 38
When converting a database with DBCS data, you must give careful consideration to functions that handle character data. Where Sybase or SQL Server count characters, DB2 UDB counts bytes in functions such as left() or length().

Related reference

"Compatibility library (INFX functions) for translations to DB2" on page 149

Expressions

In translations from Informix Dynamic Server to DB2 UDB, character expressions with subscripts (such as x[1] or y[a,b]) are not yet supported.

The following expressions are supported:

- Arithmetic (see "Operators" on page 135 for limitations)
- Casts
- Column-level (except rowid, .field and .*)
- Column subscripts: column[first] and column[first,last] (translated using the SUBSTR built-in function)
- Case
- Constant (except DBSERVERNAME, SITENAME, time units, collections, rows, and opaque type literals)
- Function
- NULL
- Parenthesized
- Implicit type conversions, with the following limitations:
  - Float or smallfloat —> char: displayed as scientific notation rather than decimal.
  - Money <—> char: represented as decimal (no currency symbol and no limit on decimal places).
  - Interval —> char: not implemented.
  - Char —> datetime: fractions not supported.
  - Datetime —> datetime: when converting to one with a more significant upper bound, for example, from day to second or year to second, any missing date

fields are not filled in with the current date and time values, as is the behavior with Informix Dynamic Server. Currently, MTK uses the arbitrary date of 1999-01-01.

– String or number <—> serial or serial8: not yet tested.
– Text <—> byte: not yet tested.
– Number, datetime, interval, or serial <—> nchar or nvarchar: not yet tested.

Conditions

All conditions are translated.

SQL statements

The following notes describe how MTK handles the translation of some statements.

ALTER TABLE

- ADD COLUMN and ADD CONSTRAINT are the only statements supported.
- Adding a primary key or unique constraint might cause a NOT NULL constraint to be added to the indicated columns.

CONNECT

- DEFAULT and database server names are ignored.
- MTK uses the database name and the user name as the current database and owner names. This association is used for any unqualified database object names for the remainder of the script (or until another statement that changes these values is encountered).

CREATE DATABASE

Not translated, but the converter interprets the given database as the current (or default) database.

Tip: (Applicable to imported scripts only.) If a reference to a database occurs in an object name and no previous CREATE DATABASE statement for the database has been given, MTK issues a warning. The converter assumes that the database is defined with default options (specifically, that the database is a NON-ANSI database). If this assumption is not correct, the converter might produce ″duplicate definition″ and ″unknown object″ messages. To correct this assumption, you can add a CREATE DATABASE statement with the correct options in the input script and then reconvert.

CREATE FUNCTION and CREATE PROCEDURE

The distinction between functions and procedures is not as strong in Informix Dynamic Server as it is in DB2 UDB. The term routines is sometimes used to describe procedures and functions collectively.

- Procedures with no return values are translated to DB2 UDB procedures.
- Functions and procedures with multiple return values are translated as DB2 UDB procedures, with additional OUT parameters for the return values. The calls to these routines are modified accordingly.
- Functions with a single return value are translated to DB2 UDB functions unless the functions contain specific features that require them to be translated as DB2 UDB procedures, as described above.

Cursor functions (RETURN . . . WITH RESUME statement)

Informix Dynamic Server routines that contain statements of the form RETURN . . . WITH RESUME are called *cursor functions*, because they return a series of values to the caller. In SPL, these functions must be called from a FOREACH statement. In applications, you can declare a cursor to contain the results of the routine call and then use FETCH to retrieve the results.

MTK translates cursor functions into stored procedures that return a result set. MTK translates FOREACH EXECUTE statements (which call cursor functions) using a DB2 UDB cursor to process the result set returned by the stored procedure.

For a function containing a RETURN . . . WITH RESUME statement, MTK translates the function to a procedure. MTK sets up a global temporary table to collect the result set, and MTK declares a cursor that selects all of the elements of the global temporary table. An OPEN cursor statement is added to the end of the procedure body. This statement is labelled *end_func*. MTK translates each RETURN *expr_list* WITH RESUME to an INSERT into the global temporary table VALUES *expr_list*. MTK translates each RETURN in the function into a GOTO end_func.

A FOREACH EXECUTE *f* INTO variables statement is translated as follows. A result_set_locator variable *loc* is declared in the DB2 UDB procedure. SQLCODE and a curs_found flag are also declared. Within the body of the procedure, the EXECUTE f(...) is translated into CALL f(...). Then, the locator variable is associated with procedure f, and a cursor is allocated to the locator variable. A fetch into the variables from the cursor occurs, followed by a while loop (while the curs_found flag is true) that contains the translated block from the FOREACH statement, followed by another FETCH.

The following is a sample translation for a cursor function and a FOREACH that calls it:

```
create table t1 (x int);

CREATE FUNCTION series (limit INT, backwards INT)
RETURNING INT;
  DEFINE i INT;
  FOR i IN (1 TO limit)
    insert into t1 values (i+10);
    RETURN i WITH RESUME;
  END FOR
  IF backwards = 0 THEN
    RETURN;
  END IF
  FOR i IN (limit TO 1 STEP -1)
    insert into t1 values (i+10);
    RETURN i WITH RESUME;
  END FOR
END FUNCTION; -- series

create table t(x int);

create procedure testSeries()
  define y int;
  foreach execute function series(10,1) into y
    insert into t1 values (y+1000);
    insert into t values (y*10);
  end foreach;
  foreach execute function series(10,1)
  end foreach;
end procedure;

execute procedure testSeries();

select * from t;

select * from t1;
-------IS TRANSLATED TO:-------

CREATE TABLE t1(x INTEGER)!

CREATE PROCEDURE series (limit INTEGER,
                         backwards INTEGER )
LANGUAGE SQL
BEGIN
    DECLARE GLOBAL TEMPORARY TABLE result_table(
        value1 INTEGER
    ) WITH REPLACE ON COMMIT PRESERVE ROWS NOT LOGGED;

    BEGIN

        DECLARE i INTEGER;
        DECLARE lower INTEGER;
        DECLARE upper INTEGER;
        DECLARE incr INTEGER;
        DECLARE temp_cursor CURSOR WITH HOLD
                WITH RETURN TO CALLER FOR
                 SELECT * FROM SESSION.result_table;
        SET i = 1;
        WHILE i BETWEEN 1 AND limit DO
            BEGIN
                INSERT INTO t1 VALUES (i + 10);
                INSERT INTO SESSION.result_table VALUES (i);
            END;
            SET i = i + 1;
        END WHILE ;
        IF backwards = 0 THEN
             GOTO end_func;
        END IF;
        SET lower = INFX.MIN(limit, 1);
        SET upper = INFX.MAX(limit, 1);
        SET incr = INFX.compute_loop_incr(limit, 1, -1);
        SET i = limit;
        WHILE i BETWEEN lower AND upper DO
            BEGIN
                INSERT INTO t1 VALUES (i + 10);
                INSERT INTO SESSION.result_table VALUES (i);
            END;
             SET i = i + incr;
        END WHILE ;
        end_func:
        OPEN temp_cursor;
    END;
END!

CREATE TABLE t(x INTEGER)!

CREATE PROCEDURE testSeries ( )
LANGUAGE SQL
BEGIN
    DECLARE y INTEGER;
    DECLARE loc RESULT_SET_LOCATOR VARYING;
    DECLARE SQLCODE INTEGER DEFAULT 0;
    DECLARE CURS_FOUND INTEGER;
    DECLARE loc1 RESULT_SET_LOCATOR VARYING;
    DECLARE CURS_FOUND1 INTEGER;
    DECLARE RETURN_VAL INTEGER;
    DECLARE limit INTEGER;
    DECLARE backwards INTEGER;
    DECLARE limit1 INTEGER;
    DECLARE backwards1 INTEGER;
    SET limit = 10;
    SET backwards = 1;
    CALL series(limit,backwards);
    ASSOCIATE LOCATORS (loc) WITH PROCEDURE series;
    ALLOCATE result_cursor CURSOR FOR RESULT SET loc;
    BEGIN
       DECLARE CONTINUE HANDLER FOR
         NOT FOUND SET CURS_FOUND = SQLCODE;
       SET CURS_FOUND = 0;
       FETCH FROM result_cursor INTO y;
    END;
    WHILE CURS_FOUND = 0 DO
        BEGIN
            INSERT INTO t1 VALUES (y + 1000);
            INSERT INTO t VALUES (y * 10);
        END;
        BEGIN
           DECLARE CONTINUE HANDLER FOR
             NOT FOUND SET CURS_FOUND = SQLCODE;
           SET CURS_FOUND = 0;
           FETCH FROM result_cursor INTO y;
        END;
    END WHILE ;
    CLOSE result_cursor;
    SET limit1 = 10;
    SET backwards1 = 1;
    CALL series(limit1,backwards1);
    ASSOCIATE LOCATORS (loc1) WITH PROCEDURE series;
    ALLOCATE result_cursor1 CURSOR FOR RESULT SET loc1;
    BEGIN
       DECLARE CONTINUE HANDLER FOR
         NOT FOUND SET CURS_FOUND1 = SQLCODE;
       SET CURS_FOUND1 = 0;
       FETCH FROM result_cursor1 INTO RETURN_VAL;
    END;
    WHILE CURS_FOUND1 = 0 DO
        BEGIN

        END;

        BEGIN
           DECLARE CONTINUE HANDLER FOR
             NOT FOUND SET CURS_FOUND1 = SQLCODE;
           SET CURS_FOUND1 = 0;
           FETCH FROM result_cursor1 INTO RETURN_VAL;
        END;
    END WHILE ;
    CLOSE result_cursor1;
END!

CALL testSeries()!

SELECT * FROM t!

SELECT * FROM t1!
```

In this translation, the cursor function produces its entire result set before the result set is processed by the while loop. This is not exactly consistent with the Informix Dynamic Server behavior. In Dynamic Server, the execution of the cursor function body and the loop body are interleaved. The loop body calls the function, which produces only one value, then halts until the loop has processed that value. Then the function is called again.

The function resumes from where it returned the previous value and continues until it produces the next value (or quits). Because the DB2 UDB execution is not interleaved, it is possible that the results of the translation to DB2 UDB are not the same as the Dynamic Server code. Specifically, if the loop or the body changes a global value that the other value refers to (such as a table or a global variable), the results will probably differ. If used in such a manner, you should manually translate the FOREACH loop.

One way to translate the FOREACH loop to preserve the original interleaved execution in DB2 UDB is to inline the cursor function code in place of the FOREACH loop. Replace RETURN *expr* WITH RESUME with the loop body (substituting the expression values for the corresponding variables). Replace RETURN with a GOTO end_func. This solution does not require a global temporary table or a cursor, but it can lead to much duplicate code if the cursor function is called more than once, and if RETURN *expr* WITH RESUME occurs more than once in the body of the function. Because of this, that MTK uses the stored procedure with a result set to translate the cursor function directly.

CREATE INDEX

- Only regular indexes are converted (not CLUSTER or GK indexes).
- No additional options are translated.

CREATE SEQUENCE

For Informix Dynamic Server Version 9.4, this statement and the expressions SEQUENCE.CURRVAL and SEQUENCE.NEXTVAL are translated

CREATE TABLE

- Adding a primary key or unique constraint might cause a NOT NULL constraint to be added to the indicated columns.

Translation of complex column defaults

DB2 UDB SQL does not allow complex column defaults (most expressions that are not literals). When translating these defaults, a special trigger is generated. When a row is inserted without specifying a value for the column, a null is inserted. The trigger then tests the value to be inserted and if it is null, it replaces the value using the default expression.

This translation works well if the column is not nullable. But if the column is nullable, the trigger cannot distinguish between a null value and an unspecified one. In either case, the default is inserted, which is not the expected behavior. Therefore, MTK issues a warning message during translation so that you can manually review the code. If you can ensure that no null values will be inserted for this column, then the translation is fine. If this is not the case, a solution is to use a special value as the default for the column (a value that is not likely to be inserted), and to test the inserted value against this special value instead of using null.

Because of the way complex defaults are translated, the converter removes a NOT NULL constraint that might exist on the column. If this is the case, a manual code review is necessary.

Translation of complex check constraints

Some complex check constraints are not allowed in DB2 UDB. These include constraints containing sub-queries, column functions, and special registers. When translating these kinds of constraints, a special trigger is generated. This trigger tests if the predicate of the constraint is true and raises an exception if not, preventing the insertion.

Note that the exception raised is an SQLSTATE 09000, which differs from the SQLSTATE 25513 raised when a real check constraint is not satisfied.

CREATE TEMP TABLE
This statement is not yet supported.

CREATE SYNONYM
No known limitations.

CREATE TRIGGER

The following features are not supported:

- SELECT clause
- DISABLED mode
- INSTEAD OF triggers (for Informix Dynamic Server Version 9.4)

ACTION clause

DB2 UDB allows only one action (BEFORE, FOR EACH ROW, or AFTER) per trigger. If a trigger in the source SQL contains a list of multiple actions, the converter translates the trigger into several DB2 UDB triggers, each of them holding one of the actions.

BEFORE or AFTER action types are translated to NO CASCADE BEFORE or AFTER triggers with the FOR EACH STATEMENT option.

FOR EACH ROW action types are translated to NO CASCADE BEFORE triggers with the FOR EACH ROW option.

BEFORE triggers

BEFORE triggers are usually translated to NO CASCADE BEFORE triggers in DB2 UDB. NO CASCADE triggers cannot fire other triggers. NO CASCADE triggers also do not allow DML statements (INSERT, UPDATE and DELETE) and the FOR EACH STATEMENT clause. If the source trigger uses one of these features, then the translator tries to translate the BEFORE trigger to an AFTER trigger, which does not have the same restrictions. However, such a translation is not possible if the body of the trigger refers to the table as the object of the trigger, or if a column of the NEW table is assigned.

BEFORE triggers without a FOR EACH ROW clause are not translated because there is no corresponding feature in DB2 UDB.

CREATE VIEW

Same limitations as the SELECT statement.

DATABASE
Not translated, but the converter interprets the given database as the current (default) database.

DROP TABLE, DROP INDEX, DROP INDEX
The DROP statement is converted for those objects that MTK supports:

- DROP FUNCTION or PROCEDURE (t1,...,t*n*). although the type list is currently ignored.
- DROP TABLE RESTRICT and CASCADE are both translated to DROP TABLE.

- DROP VIEW RESTRICT and CASCADE are both translated to DROP VIEW.

EXECUTE FUNCTION and EXECUTE PROCEDURE
These statements are translated except when the routine: (1) returns one or more values and (2) does not include an INTO clause into which to store the values.

SELECT

The default collating behavior is different in DB2 UDB than in Dynamic Server. DB2 UDB uses the collating sequence SYSTEM as its default. Once the SQL and data is deployed to DB2 UDB, run tests to determine whether to modify DB2 UDB to use the IDENTITY collating sequence to achieve similar results or whether to modify the statement logic.

All ANSI joins (including the Informix Dynamic Server Version 9.4 additions) are translated.

The specified features are not supported:
- The INTO TABLE clause
- FOR READ ONLY and FOR UPDATE clauses
- The ONLY clause
- ROWID
- Optimizer directives
- Collection sub-queries
- Collection-derived tables

INSERT

The specified features are not supported:
- The EXECUTE ROUTINE clause
- The INTO clause

UPDATE

The specified features are not supported:
- The ONLY clause
- Optimizer directives
- Collection-derived tables

DELETE

The specified features are not supported:
- The ONLY clause
- Optimizer directives
- Collection-derived tables

SPL statements

The following information highlights converter behavior and limitations in the conversion of SPL statements from Informix Dynamic Server to DB2.

DEFINE

Globals are not supported. Locals are supported for built-in data types only (REFERENCES, LIKE, and PROCEDURE are not supported).

LET

The converter can convert:

- Multiple expressions (including scalar sub-queries) on the right-hand side.
- Single selects or function calls that return multiple values on the right-hand side.

The converter does not convert SELECT statements or function calls that return multiple values with other elements on the right hand side .

SYSTEM
This statement is translated using the MTK library function INFX.system. Depending on your path settings, the full path to the executable file and the command shell that invokes it might need to be specified. For example, INFX.system('/bin/sh /export/home/migrate3/clean.sh').

DDL Statements

The CREATE statement for tables, views, and indexes is supported if the object referenced occurs earlier in the script files that are translated. Also, there can only be one CREATE statement for a given name in a given name scope. (If the same table is created in more than one procedure, MTK considers the second CREATE statement to be a duplicate).

The DROP statement is supported inside procedure and function bodies when used to drop tables, views, and indexes only.

Inside procedure and function bodies, the following statements are not supported:

- ALTER TABLE
- CREATE DISTINCT TYPE
- CREATE SYNONYM
- CREATE TRIGGER
- RENAME COLUMN
- RENAME TABLE
- TRACE

Exceptions

In Informix Dynamic Server, each exception is designated by its error code and has an error message associated to it. In DB2 UDB, each condition is designated by either its SQLCODE or its SQLSTATE and has an error message associated to it.

When the converter encounters an error number, it tries to convert it to the SQLSTATE that has a similar meaning. Given the large number of differences among the Dynamic Server exceptions and the DB2 UDB conditions, the converter only maps simple exceptions. Moreover, the exceptions that can be successfully converted might have some subtle differences that cause different behavior in the converted script. Therefore, after conversion, review the exceptions that were successfully converted and manually convert any that remain.

The translator is not able to translate special arithmetic items with error codes, because error codes are simple integers, making it very difficult to infer from the code if an integer should be considered as a regular number or as an error code. This is also true with the error messages.

ON EXCEPTION: MTK converts Informix Dynamic Server ON EXCEPTION statements into DB2 UDB condition handler declarations.

- *WITH RESUME clause* - If it is used, the handler is translated to a CONTINUE handler in DB2 UDB, otherwise it is translated to an EXIT handler.
- *IN clause* - Can be used to specify the list of exception error numbers that will be caught by the handler. The converter uses the exceptions mapping to convert the IN clause to a FOR clause containing the list of conditions (SQLSTATEs or condition classes) to be caught by the handler.
- The SET clause contains information about the exception that was raised:
  - *SQL_error_var* - the provided variable is set using the error number of the exception that was raised. This feature is converted by assigning the SQLCODE to this variable at the beginning of the handler.
  - *ISAM_error_var* - this feature is not converted because it has no equivalent in DB2 UDB.
  - *error_data_var* - the provided variable is set using the text of the exception that was raised. This feature is converted using the GET DIAGNOSTIC statement at the beginning of the handler, to assign the condition's text to the variable.

RAISE EXCEPTION: MTK converts Informix Dynamic Server RAISE EXCEPTION statements into DB2 UDB SIGNAL statements.

- *SQL_error* - Indicates the error code of the exception to be raised. If this expression is a literal, the convertor tries to convert it using the mapping of the exception. Manual conversion might be necessary.
- *ISAM_error* - Not translated because it has no equivalent in DB2 UDB. But, if the specified value is anything other than '0', MTK issues an error message.
- *error_text* - Indicates the error message to be associated to the exception being raised. MTK translates this using the SET MESSAGE_TEXT clause of the DB2 UDB SIGNAL statement.

Control flow

The following section highlights SPL statements that control the flow, or the order in which statements are executed.

FOR: The converter can convert a single expression range (a to b {step c}) and a list of expressions only. Lists containing one or more expression ranges are not supported.

FOREACH:

MTK translates Informix Dynamic Server FOREACH statements into DB2 UDB FOR statements.

- *Loop name* - DB2 UDB requires that you choose a loop name. It has no equivalent in Informix Dynamic Server. Therefore, the convertor generates a temporary loop name.
- *WITH HOLD clause* - DB2 UDB requires a cursor name to be used with the WITH HOLD option. If no cursor name is specified, the converter generates a template cursor name.
- *SELECT..INTO statement* - Informix Dynamic Server and DB2 UDB access row data differently. Informix Dynamic Server uses a SELECT INTO statement to store the row data in the provided variables. DB2 UDB has a simple SELECT statement and allows you to refer to the columns inside the FOR loop. Therefore, the converter converts the SELECT..INTO to a simple SELECT statement and assigns the row data to the provided variables at the beginning of the loop body, using the references to the column of the SELECT clause. If some columns such

as expression select items with no alias are not named, the converter automatically generates an alias name for the column.

- *EXECUTE clause* - not supported.

Transaction statements

The following statements are not yet implemented:

- BEGIN WORK
- COMMIT
- LOCK and UNLOCK TABLE
- ROLLBACK
- SET LOCK MODE
- SET ISOLATION

Compatibility library (INFX functions) for translations to DB2

Some SQL and Java functions are provided to simulate Informix Dynamic Server functions that do not have DB2 UDB equivalents. The schema called INFX contains these functions.

The INFX SQL function definitions are available in the mtkinfx.udf file that is located in the product directory. Many INFX functions, however, are implemented in Java, where SQL was not suitable. INFX Java function definitions are available in the mtkinfxdrop.udf file that is located in the product directory.

The mtkinfxdrop.udf file is a script file that is used to drop the functions created in the INFX schema with mtkinfx.udf. This file is deployed to DB2 UDB during the deploy step of the migration. You can modify this file later if desired, to improve efficiency or change the specific behavior of a function.

The following is the list of INFX functions. The descriptions address the differences between Informix Dynamic Server and DB2 UDB built-in functions, highlight any restrictions that apply to conversion, and show common usage examples where applicable.

INFX.initcap

INFX.length

INFX.lpad

INFX.mdy

INFX.replace

INFX.rpad

INFX.substr

INFX.substring

INFX.to_date

INFX.to_char

INFX.ltrim

INFX.rtrim

INFX.trim_both

INFX.trunc

INFX.weekday

Related information

"Built-in functions" on page 137

MySQL converter

The following sections detail the support that the converter provides for MySQL translations.

Converter behavior and limitations

The topics in this section describe how each particular feature of the MySQL source language is converted.

These topics are not intended to be a comprehensive overview of the features of the source and target SQL languages. The information here lists translation limitations and provides some tips when you need to provide your own modifications to the SQL.

Data types

MySQL built-in data types (including ANSI types) are converted to similar data types in the target database, according to the mapping described in this section.

Optional global type mapping values are listed in the table below and can be changed. For more information, see “Mapping data types” on page 35.

MySQL data type	DB2 data type	Informix Dynamic Server data type
INTEGER INT	INTEGER	INTEGER
INTEGER UNSIGNED	BIGINT Optional: INTEGER	INT8 Optional: INTEGER
SMALLINT	SMALLINT	SMALLINT
SMALLINT UNSIGNED	INTEGER Optional: SMALLINT	INTEGER Optional: SMALLINT
BIGINT	BIGINT	INT8
BIGINT UNSIGNED	DECIMAL Optional: BIGINT	DECIMAL Optional: INT8
TINYINT	SMALLINT	SMALLINT
TINYINT UNSIGNED	SMALLINT	SMALLINT
MEDIUMINT	INTEGER	INTEGER
MEDIUMINT UNSIGNED	INTEGER	INTEGER
BIT	SMALLINT	SMALLINT
BOOLEAN	SMALLINT	SMALLINT
REAL	DOUBLE	DOUBLE PRECISION
REAL UNSIGNED	DOUBLE	DOUBLE PRECISION
DOUBLE	DOUBLE	DOUBLE PRECISION

MySQL data type	DB2 data type	Informix Dynamic Server data type
DOUBLE UNSIGNED	DECIMAL Optional: DOUBLE	DECIMAL Optional: DOUBLE PRECISION
FLOAT	DOUBLE	DOUBLE PRECISION
FLOAT UNSIGNED	DOUBLE	DOUBLE PRECISION
NUMERIC DECIMAL DEC	DECIMAL(31,0)	DECIMAL(32,0)

NUMERIC(P)
NUMERIC(P,0)
DECIMAL(P)
DECIMAL(P,0)
DEC(P)
DEC(P,0)
	DECIMAL(min(P,31),0)	DECIMAL(min(P,32),0)

DECIMAL UNSIGNED	DECIMAL	DECIMAL
NUMERIC UNSIGNED	DECIMAL	DECIMAL

NUMERIC(p,s)
DECIMAL(p,s)
DEC(p,s)
where:
s > 0 && p >= s
s > 0 && p < s
s < 0
	DECIMAL(min(p,32), min(s,32))
	DECIMAL(min(s,32), min(s,32))
	DECIMAL(min(p,32),0)
		DECIMAL(min(p,32), min(s,32))
		DECIMAL(min(s,32), min(s,32))
		DECIMAL(min(p,32),0)

CHAR(n)
n < 256
else
	CHAR(n)
	
	Optional for iSeries: VARCHAR(n)
		CHAR(n)

VARCHAR(n)
n < 256
else
	VARCHAR(n)
	
	Optional: CLOB
		VARCHAR(n)
		Optional:
		LVARCHAR(n)
		CLOB

BINARY	CHAR (I) FOR BIT DATA	BYTE
VARBINARY	VARCHAR (I) FOR BIT DATA	BYTE
TINYBLOB	BLOB(255)	BYTE Optional: BLOB
TINYTEXT	CLOB(255)	TEXT
BLOB	BLOB(65535)	BLOB Optional: BYTE
TEXT	CLOB(65535)	TEXT
MEDIUMBLOB	BLOB(16777215)	BYTE Optional: BLOB
MEDIUMTEXT	CLOB(16777215)	TEXT
LONGBLOB	BLOB(2G)	BYTE Optional: BLOB

MySQL data type	DB2 data type	Informix Dynamic Server data type
LONGTEXT	CLOB(2G)	TEXT
ENUM	Not supported	Not supported
SET	Not supported	Not supported
DATE	DATE	DATE
TIME	TIME	DATETIME HOUR TO FRACTION
TIMESTAMP	TIMESTAMP	DATETIME YEAR TO FRACTION
DATETIME	TIMESTAMP Optional: TIME	DATETIME YEAR TO FRACTION DATE
YEAR	CHAR(4)	CHAR(4)

Statements

MTK provides limited support for translating MySQL statements CREATE TABLE, CREATE INDEX, and INSERT.

CREATE TABLE support

The MySQL CREATE TABLE statement is supported in conversions to DB2 and Informix Dynamic Server with restrictions.

The following MySQL syntax is not supported in conversions to Informix Dynamic Server:

- ON DELETE SET NULL, ON DELETE NO ACTION and ON DELETE RESTRICT actions for referential integrity constraints. A warning is issued to if any of these options are used.
- ON UPDATE clause in referential integrity constraints. The referential option clause is ignored and a warning is issued.

The following MySQL syntax is not supported in conversions to DB2 UDB:

- ON UPDATE CASCADE and ON UPDATE SET NULL clause in referential constraints. The referential option clause is ignored and a warning is issued.

The following MySQL syntax is not supported in conversions to DB2 UDB and Informix Dynamic Server:

- MATCH FULL, MATCH PARTIAL, or MATCH SIMPLE options in referential constraints. The index clause is ignored and a warning is issued.
- Specification of the type of index with the USING {BTREE | HASH} index clause.
- FULL TEXT and SPATIAL indexes.
- SELECT STATEMENT in CREATE TABLE.
- IF NOT EXISTS syntax.
- ZEROFILL attribute. For example, for a column declared as `INT(5) ZEROFILL`, a value of 4 is retrieved as 00004.
- The following table options:
  - `ENGINE|TYPE} [=]` *`engine_name`*
  - `AUTO_INCREMENT [=]` *`value`*
  - `AVG_ROW_LENGTH [=]` *`value`*

- `[DEFAULT] CHARACTER SET` *`charset_name`*
- `COLLATE` *`collation_name`*
- `COMMENT [=] '`*`string`*`'`
- `CONNECTION [=] '`*`connect_string`*`'`
- `DATA DIRECTORY [=] '`*`absolute path to directory`*`'`
- `INDEX DIRECTORY [=] '`*`absolute path to directory`*`'`
- `MAX_ROWS [=]` *`value`*
- `MIN_ROWS [=]` *`value`*
- `PASSWORD [=] '`*`string`*`'`
- `UNION [=] (`*`tbl_name`*`[,`*`tbl_name`*`]...)`

CREATE INDEX support

The MySQL CREATE INDEX statement is supported in conversions to DB2 UDB and Informix Dynamic Server with restrictions.

The following MySQL syntax is not supported in conversions to DB2 UDB and Informix Dynamic Server:

- Fulltext index clause.
- Spatial index clause.
- USING BTREE, USING HASH clause (index_type) while creating an index.

Indexes for CHAR, VARCHAR, BINARY, and VARBINARY data types that use the leading part of the column length (col_name (*length*)) are not supported. In these cases, an index is created on the entire column instead of only the leading portion. For example, the MySQL statement:

```
CREATE INDEX part_of_name ON customer (name (10));
   ----- is translated
to -----
CREATE INDEX part_of_name ON customer (name);
```

INSERT support

The MySQL INSERT statement is supported in conversions to DB2 UDB and Informix Dynamic Server with restrictions.

The following clauses of the MySQL INSERT statement are not supported in conversions to DB2 UDB and Informix Dynamic Server:

- ON DUPLICATE KEY UPDATE
- IGNORE
- LOW_PRIORITY, DELAYED, and HIGH_PRIORITY

Insert statements that contain multiple value lists are not supported in conversions to Informix Dynamic Server, however they are supported in conversions to DB2.

DB2 UDB supported features

The following tables list the MySQL features that the converter supports in conversions to DB2 UDB.

Feature	Status	Comments
Data types:		

Feature	Status	Comments
ENUM	Not supported	
SET	Not supported	
Spatial data types	Not supported	
ZEROFILL	Not supported	
COLLATE option for CHAR/VARCHAR types	Not supported	
Others	Supported	
Expressions:		
Built-in Functions	Not supported	
Auto_increment	Supported	Mapped to DB2 identity column
Others	Supported	
Conditions	Supported	
Column Definitions:		
Default expressions	Partial support	Only strings with single quotes are supported
default EXPRESSION For example: default 10 default 'MTK'	Supported	
default current_timestamp, default now, default localtime default not null, default null	Supported	
Constraints:		
Unique constraints	Partial support	Index_name, index_type, and index on partial column name are ignored.
Referential constraints MATCH {FULL|PARTIAL| SIMPLE}	Partial support	Match clause in reference definition are ignored.
ON UPDATE {CASCADE|SET NULL}	Not supported	
FULLTEXT/SPATIAL index constraints	Not supported	
CHECK constraints	Supported	Only parsed and not translated to DB2. MySQL does not enforce check constraints. To avoid data loss check constraints are not translated.
Others	Supported	

Feature	Status	Comments
Statements (except DML):		
ALTER TABLE	Partial support	
COMMENT	Not supported	
CREATE TABLE:	Partial support	
Temporary tables	Supported	
Table options	Partial support	For more information, see "CREATE TABLE support" on page 153.
Select clause	Not supported	
LIKE clause	Supported	
CREATE INDEX	Supported	
BTREE index	Not supported	
HASH index	Not supported	
Index on partial column names	Not supported	
CREATE VIEW	Not supported	
DML Statements:		
SELECT	Not supported	
INSERT	Partial support	
Select clause	Not supported	
DELETE	Not supported	
UPDATE	Not supported	

Informix Dynamic Server supported features

The following tables list the MySQL features that the converter supports in conversions to Informix Dynamic Server.

Feature	Status	Comments
Data types:		
ENUM	Not supported	
SET	Not supported	

Feature	Status	Comments
Spatial data types	Not supported	
ZEROFILL	Not supported	
COLLATE option for CHAR/VARCHAR types	Not supported	
Others	Supported	
Expressions:		
Built-in Functions	Not supported	
Auto_increment	Supported	Mapped to the Informix Dynamic Server serial data type
Others	Supported	
Conditions	Supported	
Column Definitions:		
Default expressions	Partial support	Only strings with single quotes are supported
default EXPRESSION For example: default 10 default 'MTK'	Supported	
default current_timestamp, default now, default localtime default not null, default null	Supported	
Constraints:		
Unique Constraints	Partial support	Index_name, index_type, and index on partial column name are ignored.
Referential Constraints MATCH {FULL|PARTIAL| SIMPLE}	Partial support	Match clause in reference definition are ignored.
ON UPDATE Clause	Not supported	
ON DELETE {SET NULL|RESTRICT|NO ACTION	Not supported	
FULLTEXT/SPATIAL index constraints	Not supported	
CHECK constraints	Supported	Only parsed and not translated to Informix Dynamic Server. MySQL does not enforce check constraints. To avoid data loss check constraints are not translated.

Feature	Status	Comments
Others	Supported	
Statements (except DML):		
ALTER TABLE	Partial support	
COMMENT	Not supported	
CREATE TABLE:	Partial support	
Temporary tables	Supported	
Table options	Not supported	For more information, see “CREATE TABLE support” on page 153.
Select clause	Not supported	
LIKE clause	Not supported	
CREATE INDEX	Supported	
BTREE index	Not supported	
HASH index	Not supported	
Index on partial column names	Not supported	
CREATE VIEW	Not supported	
DML Statements:		
SELECT	Not supported	
INSERT	Partial support	
Select clause	Not supported	
Multiple Inserts	Not supported	
DELETE	Not supported	
UPDATE	Not supported	

Oracle converter

The following sections detail the support that the converter provides for Oracle translations.

Converter behavior and limitations

The topics in this section describe how each particular feature of the Oracle SQL language is converted.

These reference topics are not intended to be a comprehensive overview of the features of the source and target SQL languages. The information here lists any translation limitations and provides some tips when you need to provide your own modifications to the SQL.

Built-in-functions

There are three possible scenarios when converting Oracle built-in functions to the target IBM database (either DB2 UDB or Informix Dynamic Server).

- An Oracle function has an equivalent target-database function. In this case function calls are mapped directly to the target database.
- An Oracle function does not have an equivalent target-database function, but a similar target-database function that has some slight differences in the output (such as NULL, empty strings, or maximums) is available.
- An Oracle function has no equivalent function, so an SQL user-defined function or a Java function must be used. For conversions to UDB DB2, these functions have names of the form `ORA`.function. See "Compatibility library (ORA functions)" on page 237 for function descriptions.

The following table describes the functions MTK used in converting Oracle built-in functions to DB2 UDB or IBM Informix Dynamic Server. See the notes below the table for details about a few of the conversions.

MTK does not convert Oracle analytic functions into IBM Informix Dynamic Server functions.

Oracle function	Type	DB2 UDB implementation	IBM Informix implementation
ABS	Number	ABS	ABS
ACOS	Number	ACOS	ACOS
ADD_MONTHS	Date	ORA.ADD_MONTHS	ORA.ADD_MONTHS
ASCII	String	ASCII	ORA.ASCII
ASIN	Number	ASIN	ASIN
ATAN	Number	ATAN	ATAN
ATAN2	Number	ORA.ATAN2	ATAN2
AVG	Aggregate	AVG	AVG
BFILENAME	Misc	(not supported)	(not supported)
BITAND	Number	(not supported)	(not supported)
CAST	Conv	CAST (partial) ; 1	CAST
CEIL	Number	CEIL	ORA.CEIL

Oracle function	Type	DB2 UDB implementation	IBM Informix implementation
CHARTOROWID	Conv	(not supported)	(not supported)
CHR	String	ORA.CHR	(not supported)
COALESCE	Misc	(not supported)	NVL
CONCAT	String	ORA.CONCAT	(not supported)
CONVERT	Conv	(not supported)	(not supported)
CORR	Aggregate	(not supported)	(not supported)
COS	Number	COS	(not supported)
COSH	Number	ORA.COSH	ORA.COSH
COUNT	Aggregate	COUNT	COUNT
COVAR_POP	Aggregate	(not supported)	(not supported)
COVAR_SAMP	Aggregate	(not supported)	(not supported)
CUME_DIST	Aggregate	(not supported)	(not supported)
DECODE^2	Language Construct	CASE Expression	DECODE
DENSE_RANK	Number	(not supported)	(not supported)
DEREF	Object Ref	(not supported)	(not supported)
DUMP	Misc	(not supported)	(not supported)
EMPTY_BLOB	Misc	(not supported)	(not supported)
EMPTY_CLOB	Misc	(not supported)	(not supported)
EXP	Number	EXP	EXP
FIRST_VALUE		(not supported)	(not supported)
FLOOR	Number	FLOOR	ORA. FLOOR
GREATEST	Number	(not supported)	(not supported)
GREATEST_LB	Number	(not supported)	(not supported)
GROUPING	Aggregate	(not supported)	(not supported)
HEXTORAW	Conv	ORA.HEXTORAW	(not supported)
INITCAP	String	ORA.INITCAP	INITCAP
INSTR	String	ORA.INSTR	(not supported)
INSTRB	String	(not supported)	(not supported)
LAG	Analytical	(not supported)	(not supported)
LAST_DAY	Date	ORA.LAST_DAY	ORA. LAST_DAY
LAST_VALUE	Analytical	(not supported)	(not supported)
LEAD	Analytical	(not supported)	(not supported)
LEAST	Number	(not supported)	(not supported)
LEAST_LB	Number	(not supported)	(not supported)
LENGTH	String	LENGTH	LENGTH
LENGTHB	String	(not supported)	(not supported)
LN(x)	Number	LN	LN
LOG(x,y)	Number	ORA.LOG(x,y)	ORA.LOG(x,y)

Oracle function	Type	DB2 UDB implementation	IBM Informix implementation
LOWER	String	LOWER	LOWER
LPAD	String	ORA.LPAD	(not supported)
LTRIM	String	ORA.LTRIM	TRIM(LEADING from)[3]
MAKE_REF	Object Ref	(not supported)	(not supported)
MAX	Analytical	MAX	MAX
MIN	Analytical	MIN	MIN
MOD	Number	ORA.MOD	MOD
MONTHS_BETWEEN	Date	ORA.MONTHS_ BETWEEN	ORA. MONTHS_BETWEEN
NEW_TIME	Date	(not supported)	(not supported)
NEXT_DAY	Date	ORA.NEXT_DAY	ORA.NEXT_DAY
NLS_CHARSET_ DECL_LEN	Misc	(not supported)	(not supported)
NLS_CHARSET_ID	Misc	(not supported)	(not supported)
NLS_CHARSET_NAME	Misc	(not supported)	(not supported)
NLS_INITCAP	Misc	(not supported)	(not supported)
NLS_LOWER	String	(not supported)	(not supported)
NLS_UPPER	String	(not supported)	(not supported)
NLSSORT	String	(not supported)	(not supported)
NTILE	Misc	(not supported)	(not supported)
NUMTODSINTERVAL	Conv	(not supported)	(not supported)
NUMTOYMINERTVAL	Conv	(not supported)	(not supported)
NVL	Misc	COALESCE	NVL
NVL2	Misc	CASE Expression	(not supported)
PERCENT_RANK	Analytical	(not supported)	(not supported)
POWER	Number	POWER	POW
RANK	Analytical	(not supported)	(not supported)
RATIO_TO_REPORT	Conv	(not supported)	(not supported)
RAWTOHEX	Conv	ORA.RAWTOHEX	(not supported)
REF	Misc	(not supported)	(not supported)
REFTOHEX	Conv	(not supported)	(not supported)
REGR_(linear regression functions)	Aggregate	(not supported)	(not supported)
REPLACE	String	ORA.REPLACE	(not supported)
ROUND (number)	Number		ROUND (number)
ROUND (number function)	Number	ORA.ROUND	(not supported)
ROUND (date function)	Date	ORA.ROUND	ORA.ROUND (date)
ROW_NUMBER	Conv	(not supported)	(not supported)
ROWIDTOCHAR	Misc	(not supported)	(not supported)
RPAD	String	ORA.RPAD	(not supported)

Oracle function	Type	DB2 UDB implementation	IBM Informix implementation
RTRIM	String	ORA.RTRIM	TRIM(Trailing from)
SIGN	Number	SIGN	ORA.SIGN
SIN	Number	SIN	(not supported)
SINH	Number	ORA.SINH	SINH
SOUNDEX	String	(not supported)	(not supported)
SQRT	Number	SQRT	SQRT
STDDEV	Aggregate	(not supported)	(not supported)
STDDEV_POP	Aggregate	(not supported)	(not supported)
STDDEV_SAMP	Aggregate	(not supported)	(not supported)
SUBSTR	String	ORA.SUBSTR	SUBSTR
SUBSTRB	String	(not supported)	(not supported)
SUM	Aggregate	SUM	SUM
SYS_CONTEXT	Misc	(not supported)	(not supported)
SYS_GUID	Misc	(not supported)	(not supported)
SYSDATE	Date	CURRENT TIMESTAMP	CURRENT DATETIME YEAR TO FRACTION(5)
TAN	Number	TAN	(not supported)
TANH	Number	ORA.TANH	ORA.COSH
TO_BINARY_INTEGER	Conv	(not supported)	(not supported)
TO_CHAR(date)	Conv	ORA.TO_CHAR	TO_CHAR(date)
TO_CHAR(num)	Conv	ORA.TO_CHAR and ORA.TO_CHAR_ DECIMAL	CAST(number AS CHAR(10)
TO_DATE	Conv	ORA.TO_DATE	TO_DATE[4]
TO_LOB	Conv	(not supported)	(not supported)
TO_MULTI_BYTE	Conv	(not supported)	(not supported)
TO_NUMBER	Conv	ORA.TO_NUMBER and ORA.TO_NUMBER _DECIMAL	CAST('string' as INTEGER)
TO_SINGLE_BYTE	Conv	(not supported)	(not supported)
TO_TIMESTAMP[5]	Date Number	(not supported)	ORA.TO_TIMESTAMP
TRANSLATE	String	ORA.TRANSLATE	(not supported)
TRANSLATE...USING	Conv	(not supported)	(not supported)
TRIM	String	• ORA.TRIM (DB2 V8.2, DB2 Iseries, DB2 Zseries) • TRIM (DB2 V9)	TRIM
TRUNC (number)	Number		TRUNC (number)
TRUNC (number function)	Number	ORA.TRUNC	(not supported)

Oracle function	Type	DB2 UDB implementation	IBM Informix implementation
TRUNC (date function)	Number	ORA.TRUNC	ORA.TRUNC (date)
UID	Misc	(not supported)	(not supported)
UPPER	String	UPPER	UPPER
USER	Misc	USER	(not supported)
USERENV	Misc	(not supported)	(not supported)
VALUE	Object Ref	(not supported)	(not supported)
VAR_POP	Analytical	(not supported)	(not supported)
VAR_SAMP	Analytical	(not supported)	(not supported)
VARIANCE	Analytical	(not supported)	(not supported)
VSIZE	Misc	(not supported)	(not supported)

Table notes

1. For more information on the conversion of Oracle CASTs, see“Expressions” on page 185.
2. Although Oracle’s DECODE function syntactically resembles a function, it can be considered a language construct. Therefore, it is translated to DB2 UDB using a CASE expression. For example:

```
select decode('x','x','This is an x','y','This is a y',
              'This is something else') from dual;
   ----- is translated
to ----- ------
SELECT CASE 'x'
         WHEN 'x' THEN 'This is an x'
         WHEN 'y' THEN 'This is a y'
         ELSE 'This is something else'
       END
FROM SYSIBM.SYSDUMMY1!
```

   However, for NULL comparison, the Oracle DECODE function differs from a simple DB2 UDB case expression. In Oracle, two NULL values are considered equal when evaluating a DECODE expression. In order to simulate this behavior for DB2 UDB in cases where parameter values can be set to NULL, the converter uses a DB2 UDB searched case instead of a simple case. For example, considering a table t defined by:

```
create table t(x int, z int);

select DECODE(x,z,'equal','not equal') from t;

   ----- is translated
to -----

SELECT CASE
       WHEN X = Z
            OR (X IS NULL
                AND Z IS NULL) THEN 'equal'
       ELSE 'not equal'
       END
FROM T!
```

3. In situations where the trim string is greater than length 1, MTK converts the function using a user-defined function instead of TRIM, as shown in this example:

```
SELECT LTRIM('xyxXxyLAST WORD','xy') "LTRIM example"

   ----- is translated
to -----

SELECT ORA.LTRIM('xy', 'xyxXxyLAST WORD') "LTRIM example"
```

4. In translations from Oracle to Informix Dynamic Server, if TO_DATE has three arguments and the third argument is an NLS format, MTK returns an invalid number of arguments.
5. In translations of TO_TIMESTAMP statements, MTK uses DD-MON-RR an internal default format. For example, MTK uses the DD-MON-RR format when translating this statement:

   ```
   select to_timestamp(col1) from dual;
   ```

   If you specify a date format, the MTK translator converts the format into one of the Informix Dynamic Server date formats specified in the table in the ″TO_DATE Formats″ section below.

Some Oracle to Informix Dynamic Server conversion examples

The Oracle SQL statement

```
SELECT TO_CHAR(ADD_MONTHS(hire_date,1),
     'DD-MON-YYYY') "Next month"
     FROM employees ;
   ----- is translated
to -----
SELECT TO_CHAR(ORA.ADD_MONTHS(hire_date, 1),
     'DD-MON-YYYY') AS NEXT_MONTH FROM employees;
```

The Oracle SQL statement

```
SELECT ASCII('Q') FROM EMPLOYEES;
   ----- is translated
to -----
SELECT ORA.ASCII('Q') from EMPLOYEES;
```

The Oracle SQL statement

```
SELECT LTRIM('xyxXxyLAST WORD','xy') "LTRIM example"
   FROM DUAL;
   ----- is translated
to -----
SELECT ORA.LTRIM('xy', 'xyxXxyLAST WORD')
   "LTRIM example"
```

The Oracle SQL statement

```
SELECT MONTHS_BETWEEN
   (TO_DATE('02-02-1995','MM-DD-YYYY'),
    TO_DATE('01-01-1995','MM-DD-YYYY') ) "Months"
    FROM DUAL;
   ----- is translated
to -----
SELECT ORA.MONTHS_BETWEEN
   (DATETIME(1995-02-02-00.00.00.00000) YEAR TO FRACTION(5) ,
    DATETIME(1995-01-01-00.00.00.00000) YEAR TO FRACTION(5) )
    AS "Months"FROM informix.dual;
```

MTK translates the TO_DATE with a literal as an argument DATETIME() year to fraction(5). For example,

```
TO_DATE('02-02-1995', 'MM-DD-YYYY')
   ----- is translated
to -----
DATETIME(1995-02-02-00.00.00.00000) YEAR TO FRACTION(5)
```

However, if TO_DATE receives a column name as an argument, the format part is modified according to Informix Dynamic Server format. For example, `MM` is mapped to `%m DD - %d` and `YYYY` is mapped to `%iYso`. If the column name is `col1` in a table, the converted output is `to_date(col1, "%m-%d-%iY')`.

The Oracle SQL statement

```
SELECT TO_NUMBER('10') from dual;
   ----- is translated
to -----
SELECT CAST('10' AS INTEGER) from DUAL;
```

The Oracle SQL statement

```
SELECT SYSDATE
   ----- is translated
to -----
SELECT CURRRENT YEAR TO FRACTION(5)
```

However, if CURRENT is used as a default clause in a CREATE TABLE statement for a DATETIME YEAR TO FRACTION(5) column, Dynamic Server returns an error

TO_Date formats

Oracle format	Informix Dynamic Server Format
YYYY	%Y
YY RR	%iy
MM	%m
MON	%b
MONTH	%B
D	%w
DD	%d
DAY	%A
DY	%a
EE	%EC
E	%Eg
HH24	%H
HH12 HH	%I
AM PM A.M. P.M.	%p
MI	%M
SS	%S
AD	AD

Oracle format	Informix Dynamic Server Format
BC	BC
A.D.	A.D.
B.C.	B.C.
CC SCC	CC

For example, MTK translates the statement

```
SELECT to_date(col1, "DD-MON-RR") from dual
   ----- is translated
to -----
SELECT to_date(col1, "%d-%b-%iy") from informix.dual;
```

Related concepts

"Data format recommendations" on page 38
When converting a database with DBCS data, you must give careful consideration to functions that handle character data. Where Sybase or SQL Server count characters, DB2 UDB counts bytes in functions such as left() or length().

Related information

"Compatibility library (ORA functions)" on page 237
Some SQL and Java functions are provided to simulate Oracle functions that do not have DB2 UDB or Informix Dynamic Server equivalents. A schema called ORA has been created to contain these functions.

Conditions and predicates

The following information describes various aspects of the conversion of Oracle conditions to DB2 UDB predicates.

An Oracle condition is equivalent to an expression that has a boolean data type. DB2 UDB has no boolean data type and predicates cannot be used in expression contexts. The converter must perform special actions, as described in this section, to produce a correct translation for Oracle conditions.

Boolean expressions

The PL/SQL Boolean data type is not part of Oracle SQL and is not supported by DB2 UDB or Informix Dynamic Server.

Boolean expressions and boolean variables are translated to integer expressions and integer variables with the values 1 = true and 0 = false. When a boolean expression (translated to an integer expression) is referred to and DB2 UDB or Informix Dynamic Server expects a predicate, the integer expression is compared to 1.

The following example shows how Oracle PL/SQL boolean integer expressions are handled in conversions to DB2 UDB and Informix Dynamic Server:

Oracle PL/SQL

```
declare
        b boolean := true;
begin
        if b then null; end if;
end
```

DB2 UDB conversion

```
BEGIN ATOMIC
      DECLARE B INTEGER DEFAULT 1;
      IF B = 1 THEN END IF;
END!
```

Informix Dynamic Server conversion

```
DEFINE b INTEGER;
LET b = 1;
IF b = 1 THEN
   BEGIN
   END
END IF;
```

References to conditional expressions where DB2 UDB or Informix Dynamic Server expect an expression result in the Oracle condition being converted to a DB2 UDB or Informix Dynamic Server CASE expression that tests the condition and returns 1 if it is true and false otherwise.

The following example shows how Oracle PL/SQL boolean expressions are handled in conversions to DB2 UDB and Informix Dynamic Server.

Oracle PL/SQL boolean expression

```
declare
        b boolean := true;
        i int := 3;
begin
        b := (i=5);
end;
```

DB2 UDB conversion

```
BEGIN ATOMIC
       DECLARE B INTEGER DEFAULT 1;
       DECLARE I INTEGER DEFAULT 3;
       SET B = CASE
               WHEN (I = 5) THEN 1
               ELSE 0
               END;
END!
```

Informix Dynamic Server conversion

```
DEFINE b INTEGER;
DEFINE i INTEGER;
LET b = 1;
LET i = 3;
LET b = CASE
            WHEN (i = 5) THEN 1
            ELSE 0
            END;
```

An assignment between two boolean variables will be converted to a simple assignment between the corresponding DB2 UDB or Informix Dynamic Server integer variables.

Blank and non-blank padding

Oracle blank and non-blank padded semantics are mimicked in translations to IBM Informix Dynamic Server. String comparison is handled differently in DB2 UDB and Oracle, in particular when the string is of type VARCHAR.

In DB2 UDB, blank-padding semantics *always* apply to string comparisons. When comparing two strings, the shortest is padded up to the length of the longest string with blanks; therefore, the following is true:

```
'a' = 'a '
```

In Oracle, blank padding will not occur when at least one of the strings being compared is of the type VARCHAR. In this case trailing blanks become significant.

By default, the converter will not enforce non-blank-padding in the generated DB2 UDB code.

Migration strategies

To imitate the Oracle behavior, you can explicitly select the **enforce non-blank-padded comparisons** option to invoke the ORA.NO_PAD utility function provided by MTK, which makes the DB2 UDB blank-padding ineffective.

The use of this option can cause a significant loss in performance during the execution of a query and should be avoided if you do not require trailing blanks to be significant in string comparisons. See ORA.NO_PAD in"Compatibility library (ORA functions)" on page 237 for more information on enforcing non-blank-padding with the ORA.NO_PAD function.

The following examples identify cases in which certain membership conditions can result in a complex conversion when the **enforce non-blank-padded comparisons** option is selected. Certain conditions may need to be translated manually.

Conditions of the form:

```
a IN/NOT IN (b, c, ...)

   ----- is translated
to -----

ORA.NO_PAD(a) IN (ORA.NO_PAD(b), ORA.NO_PAD(c), ...)
```

where b, c, ... are all of the type VARCHAR, or if a is of the type VARCHAR.

However, if a is of the type CHAR and b, c, ... have heterogeneous string types (both CHAR and VARCHAR), the membership comparison needs to be expanded into basic comparisons to account for the fact that this membership condition really contains both a CHAR - CHAR and a CHAR - VARCHAR comparison. For instance, if b is VARCHAR and c is CHAR, the DB2 UDB generated code would be:

```
ORA.NO_PAD(a) = ORA.NO_PAD(b)  OR a = c OR ...
```

Notice that non-blank-padding is enforced for CHAR - VARCHAR (a-b) but not for CHAR - CHAR (a-c).

Conditions of the form:

```
(a, ...) IN/NOT IN ((b, ...), (c, ...), ...)
```

where a and b are of the type CHAR, c is VARCHAR, and xn is any other data type so that

```
(a, x1) IN ((b, x2), (c, x3) )
```

are not handled by the converter. Conditions of this form are not handled because a compared to b should be a non-blank-padded comparison whereas a compared to c should be a blank-padded comparison. Any attempt to convert conditions of this form will result in a warning message from the converter. This type of condition can be manually converted as follows:

```
(a = b AND x1 = x2) OR (ORA.NO_PAD(a) = ORA.NO_PAD(c) AND x1 = x3)
```

Related reference

"LIKE"
For translations from Oracle to IBM Informix Dynamic Server, MTK retains either retains the LIKE clause or changes the clause to mimic the behavior in the source function. Translations to DB2 UDB occur as described in this section.

"List of functions that simulate Oracle built-in functions" on page 238

LIKE

For translations from Oracle to IBM Informix Dynamic Server, MTK retains either retains the LIKE clause or changes the clause to mimic the behavior in the source function. Translations to DB2 UDB occur as described in this section.

For predicates of the form `expression LIKE pattern`, when `pattern` is a constant expression (does not contain a column reference), DB2 UDB and Oracle both use non-blank-padded semantics when comparing two strings using the LIKE operator.

For example, the following is false in both DB2 UDB and Oracle in most cases.

```
'ba  ' like '%a'
```

The exception to this rule occurs when, in Oracle, the pattern does not contain any pattern matching symbol (% or _). When this occurs the operation is optimized in a simple equality comparison where blank-padded semantics do apply so that the following is true in Oracle but still false in DB2 UDB.

```
'ba  ' like 'ba'
```

The translator converts the previous example to:

```
'ba  ' = 'ba'
```

Non constant patterns are not likely to be found in Oracle input source. In cases where a pattern is *not* a constant expression (contains a column reference), the converter does not apply any particular conversion and the generated DB2 UDB code might be incorrect. This will happen when the pattern expression evaluates to a value that contains no '%' or '_' and the string being compared has trailing blanks.

Related reference

"Blank and non-blank padding" on page 167
Oracle blank and non-blank padded semantics are mimicked in translations to IBM Informix Dynamic Server. String comparison is handled differently in DB2 UDB and Oracle, in particular when the string is of type VARCHAR.

Cursors

The converter supports a limited number of features concerning Oracle cursors in conversions to DB2 UDB and Oracle PL/SQL cursors in conversions to Informix Dynamic Server.

Cursor support in conversions to DB2 UDB

The converter supports a limited number of features concerning Oracle cursors in conversions to DB2 UDB.

The converter supports the following features:

- Oracle cursor declarations are converted directly into DB2 UDB cursor declarations.
- OPEN and CLOSE statements are converted directly into DB2 UDB OPEN and CLOSE statements.

- The FETCH statement is converted into a DB2 UDB compound statement containing a FETCH statement and a continue handler to avoid throwing a NOT FOUND exception when the cursor reaches the end of the result set.

The converter supports the following attributes:
- FOUND
- NOTFOUND
- SQL%ROWCOUNT
- SQL%FOUND
- SQL%NOTFOUND

To support the FOUND and NOTFOUND attributes, the translator creates a special integer variable associated with the DB2 UDB cursor and stores the SQLCODE after each FETCH statement. The attributes are then converted with a comparison between the variable and the standard SQLCODE values (FOUND=0, NOTFOUND=100).

Cursor FOR loops are converted into:
- OPEN statement
- WHILE loop
- 2 FETCH statements (one before the loop, one inside)
- CLOSE statement

The condition of the WHILE loop involves the creation of a variable and an assignment to the SQLCODE after each FETCH (as for the translation of the FOUND and NOTFOUND attributes).

The Oracle select FOR loop is converted into a DB2 UDB FOR loop. The use of aliases for the select statement is supported.

Parameterized cursors are translated using a prepare statement and the USING clause of the OPEN statement. The USING clause accepts only variables; therefore, if the cursor parameters are constant literals or complex expressions, then they are first stored in a variable. For example:

```
create procedure p is
  cursor curs(x int) is select * from t where t.i=x;
begin
  open curs(2);
  close curs;
end;

   ----- is translated
to -----

CREATE PROCEDURE P()
LANGUAGE SQL
BEGIN
    DECLARE CURS_STMTSTR VARCHAR(32672) DEFAULT
        'SELECT *  FROM T WHERE T.I = CAST(? AS INTEGER)';
    DECLARE X INTEGER;
    DECLARE CURS_STMT STATEMENT;
    DECLARE CURS CURSOR FOR CURS_STMT;
    PREPARE CURS_STMT FROM CURS_STMTSTR;
    SET X = 2;
    OPEN CURS USING X;
    CLOSE CURS;
END!
```

Translation limitations

The following cursor features are not supported by the converter:

- Cursor variables and host variables
- Cursor return type
- BULK COLLECT INTO clause

The following attributes are not supported:

- ISOPEN
- ROWCOUNT
- SQL%ISOPEN
- SQL%BULKROWCOUNT

Related reference

"Control flow" on page 222
This section highlights PL/SQL statements that control the flow, or order in which, statements are executed.

Cursor support in conversions to Informix Dynamic Server

The converter supports a limited number of features concerning Oracle PL/SQL cursors in conversions to Informix Dynamic Server.

The converter supports:

- Simple FOR loop statements.
- OPEN statements.
- CLOSE statements.
- Only one FETCH statement from an open cursor. Subsequent FETCH statements, if they occur, are not translated. The single FETCH statement must be outside of looping statements (for example: while, loop).
- Cursor declarations, including parameters and the RETURN clause.

Limitations

The following cursor features are not supported:

- FETCH statements into different variables. All FETCH statements must be into the same variables.
- More than one cursor per PL/SQL block.
- Cursors in nested PL/SQL blocks.
- Looping statements that contain FETCH statements.
- Cursor attributes.
- The DELETE WHERE CURRENT OF and UPDATE WHERE CURRENT OF statements.

If these limitations greatly impact the Oracle PL/SQL to be migrated, please consider using Oracle cursor FOR loops, which are supported at a high level in MTK, or Informix Dynamic Server's ESQL/C programming language.

Data types

Oracle built-in data types (including ANSI types) are converted to similar data types in the target database, according to the mapping described here.

Oracle Data Type	DB2 UDB Data Type	Informix Dynamic Server
BOOLEAN (PL/SQL only)	INT (See “Boolean expressions” on page 166 for more information)	INTEGER
NATURAL	DECIMAL(31,0)	DECIMAL(32,0)
NUMBER[4]	FLOAT (negative scale not supported)	DOUBLE PRECISION
NUMBER(P) or NUMBER (p,0) where: p < 5 5 <= p < 10 10 <= p <19 19 <= p < 32 32 <= p	SMALLINT INTEGER BIGINT DECIMAL(p,0) DECIMAL(31,0)	SMALLINT INTEGER INT8 DECIMAL(p,0) DECIMAL(32,0)
NUMBER(P) or NUMBER (p,0) where: p < 5 5 <= p < 10 10 <= p <19 19 <= p < 33 33 <= p		SMALLINT INTEGER INT8 DECIMAL(p,0) DECIMAL(32,0)

NUMBER(p,s)[1]
 	
where:
s > 0 and p >= s
s > 0 and p< s
s < 0
	DECIMAL(min(p,31),min(s,31))
	DECIMAL(min(s,31),min(s,31))
	DECIMAL(min(p-s,31),0)
		DECIMAL(min(p,32),min(s,32))
		DECIMAL(min(s,32),min(s,32))
		DECIMAL(min(p-s,32),0)

VARCHAR2(n)[3]
VARCHAR(n)
CHAR VARYING(n)
CHARACTER VARYING(n)
[3]
	VARCHAR(n)	VARCHAR(n)
		LVARCHAR(n)

NVARCHAR2(n)
NCHAR VARYING(n)
NATIONAL CHAR VARYING(n)
NATIONAL CHARACTER VARYING(n)
	VARGRAPHIC (n)	NVARCHAR(n)
		NCHAR(n)

CHAR(n)
CHARACTER(N)
 	
where:
n < 255
else
	CHAR(n)
	where:
	1 <= n < 254 CHAR(n)
	255 <= n VARCHAR(n)
	VARCHAR(n)
		CHAR(n)
		VARCHAR(n)

NCHAR(n)
NATIONAL CHAR(n)
NATIONAL CHARACTER(n)
 	
where:
n < 128
else
	GRAPHIC(n)
	VARGRAPHIC(n)
		NCHAR(n)

Oracle Data Type	DB2 UDB Data Type	Informix Dynamic Server
DATE	TIMESTAMP	DATETIME YEAR TO FRACTION (5)
TIMESTAMP(p)[8]	TIMESTAMP	DATETIME YEAR TO FRACTION (5)
BFILE LONG RAW BLOB	BLOB (2G) NOT LOGGED[2]	BLOB
LONG CLOB	CLOB (2G) NOT LOGGED[2]	CLOB
NCLOB	DBCLOB (1G) NOT LOGGED[2]	CLOB
RAW(n)	VARCHAR(n) FOR BIT DATA	BLOB
ROWID	INTEGER	INTEGER
UROWID(n)	INTEGER	INTEGER
NUMERIC DECIMAL DEC	DECIMAL(31,0)	DECIMAL(32,0)

NUMERIC(p), NUMERIC(p,0)
DECIMAL(p), DECIMAL(p,0)
DEC(p), DEC(p,0)
	DECIMAL(min(p,31),0)	DECIMAL(min(p,32),0)

NUMERIC(p,s)
DECIMAL(p,s)
DEC(p,s)
 	
where:
s > 0 && p >= s
s > 0 && p < s
s < 0
	DECIMAL(min(p,31), min(s,31))
	DECIMAL(min(s,31), min(s,31))
	DECIMAL(min(p,31),0)
		DECIMAL(min(p,32), min(s,32))
		DECIMAL(min(s,32), min(s,32))
		DECIMAL(min(p,32),0)

INTEGER INT	INTEGER	INTEGER
SMALLINT	SMALLINT	SMALLINT
INTERVAL	(not converted)	(not converted)
FLOAT	FLOAT	DOUBLE PRECISION
FLOAT(n)	DOUBLE	DOUBLE PRECISION
DOUBLE PRECISION	DOUBLE	DOUBLE PRECISION
REAL	DOUBLE	DOUBLE PRECISION
LONG VARCHAR (xxx)	CLOB (2G) NOT LOGGED[2]	TEXT
BINARY_INTEGER PLS_INTEGER (PL/SQL only)	DECIMAL(31,0)	DECIMAL(32,0)

Table notes

1. When MTK converts NUMBER(p,s) to DB2, if s-p > 31 or s < -31 to DB2, the translation will be respectively DECIMAL(31,31) or DECIMAL(31,0). However,

the resulting data type will not be able to capture any digit of the source data type. In this particular case the translator will issue an error.

The maximum precision for Informix Dynamic Server is 32 and that for DB2 is 31.

When MTK converts NUMBER(p,s) NUMERIC(p,s), DECIMAL(p,s) , DECIMAL(p,s) to Informix Dynamic Server, if s< 0, the translation will always have s = 0. During the conversion of NUMBER(p,s), if s-p > 32 or s < -32, the translation will be respectively DECIMAL(32,32) or DECIMAL(32,0). However, the resulting data type will not be able to capture any digit of the source data type. In this particular case, MTK issues an error.

2. For procedures, variables and parameters generated with a DB2 UDB data type for large objects (LOB) are assigned an arbitrary size of 255K, not 1G or 2G as stated in this table. In these cases a warning is generated. DB2 UDB allocates memory for variables and parameters of these types. You can limit the size of these parameters to improve performance.
3. While Oracle VARCHAR data types have a limit of 4000 bytes, an Informix Dynamic Server VARCHAR data types has a limit of 255 bytes. An Informix Dynamic Server LVARCHAR has a limit of 32,739 Bytes.
4. MTK maps an Oracle VARCHAR2 data type to an Informix VARCHAR data type that is up to 255 bytes; MTK maps larger VARCHAR data types LVARCHAR data types.
5. Because an Informix Dynamic Server NVARCHAR data type has a limit of 255 bytes, MTK maps NVARCHAR to NCHAR when n> 255.
6. MTK maps the Oracle NUMBER data type to an Informix DOUBLE PRECISION data type. The range of values for the DOUBLE PRECISION data type is the same as the range of the C double data type on your computer.
7. MTK translates an Oracle NCHAR data type to an Informix NCHAR data type for all values of *n*. When translating to DB2, if n< 128, MTK translates NCHAR to DB2 GRAPHIC; otherwise MTK translates NCHAR to DB2 VARGRAPHIC.
8. MTK converts the Oracle TIMESTAMP to DB2 TIMESTAMP with a default precision of 6 and to an Informix Dynamic Server DATETIME YEAR TO FRACTION data type, with the fractional part having a precision of 5. MTK does not support the translation of the WITH TIME ZONE and WITH LOCAL TIME ZONE clauses of the original Oracle 9i TIMESTAMP data type.

MTK automatically maps source data types to target database data types. The default mappings are listed in a table provided in the Global Type Mapping window. You can use this window to change the mapping.

Important: If you are migrating from Oracle to Informix Dynamic Server and have columns that store very large amounts of binary or text data in each field, map these columns to BLOB or CLOB columns. Mapping columns with very large data fields to TEXT or BYTE columns might cause MTK to run out of memory. In addition, if you are mapping source data types to Informix Dynamic Server large object data types, check to be sure you created spaces for the large objects in Dynamic Server. You must do this before you can deploy data.

Related reference

DB2 UDB Version 8 - Data types

Implicit and explicit data type conversion

Oracle PL/SQL supports explicit and implicit data type conversions. These explicit and implicit data types are converted to Informix Dynamic Server syntax using Informix Dynamic Server's explicit and implicit data type conversion mechanism.

For explicit data type conversions, Oracle uses built-in functions. However, in conversions to Informix Dynamic Server, MTK only supports the following built-in functions:

- CAST
- TO_CHAR(date)
- TO_CHAR(number)
- TO_DATE
- TO_TIMESTAMP

For implicit data type conversions, MTK does not support BLOB, CLOB, LONG, RAW, or UROWID conversions.

For more information Oracle PL/SQL explicit and implicit data type conversion, see the Oracle documentation.

ROWID

Oracle databases support a pseudocolumn named ROWID, such that every row in every table is assigned an auto-generated ID. Oracle also allows you to define a column of type ROWID, which can be used as a foreign key to a ROWID pseudocolumn.

In Oracle databases, you can also create tables and clusters that contain actual columns that have the ROWID data type. However, the values of such columns are not necessarily valid ROWIDs. Informix Dynamic Server does not have the ROWID data type.

Conversions to Informix Dynamic Server

Informix Dynamic Server does not have the ROWID data type. Instead, it has the ROWID keyword. This can be used to query the pseudocolumn ROWID of a Dynamic Server table. However, since there is no such data type as ROWID, creating a table column with data type ROWID is not possible.

In conversions to Dynamic Server, you do not need to use the **Add ROWID column in every table** option on the Convert page of MTK to create a special ROWID column. This is because all non-fragmented tables in Dynamic Server have the ROWID virtual column. In the case of fragmented or partitioned tables, Dynamic Server needs a With ROWID clause in the CREATE TABLE statement.

Conversions to DB2 UDB

DB2 UDB does not have an auto-generated ROWID pseudocolumn. See the ″Migration strategies″ section below for information on simulating ROWID pseudocolumn behavior in DB2

In conversions to DB2 UDB, MTK converts all fragmented CREATE TABLES to non-fragmented tables.

Translation limitations

Because the converter can be used to translate many scripts separately in the same conversion, it is not possible to make an accurate assumption of whether ROWID is used in the source database.

Migration strategies

Migrating to DB2 UDB

Evaluate the Oracle source script and determine if it makes use of the auto-generated ROWID, such as SELECT ROWID FROM X. If the Oracle source script makes use of this pseudocolumn, you can select the **Add ROWID column in every table** option on the Convert page of MTK to simulate the behavior in DB2 UDB. When this option is selected, the translator does the following:

- A column named ROWID is added to every table:

```
create table people (lastname char(24), firstname char(18));
   ----- is translated
to -----
CREATE TABLE PEOPLE (LASTNAME CHAR(24), FIRSTNAME CHAR(18),
                     ROWID INTEGER GENERATED ALWAYS AS IDENTITY)!
```

- Statements referring to the table are translated using the explicit column names from the table, excluding the ROWID column.

```
insert into people values ('Jones','Sam');
   ----- is translated
to -----
INSERT INTO PEOPLE (LASTNAME,FIRSTNAME) VALUES ('Jones','Sam')!

select * from people;
   ----- is translated
to -----
SELECT PEOPLE.LASTNAME, PEOPLE.FIRSTNAME FROM PEOPLE!
```

- As a result of the ROWID pseudocolumn being translated to a DB2 UDB column named ROWID, statements that refer to the ROWID pseudocolumn are allowed and translated without error.

```
select rowid from people;
   ----- is translated
to -----
SELECT ROWID FROM PEOPLE!
```

Whether or not you specify to add the ROWID column, the translator recognizes the ROWID type and either translates the code or generates an error accordingly.

- If the script defines a column to be of type ROWID, the converter translates it to DB2 UDB type INTEGER.

```
create table t (x int, r rowid);
   ----- is translated
to -----
CREATE TABLE T (X INTEGER, R INTEGER)!
```

- Oracle uses a non-standard type to encode ROWID values. Therefore, if a ROWID *literal* value is encountered in the source script, the translator converts the value to NULL and issues an error.

```
insert into t (select y, 'AAAM/WAABAACc90AAA' from u);
   ----- is translated
to -----
--* [700116] Oracle Rowid values are ignored.
INSERT INTO T SELECT Y, CAST (NULL AS INTEGER) FROM U!
```

Record

The converter uses DB2 UDB structured types to translate the declaration and use of Oracle PL/SQL record data types. The converter simulates record variables.

The DB2 UDB procedures language does not permit user-defined types (including structured types).

The converter handles record types as follows:

Type declarations

The converter does not generate any SQL statement for record type declarations, but the declarations are processed to be referenced later.

```
type Rec1 is record (i int, j char(2));
type Rec2 is record (a int, b Rec1);
   ----- no translation
-----
```

Variable declaration

Some PL/SQL scripts contain variable declarations that use previously declared record types. In such cases, the declaration is translated by the converter into multiple DB2 UDB variable declarations. One variable declaration is produced for each field of the record type. If a field has a record type then the converter will generate additional declarations for the sub-fields of that field. The name of the DB2 UDB variable is obtained by appending the name of the field to the name of the Oracle variable if the name is not already in use (the following example uses the record types defined in the previous example).

```
var1 Rec1;
var2 Rec2;
   ----- is translated
to -----
DECLARE VAR1_I INTEGER;
DECLARE VAR1_J CHAR(2);
DECLARE VAR2_A INTEGER;
DECLARE VAR2_B_I INTEGER;
DECLARE VAR2_B_J CHAR(2);
```

Field selection

In PL/SQL, you can refer to the individual fields of a variable. This type of reference expression is converted to a reference to the DB2 UDB variable corresponding to the field. If the field itself has a record type, there will be no corresponding DB2 UDB variable, but rather a list of variables.

Assignments and SELECT INTO statements

PL/SQL scripts that contain an assignment involving record types, are not directly converted. The converter will generate a set of assignments using the DB2 UDB variables corresponding to the fields of the record type being used in the assignment.

```
var1.i := 2;
var1.j := 'test';
var2.a := 1;
var2.b := var1;

   ----- is translated
to -----

SET VAR1_I = 2;
SET VAR1_J = 'TEST';
SET VAR2_A = 1;
SET VAR2_B_I = VAR1_I;
SET VAR2_B_J = VAR1_J;
```

In PL/SQL, you can use SELECT INTO with a variable (or a field of a variable) that has a record type. This kind of statement is converted into a SELECT INTO using the list of DB2 UDB variables corresponding to the fields of the record type.

```
CREATE TABLE T (i INT, j CHAR(2));
SELECT * INTO var1 FROM T;
   ----- is translated
to -----
CREATE TABLE T (i INT, j CHAR(2));
SELECT *
INTO VAR1_I,
   VAR1_J
FROM T;
```

%TYPE

In PL/SQL, you can refer to the type of an existing variable, field, or column using the %TYPE attribute. The converter replaces this reference directly by the type it refers to.

Translation limitations

The converter currently supports %TYPE for variables, columns, and fields of record variables.

Examples

```
CREATE TABLE T (i INT, j CHAR(2));

i INT;
TYPE Rec1 IS RECORD (i INT, j CHAR(2));
var1 Rec1;
var3 i%TYPE;
var4 var1%TYPE;
var5 t.i%TYPE;
   ----- is translated
to -----
CREATE TABLE T (i INT, j CHAR(2));

DECLARE I INTEGER;
DECLARE VAR1_I INTEGER;
DECLARE VAR1_J CHAR(2);
DECLARE VAR3 INTEGER;
DECLARE VAR4_I INTEGER;
DECLARE VAR4_J CHAR(2);
DECLARE VAR5 INTEGER;
```

%ROWTYPE

In PL/SQL, you can refer to the record type that represents a row of a table, a cursor, or a cursor variable. PL/SQL scripts containing the %ROWTYPE attribute are handled in the same way as a record type that has been previously declared.

If the %ROWTYPE attribute is used for a variable declaration, the converter produces a list of DB2 UDB variables matching the record fields.

```
CREATE TABLE T (i INT, j CHAR(2));

var6 t%ROWTYPE;
   ----- is translated
to -----
CREATE TABLE T (i INT, j CHAR(2));

DECLARE VAR6_I INTEGER;
DECLARE VAR6_J CHAR(2);
```

Translation limitations

The converter only supports the use of %ROWTYPE for columns and cursors.

Subtypes

References to subtypes declared in packages and PL/SQL blocks are converted to the DB2 UDB and Informix Dynamic Server equivalent base types.

Examples

```
create package p1 as:
 SUBTYPE newInt is Number(5,0);
end p1;
/
...
declare
 var1 p1.newInt;
   ----- is translated
to -----
DECLARE VAR1 INTEGER:
```

VARRAY

The Oracle VARRAY functionality is supported in conversions to DB2 Viper II.

Conversions of VARRAYs have the following limitations:

- Oracle VARRAYs that contain data types that map to LONG VARCHAR, LONG VARGRAPHIC, XML, REFERENCE, UDT, and ARRAY are not supported.
- The NOT NULL clause of Oracle syntax is not supported.
- Record variables and Oracle_supplied_types syntax are not supported.
- The ARRAY data type is not a valid table column type.
- ARRAY data types cannot be parameters or the return type of a user-defined function.
- Global variables cannot be of the ARRAY data type.
- BULK COLLECT clauses found in anonymous PL/SQL blocks are not translated.
- Casting into a varray from outside the procedure context is not translated.
- User-defined varray types are not supported.
- FORALL statements are translated only when varray data types are used as indices.
- Varray elements are not supported in FETCH statements. In the example below, *n*(loop_cnt) is a varray element which is not supported in conversions to DB2 Viper II:

  ```
  FETCH c1 INTO n(loop_cnt)
  ```
- Varrays in cursors are only supported when there is a single FETCH statement with a BULK COLLECT clause. For example:
  - The following Oracle statement is correctly translated:

    ```
    DECLARE
    TYPE NameList IS VARRAY(50) OF CHAR(50);
    CURSOR c1 IS SELECT ename FROM emp WHERE sal > 1000;
    names NameList;
    BEGIN
    OPEN c1;
    FETCH c1 BULK COLLECT INTO names;
    END;

       ----- is translated
    to -----

    DECLARE
    TYPE NameList IS CHAR(50) ARRAY[50];
    CURSOR c1 IS SELECT ARRAY_AGG(ename) FROM emp WHERE sal > 1000;
    names NameList;
    ```

```
BEGIN
OPEN c1;
FETCH c1 INTO names;
END;
```

– The following Oracle statement is not correctly translated:

```
DECLARE
TYPE NameList IS VARRAY(50) OF CHAR(50);
CURSOR c1 IS SELECT ename FROM emp WHERE sal > 1000;
names NameList;
nm char(50);
BEGIN
OPEN c1;
FETCH c1 BULK COLLECT INTO names;
FETCH c1 INTO nm;
END;
```

- The LIMIT clause is ignored and all the rows are fetched into varray.
- DB2 has same SQLSTATE for different Oracle exceptions. Therefore, if you have multiple VARRAY exception declarations in the same procedure, they will not work on DB2, You need to manually correct these before deploying to DB2. VALUE_ERROR, SUBSCRIPT_OUTSIDE_LIMIT, and SUBSCRIPT_BEYOND_COUNT are mapped to DB2 SQLSTATE=2202E.
- COLLECTION_NULL is not supported.

In conversions to DB2 Viper II, the Oracle type declaration is moved outside of the procedure or function. The means that the scope of the type declaration is now global, which is different than that of Oracle. MTK generates a warning, for example: `The type declaration has been moved to the global scope`. The following example shows the Oracle TYPE declaration being migrated to DB2 Viper II:

```
CREATE TABLE tab(col1 char(20),
  col2 date);
CREATE OR REPLACE PROCEDURE p2_2 AS

TYPE cust_id_list_32 IS VARRAY(10) OF integer;

     j cust_id_list_32;

begin

j := cust_id_list_32(1,3,NULL,8,9);

end
/

   ----- is translated
to -----

CREATE TABLE tab(
    col1 CHAR(20),
    col2 DECIMAL(31,0)
)!

CREATE TYPE cust_id_list_32 AS DECIMAL(31,0) ARRAY[10]!

(TYPE declaration had been moved to the global scope)

CREATE PROCEDURE p2_2()
LANGUAGE SQL

BEGIN

    DECLARE j CUST_ID_LIST_32;
  SET j = ARRAY[1, 3, NULL, 8, 9];

END!
```

Function calls on VARRAY:

Oracle functions that are called on VARRAYs are supported in migrations to DB2 Viper II.

The following table shows the Oracle functions and their translated DB2 Viper II equivalent:

Oracle function call	DB2 translation	Explanation
`numbers.COUNT`	`CARDINALITY(numbers)`	This returns the number of elements in the array.
`numbers.LIMIT`	`MAX_CARDINALITY(numbers)`	This returns the maximum number of elements in the array.
`first_index := numbers.FIRST;`	`SET first_index = 1;`	This returns the first index.
`last_index := numbers.LAST;`	`SET last_index = CARDINALITY(numbers);`	This returns the last index. For VARRAYs, COUNT and LAST are always equal.
`n := numbers.NEXT(n);`	`SET n = n + 1;`	This returns the next index.
`n := numbers.PRIOR(n);`	`SET n = n - 1;`	This returns the previous number.

`numbers.EXISTS(i)`	`CASE`
	`WHEN i >= 1`
	`AND i <= CARDINALITY`
	`(numbers) THEN 1`
	`ELSE 0`
	`END;`
		This tests for existence of the element value.

`numbers.EXTEND();`
`numbers.EXTEND(`*n*`);`
	No translation required.	This increases the array size.

`numbers.EXTEND(`*n*`,i);`	`DECLARE extend_count`
	` INTGER DEFAULT 0;`
	`WHILE extend_count < n DO`
	`SET numbers[CARDINALITY`
	`(numbers) + 1] =`
	` numbers[i];`
	`SET extend_count =`
	` extend_count + 1;`
	`END WHILE;`
		This append ith element *n* times.

`numbers.TRIM();`	`SET numbers = TRIM_ARRAY(numbers, 1);`	This removes one element.
`numbers.TRIM(`*n*`);`	`SET var = TRIM_ARRAY(var,` *n*`);`	This removes *n* elements.
`numbers.DELETE;`	`SET numbers = ARRAY[];`	This remove all elements.
PRIOR(*n*)	*n* + 1	In conversions from Oracle, if *n* has no predecessor, MTK does not return NULL.
NEXT(*n*)	*n* - 1	In conversions from Oracle, if *n* has no successor, MTK does not return NULL.

Exceptions in PL/SQL

MTK supports Oracle PL/SQL predefined exceptions and user-defined exceptions.

Predefined exceptions

Oracle PL/SQL predefines some common Oracle errors as exceptions. For example, PL/SQL raises the predefined exception NO_DATA_FOUND if a SELECT INTO statement returns no rows. Informix Dynamic Server also supports predefined and user defined exceptions, however, they are represented numerically, not with labels.

Oracle has many predefined and user defined exception labels such as CURSOR_ALREADY_OPEN and NO_DATA_FOUND, ZERO_DIVIDE. The labels can be used in the logic to identify errors. Only one exception check is allowed per BEGIN and END block using the EXCEPTION construct, and it is normally placed just before the END statement.

With Informix Dynamic Server, all of the exceptions checked in the stored procedure control blocks, delimited with BEGIN and END statements, must be declared explicitly at the top of each control block with the Informix Dynamic Server EXCEPTION construct. Therefore, Oracle procedure code must be restructured to manipulate these exceptions.

The following table provides mapping between the Oracle predefined exceptions and the Informix Dynamic Server errors:

Oracle	Informix Dynamic Server	Description
ACCESS_INTO_NULL	Not supported	See Oracle documentation for error description. MTK does not support objects.
CASE_NOT_FOUND	Not supported	See Oracle documentation for error description. Informix Dynamic Server does not return any error.
COLLECTION_IS_NULL	Not supported	See Oracle documentation for error description. MTK does not support varray in migrations to Informix Dynamic Server.
CURSOR_ALREADY_OPEN	Not supported	See Oracle documentation for error description. Informix Dynamic Server does not return any error.
DUP_VAL_ON_INDEX	-239, -268	See Informix Dynamic Server documentation for error description.

Oracle	Informix Dynamic Server	Description
INVALID_CURSOR	-285	See Informix Dynamic Server documentation for error description.
INVALID_NUMBER	-1213	See Informix Dynamic Server documentation for error description.
LOGIN_DENIED	-950, -951, -952	See Informix Dynamic Server documentation for error description.
NOT_LOGGED_ON	-349	See Informix Dynamic Server documentation for error description.
PROGRAM_ERROR	-959	See Informix Dynamic Server documentation for error description.
ROWTYPE_MISMATCH	-1279	See Informix Dynamic Server documentation for error description.
SELF_IS_NULL	Not supported	See Oracle documentation for error description. MTK does not support objects.
STORAGE_ERROR	-837	See Informix Dynamic Server documentation for error description.
SUBSCRIPT_BEYOND_COUNT	Not supported	See Oracle documentation for error description. MTK does not support varray in migrations to Informix Dynamic Server.
SUBSCRIPT_OUTSIDE_LIMIT	Not supported	See Oracle documentation for error description. MTK does not support varray in migrations to Informix Dynamic Server.
SYS_INVALID_ROWID	Not supported	See Oracle documentation for error description. Informix Dynamic Server does not support this rowid datatype
TIMEOUT_ON_RESOURCE	Not supported	See Oracle documentation for error description.

Oracle	Informix Dynamic Server	Description
TOO_MANY_ROWS	-284	See Informix Dynamic Server documentation for error description.
VALUE_ERROR	-1279, -1213	See Informix Dynamic Server documentation for error description.
ZERO_DIVIDE	-1202	See Informix Dynamic Server documentation for error description.

Here is an example of how an Oracle predefined exception is converted in migrations to Informix Dynamic Server:

```
CREATE PROCEDURE   P1 AS
h1 INT;
BEGIN
SELECT c1 INTO h1 FROM t1;
EXCEPTION
WHEN TOO_MANY_ROWS THEN
DBMS_OUTPUT.PUT_LINE ('Two many rows');
END;
/

   ----- is translated
to -----

CREATE PROCEDURE   P1 ()
SUBSCRIPT_BEYOND_DEFINE h1 INT;
ON EXCEPTION IN (-284)
CALL ORA.PUT_LINE ('Two many rows');
END EXCEPTION
SELECT c1 INTO h1 FROM t1;
END PROCEDURE
```

User-defined exceptions

PL/SQL allows you to create user-defined exceptions of your own. Informix Dynamic Server (IDS) also allows you to generate your own exceptions using integer numbers.

MTK generates unique integer numbers that correspond to the Oracle PL/SQL user-defined exception labels.

The following example shows how the Oracle PL/SQL user-defined exceptions are converted to IDS exceptions:

```
minimum_balance EXCEPTION ;

RAISE minimum_balance;
...
EXCEPTION WHEN minimum_balance THEN
DBMS_OUTPUT.PUT_LINE('Handling minimum_balance exception.');
...
WHEN OTHERS THEN
DBMS_OUTPUT.PUT_LINE('Could not recognize minimum_balance exception.');
END; /

   ----- is translated
to -----

DEFINE minimum_balance INT;
DEFINE others INT;
ON EXCEPTION SET others
ORA.PUT_LINE('Could not recognize minimum_balance exception.');
END EXCEPTION;

ON EXCEPTION IN ( 10 )
ORA.PUT_LINE('Handling minimum_balance exception.');
END EXCEPTION;

LET minimum_balance = 10;

RAISE EXCEPTION minimum_balance;
```

PL/SQL exceptions and Informix Dynamic Server error numbers

Oracle error messages do not directly map to Informix Dynamic (IDS) error numbers and so you need to manually correct these values.

To caution you during translation, MTK generates the following message during translation:

```
Oracle and Informix Dynamic Server database error numbers are different,
check the documentation and manually correct the error number.
```

Here is an example of how to manually change the error message value:

```
CREATE PROCEDURE   p1 AS
table_not_found EXCEPTION;
PRAGMA EXCEPTION_INIT(table_not_found, -942);
BEGIN
INSERT INTO emp_phone VALUES ('800-555-1234') ;
EXCEPTION
WHEN table_not_found THEN
CREATE TABLE tab1(c1 CHAR(10));
END;
/

   ----- is translated
to -----

CREATE PROCEDURE p1()
DEFINE table_not_found INT;

ON EXCEPTION IN ( -206 )
CREATE TABLE tab1(c1 CHAR(10));
END EXCEPTION;

LET table_not_found = -206;
INSERT INTO emp_phone VALUES ('800-555-1234') ;
END PROCEDURE;
```

Expressions

All the basic expressions are supported by the converter. Some expressions require special handling, as detailed here.

Informix Dynamic Server supports one item in the expression list for an IN predicate only if the expression list is on the left-hand side. If the expression list is on the right-hand side, there are no restrictions. For example:

```
...

select * from t1 where (1) in col1; --> right
select * from t1 where (1,2) in col1; --> error
select * from t1 where col1 in (1); --> right
select * from t1 where col1 in (1,2); --> right
```

Cast expression conversion

Cast expressions are supported in Informix Dynamic Server. These expressions are also supported in DB2 UDB, except for those for non-scalar sub-queries, where the result set size is greater than one.

For example, in a conversion to Informix Dynamic Server:

```
SELECT CAST ('1997-10-22' AS DATE) FROM DUAL;
   ----- is translated
to -----
SELECT DATETIME(1997-03-10-00.00.00.00000) YEAR TO FRACTION(5)
FROM informix.dual;
```

Character literals

In converting Oracle PL/SQL to Informix Dynamic Server, case sensitivity of character literals is preserved.

CURRVAL and NEXTVAL

In conversions to DB2 UDB, generally CURRVAL is converted to PREVVAL and NEXTVAL is converted to NEXTVAL. However, if references to CURRVAL and NEXTVAL occur in the same statement, CURRVAL is converted to NEXTVAL in order to preserve Oracle behavior.

When SEQUENCE is converted to Informix Dynamic Server, CURRVAL remains ″CURRVAL and NEXTVAL remains NEXTVAL.

Datetime literals

INTERVAL data types and INTERVAL literals are not supported.

Date and timestamp literals

In translations to DB2 UDB, MTK reformats string literals that represent DATE to fit the DB2 UDB format.

In translations to Informix Dynamic Server, MTK converts each Oracle DATE and TIMESTAMP literals to an Informix Dynamic Server DATETIME(<literal>) YEAR TO FRACTION(5).

For example, in conversions to Informix Dynamic Server:

```
SELECT DATE'1911-11-11' FROM DUAL;
   ----- is translated
to -----
SELECT DATETIME(1911-11-11-00.00.00.00000)
 YEAR TO FRACTION(5)FROM dual;
```

```
INSERT INTO T2(T1) VALUES
 (TIMESTAMP'1911-11-11 11:11:11');
   ----- is translated
to -----
INSERT INTO t2(t1) VALUES
 (DATETIME(1911-11-11-11.11.11.00000)
 YEAR TO FRACTION(5) );
```

Exception: In the CREATE-DEFAULT clause, MTK converts Oracle DATE literals into Informix Dynamic Server DATE literals in the GL_DATE or GL_DATETIME format.

MTK sets the following GL_DATE and GL_DATETIME variables as an environment or JDBC connection attributes during the deployment phase of conversion.

- GL_DATE=″%Y-%m-%d″

- GL_DATETIME=″%Y-%m-%d-%H.%M.%S%iF5″

Date and time special registers

MTK converts the following Oracle date/time special registers to their Informix Dynamic Server equivalents:

- sysdate
- current_date
- localtimestamp(p)

For example, the Oracle statement

```
insert into t2(d) values(sysdate);
   ----- is translated
to -----
insert into t2(d) values(CURRENT YEAR TO FRACTION(5));
```

However, MTK does not convert the following date/time special registers:

- current_timestamp(p)
- dbtimezone
- sessiontimezone
- systimestamp

Both current_timestamp(p) and systimestamp return a timestamp with a time zone value, which is not supported.

You can map an Oracle DATE to an Informix Dynamic Server date manually through global type mapping or using this pragma:

```
/*PRAGMA translate_type("DATE","DATE")*/
```

Date and timestamp arithmetic in expressions

MTK supports conversion of Oracle date and timestamp arithmetic in expressions, except MTK does not support the conversion from type INTERVAL DAY TO SECOND to type NUMBER.

Null literals

Null literals in most contexts in DB2 UDB must be cast to the type they are representing. The converter inserts these casts where necessary (they are not needed in INSERT, so NULL is not cast in INSERT). If the converter is unable to determine the type for the NULL then it uses VARCHAR(1).

Numeric literals

In converting from Oracle PL/SQL to Informix Dynamic Server:

- If a numeric literal value is composed of only digits and is within the range -2 147 483 648 to 2 147 483 647 it has a PLS_INTEGER data type; otherwise this literal has the NUMBER data type.
- Because BINARY_FLOAT and BINARY_DOUBLE data types are not supported, numeric literals with the suffix of ‘f’ or ‘d’ (for example, 2.0f or 2.0d) are not supported.

String literals

In conversions from Oracle PL/SQL to Informix Dynamic Server, case sensitivity of string literals is preserved. However, the following Oracle PL/SQL delimiter notations are not supported in conversions to Informix Dynamic Server:

- The `q'!...!'` notation, which allows the use of single quotes inside the string literal.

- The `nq'...'` notation, which is used for NCHAR and NVARCHAR2 literals.

The following sections require more detailed explanation.

NULL and empty string values

Empty string and NULL values are recognized as equivalents in Oracle, but not in DB2 UDB or Informix Dynamic Server.

During the migration process, empty strings are converted to NULL values to ensure consistent target server output.

The following example illustrates how Oracle and DB2 UDB behave differently regarding empty strings. Oracle returns 1, and if converted literally, DB2 UDB would return 0. Therefore, the code is translated to DB2 UDB with a NULL value in place of the empty string.

```
create table T ( val char(10));
insert into T values ('');

select count (*) from T where val is NULL;

   ----- is translated
to -----

CREATE TABLE T(VAL CHAR(10))!
INSERT INTO T VALUES (CAST (NULL AS CHAR(1)))!

SELECT COUNT(*) FROM T WHERE VAL IS NULL!
```

Related reference

"Concatenation" on page 194
Oracle and DB2 UDB and Informix Dynamic Server have different rules for using null and empty string values with the concatenation operator.

Implicit type conversion

Implicit type conversion occurs whenever the context expects a certain type that is different from the type of the expression. Oracle performs more implicit type conversion (such as implicit conversion of numbers to strings) than DB2 UDB or Informix Dynamic server performs.

For example, in Oracle, CHAR or VARCHAR2 can be used where NUMBER is expected and vice-versa. However, MTK does not convert most of these implicit conversions are not made. Instead, the converter applies explicit conversions either during the conversion (as with literals) or with a conversion function (for example, from ORA.TO_CHAR or ORA.TO_NUMBER) when converting to DB2 UDB.

Each of the following examples show how the converter translates the implicit Oracle conversions into explicit DB2 UDB conversions. Similar conversions occur for translations to Informix Dynamic Server.

NUMBER to STRING conversions

In the context of a call to the built-in function SUBSTR, a number can be passed as first argument, although a CHAR or VARCHAR is expected:

```
select substr(1000,2) from dual;

   ----- is translated
to -----

SELECT ORA.substr('1000', 2)
FROM SYSIBM.SYSDUMMY1!
```

Notice that quotes have been added around the number literal 1000. When the expression is not a literal, like in the case of a column reference of type NUMBER or a function that returns a NUMBER, the conversion function ORA.TO_CHAR will be applied. In the following example, T is a table with a column C of type NUMBER:

```
selec substr(C,2), substr(COS(.5),2) fromT;

   ----- is translated
to -----

SELECT ORA.SUBSTR(ORA.TO_CHAR(C), 2), SUBSTR(ORA.TO_CHAR(COS(.5)),2)
FROM T!
```

STRING to NUMBER conversions

Here, 1 is expected to be a NUMBER:

```
select '1' + 1000 from dual;
```

It is converted to:

```
SELECT 1 + 1000
FROM SYSIBM.SYSDUMMY1!
```

If T is a table with a column C of type VARCHAR:

```
select c + 1000 from dual;

   ----- is translated
to -----

SELECT ORA.TO_NUMBER(C) + 1000
FROM SYSIBM.SYSDUMMY1!
```

DATE to STRING conversions

In this case both Oracle and DB2 UDB provide implicit conversion between these types but the formats differ for the STRING version. In this case the ORA UDF is used to convert between Oracle and DB2 UDB string date formats. If T is a table with a column of type VARCHAR:

```
insert into T values (SYSDATE);

   ----- is translated
to -----

INSERT INTO T VALUES (ORA.TO_CHAR( CURRENT TIMESTAMP ))!
```

In the previous example, the DB2 UDB code would work without the ORA.TO_CHAR function, but a different string value would appear in the table.

STRING to DATE conversions

Here a date value is expected to the right of the operand:

```
select 'Today is Y2k' from dual
where SYSDATE = '01-jan-2000';

   ----- is translated
to -----

SELECT 'Today is Y2k' FROM SYSIBM.SYSDUMMY1
WHERE  CURRENT TIMESTAMP  = '2000-01-01-00.00.00.000000'!
```

In the previous example, the Oracle date string is converted directly to the DB2 UDB expected format.

FLOAT or DECIMAL to INT conversions

Although DB2 UDB does implicitly convert FLOAT or decimal values to INT, the conversion is handled differently in that Oracle rounds the value and DB2 UDB truncates it. Therefore, an explicit rounding must be applied. Here, T is a table with one column of type INT:

```
insert into T values (5.9);

   ----- is translated
to -----

INSERT INTO T  VALUES (ORA.ROUND(5.9))!
```

Table 9. Handling of Oracle implicit-type conversions

	Literals	Other cases
NUMBER to STRING	The number literal is enclosed in single-quotes.	ORA.TO_CHAR(number)
STRING TO NUMBER	Quotation marks are removed from the string literal.	ORA.TO_NUMBER(string)
DATE to STRING	N/A	ORA.TO_CHAR(date)
STRING to DATE	The string is converted to the DB2 UDB format for date literals then DB2 UDB implicitly converts the literal to DATE (for example 01-jan-2001 becomes 2001-01-01-00.00.00.000000)	ORA.TO_DATE(string)
FLOAT to INT	ORA.ROUND(float)	ORA.ROUND(float)

Explicit type conversion takes place in sub-queries as well. In sub-queries, the explicit type-conversion is applied to each SELECT item when needed. For example, T is a table with a column C of type CHAR:

```
select 'foo' from dual where (1,SYSDATE,0) in (Select * from t);

   ----- is translated
to -----

SELECT 'foo'
FROM SYSIBM.SYSDUMMY1
WHERE CURRENT TIMESTAMP IN (SELECT ORA.TO_DATE(C)
                            FROM T)!
```

In the case of a SELECT * in a sub-query, the conversion might need to query separately each column that requires an explicit conversion. For example, T contains columns A, B, and C, and B requires an explicit conversion:

```
select 'foo' from dual where SYSDATE IN (SELECT * from T);

   ----- is translated
to -----

SELECT 'foo'
FROM SYSIBM.SYSDUMMY1
WHERE CURRENT TIMESTAMP IN (SELECT ORA.TO_DATE(T.B)
                            FROM T)!
```

ROWNUM

ROWNUM is not supported in Informix Dynamic Server. In translations to DB2 UDB, ROWNUM is converted differently for various statements.

Translation limitations

For example, in translations to DB2 UDB SELECT statements, ROWNUM is converted using the OLAP function: ROW_NUMBER () OVER(). In UPDATE and DELETE statements, ROWNUM is converted using a FOR loop.

- Outer joins with ROWNUM are not converted.
- Ordering may be different (unless ROWNUM is originally selected from a sub-query with an ORDER BY clause).
- Top-level UPDATE and DELETE statements with ROWNUM are not converted (top-level cursor FOR loops are not supported).
- ROWNUM inside the SET clause of an UPDATE statement is not converted.

Short-circuit evaluations

Informix Dynamic Server supports Oracle PL/SQL short-circuit evaluations.

SQLCODE and SQLERRM

Oracle SQLCODE is translated to DB2 UDB SQLCODE, and Oracle SQLERRM is translated to a variable, the value of which is retrieved using the DB2 UDB GET STATISTICS statement.

DB2 UDB and Oracle differ in how these variables are set. The Oracle SQLCODE refers to the last exception that was made and retains its value throughout the exception handling routine, whereas the DB2 UDB SQLCODE is set by each statement that is executed, successful or not. For this reason, the translation of the exception handling routine is not straightforward. Two variables are declared to contain the original values of SQLCODE and the explanation retrieved from the GET STATISTICS statement. The values of these variables are then used later in the handler as they are in the Oracle handler.

```
create procedure p is
 mes varchar(255);
 val int;
 a int;
 b int;
begin
 a:=0;
 b:=1/a;
exception
  when others then
    val := sqlcode;
    mes := sqlerrm;
    insert into t values(val, mes);
end;
/

   ----- is translated
to -----

CREATE PROCEDURE P()
LANGUAGE SQL
BEGIN
    DECLARE MES VARCHAR(255);
    DECLARE VAL INTEGER;
    DECLARE A INTEGER;
    DECLARE B INTEGER;
    DECLARE SQLCODE INTEGER DEFAULT 0;
    DECLARE SQLCODE1 INTEGER;
    DECLARE SQLERRM VARCHAR(255);
    DECLARE EXIT HANDLER FOR SQLEXCEPTION, SQLWARNING, NOT FOUND
        BEGIN
            GET DIAGNOSTICS EXCEPTION 1 SQLERRM = MESSAGE_TEXT;
            SET SQLCODE1 = SQLCODE;
            SET VAL = SQLCODE1;
            SET MES = SQLERRM;
            INSERT INTO T VALUES (VAL,MES);
        END;
    SET A = 0;
    SET B = ORA.ROUND(CAST (1 AS FLOAT) / A);
END!
```

Since in Oracle the execution of a procedure stops as soon as an exception is raised, any reference to an SQLCODE or SQLERRM that is not within an exception handler is translated to 0 or an empty string.

Translation limitations

Most of the time the error codes and messages between Oracle and DB2 UDB don't have the same meaning, and the converter does not match them together. If the error code or message is only inserted in a table to log the error, it is not necessary to determine the meaning of the message during migration. However, if the SQLCODE is compared to a literal value, you should refer to the Oracle and DB2 UDB message references to map the Oracle SQL code to the corresponding DB2 UDB SQL code.

Identifiers

Oracle SQL, DB2 UDB SQL, and IBM Informix Dynamic Server SQL have different length specifications for identifiers. The converter adjusts identifiers as necessary to meet the length limitations.

Oracle PL/SQL identifiers are mapped to their Informix Dynamic Server equivalents in most cases. However, MTK cannot process Oracle PL/SQL identifiers that are defined in blocks within the procedures. This limitation will be fixed in a future release.

The converter also detects instances of DB2 UDB keywords that are used as identifiers in Oracle. The keywords are marked with double quotation marks so they can be interpreted as identifiers.

When converting to IBM Informix Dynamic Server, MTK operates as follows:
- MTK replaces all instances of a space (" "), of "#", and of "@" in an identifier with an underscore ("_").
- MTK converts Oracle non-reserved words that are reserved words in IBM Informix Dynamic Server into delimited identifiers.
- Oracle delimited identifiers (identifiers such as ″TABLE″ that are delimited by double quotation marks and appear as quoted identifiers) remain delimited identifiers in Informix Dynamic Server.

Variable name conflicts can occur when converting from Oracle PL/SQL to IBM Informix Dynamic Server, if variables are moved from different scopes to the same scope. To avoid this, MTK renames the variables.

Migration strategies

The converter renames identifiers by retaining the first few characters and appending a number. To help you identify which objects have been renamed, the converter issues a message for each rename. If you are satisfied with the generated names, you can hide the message when refining the translation. If you prefer your own naming scheme, you can use the object renaming features on the Refine page of MTK.

Related concepts

"Object renaming" on page 129
Because of differences in rules for identifiers, some objects must be renamed during the translation. The converter generates new names in such a way that they do not conflict with any preexisting names.

Related tasks

"Changing an object name" on page 41

If an edit icon ( ) exists in the target column, you can change the name of the object.

Related reference

DB2 UDB Version 8 - SQL Limits

DB2 UDB Version 8 - Identifiers

Naming convention

Oracle PL/SQL names can be simple, qualified, remote, or both qualified and remote. However, MTK only supports translation of simple and qualified names to Informix Dynamic Server. Any kind of remote object names or synonyms for remote object names will return an unsupported syntax error.

Here are some examples of supported and unsupported Oracle PL/SQL names:

Oracle PL/SQL name	Type	Supported by MTK
`raise_salary(...);`	simple	yes
`emp_actions.raise_salary(...);`	qualified	yes
`raise_salary@newyork(...);`	remote	no
`emp_actions.raise_salary@newyork(...);`	qualified and remote	no

Operators

The converter behavior and limitation information in this section details how certain operators manipulate operands to return results differently in Oracle and the target database, either DB2 UDB or IBM Informix Dynamic Server.

The MINUS and INTERSECT operators in SELECT statements are mapped to the WHERE IN and WHERE NOT IN clause for Informix Dynamic Server. However, there are conditions where this mapping does not work:

- If the columns contain a NULL value, the output returned by Oracle is different than Informix Dynamic Server.
- If there is more than one column in the Oracle WHERE IN and WHERE NOT IN clause, Informix Dynamic Server does not support this. However, Oracle INTERSECT and MINUS can support more than one column.

See "Built-in-functions" on page 159, "Concatenation" on page 194, and "Division" on page 194for additional information on operators.

Division

Division is performed differently in Oracle and DB2 UDB. To ensure that the converted DB2 UDB output performs float division when it would normally perform integer division, the converter casts one of the operands (the numerator) to float.

In DB2 UDB, if both operands are integers, integer division is performed and the remainder is dropped so that an integer is returned as a result. Oracle uses float division which can result in non-truncated float values.

```
CREATE TABLE T(x INT);
INSERT INTO T VALUES(7);
SELECT 2/x, x/2 FROM T;

   ----- is translated
to -----

CREATE TABLE T(X INTEGER )!
INSERT INTO T VALUES(7)!
SELECT CAST(2 AS FLOAT)/X, CAST(X AS FLOAT)/2 FROM T!
```

The correct result = +2.85714285714286E-001 +3.50000000000000E+000. If the values were not cast to float, DB2 UDB would return 0 and 3.

Concatenation

Oracle and DB2 UDB and Informix Dynamic Server have different rules for using null and empty string values with the concatenation operator.

Translations to DB2 UDB

In Oracle a string concatenated with a null value returns the string. In DB2 UDB, a string concatenated with a null value returns NULL.

```
ORACLE:
NULL || 'a'  and 'a' || NULL return 'a'

DB2 UDB:

NULL || 'a'  and 'a' || NULL return 'null'
```

To maintain the behavior of the original Oracle source, operands for the concatenation operator are wrapped with the coalesce function to replace null with the empty string (''):

```
op1 || op2

   ----- is translated
to -----

COALESCE(op1,'') || COALESCE(op2,'')
```

The coalesce function is not used when it is not needed, for example, when an operand is a NOT NULLABLE column.

In Oracle, a NULL value concatenated with another returns NULL. In this case, concatenation expressions need to be wrapped with ORA.EMPTY_TO_NULL which will return a NULL instead of an empty string literal. (unless at least one of the operands is NOT NULL):

```
op1 || op2

   ----- is translated
```

```
to -----

ORA.EMPTY_TO_NULL(COALESCE(op1,'') || COALESCE(op2,''))
```

The generated DB2 UDB code never produces an empty string value because they are converted to NULL. Conversion of Oracle's built-in functions (with equivalent DB2 UDB built-in functions or UDFs in the ORA schema) will always return a NULL instead of an empty string literal. Oracle's CONCAT built-in function is converted to ORA.CONCAT which performs similar checking for NULL values and empty strings.

The following table illustrates common conversion examples regarding the use of null and empty string with the concatenation operator. The following examples are based on the CREATE TABLE statement:

```
CREATE TABLE T (str_nullable CHAR(10),
  str_not_null CHAR(10) NOT NULL, timestmp TIMESTAMP)
```

Oracle	DB2 UDB	Comments
'a' || 'b'	'a' || 'b'	No COALESCE or ORA.EMPTY_TO_NULL wrapping needed as 'a' and 'b' are not null string literals
SUBSTR('a',2) || 'b'	COALESCE(ORA.SUBSTR('a',2),'') || 'b'	COALESCE protects ORA.SUBSTR('a',2) which will eventually be NULL. No ORA.EMPTY_TO_NULL wrapping is necessary as both operands cannot be NULL.
'a' || str_not_null	'a' || str_not_null	No COALESCE or ORA.EMPTY_TO_NULL wrapping needed as 'a' is a not null string literal and str_not_null is a not nullable column
'a' || str_nullable	'a' || COALESCE(str_nullable,'')	str_nullable can be NULL and is therefore protected with the COALESCE function. ORA.EMPTY_TO_NULL is not needed as 'a' is a non-null string literal
str_nullable || str_nullable	ORA.EMPTY_TO_NULL (COALESCE(str_nullable,'') || COALESCE(str_nullable,''))	ORA.EMPTY_TO_NULL prevents NULL || NULL in Oracle from being converted to an expression that would return an empty string literal in DB2 UDB

Translations to Informix Dynamic Server

Translations to Informix Dynamic Server are similar, expect that Informix uses NVL instead of COALESCE.

For example:

```
SUBSTR('a',2) || 'b'

   ----- is translated
to -----

NVL(ORA.SUBSTR('a',2),'') || 'b'
```

Related reference

"NULL and empty string values" on page 188
Empty string and NULL values are recognized as equivalents in Oracle, but not in DB2 UDB or Informix Dynamic Server.

Operator precedence

Operations within an expression are completed in a particular order that depends on their precedence. Oracle and Informix have the same operator precedence.

The following table shows the Oracle and Informix Dynamic Server precedence of the operators in descending (highest to lowest) order.

Operation	Oracle operator	Informix Dynamic Server operator
Exponentiation	**	Pow( ) builtin function
Identity, negation	+, -	+, -
Multiplication, division	*, /	*, /
Addition, subtraction, concatenation	+, -, ||	+, -, ||
Comparison	=, <, >, <=, >=, <>, !=, ~=, ^=, IS NULL, LIKE, BETWEEN, IN	=, <, >, <=, >=, <>, !=, <>, <>, IS NULL, LIKE, BETWEEN, IN
Logical negation conjunction	NOT AND	NOT AND
Inclusion	OR	OR

Here is an example of how precedence works with Oracle and Informix Dynamic Server.

Oracle	Informix Dynamic Server
`CREATE PROCEDURE r2 AS`	`CREATE PROCEDURE r1( )`
`x INT ;`	`DEFINE x INTEGER ;`
`num1 NUMBER;`	`DEFINE num1 DOUBLE PRECISION;`
`num2 NUMBER;`	`DEFINE num2 DOUBLE PRECISION;`
`num3 NUMBER;`	`DEFINE num3 DOUBLE PRECISION;`
`num4 NUMBER;`	`DEFINE num4 DOUBLE PRECISION;`
`num5 NUMBER;`	`DEFINE num5 DOUBLE PRECISION;`
`pi NUMBER;`	`DEFINE pi DOUBLE PRECISION;`
`radius NUMBER;`	`DEFINE radius DOUBLE PRECISION;`
`bonus NUMBER;`	`DEFINE bonus DOUBLE PRECISION;`
`salary NUMBER;`	`DEFINE salary DOUBLE PRECISION;`
`commission NUMBER;`	`DEFINE commission DOUBLE PRECISION;`
`circle NUMBER;`	`DEFINE circle DOUBLE PRECISION;`
`ch char(200);`	`DEFINE ch CHAR(200);`

Oracle	Informix Dynamic Server
x := 1;	LET x = 1;
pi := 3.14;	LET pi = 3.14;
radius := 4.2;	LET radius = 4.2;
salary := 15000;	LET salary = 15000;
commission := 2;	LET commission = 2;
num1 := -x / 2 + 3;	LET num1 = -x / 2 + 3;
num2 := 5 + 12 / 4;	LET num2 = 5 + 12 / 4;
num3 := 12 / 4 + 5;	LET num3 = 12 / 4 + 5;
num4 := (8 + 6) / 2;	LET num4 = (8 + 6) / 2;
num5 := 100 + (20 / 5 + (7 - 3));	LET num5 = 100 + (20 / 5 + (7 - 3));
bonus := (salary * 0.05) + (commission * 0.25);	LET bonus = (salary * 0.05) + (commission * 0.25);
circle := pi * (radius ** 2);	LET circle = pi * (POW(radius, 2));
ch := 'abc' || 'def';	LET ch = 'abc' || 'def';
DBMS_OUTPUT.PUT_LINE('num1='||num1 || ' num2='||num2 || ' num3='||num3 || ' num4='||num4 || ' num5='||num5);	CALL ORA.PUT_LINE('num1=' || NVL(num1, '') || ' num2=' || NVL(num2, '') || ' num3=' || NVL(num3, '') || ' num4=' || NVL(num4, '') || ' num5=' || NVL(num5, ''));
DBMS_OUTPUT.PUT_LINE(' bonus='||bonus || ' circle='||circle || ' ch='||ch);	CALL ORA.PUT_LINE(' bonus=' || NVL(bonus, '') || ' circle=' || NVL(circle, '') || ' ch=' || ch);
END; /	END PROCEDURE;

Packages

MTK converts Oracle packages to their DB2 UDB and Informix Dynamic Server equivalents.

Packages in conversions to DB2 UDB

Oracle packages are converted to DB2 UDB using schemas.

A package *p* located in the schema *s* will be converted to a schema *p* in DB2 UDB. It is not possible to store variables or cursors in schemas, because schemas can only contain objects that can be created using DDL statements. Therefore, only package functions, procedures, and constants are converted in this manner. Constants in packages are not translated as variables; instead, they are translated as functions that return the value of the constant. References to the package constant are replaced by a call to the function.

Order of functions and procedures:

In Oracle a package is created using a package specification (create package statement) and a package body (create package body statement). The package specification can contain specifications for various database objects including functions and procedures. The complete definitions of these objects are located inside the package body.

Because DB2 UDB does not support forward declaration, the separation of specification from body presents two problems for translation:

- Since the functions and procedures are usually declared in the Oracle package specification, they can be defined in any order inside the Oracle package body. The converter will try to reorder the functions and procedures so that DB2 UDB can compile them without error. If these items contain a circular dependency, the converter tries to find the best ordering and issues a warning.
- If, in a source statement located between the specification and body, a reference is made to an object that has been specified but not yet defined, the reference is not resolved by the converter, and the code must be altered manually.

```
create package pk is
 procedure proc;
 function func(i int) return int;
end;
/

create package body pk is
 procedure proc is
 begin
  null;
 end;
 function func(i int) return int is
 begin
  return i+1;
 end;
end;
/

create procedure test is
 i int := 0;
begin
 pk.proc();
 i := pk.func(i);
end;
/

   ----- is translated
to -----

CREATE PROCEDURE PK.PROC()
LANGUAGE SQL
BEGIN
END!

CREATE FUNCTION PK.FUNC (I INTEGER )
RETURNS INTEGER
BEGIN ATOMIC
    RETURN I + 1;
END!

CREATE PROCEDURE TEST()
LANGUAGE SQL
BEGIN
    DECLARE I INTEGER DEFAULT 0;
    CALL PK.PROC();
    SET I = PK.FUNC(I);
END!
```

Packages in conversions to Informix Dynamic Server

Oracle PL/SQL packages are mapped to Informix Dynamic Server equivalents.

The Oracle PL/SQL package reference within the SQL file is noted by MTK and when the actual procedure or function within the package body is translated, the procedure or function name is prefixed with the name of the package.

Only package functions, procedures, variables, and constants are supported for Informix Dynamic Server. The procedures and functions within the create package body statement are mapped to standalone procedures and the name of the package is added as the schema name to the procedure or the function created.

The CREATE OR REPLACE PACKAGE syntax is taken into account, however, because there is not an equivalent statement in Informix Dynamic Server, it is not mapped to anything. MTK issues the following message:

```
No translation available, but the statement has been taken into account.
```

The invoker_rights_clause is not supported and the following message is issued:

```
This package item is not translated.
```

The CREATE OR REPLACE PACKAGE BODY statement is considered, however, because there is not an equivalent statement in Informix Dynamic Server it is not mapped to anything.

Example

The following example shows an Oracle package and its resulting conversion to Informix Dynamic Server.

```
create table tab1(c1 int, c2 int, c3 char(30), c4 date);
insert into tab1 values(1,1,'hello','1-JAN-2000');

create or replace package pack1
as
function simple_func(vc1 int) return number;
procedure simple_proc(vc1 int);
end;
/

create or replace package body pack1
as
v_date date;
v_char char(30);
v_int int;
function simple_func(vc1 int)
 return number
 as
 begin
 return vc1;
 end;

procedure simple_proc(vc1 int)
 as vc2 int;
 begin
 vc2 :=simple_func(5);
 insert into tab1 values(vc1,vc2,2,null);
 end;
begin
-- the following will be executed every time the package is executed.
 v_int:=simple_func(1);
 v_date :='2-JAN-2000';
 v_char:='from block';
 insert into tab1 values(3,v_int,v_char,v_date);
 simple_proc(30);
end;
/

call pack1.simple_proc(30);

   ----- is translated
```

to -----

```
CREATE TABLE tab1(
    c1 INTEGER,
    c2 INTEGER,
    c3 CHAR(30),
    c4 DATETIME YEAR TO FRACTION (5)
);
INSERT INTO tab1 VALUES (1,1,'hello',DATETIME(2000-01-01-00.00.00.00000)
YEAR TO FRACTION(5) );

CREATE PROCEDURE pack1.decl_pack1_private_vars()
        CREATE TEMP TABLE pack1_private_vars(
        v_date DATETIME YEAR TO FRACTION (5),
        v_char CHAR(30),
        v_int INTEGER
    ) WITH NO LOG ;
    INSERT INTO pack1_private_vars VALUES (NULL,NULL,NULL);
END PROCEDURE;

EXECUTE PROCEDURE pack1.decl_pack1_private_vars();
CREATE FUNCTION pack1.get_v_date (return_val DATETIME YEAR TO FRACTION (5) )
RETURNING DATETIME YEAR TO FRACTION (5)
    SELECT v_date
    INTO return_val
    FROM pack1_private_vars;
    RETURN return_val;
END FUNCTION ;

CREATE PROCEDURE pack1.set_v_date (new_val DATETIME YEAR TO FRACTION (5) )
    UPDATE pack1_private_vars
          SET v_date = new_val;
END PROCEDURE;

CREATE FUNCTION pack1.get_v_char (return_val CHAR(30) )
RETURNING CHAR(30)
    SELECT v_char
    INTO return_val
    FROM pack1_private_vars;
    RETURN return_val;
END FUNCTION ;

CREATE PROCEDURE pack1.set_v_char (new_val CHAR(30) )
    UPDATE pack1_private_vars
          SET v_char = new_val;
END PROCEDURE;

CREATE FUNCTION pack1.get_v_int (return_val INTEGER )
RETURNING INTEGER
    SELECT v_int
    INTO return_val
    FROM pack1_private_vars;
    RETURN return_val;
END FUNCTION ;

CREATE PROCEDURE pack1.set_v_int (new_val INTEGER )
    UPDATE pack1_private_vars
          SET v_int = new_val;
END PROCEDURE;

CREATE FUNCTION pack1.simple_func (vc1 INTEGER )
RETURNING DOUBLE PRECISION
    RETURN vc1;
END FUNCTION ;

CREATE PROCEDURE pack1.simple_proc (vc1 INTEGER )
    --|  vc2 int;
    DEFINE vc2 INTEGER;
    LET vc2 = NULL;
    LET vc2 = ROUND(pack1.simple_func(5));
    INSERT INTO tab1 VALUES (vc1,vc2,'2',NULL);
END PROCEDURE;

CREATE PROCEDURE pack1.initialization_block()
    DEFINE RETURN_VAL INTEGER;
    DEFINE RETURN_VAL1 CHAR(30);
    DEFINE RETURN_VAL2 DATETIME YEAR TO FRACTION (5);
    EXECUTE PROCEDURE pack1.set_v_int(ROUND(pack1.simple_func(1)));
    EXECUTE PROCEDURE pack1.set_v_date(DATETIME(2000-01-02-00.00.00.00000)
    YEAR TO FRACTION(5) );
    EXECUTE PROCEDURE pack1.set_v_char('from block');
    LET RETURN_VAL = NULL;
    LET RETURN_VAL = pack1.get_v_int(RETURN_VAL);
    LET RETURN_VAL1 = NULL;
    LET RETURN_VAL1 = pack1.get_v_char(RETURN_VAL1);
    LET RETURN_VAL2 = NULL;
    LET RETURN_VAL2 = pack1.get_v_date(RETURN_VAL2);
    INSERT INTO tab1 VALUES (3,RETURN_VAL,RETURN_VAL1,RETURN_VAL2);
    EXECUTE PROCEDURE pack1.simple_proc(30);
END PROCEDURE;

EXECUTE PROCEDURE pack1.initialization_block();

CREATE PROCEDURE pack1.init_pkg()
    EXECUTE PROCEDURE pack1.decl_pack1_private_vars();
    EXECUTE PROCEDURE pack1.initialization_block();
END PROCEDURE;

EXECUTE PROCEDURE pack1.simple_proc(30);
```

Package variables

Public and private package variables are converted to columns in global temporary tables, unless you are migrating to DB2 Viper II. In migrations to DB2 Viper II, private package variables are converted to columns in global temporary tables, but public package variables are converted to DB2 global variables.

The tables are not created if package variables do not exist. Given a package named *pk*, the following global temporary tables are used in the DB2 UDB and Informix Dynamic Server script, if *pk* contains package variables:

SESSION.PK_PUBLIC_VARS
Contains a column for each variable in the CREATE PACKAGE pk statement (specification). This is not applicable in migrations to DB2 Viper II. For an example, see “Example of migration to DB2 Viper II” on page 203.

SESSION.PK_PRIVATE_VARS
contains a column for each variable in the CREATE PACKAGE BODY pk statement (implementation).

IDS **PK_PUBLIC_VARS**
Contains a column for each variable in the CREATE PACKAGE pk statement (specification).

IDS **PK_PRIVATE_VARS**
Contains a column for each variable in the CREATE PACKAGE BODY pk statement (actual implementation).

A table is created only if the package contains the specified type of variable. The mechanism required to manage the variables is implemented using stored procedures, which are written to the DB2 UDB and Informix Dynamic Server script as needed:

pk.**decl_***pk***_public_vars()**

Creates the table for the variables in the specification of package *pk*. In migrations to DB2 Viper II, the stored procedure is not created. Instead the Oracle public package variables are translated to DB2 CREATE VARIABLE statements which are global variables. Each of these variables are accessible from within the package body by specifying the package name. For an example, see “Example of migration to DB2 Viper II” on page 203.

pk.**decl_***pk***_private_vars()**

Creates the table for the variables in the body of package *pk*.

IDS *pk*.**get_** *v* **(return_val type)**

The *return_val* is set to the current value of *v* in the appropriate table for package *pk*.s

pk.**get_***v* **(OUT return_val type)**

Sets the OUT argument *return_val* to the current value of *v* in the appropriate table for package *pk*.

IDS *pk*.**set_***v* **(new_val type)**

Sets the current value of *v* to the value of *new_val* in the appropriate table for package *pk*.

pk.**set_***v* **(IN new_val type)**

Sets the current value of *v* to the value of *new_val* in the appropriate table for package *pk*.

pk.**initialization_block()**

Contains the DB2 UDB and Informix Dynamic Server translation of the initialization block, if present, of the implementation of package *pk*.

pk.**init_pkg()**

Used in applications, this procedure calls the procedures pk.decl_pk_public_vars(), pk.decl_pk_private_vars(), and pk.initialization_block(), for those that exist in this package. In migrations to DB2 Viper II and higher, procedure pk.decl_pk_public_vars() is not generated.

Important: This procedure must be called by the application that uses this package before any of the items in the package can be used. Oracle does this implicitly when the package is first referenced, but this translation requires that it be done explicitly.

The procedures *pk*.decl_pk_public_vars(), *pk*.decl_pk_private_vars(), and *pk*.initialization_block() are called from within the DB2 UDB and Informix Dynamic Server translated script so that package procedures that refer to package variables compile correctly in DB2 UDB and Informix Dynamic Server, and so that any top level calls to these procedures and functions have the global temporary tables declared and initialized for them.

The procedure pk.init_pkg() is not called from the DB2 UDB and Informix Dynamic Server script. It is for application use only.

All references to package variables in the input script are translated using the procedures *pk*.get_*v*(type) and *pk*.set_*v*(type). References in applications must be manually translated.

Global temporary tables in DB2 UDB and Informix Dynamic Server are subject to transaction rollback. If package variables are used in a context where rollbacks may occur, manual intervention may be necessary to preserve the original behavior from Oracle.

Example of migration to DB2 Viper II

The following example shows how MTK inserts a corresponding CREATE VARIABLE statement, in migrations to DB2 Viper II, when it encounters a public package variable in the declaration section of the package.

```
CREATE TABLE GLOBALV_INSERT_TAB3 (ID INTEGER, c1 INTEGER);

CREATE OR REPLACE PACKAGE GV_PACKAGE1 IS
 myint integer:=33;
 PROCEDURE PROC1 (C1 IN OUT integer);
END GV_PACKAGE11;
/
CREATE OR REPLACE PACKAGE BODY GV_PACKAGE1 IS
 PROCEDURE PROC1 (C1 IN OUT INTEGER) IS
  BEGIN
   C1:=C1+100;
   INSERT INTO GLOBALV_INSERT_TAB3 VALUES (1, myint);
   INSERT INTO GLOBALV_INSERT_TAB3 VALUES (2, C1);
  END;
END GV_PACKAGE1;
/
   ----- is translated
to -----

CREATE TABLE GLOBALV_INSERT_TAB3(
    ID DECIMAL(31,0),
    c1 DECIMAL(31,0)
)!

CREATE SCHEMA GV_PACKAGE1!

CREATE VARIABLE GV_PACKAGE1.myint DECIMAL(31,0) DEFAULT (33)!

CREATE PROCEDURE GV_PACKAGE1.PROC1 ( INOUT C1 DECIMAL(31,0) )
LANGUAGE SQL

BEGIN

    DECLARE C11 DECIMAL(31,0);

    SET C1 = C1 + 100;

    INSERT INTO GLOBALV_INSERT_TAB3 VALUES (1,GV_PACKAGE1.myint);

    SET C11 = C1;

    INSERT INTO GLOBALV_INSERT_TAB3 VALUES (2,C11);

END!
```

Translation limitations

- Package variables having record types or collection types are not supported.
- In conversions to DB2 Viper II, LONG types, LOBs, XML, ARRAY, and structured data types are not supported.

System packages: DBMS_

Oracle comes with a set of packages, having names starting with DBMS_. The most commonly used is DBMS_OUTPUT, in particular its put_line procedure. The converter provides basic support for this procedure.

```
dbms_output.put_line('test');

   ----- is translated
to -----

CALL ORA.PUT_LINE('test')!
```

Translation limitations

Calls to other procedures in the DBMS_OUPUT package will be recognized but not translated. An error message stating that the procedure is not supported will be issued.

Another commonly used package is DBMS_SQL, which provides dynamic SQL functionality. Objects in this package are recognized, but as they are not supported an appropriate error message will be issued.

Objects in other system packages are not recognized. An error message is issued stating that a reference to an unknown object was found.

Migration strategies

A stub definition of ORA.PUT_LINE is provided in the:

- mtkdbmspack.db2 file, located in the product directory, for deployments to DB2 UDB.
- mtkdbmspack.ids file, located in the product directory, for deployments to Informix Dynamic Server.

The contents of the applicable file are loaded to DB2 UDB or Informix Dynamic Server during deployment. Since there is no equivalent feature to put_line in DB2 UDB or Informix Dynamic Server, you must provide the implementation for this procedure. You can use the strategies that are described in the comments of the file.

UTL_FILE package:

MTK provides limited support for the Oracle UTL_FILE package on DB2 LUW and Informix Dynamic Server.

DB2 and Informix Dynamic Server support UTL_FILE functionality through MTK user-defined functions (UDFs). The number and type of parameters of these UDFs may differ from the Oracle syntax.

Support for the UTL_FILE package is limited to conversions from Oracle 9.2. The following APIs are supported:

- FOPEN

- FCLOSE
- FRENAME
- FREMOVE
- PUT_LINE
- GET_LINE

Queries

The converter provides support for standard-conforming queries.

However, MTK does not support the translation of Oracle distributed queries into IBM Informix Dynamic Server queries and, in translations to Dynamic Server, does not currently support the following clauses and parameters:

- Optimizer hints
- Selecting from or using a snapshot or materialized view
- A PARTITION clause in a query_table_expression_clause
- A SUBPARTITION clause in a query_table_expression_clause
- A @dblink clause in a query_table_expression_clause
- A sample clause in a query_table_expression_clause
- A WITH clause in a query_table_expression_clause

  However, a WITH CHECK OPTION clause in a CREATE VIEW statement is supported.
- A hierarchical query clause
- The WAIT and NOWAIT keyword of a FOR UPDATE clause
- The NULLS subclass of an ORDER BY clause

The WAIT and NOWAIT keyword of a FOR UPDATE clause is also not supported in translations to DB2 UDB.

Renaming with SELECT and SUBQUERY

Some differences can occur in translations from Oracle to DB2 UDB and IBM Informix Dynamic Server.

In Oracle PL/SQL, a column reference that is qualified by a table or view only (and not a schema name) can resolve to a table in the FROM list in *any* schema. In DB2 UDB SQL and in IBM Informix Dynamic Server SQL , such a reference resolves only to the table in the FROM list in the *current* schema.

For example, assume that the current schema is alpha and that the following table definitions exist:

```
create table alpha.t (x int);
create table beta.t (x int);
```

The following works in Oracle PL/SQL (t.x resolves to beta.t.x):

```
select t.x from beta.t;
```

A direct translation gives an error in DB2 UDB SQL, because t.x resolves to alpha.t.x (since alpha is the current schema). This is an error since alpha.t is not in the from list. The problem is solved by qualifying such references with their schemas:

```
select beta.t.x from beta.t!
```

In Oracle PL/SQL, references to correlation names (in the select list or where clause, for example) can be qualified by the schema of the table they represent. In DB2 UDB SQL and in IBM Informix Dynamic Server SQL, references to correlation names *cannot* be qualified. Dropping the qualifier is not a sufficient solution because there might be more than one instance of the correlation name, as in the following example:

```
 select beta.tt.x from beta.t as tt, alpha.t as tt;
```

beta.tt.x resolves to beta.t.x, but in DB2 UDB SQL this would give an error because in the select list, tt.x cannot be qualified by a schema, since tt is a correlation name. If the qualifier beta from beta.tt.x is dropped, then the query in DB2 UDB SQL becomes ambiguous (because x is found in both beta.t and alpha.t). This problem is solved by mapping each schema correlation pair to a separate correlation name. The qualified correlation name in the select list is replaced with the new one, and the old correlation names in the from list are translated to the new ones. The above select statement will be translated as follows:

```
select tt.x from beta.t.x as tt, alpha.t as tt1!
```

Outer joins

The converter provides basic support for Oracle outer joins

In Oracle, outer joins are defined in the WHERE clause. In DB2 UDB they are defined in the FROM clause.

```
create table t1 (x int, y int);
create table t2 (y int, z int);

select * from t1, t2 where t1.y=t2.y(+) and t2.z(+)=10;

   ----- is translated
to -----

SELECT *
FROM T2,  T1 LEFT OUTER JOIN   T2 ON T2.Y = T1.Y
WHERE T2.Z = 10!
```

Translation limitations

In translations to DB2 UDB and IBM Informix Dynamic Server:

- Only the equality (=) operator is supported.
- The (+) operator cannot follow a complex expression, it can follow a column reference only.

Related concepts

DB2 UDB Version 8 - Join types

Statements

The converter accepts the top-level statements and clauses shown in the table below.

Top-level statements and clauses	Converter accepts statements when translating to DB2 UDB	Converter accepts statements when translating to IBM Informix Dynamic Server
ALTER TABLE —ADD and MODIFY (column only)	Yes	No

Top-level statements and clauses	Converter accepts statements when translating to DB2 UDB	Converter accepts statements when translating to IBM Informix Dynamic Server
ALTER SESSION — SET: NLS_DATE_FORMAT CURRENT_SCHEMA NLS_CURRENCY NLS_ISO_CURRENCY NLS_NUMERIC_CHARACTERS	Yes	No
BEGIN...END	Yes	No
CALL	Yes	Yes
COMMENT	Yes	No
CREATE FUNCTION	Yes	Yes
CREATE INDEX	Yes	Yes
CREATE PROCEDURE	Yes	Yes
CREATE SEQUENCE	Yes	Yes
CREATE SYNONYM	Yes	Yes
CREATE TABLE	Yes	Yes
CREATE TRIGGER	Yes	Yes
CREATE TYPE	Yes	No
CREATE VIEW	Yes	Yes
DECLARE...BEGIN...END	Yes	Yes
DELETE	Yes	Yes
DROP	Yes	Yes
END	Yes	No
MERGE	Yes	No
INSERT	Yes	Yes
SELECT	Yes	Yes
UPDATE	Yes	Yes

The following statements have unique characteristics, notations, and limitations that should be addressed during the conversion process:

Connect

The connect statement is a SQL-PLUS statement that is interpreted by the converter to determine the current default schema for the Oracle source code. It is used as the default until the next connect statement is encountered, which provides a new default schema.

The converter uses ″dba″ as the default Oracle schema prior to the first connect statement. The schema of the first created object is used as the default schema for the converter output. For example:

```
create table scott.t (x int); select * from t;
```

This statement gives the following output (with an error because the Oracle default schema is ″dba″ and dba.t is an unknown table).

```
CREATE TABLE T(X INTEGER )!
 --* [200011]"C:\connect.sql"(5:15)-(5:16)
Reference to unknown object: t
SELECT *
 FROM DBA.t!
```

This problem is solved when the connect command is included:

```
connect scott/tiger
 create table scott.t (x int);
 select * from t;

   ----- is translated
to -----

CREATE TABLE T(X INTEGER )! SELECT * FROM T!
```

CREATE FUNCTION

The Oracle SQL CREATE FUNCTION statement is translated to a CREATE FUNCTION statement in Informix Dynamic Server.

Here is an example of the translation from an Oracle CREATE FUNCTION statement to an Informix Dynamic Server CREATE FUNCTION statement.

Oracle	Informix Dynamic Server

`CREATE FUNCTION f100 RETURN INT AS`
`  i1 INT;`
`BEGIN`
`  SELECT COUNT(*) INTO i1 FROM dept;`
`  RETURN i1;`
`END f100;`
` SELECT f100 FROM DUAL;`
	`CREATE FUNCTION f100() RETURNING INT`
	`  DEFINE i1 INT;`
	`  SELECT COUNT(*) INTO i1 FROM dept;`
	`  RETURN i1;`
	`END FUNCTION;`

The following Oracle CREATE FUNCTION statement features are not supported by Informix Dynamic Server:

- Oracle PL/SQL PARALLEL_ENABLE option. Translated to PARALLELIZABLE in external routines only.
- Oracle PL/SQL DETERMINISTIC option. Although Informix Dynamic Server does support the NOT VARIANT clause, NOT VARIANT functions cannot contain any SQL statements. The Oracle DETERMINISTIC option does not have any such limitations.
- PIPELINED clause.
- AGGREGATE USING clause.

Important: If the column name and the parameter name are identical, Informix Dynamic Server and Oracle return different results.

In Oracle, the RETURN clause supports any data type that is supported by PL/SQL. However, Informix Dynamic Server has limited support per the following diagram:

Data type	C	Java	SPL
BLOB	X		X
BYTE	X	X	X
COLLECTION		X	
CLOB	X		X
LIST		X	

Data type	C	Java	SPL
MULTISET		X	
ROW		X	X
SET		X	X
SERIAL	X	X	X
SERIAL8	X	X	X
TEXT	X	X	X

CREATE PROCEDURE

The Oracle CREATE PROCEDURE statement is translated to a CREATE PROCEDURE statement in Informix Dynamic Server.

Consider the following restrictions when translating Oracle CREATE PROCEDURE statements to Informix Dynamic Server:

- The entire text of an Informix Dynamic Server SPL routine, including spaces and tabs, must not exceed 64 KB.
- The Oracle OR REPLACE statement is ignored in translations to Informix Dynamic Server (see the example below for more information).
- Oracle invoker rights versus definer rights [AUTHID {DEFINER | CURRENT_USER}] is not supported in Informix Dynamic Server.
- The Oracle [local declarations] are translated to Informix Dynamic Server DEFINE statements.
- Informix Dynamic Server supports positional and named notation parameters, but not mixed notation.
- Only Oracle IN parameters are supported. Although Informix Dynamic Server has IN, OUT, and INOUT parameters, OUT and INOUT must be statement local variables. The INOUT parameters are valid only for the life of the SQL statement.
- The NOCOPY compiler hint is not supported.
- The Oracle default values for parameters are supported except in the case of references.
- Overloading subprogram names is supported.
- Recursion is supported.

Important: If the column name and the parameter name are identical, Informix Dynamic Server and Oracle return different results.

Here is an example of the translation from an Oracle CREATE PROCEDURE statement to an Informix Dynamic Server CREATE PROCEDURE statement:

```
CREATE OR REPLACE PROCEDURE father (p1 IN INT) AS
local1 INT := 10;
PROCEDURE son(param1 INT) AS
i1 INT;
BEGIN
SELECT COUNT(*) INTO i1 FROM dept
where deptno = param1;
RETURN i1;
END son;
BEGIN
son(p1);
INSERT INTO dept VALUES(1, 'done', 'done');
END father;
/
```

```
   ----- is translated
to -----

CREATE PROCEDURE son (param1 INTEGER,
local1 INTEGER,
p1 INTEGER )
DEFINE i1 INTEGER;
LET i1 = NULL;
SELECT COUNT(*) INTO i1 FROM dept WHERE deptno = param1;
;
END PROCEDURE;

CREATE PROCEDURE father (p1 INTEGER )
EXECUTE PROCEDURE SON(p1 ,local1,p1);
INSERT INTO dept VALUES (1,'done','done');
END PROCEDURE;
```

CREATE TRIGGER

The Oracle CREATE TRIGGER statement is supported in conversions to Informix Dynamic Server with some restrictions.

The following Oracle syntax is not supported in conversions to Informix Dynamic Server:

- Database event and DDL event triggers
- The ON DATABASE clause
- The nested table clause in the dml_event_clause
- The PARENT clause in referencing_clause
- INSTEAD OF triggers

Other restrictions include:

- The OR REPLACE clause is ignored.
- Multiple triggers cannot be declared on the same event for the same table.

The Oracle trigger body always requires a PL/SQL block. This PL/SQL block can consist of a procedure call statement with or without valid PL/SQL syntax.

DECLARE

You can declare variables and constants in the declarative part of any Oracle PL/SQL block, subprogram, or package. However, because Informix Dynamic Server does not have a declarative part, all of the declarations in the Oracle PL/SQL DECLARE statement are translated to DEFINE statements with the Informix Dynamic Server.

Informix Dynamic Server does not have the CONSTANT keyword and variables are not assigned value using the DEFINE statement. Therefore the Oracle PL/SQL DECLARE statement is translated into two Informix Dynamic Server statements. One statement declares the variable and the other assigns value to the variable.

Informix Dynamic Server SPL local variables do not support Oracle PL/SQL declarations using DEFAULT. The translator explicitly initializes these variables using the LET statement.

MTK ignores the NOT NULL constraint.

The MTK converter and Informix Dynamic Server replace %TYPE references directly by the type that they refer to.

The %ROWTYPE attribute is translated to a list of Informix Dynamic Server variables that match the record fields. Aggregate assignment of %ROWTYPE variables is achieved by individual LET statements. The use of %ROWTYPE is only supported for table columns and cursors.

Oracle PL/SQL select-list item aliases for expressions are not supported because the Informix Dynamic Server SELECT statement for cursors cannot contain an expression in the select list.

The following table shows special cases of the Oracle DECLARE statement and the equivalent Informix Dynamic Server statements:

Oracle DECLARE statement	Informix Dynamic Server DEFINE statement

`DECLARE credit_limit CONSTANT INT := 1500;`	`DEFINE credit_limit INTEGER;`
	`LET credit_limit = 1500;`

`DECLARE credit_limit INT DEFAULT 1500;`	`DEFINE credit_limit INTEGER;`
	`LET credit_limit = 1500;`

`DECLARE`
`var1 int;`
`var2 var1%TYPE;`
	`DEFINE var1 INTEGER;`
	`DEFINE var2 INTEGER;`

`create table tab(c1 int, c2 int);`
`DECLARE z tab%ROWTYPE;`
	`CREATE TABLE tab(c1 INTEGER,c1 INTEGER);`
	`DEFINE z_c1 INTEGER;`
	`DEFINE z_c2 INTEGER;`

`create table tab(c1 int, c2 int);`
`DECLARE z tab%ROWTYPE;`
`DECLARE w tab%ROWTYPE; z := w;`
	`CREATE TABLE tab(c1 INTEGER,c1 INTEGER);`
	`DEFINE z_c1 INTEGER;`
	`DEFINE z_c2 INTEGER;`
	`DEFINE w_c1 INTEGER;`
	`DEFINE w_c2 INTEGER;`
	`SET z_c1 = w_c1; SET z_c2 = w_c2;`

DDL

This information covers converter behavior and limitations related to DDL statement conversion.

See Static DDL restrictions for special restrictions regarding DDL migration.

Implicit commit:

Oracle issues an implicit commit before and after each DDL statement, and DDL statements may appear only at the top-level. When DB2 UDB is run in auto-commit mode, a commit is issued after every top-level statement.

The converter assumes that the DB2 UDB script will be run in the auto-commit mode (During deployment, MTK runs the DB2 UDB scripts in auto-commit mode).

Related reference

"Transaction statements" on page 235
The transaction statements COMMIT, ROLLBACK, and SAVEPOINT are translated directly to DB2 UDB, with a few exceptions as noted here.

Defaults in columns:

In DB2 UDB SQL, only constants, date or time special registers, USER, NULL, and cast functions can be used as defaults.

When the Oracle default expression is converted, the result is checked to ensure that it meets the DB2 UDB default restrictions. Some manual modifications of the result might be required.

CREATE TABLE:

In a few situations, differences can occur in translations from Oracle to DB2 UDB and IBM Informix Dynamic Server.

In DB2 UDB, columns involving a primary key or unique constraint are required to have NOT NULL constraints over them. These constraint requirements do not exist in Oracle. Also, some complex check constraints are not allowed in DB2 UDB, including constraints containing sub-queries, column functions, and special registers.

Translation of primary key or unique constraints when converting to DB2 UDB

The conversion of a primary key or unique constraint might cause new NOT NULL constraints to be added to column definitions in the DB2 UDB output in order to satisfy the DB2 UDB requirement:

```
create table t10 (x int, y int not null, primary key (x,y));

   ----- is translated
to -----

CREATE TABLE T10(X INTEGER NOT NULL,Y INTEGER NOT NULL,PRIMARY KEY(X,Y) )!
```

Translation of complex column defaults when converting to DB2 UDB

The default clause and constraints, in general, are supported by the converter. DB2 UDB SQL does not allow complex column defaults (most expressions that are not literals). When translating these defaults, a special trigger is generated. When a row is inserted without specifying a value for the column, a null is inserted. The trigger then tests the value to be inserted; if the value is null, it is replaced with the default expression.

This translation works well if the column is not nullable. But if the column is nullable, the trigger cannot distinguish between a null value and an unspecified one and inserts the default value, which is not the expected behavior. Therefore, a warning message is issued during translation so that you can manually review the code. If you can ensure that no null values will be inserted for this column then the translation is fine. If this is not the case, a solution is to use a special value as the default for the column (a value that is not likely to be inserted), and to test the inserted value against this special value instead of using null.

Because of the way complex defaults are translated, the converter removes the NOT NULL constraint that might exist on the column. If this is the case, a manual code review is necessary.

Translation of complex check constraints when converting to DB2 UDB

When translating these kinds of constraints, a special trigger is generated. This trigger tests if the predicate of the constraint is true and raises an exception if not, preventing the insertion.

Note that the exception raised is an SQLSTATE 09000, which differs from the SQLSTATE 25513 raised when a real check constraint is not satisfied.

Translation of tables containing a "from dual" expression when converting to IBM Informix Dynamic Server

When you select a dual table for the first time, MTK generates these statements:

```
CREATE TABLE dual(dummy varchar)
INSERT into dual value('X')
```

Dual tables are created as user "informix" tables. For example, the Oracle SQL statement

```
SELECT 8/4 FROM dual
   ----- is translated
to -----
SELECT 8/4 FROM informix.dual
```

Syntax that MTK ignores or does not support when converting to IBM Informix Dynamic Server

IBM Informix Dynamic Server does not support the following statements or clauses:

- ON COMMIT DELETE/PRESERVE ROWS
- ON DELETE SET NULL
- Much of the physical_properties clause
- Much of the table_properties clause
- CREATE GLOBAL TEMPORARY TABLE (In Informix Dynamic Server, a temporary table is visible only to the user who created it and only for that session.)

In addition, in Informix Dynamic Server, the IN clause is only supported for a dbspace.

Related tasks

"Step 2: Converting source metadata" on page 28
The Convert step converts source metadata to target database metadata. You can specify various options that affect the converted output before you convert source SQL into DB2 UDB or Informix Dynamic Server SQL.

Related reference

"ALTER TABLE" on page 216
The converter translates ADD and MODIFY clauses of the statement.

"Temporary tables" on page 300
For translations to Informix Dynamic Server, MTK supports the DECLARE LOCAL TEMPORARY TABLE statement, not the DECLARE GLOBAL TEMPORARY TABLE statement.

Range partitioning:

MTK provides limited support of Oracle range partitioning (also known as table partitioning) in conversions from Oracle to DB2 9.1.

The range_values_clause and the built-in function PARTITION (<partition_name>) is supported. This feature is supported for DB2 9.1 and higher.

The following features are not supported:

- Table_partition_description.
- Range partition with dbspaces.

- MTK extraction for the range_values_clause; instead use the MTK input file option.

DB2 does not allow MAXVALUE to be followed by any other values and the range must be valid for each partition value. As a workaround, you can change the values in the ENDING clause as follows: (`MAXVALUE, MAXVALUE, MAXVALUE`).

Here is an example of a conversion from Oracle to DB2 9.1:

```
CREATE TABLE sales
 ( prod_id NUMBER(6)
 , cust_id INT
 , time_id DATE
 , channel_id CHAR(1)
 , promo_id NUMBER(6)
 , quantity_sold NUMBER(3)
 , amount_sold NUMBER(10,2)
)
 PARTITION BY RANGE (cust_id, prod_id)
 (PARTITION part1 VALUES LESS THAN (1, 100),
  PARTITION part2 VALUES LESS THAN (101, 200),
  PARTITION  part3 VALUES LESS THAN (MAXVALUE, MAXVALUE));

   ----- is translated
to -----

CREATE TABLE sales(
    prod_id INTEGER,
    cust_id INTEGER,
    time_id TIMESTAMP,
    channel_id CHAR(1),
    promo_id INTEGER,
    quantity_sold SMALLINT,
    amount_sold DECIMAL(10,2)
)
 PARTITION BY RANGE(
    cust_id,
    prod_id
)(
PARTITION part1 STARTING (MINVALUE, MINVALUE) EXCLUSIVE   ENDING (1, 100) INCLUSIVE,
PARTITION part2 STARTING (1, 100) EXCLUSIVE   ENDING (101, 200) INCLUSIVE,
PARTITION part3 STARTING (101, 200) EXCLUSIVE   ENDING (MAXVALUE, MAXVALUE) INCLUSIVE)!
```

Here is another example of a conversion from Oracle to DB2 9.1:

```
SELECT * FROM range_sales1 PARTITION(part1) ;

   ----- is translated
to -----

SELECT *
 FROM range_sales1 WHERE DATAPARTITIONNUM(prod_id) =
      ( SELECT SEQNO FROM SYSCAT.DATAPARTITIONS WHERE TABNAME = UPPER('range_sales1')
     AND  DATAPARTITIONNAME = UPPER('part1'))!
```

table_compression:

MTK provides limited support for data compression using the table_compression clause.

The table_compression clause is only supported in extractions from Oracle 10g and higher to DB2 9.1 and higher. If you are migrating from an earlier version of Oracle, use the input file option instead of the table_compression clause.

In conversions to DB2 9.1, the ALTER TABLE clause is supported and limited support is provided for CREATE TABLE statement. In the CREATE TABLE statement, the key_compression clause and partitioned tables with compression are not supported.

In DB2 9.1, the compression takes effect only after the table dictionary is built, which is usually during the table REORG phase.

In conversions to DB2 9.1, data compression is not supported for the following Oracle statements:

- ALTER MATERIALIZED VIEW
- CREATE TABLESPACE

Here is an example of a conversion from Oracle 10g to DB2 9.1:

```
ALTER TABLE Sales MOVE COMPRESS;

   ----- is translated
to -----

ALTER TABLE Sales COMPRESS YES
REORG TABLE Sales
```

The TABLESPACE clause:

When evaluating a CREATE TABLE statement, the translator will ignore any physical and table properties specified, without issuing a warning message.

The properties include:

- ORGANIZATION HEAP and INDEX
- CLUSTER
- LOB storage clause
- VARRAY storage clause
- NESTED TABLE storage clause
- LOGGING and NOLOGGING
- PCTFREE and PCTUSED
- INITRANS
- MAXTRANS
- STORAGE
- PARTITION clause
- CACHE and NOCACHE
- MONITORING and NOMONITORING
- PARALLEL and NOPARALLEL
- ENABLE and DISABLE

The TABLESPACE clause used in a CREATE INDEX statement is not translated. In such occurrences, an ″ignored input″ message will be issued.

Migration strategies

By selecting the appropriate advanced option on the Convert page of MTK, you can have the TABLESPACE clause translated as is, and a message is given to remind you to add the desired properties as necessary to work properly in the DB2 UDB environment.

```
create table t (x int)
tablespace ts;

   ----- is translated
to -----

CREATE REGULAR TABLESPACE TS
MANAGED BY SYSTEM
USING ('TS')!

CREATE TABLE T(
    X INTEGER
)
IN TS!
```

Related tasks

"Step 2: Converting source metadata" on page 28
The Convert step converts source metadata to target database metadata. You can specify various options that affect the converted output before you convert source SQL into DB2 UDB or Informix Dynamic Server SQL.

CREATE TABLE AS SUBQUERY:

In translations to DB2 UDB, this statement is converted to a CREATE TABLE statement followed by an INSERT statement that uses the sub-query to load the data into the table.

ALTER TABLE:

The converter translates ADD and MODIFY clauses of the statement.

The ADD clause, which includes column definitions and table constraints, is converted directly to the corresponding target ALTER TABLE statement.

In the MODIFY column clause, data types and default values are converted by making changes to the corresponding CREATE TABLE statement for the table being altered. For example, for a conversion to UDB DB2:

```
CREATE TABLE T1
 (id NUMBER NOT NULL,
  creation_date DATE NOT NULL
 );
ALTER TABLE T1 MODIFY
 (creation_date DEFAULT SYSDATE,
  id NUMBER(30)
 );

   ----- is translated
to -----

CREATE TABLE T1(
 ID DECIMAL(30,0) NOT NULL,
 CREATION_DATE TIMESTAMP DEFAULT CURRENT TIMESTAMP NOT NULL
)!
```

Translation limitations

- ADD and MODIFY are the only ALTER TABLE clauses supported by the converter.
- Any references made to the table between the CREATE TABLE and the ALTER TABLE commands in the source script might cause problems.

- Adding a primary key or unique constraint in an ALTER TABLE statement may cause a NOT NULL constraint to be added to a column definition in a previous CREATE TABLE statement.

  Related reference

  "CREATE TABLE" on page 212
  In a few situations, differences can occur in translations from Oracle to DB2 UDB and IBM Informix Dynamic Server.

CREATE INDEX:

Most indexes are translated.

In DB2 UDB, no more than one index with its particular set of properties can be specified per column. Oracle allows duplicate indexes that have the same properties to be defined as long as the names of the indexes are different.

The converter issues a warning only when it encounters an index that has the same name as a previously defined index. If you attempt to deploy differently named indexes that have the same properties to DB2 UDB, only one of the indexes is deployed, causing any references to the undefined indexes to fail.

Syntax that MTK ignores or does not support when converting to IBM Informix Dynamic Server

IBM Informix Dynamic Server does not support the following statements or clauses:
- cluster_index clause
- local_index clause
- domain_index_clause
- BITMAP clause (as in CREATE BITMAP INDEX)
- The following clauses are partially supported in translations to Dynamic Server:
- global_index clause
- index_attributes clause

The following example of creating an index on a global partition is not supported because IBM Informix Dynamic Server does not have a concept of MAXVALUE:

```
CREATE INDEX idx6 ON tab1(col1)
GLOBAL PARTITION BY RANGE (col1)
(PARTITION sx1992 VALUES LESS THAN (1) TABLESPACE ts0,
PARTITION sx1993 VALUES LESS THAN (MAXVALUE) TABLESPACE ts1);
```

CREATE SYNONYM:

The CREATE SYNONYM statement is fully supported.

Translations to DB2 UDB occur as follows:
- To simulate Oracle name-scoping rules, public synonyms are translated to the PUBLIC schema of DB2 UDB. All references to the translated synonyms are qualified by the schema name PUBLIC.
- Synonyms for tables, views and similar objects are translated using DB2 UDB aliases.
- For sequence, procedure, function, and package objects, it is not possible to use DB2 UDB aliases. The converter therefore translates any references to such synonyms to references to the actual objects.

Here is an example of how MTK converts a CREATE SYNONYM into a DB2 UDB statement:

```
create table t(i int);
create public synonym s1 for t;
select i from s1;

create procedure p is
begin null; end;
/
create synonym s2 for p;
call s2();

   ----- is translated
to -----

CREATE TABLE T(I INTEGER)!
CREATE ALIAS PUBLIC.S1 FOR T!
SELECT I FROM PUBLIC.S1!

CREATE PROCEDURE P() LANGUAGE SQL
BEGIN END!
CALL P()!
```

Translations to IBM Informix Dynamic Server occur as follows:

- If source CREATE SYNONYM are not qualified with the schema name PUBLIC, translated statements are qualified with the schema name PRIVATE.
- Dblink is not supported.
- Package, materialized view, and Java class schema object synonyms are not supported.

The following examples show how CREATE SYNONYM statements are translated into similar Dynamic Server statements:

```
CREATE SYNONYM s1.syn1 FOR tab1;
   ----- is translated
to -----

CREATE PRIVATE SYNONYM s1.syn1 FOR tab1;
```

```
CREATE PUBLIC SYNONYM s1.syn2 FOR view1;
   ----- is translated
to -----

CREATE PUBLIC SYNONYM s1.syn2 FOR view1;
```

CREATE VIEW:

MTK generally translates views to the target database. However, certain clauses or views are not supported in translations to IBM Informix Dynamic Server.

The following clauses or views are not supported in translations to IBM Informix Dynamic Server:

- OR REPLACE clause
- NO FORCE clause
- READ ONLY clause
- CONSTRAINT clause with CHECK OPTION
- Object views
- inline_constraint
- out_of_line_constraint

- object_view_clause
- XML Type_view_clause
- READ ONLY option of the subquery_restriction_clause
- CONSTRAINT option with the subquery_restriction_clause

In migrations from Oracle to Informix Dynamic Server, the deployment of VIEW fails when the CREATE VIEW statement involves column expressions in the SELECT statements.

CREATE TYPE:

MTK generally translates CREATE TYPE statements. Some restrictions occur in translations to IBM Informix Dynamic Server.

In translations to Dynamic Server, the following clauses of the CREATE TYPE statement for object types are not supported or ignored:
- REPLACE clause
- invoker_rights_clause
- procedure_spec and function_spec (for specifying procedure or function specifications in the type)
- pragma_clause
- REPLACE clause for varray types
- REPLACE CLAUSE for nested tables
- ON COMMIT DELETE/PRESERVE ROWS
- physical_properties clause
- table_properties clause
- OID_clause
- OID_index_clause

DROP statements:

DROP statements are translated for the following objects: FUNCTION, INDEX, PROCEDURE, SEQUENCE, ALIAS, TABLE, TRIGGER, and VIEW.

Translation limitations

Limitations in translation to DB2 UDB are:
- The converter allows DROP statements for objects that have not yet been defined. An occurrence of this causes DB2 UDB to issue a message during deployment.
- When a DROP statement is encountered, the converter does not actually remove the declaration of the object, so if the object is declared again after the DROP, the converter issues an error.
- The translation of a DROP TRIGGER can result in several DROP TRIGGER statements in DB2 UDB, because the Oracle converter sometimes produces multiple triggers for a single Oracle trigger.

Limitations in translation to IBM Informix Dynamic Server are:
- MTK ignores the FORCE clause in a DROP INDEX statement.
- MTK ignores the PUBLIC clause in a DROP SYNONYM statement.

DML statements

This information covers converter behavior and limitations specifically related to DML statements.

INSERT:

For translations to DB2 UDB, MTK converts INSERT statements into base tables and views only. The converter does not support INSERT statements into sub-queries or expressions that return nested tables or arrays.

In DB2 UDB, variables used in INSERT statements cannot have the same name as a column in the table inserted into. The converter therefore uses a different variable name.

```
create table t (c int);
create procedure p
as
  c int := 10;
begin
  insert into t values (c);
end;

   ----- is translated
to -----

CREATE TABLE T(C INTEGER)!
CREATE PROCEDURE P()
LANGUAGE SQL
BEGIN
 DECLARE C INTEGER DEFAULT 10;
 DECLARE C1 INTEGER;
   SET C1 = C;
   INSERT INTO T VALUES (C1);
END!
```

Oracle INSERT statements are generally translated into IBM Informix Dynamic Server INSERT statements, with the translation limitations noted below.

Translation limitations

The converter does not support INSERT statements that involve literal values being inserted into a column with a DB2 UDB binary type (BLOB or CHAR FOR BIT DATA). Other forms of column inserts are supported.

MTK converts INSERT statements to IBM Informix Dynamic Server with the following limitations:

- The insertion of literal values into Dynamic Server BLOB, CLOB, BYTE and TEXT type columns is not supported.
- INSERT statements involving user-defined types, REFs, and nested tables are not supported.
- The following clauses or parts of clauses, as defined in Oracle 9i SQL grammar, are not supported:
  - returning_clause
  - hint
  - subquery
    - ORDER BY
    - UNION
    - FIRST

- the following parts of a DML_table_expression_clause:
  - Subquery
  - Table_collection_expression
  - PARTITION
  - SUBPARTITION
- Values_clause subquery (The values_clause can have a subquery in Oracle 8i, not Oracle 9i.)
- Multi_table_insert

When a table name is followed by an alias, MTK ignores the alias name. An example of an INSERT statement with an alias is:

```
INSERT into employee e VALUES('Mike'),
```

In INSERT statements, MTK converts Oracle TIMESTAMP literals into DATETIME(<literal>) YEAR TO FRACTION(5) literals in the following GL_DATETIME format:

```
GL_DATETIME="%Y-%m-%d-%H.%M.%S%iF5"
```

UPDATE:

MTK converts UPDATE statements into DB2 UDB base tables and views only.

Translation limitations

MTK converts UPDATE statements to DB2 UDB with the following limitations:

- The converter does not support the conversion of DB2 UDB UPDATE statements into sub-queries or expressions that return nested tables or arrays.
- The RETURNING clause is not converted.

MTK converts UPDATE statements to IBM Informix Dynamic Server with the following limitations:

- Informix Dynamic Server does not support updating a table queried in the same UPDATE statement.
- Informix Dynamic Server does not support UPDATE statements with aggregate functions in the SET clause.
- The following clauses or parts of clauses, as defined by the Oracle 9i SQL, grammar are not supported:
  - returning_clause
  - hints, which are ignored
  - update_set_clauses that contain VALUE
  - the following parts of the dml_table_expression clause:
    - subquery
    - table_collection_expression
    - PARTITION
    - SUBPARTITION

In a translation to Informix Dynamic Server:

```
UPDATE ng_t1 /*+ ALL_ROWS */  SET c1 = c1 * 100 WHERE c1 = 1;
   ----- is translated
to -----

UPDATE ng_t1 SET c1 = c1 * 100 WHERE c1 = 1;
```

DELETE:

For conversions to DB2 UDB, MTK converts DELETE statements into base tables and views only.

Translation limitations

MTK converts DELETE statements to DB2 UDB with the following limitations:
- The converter does not support DELETE statements into sub-queries or expressions that return nested tables or arrays.
- The RETURNING clause is not converted.

MTK supports the exact Oracle syntax for deleting all rows in a table.

In translations to Informix Dynamic Server, the following clauses or parts of clauses, as defined by the Oracle 9i SQL, grammar are not supported:
- returning_clause
- hints, which are ignored
- the following parts of the dml_table_expression clause:
  - subquery
  - table_collection_expression
  - PARTITION
  - SUBPARTITION

GOTO

The Oracle GOTO statement is not supported in translations to Informix Dynamic Server. However, it is supported in translations to DB2 UDB.

If the GOTO statement is passed to MTK, the message MTKO0269 is issued stating that this is not supported. It is your responsibility to ensure that the translated stored procedure semantically matches with the source database.

PL/SQL

This information highlights converter behavior and limitations for statements relating to PL/SQL conversion.

Control flow:

This section highlights PL/SQL statements that control the flow, or order in which, statements are executed.

These structures can implement the following statements:
- Iteration (LOOP, WHILE, FOR)
- Selection (IF-THEN-ELSIF-END IF, CASE)
- Sequence (EXIT, GOTO)

  Related reference

  "Cursor support in conversions to DB2 UDB" on page 169
  The converter supports a limited number of features concerning Oracle cursors in conversions to DB2 UDB.

Range FOR:

The Oracle range FOR loop does not exist in DB2 UDB or Informix Dynamic Server and is converted using a WHILE loop.

```
 create table t(x int);

BEGIN
 for i in 1..3 loop
 insert into t values(i);
 end loop;
 END;

   ----- is translated
to -----

CREATE TABLE T(
    X INTEGER
)!

BEGIN ATOMIC
    DECLARE I INTEGER DEFAULT 1;
    WHILE I <= 3 DO
        INSERT INTO T  VALUES (I);
        SET I = I + 1;
    END WHILE ;
```

In this example, the index variable "i" is initialized in the declaration using DEFAULT, and a statement is added at the end of the while loop to be incremented. DB2 UDB and Informix Dynamic Server require explicit declaration of index variables, while Oracle index variables are implicit. Naming conflicts that might occur as a result of index variable declaration inconsistencies are handled by renaming.

EXIT:

The Oracle EXIT statement is used to exit a loop and is similar to the DB2 UDB LEAVE statement and the Informix Dynamic Server EXIT statement, but differences exist.

The DB2 UDB LEAVE statement requires an explicit label to specify which loop must be exited (which is useful in the case of nested loops).

The Informix Dynamic Server EXIT statement requires the type of loop to be specified along with the EXIT statement. For example, EXIT WHILE and EXIT FOR.

Explicit labels can be used with Oracle EXIT statements, but they are not required when the closest enclosing loop is the default target. The converter uses the closest loop's label (creating one if it doesn't already exist) when converting an EXIT statement where the label is not explicitly defined:

```
 begin
  for i in 1..10 loop
   exit;
  end loop;
 end;

   ----- is translated
to -----

BEGIN ATOMIC
    DECLARE I INTEGER DEFAULT 1;
    LOOP_LABEL:
    WHILE I <= 10 DO
```

```
        LEAVE LOOP_LABEL;
        SET I = I + 1;
    END WHILE LOOP_LABEL;
END!
```

In this example the range for loop is translated to a while loop. In Oracle, the range for loop is not labeled but the converter adds a label to the while loop in DB2 UDB so that it can translate the EXIT statement properly.

The Oracle EXIT statement has an optional WHEN clause to specify a condition for the EXIT statement. There is no such clause for the DB2 UDB LEAVE statement or the Informix Dynamic Server EXIT statement. The converter will generate an IF statement as follows to compensate for this difference:

```
 declare
   i int := 1;
 begin
  <<while_loop>>
  while i < 100 loop
    exit when i>4;
    i := i+1;
  end loop while_loop;
 end;

   ----- is translated
to -----

BEGIN ATOMIC
    DECLARE I INTEGER DEFAULT 1;
    WHILE_LOOP:
    WHILE I < 100 DO
        IF I > 4 THEN
            LEAVE WHILE_LOOP;
       END IF;
        SET I = I + 1;
    END WHILE WHILE_LOOP;

END!
```

Labels:

Labels, as applied to statements and variable prefixes, differ between DB2 UDB and Oracle.

Labels are not supported in translations to Informix Dynamic Server. If a label is passed to MTK, the message MTKO0270 is issued stating that this is not supported. It is your responsibility to ensure that the translated stored procedure semantically matches with the source database.

In general, statements are converted as follows:

```
 create procedure p
 as
  i int := 1;
 begin
  <<label>>
  declare
   i int := 1;
  begin
 p.i := label.i+1;
 end;
  if i > 10 then
   goto label;
  end if;
```

```
end;

   ----- is translated
to -----

CREATE PROCEDURE P()
LANGUAGE SQL
BEGIN
    DECLARE I INTEGER DEFAULT 1;
    LABEL:
    BEGIN
        DECLARE I INTEGER DEFAULT 1;
        SET P.I = LABEL.I + 1;
     END LABEL;
    IF I > 10 THEN
         GOTO LABEL;
    END IF;
END!
```

Translation limitations

The following differences between DB2 UDB and Oracle label implementation should be noted:

- Scope differences can cause the converter to automatically rename some labels in DB2 UDB. If this happens, a warning message appears.
- Name collision for labels is supported in Oracle as long as the colliding labels are not referenced (the ″GOTO label″ cannot be used because it would cause ambiguity). In DB2 UDB, labels must be unique.
- Labeling of top-level blocks is supported in Oracle, but not in DB2 UDB. The following top-level block label would not convert properly and a warning would be issued:

  ```
  <<label>>
  BEGIN
   NULL;
  END;
  /
  ```
- In general, labels are not permitted inside DB2 UDB dynamic compound statements. The converter skips these labels and issues a warning message. Labels are permitted in FOR loops so that the LEAVE statement can be supported. See “DB2 UDB dynamic compound statements” on page 235 for more information.

Migration strategies

The following example provides converter output for a statement that uses ″label″ twice. The converter would return an input error message under these circumstances. To avoid such errors you should modify the input by renaming the second label to something unique.

```
create procedure p
as
   i int;
begin
   <<label>>
   i := 0;
   <<label>>
   i := 1;
end;

   ----- is translated
to -----
```

```
CREATE PROCEDURE P()
LANGUAGE SQL
BEGIN
    DECLARE I INTEGER;
    LABEL:
    SET I = 0;

--* [200016]"test.sql"(7:2)-(8:9)Duplicate definition of label

     LABEL:
     SET I = 1;

END!
```

Related reference

"DB2 UDB dynamic compound statements" on page 235
A DB2 UDB dynamic compound statement is a block of statements (between BEGIN and END) that is used for top-level anonymous blocks, user-defined functions bodies, and trigger bodies.

Triggers:

There are significant differences between Oracle triggers and DB2 UDB and Informix Dynamic Server triggers.

Translation limitations

Oracle and DB2 UDB triggers differ in many ways, for example, the body of a DB2 UDB trigger can only consist of a dynamic compound statement.

Oracle In conversions from Oracle to DB2 UDB:

- INSTEAD OF triggers are translated only if DB2 UDB Version 8 is the specified target.
- For each view or event, only one INSTEAD OF trigger is translated because DB2 UDB does not allow more than one to be declared for each.

IDS In conversions from Oracle to Informix Dynamic Server:

- Exceptions that are defined inside of triggers are not supported.
- The Oracle AFTER with FOR EACH clause is not supported.

BEFORE triggers:

BEFORE triggers are usually translated to NO CASCADE BEFORE triggers in DB2 UDB.

NO CASCADE triggers cannot fire other triggers. NO CASCADE triggers also do not allow DML statements (INSERT, UPDATE and DELETE) and the FOR EACH STATEMENT clause. If the source trigger uses one of these features, then the translator will try to translate the BEFORE trigger to an AFTER trigger instead, which doesn't have the same restrictions. However, such a translation is not possible if the body of the trigger refers to the table as the object of the trigger, or if a column of the NEW table is assigned.

BEFORE triggers without a FOR EACH ROW clause are not translated because there is no corresponding feature in DB2 UDB.

Multiple trigger events:

Oracle triggers have the capability of specifying several trigger events (ON INSERT OR DELETE) on one trigger. The keywords 'inserting', 'updating' or 'deleting' are used inside the body to produce different results, depending on the event that has been fired. DB2 UDB triggers support only one event per trigger

MTK translates trigger events as follows:

- The translated contents of the Oracle trigger is copied to separate DB2 UDB and Informix Dynamic Server triggers, one for each specified event type.
- The 'inserting', 'updating', and 'deleting' keywords are handled using three integer variables, also called INSERTING, UPDATING, and DELETING. These variables are set to `1` (true) or `0` (false), depending on the event of the trigger.
- Within the translated trigger body, these variables are compared to the value `1` to test which case is relevant.

```
create trigger t after insert or delete on t
 declare
  i int;
 begin
  if (inserting) then i:=2;
  else i:=3; end if;
 end;

   ----- is translated
to -----

CREATE TRIGGER T_FOR_INSERT AFTER INSERT ON T
 FOR EACH STATEMENT MODE DB2SQL
 BEGIN ATOMIC
  DECLARE I INTEGER;
  DECLARE INSERTING INTEGER DEFAULT 1;
  DECLARE UPDATING INTEGER DEFAULT 0;
  DECLARE DELETING INTEGER DEFAULT 0;
  IF (INSERTING = 1) THEN
   SET I = 2;
  ELSE
   SET I = 3;
  END IF;
 END!

 CREATE TRIGGER T_FOR_DELETE AFTER DELETE ON T
 FOR EACH STATEMENT MODE DB2SQL
 BEGIN ATOMIC
  DECLARE I INTEGER;
  DECLARE INSERTING INTEGER DEFAULT 0;
  DECLARE UPDATING INTEGER DEFAULT 0;
  DECLARE DELETING INTEGER DEFAULT 1;
  IF (INSERTING = 1) THEN
   SET I = 2;
  ELSE
   SET I = 3;
  END IF;
 END!
```

Procedure calls within triggers:

Triggers in DB2 UDB Version 8.1 Fix Pack 6 and earlier cannot contain procedure calls; therefore, for deployments to UDB 8.1, procedure calls within an Oracle trigger are converted by copying the translated contents of the procedure body into the trigger. Any parameters are declared as local variables and assigned at the beginning and end of the trigger to maintain the correct behavior.

Translation limitations

Since a DB2 UDB trigger can only consist of a dynamic compound statement, the converted contents might require manual editing and conversion (for example, if the trigger contains nested blocks, loops, or RETURN statements).

Functions:

Oracle supports user-defined functions (UDFs) with output or input/output parameters (with the OUT or INOUT keywords). These UDFs can only be called inside a function, procedure, or trigger body and not from the top-level. UDFs in DB2 UDB can only contain input parameters.

The converter translates Oracle functions to DB2 UDB functions unless the Oracle function uses a feature that is not supported inside a DB2 UDB function, such as DML statements, exception blocks, cursors, or OUT parameters. If these features are present in the function, then it is translated to a procedure.

```
 create function f(x IN INT, y OUT INT)
 return INT
 is
 begin
    y := x;
    return x;
 end;

   ----- is translated
to -----


CREATE PROCEDURE F ( IN X INTEGER,
                     OUT Y INTEGER,
                     OUT RETURN_VAL INTEGER )
LANGUAGE SQL

BEGIN
    SET Y = X;
    SET RETURN_VAL = X;
END!
```

In the conversion, the return value of the function is converted to an output parameter (named ″RETURN_VAL″) and a return of 0 is added to exit the procedure. If the RETURN statement occurs inside an exception handler, it is translated to a RESIGNAL statement that refers to a generated empty exit handler. For these conversions, calls to routines are modified and the statements are arranged so that an Oracle function call becomes a DB2 UDB procedure call.

The following example of statements located inside a procedure body illustrates the converter output with the Oracle function F defined in the previous example:

```
 a int default 1;
 b int;
 a := f(a, b);

   ----- is translated
to -----

DECLARE A INTEGER DEFAULT 1;
DECLARE B INTEGER;
DECLARE RETURN_VAL INTEGER;
CALL F(A,B,RETURN_VAL);
SET A = RETURN_VAL;
```

Translation limitations

Language structures such as IF, THEN, ELSE, and WHILE are too complex to be rearranged as in the previous example. Conversion of these structures will result in an error message:

```
while (f(a,b)!=5) loop
   a := a+1;
   insert into t values (b);
end loop;

--* [400096]"m_function_with_OUT_params.sql"(29:11)-(29:17)
--*        Call to function with OUT parameter too complex
--*        to be translated
WHILE ((Untranslated Expression) <> 5) DO
   SET A = A + 1;
   INSERT INTO T VALUES (B);
END WHILE ;
```

Procedures:

Procedure translation is supported by the converter.

A general procedure is converted as follows:

```
 CREATE PROCEDURE P(x IN INT)
 AS
 BEGIN
  INSERT INTO T VALUES(x);
 END;

   ----- is translated
to -----

CREATE PROCEDURE P ( IN X INTEGER )
LANGUAGE SQL
BEGIN
     INSERT INTO T  VALUES (X);
END!
```

Overloading is supported; however, in some cases DB2 UDB will require that the procedure be renamed.

In DB2 UDB, all arguments passed to the procedure must be variables. The following example illustrates how the converter might handle a situation where something other than a variable is passed to a procedure in PL/SQL. A simple procedure, p, inserts a value in a table. When that procedure is called from within another procedure, p2, with an expression as a parameter (10 * x), then the resulting DB2 UDB code will first evaluate and assign the expression to a variable before passing it to the procedure (SET X1 = 10 * x).

```
create procedure p(x in int)
as
begin
  insert into t values (x);
end;

create procedure p2(x in int)
as
begin
  p(10 * x);
end;

   ----- is translated
to -----
```

```
CREATE TABLE T(C INTEGER)!

CREATE PROCEDURE P ( IN X INTEGER )
LANGUAGE SQL
BEGIN
  INSERT INTO T VALUES (X);
END!

CREATE PROCEDURE P2 ( IN X INTEGER )
LANGUAGE SQL
BEGIN
  DECLARE X1 INTEGER;
  SET X1 = 10 * X;
  CALL P(X1);
END!
```

Procedure call resolution

DB2 UDB does not support default values for procedures and function parameters. Default values for parameters are not translated in the parameter list, but the converter remembers them and adds them to each procedure call when the corresponding argument is missing. The following sample converter output shows a typical conversion involving a parameter with a default value:

```
CREATE OR REPLACE PROCEDURE P(x IN INT, y IN INT DEFAULT 0)
AS
BEGIN
  INSERT INTO T VALUES(x,y);
END;

   ----- is translated
to -----

CREATE PROCEDURE P ( IN X INTEGER,
                     IN Y INTEGER )
LANGUAGE SQL
BEGIN
     INSERT INTO T  VALUES (X,Y);
END!

--| call p(1);

CALL P(1,0)!
```

In this example, y is an optional parameter with a default value of 0. In the call to P, a value for y is missing, so the converter adds the default value for that parameter, 0, to the argument list.

The passing of parameters in a function or procedure call is handled differently in Oracle and DB2 UDB. Oracle has three possible ways of passing parameters in a function or procedure call:

- positional
- named
- mixed

DB2 UDB only supports the passing of positional parameters. However, all three methods of Oracle parameter passing are supported by the converter by putting the arguments in the expected order and filling in default values for missing arguments. The following sample converter output shows how the converter orders parameters that are named according to the definition of the procedure:

```
CREATE OR REPLACE PROCEDURE P(x IN INT, y IN INT)
AS
BEGIN
  INSERT INTO T VALUES(x,y);
END;

   ----- is translated
to -----

CREATE PROCEDURE P ( IN X INTEGER,
                     IN Y INTEGER )
LANGUAGE SQL

BEGIN
     INSERT INTO T  VALUES (X,Y);
END!
```

A procedure call using named parameters in Oracle is converted as follows:

```
call p(y=>1,x=>2);

   ----- is translated
to -----

CALL P(2,1)!
```

Translation limitations

- The replacement of procedures is not supported. Any statement with ″REPLACE″ is handled by the converter as follows:

  ```
  create or replace procedure p...

     ----- is translated
  to -----

  CREATE PROCEDURE P...
  ```

- Local procedures are procedures nested in the declaration section of a block. These procedures are converted but the generated code is commented out as there is no equivalent feature in DB2 UDB. See Migration strategies below for an example of how you might handle this case.

Migration strategies

Calls to local procedures are converted and you can move the code that is commented out outside of the block as the following sample converter output demonstrates.

```
create procedure p
as
  procedure local_proc
    as
    BEGIN
      insert into t values (1);
    END;
  BEGIN
    local_proc();
END;

   ----- is translated
to -----

CREATE PROCEDURE P()
LANGUAGE SQL
BEGIN
```

```
--* [500005]"test.sql"(3:2)-(7:6)This feature is not yet translated

     CREATE PROCEDURE LOCAL_PROC()
     LANGUAGE SQL
     BEGIN
          INSERT INTO t  VALUES (1);
     END;
    CALL LOCAL_PROC;
END!
```

In this example, ″local_proc″ is a local procedure. It is converted but commented out. To make the procedure call work properly, remove it and place it before the ″CREATE PROCEDURE P″ statement so that the call to this procedure (″call local_proc″ in DB2 UDB) will work. More changes may be necessary if the procedure name conflicts or if the procedure references a variable declared in an enclosing block.

Collection types:

Collection types are supported for local variables and parameters.

A local variable declaration is translated to a DB2 UDB DECLARE GLOBAL TEMPORARY table in the body of the procedure (this may move the declaration further down in the translated script).

For a procedure or function containing parameters of collection type, an addition procedure is declared before the translation of the original. If the procedure is named ″p″, the additional procedure is named ″declare_p_collection_tables″. This procedure declares a separate DB2 UDB DECLARE GLOBAL TEMPORARY TABLE for each collection-type parameter. This procedure must be called before each invocation of the corresponding original procedure or function (if the procedure is called more than once, the table is dropped and recreated by DB2 UDB).

For every IN parameter, an application calling the procedure must then fill the parameter's global temporary table with appropriate values for that collection. Then the application can call the DB2 UDB translation of the original procedure or function. After the call, for every OUT parameter, the application calling parameter must either copy the values from the global temporary table into it's own corresponding collection variable, or it may use the global temporary table directly in the application (at least until the corresponding procedure is called again). Note that for other stored procedures calling this original procedure or function, this entire process is incorporated automatically by MTK.

The temporary tables for the variables and parameters have the following columns:

- INDEX of type integer
- VALUE of the type of the collection's element type

Each element in the collection is represented by a row in the global temporary table. The INDEX column acts as a primary key for the table and is used to simulate collection access expressions in Oracle (for example, the expression ″x(10),″ which refers to the 10th element of collection x.

```
create type charArrayTyp is table of varchar2(10);
/
create  procedure get_employees(
        dept_name   in     charArrayTyp,
        emp_name    out    charArrayTyp
) is
    subtype INTEGER1 IS NUMBER(38,0);
    TYPE newList IS TABLE OF INTEGER1;
    subtype newList1 is newList;
    newArr newList;
    newArr1 newList1;
begin
    insert into t1 values (dept_name(1),10);
    emp_name := charArrayTyp('jim','joe','jon');
    emp_name(2) := 'jen';
end;
/

   ----- is translated
to -----

CREATE PROCEDURE declare_get_employees_collection_tables()
LANGUAGE SQL
BEGIN
    DECLARE GLOBAL TEMPORARY TABLE dept_name(
        index INTEGER,
        value VARCHAR(10)
    ) WITH REPLACE ON COMMIT PRESERVE ROWS NOT LOGGED;

    DECLARE GLOBAL TEMPORARY TABLE emp_name(
        index INTEGER,
        value VARCHAR(10)
    ) WITH REPLACE ON COMMIT PRESERVE ROWS NOT LOGGED;
END!

CALL declare_get_employees_collection_tables()!

CREATE PROCEDURE get_employees ( )
LANGUAGE SQL
BEGIN
    DECLARE GLOBAL TEMPORARY TABLE newArr(
        index INTEGER,
        value DECIMAL(31,0)
    ) WITH REPLACE ON COMMIT PRESERVE ROWS NOT LOGGED;

    DECLARE GLOBAL TEMPORARY TABLE newArr1(
        index INTEGER,
        value DECIMAL(31,0)
    ) WITH REPLACE ON COMMIT PRESERVE ROWS NOT LOGGED;

    INSERT INTO t1 VALUES ((SELECT VALUE
                            FROM SESSION.dept_name
                            WHERE INDEX = 1),10);

    DELETE FROM SESSION.emp_name;
    INSERT INTO SESSION.emp_name VALUES (1,'jim');
    INSERT INTO SESSION.emp_name VALUES (2,'joe');
    INSERT INTO SESSION.emp_name VALUES (3,'jon');

    UPDATE SESSION.emp_name
          SET VALUE = 'jen'
          WHERE INDEX = 2;
END!
```

Collection types in Oracle may be declared as schema objects, types local to a block, package types, or subtypes. Each of these collection type declarations is supported by MTK. Collection element types supported by MTK are built-in types and subtypes.

MTK supports only the following contexts where references to expressions having collection types may occur. These operations are translated using the appropriate DML statements over the corresponding global temporary table.

- Collection constructors as local variable default values. For example:

```
x collection_type := collection_type('a','b','c');
```

- Assignment of collection constructors or a local variable or parameter to another local variable or parameter of collection type.
- Indexed reference to an element of a collection variable or parameter. For example:

```
x(n) := x(n+1) * 2;
```

- Local variables, parameters, and collection constructors as actual arguments to calls to functions or procedures.
- Collection methods: COUNT, FIRST, and LAST only, when the collection object is a local variable or parameter.

Note:

1. A function containing a local variable or parameter of collection type is translated to DB2 UDB as a procedure.
2. Global temporary tables in DB2 UDB are subject to transaction rollback. If collection-typed local variables and parameters are used in a context where rollbacks may occur, manual translation might be necessary to preserve the original behavior.

Translation limitations

Other collection expression contexts are not translated by MTK, such as the following:

- Any SELECT or FETCH into an indexed reference to collection elements
- Functions that return collections (nor calls to these functions)
- Package variables of collection type
- Collection types are currently supported for only local variables and parameters.

Wrapped objects:

In Oracle, certain objects can be wrapped or encrypted for security purposes so that they cannot be recovered from the catalog.

All objects that are made, including packages, stored procedures, triggers, and views and that are extracted in binary mode cannot be converted. The original PL/SQL scripts need to be imported to convert these objects.

MERGE

The Oracle MERGE statement is translated to the DB2 MERGE statement.

The following Oracle MERGE statement clauses are not supported:

- The error_logging_clause
- Optimizer hints

DB2 and Oracle MERGE statements perform differently in situations where a DELETE clause is involved with a MERGE statement that contains an UPDATE clause. DB2 rows that were updated by the MERGE statement containing an UPDATE clause will not be deleted if they match the condition in the DELETE clause. You need to keep this in mind when using a translated MERGE statement.

The DELETE clause of the merge_update_clause is translated to a separate DB2 matching-condition and search-condition.

DB2 UDB dynamic compound statements

A DB2 UDB dynamic compound statement is a block of statements (between BEGIN and END) that is used for top-level anonymous blocks, user-defined functions bodies, and trigger bodies.

The features available inside dynamic compound statements contain some restrictions:

- local variables with defaults
- condition SELECT (no INTO)
- VALUES (no INTO)
- SET (multi column)
- UPDATE (no CURRENT OF)
- DELETE (no CURRENT OF)
- INSERT SIGNAL
- GET DIAGNOSTICS .. ROWCOUNT
- IF THEN ELSEIF END IF
- WHILE
- FOR
- ITERATE
- LEAVE
- multiple-line comments: /* ... */
- single-line comments: --

All other features cannot be used in the DB2 UDB Dynamic Compound Statement (for example, CASE, LOOP, Nested blocks, SELECT INTO, VALUES INTO, and cursors). The converter will convert features that cannot be used in Oracle, but it will comment them out and issue an error message (″This statement is not supported in a DB2 UDB Dynamic Compound Statement″). Refer to the DB2 UDB documentation for a more comprehensive description of the dynamic compound statement.

Related reference

DB2 UDB Version 8 - Compound SQL (Dynamic) statement

Transaction statements

The transaction statements COMMIT, ROLLBACK, and SAVEPOINT are translated directly to DB2 UDB, with a few exceptions as noted here.

Translation limitations

Commit

The COMMIT FORCE statement and the COMMENT clause of the COMMIT statement are not converted.

The Oracle COMMIT statement does not close currently open cursors. The DB2 UDB COMMIT statement does close currently open cursors (except those declared using ″WITH HOLD″). Manual conversion might be required if open cursors are used after a COMMIT in the Oracle source code. COMMIT in both Oracle and DB2 UDB, releases the locks held by the cursors, except in DB2 UDB for ″WITH HOLD″ cursors. See “Implicit commit” on page 211for additional information.

Rollback and savepoint

Oracle can have many savepoints active at one time, while DB2 UDB allows only one savepoint to be active at a time. The converter is unable to determine if any other savepoints are active when a SAVEPOINT is encountered, so the conversion causes an error in DB2 UDB when this is the case.

Autonomous transactions

Oracle allows subprograms to behave as ″autonomous transactions″ when the corresponding pragma is given. In this case the transaction statements in the subprogram behave independently of the enclosing environment. DB2 UDB does not provide for autonomous transactions and the converter flags any usage of the AUTONOMOUS_TRANSACTION pragma. The transaction statements inside the subprogram are translated without warning, though they may have different behavior in DB2 UDB, in combination with the outer environment.

Related reference

"Implicit commit" on page 211
Oracle issues an implicit commit before and after each DDL statement, and DDL statements may appear only at the top-level. When DB2 UDB is run in auto-commit mode, a commit is issued after every top-level statement.

Variable assignment

The LET statement is used to assign values to variables when converting from Oracle PL/SQL to Informix Dynamic Server. Oracle boolean variables TRUE, FALSE, and NULL are translated to 1, 0, and NULL.

Here is an example of an Oracle block of code:

```
create table tab(no int);
create procedure ora.detail_proc_type_num as
done boolean;
myno int;
begin
done := TRUE;
done := FALSE;
done := NULL;
myno := 1500;
select no into myno from tab where no = 5;
end;
```

The following table shows how the variables in the above example are converted from Oracle PL/SQL variables to Informix Dynamic Server variables.

Oracle variable assignment	Informix Dynamic Server variable assignment
`myno:= 1500;`	`LET myno = 1500;`
`done := FALSE;`	`LET done = 0;`
`done := TRUE;`	`LET done = 1;`
`done := NULL;`	`LET done = NULL;`
`select no into myno from tab where no = 5;`	`SELECT "no" INTO myno FROM tab WHERE no" = 5;`

Compatibility library (ORA functions)

Some SQL and Java functions are provided to simulate Oracle functions that do not have DB2 UDB or Informix Dynamic Server equivalents. A schema called ORA has been created to contain these functions.

Functions of this type that correlate directly to an Oracle equivalent are described in "List of functions that simulate Oracle built-in functions" on page 238. Functions in the ORA schema that are not used as direct entry points are not described. However, function definitions are available in the mtkora.udf file located in the product directory. Where possible, SQL was used, but it was necessary in some cases to implement the UDF in Java. The mtkoradrop.udf file is a script file used to conveniently drop any functions created in the ORA schema with mtkora.udf. These files are deployed to the target database server during the deploy step of the migration. You can change the files to improve efficiency or change the specific behavior of a function.

Related reference

"Built-in-functions" on page 159
There are three possible scenarios when converting Oracle built-in functions to the target IBM database (either DB2 UDB or Informix Dynamic Server).

National language support

The behavior of some of the Oracle built-in functions depend on the current state of several national language support (NLS) session parameters.

These parameters have the following characteristics:

- They have default values.
- They can be changed using ALTER SESSION SET statements.
- They are reset to their default values with CONNECT statements.
- Some can be temporarily overridden with an NLS parameter string.

When an Oracle database is converted to DB2 UDB, the states of the session parameters are not stored in the DB2 UDB database, but they are tracked by the Oracle converter, which produces code to provide the same behavior as the Oracle code. The converter reacts to the states as follows:

- If all the arguments of a function are literals, then the converter will try to directly evaluate the function using the current session parameters and the NLS parameter string if it has been provided. For example:

  ```
  ALTER SESSION SET NLS_NUMERIC_CHARACTERS = 'dg';
  SELECT TO_CHAR(1000.1, '9G999D999') FROM DUAL;

     ----- is translated
  to -----

  SELECT ' 1g000d100' FROM SYSIBM.SYSDUMMY1!
  ```

- If the first argument (that is, the object to be converted) is not a literal, the converter will generate a function call to an ORA conversion UDF.
  - If the parameter string is a literal or is not specified, then the converter will produce an NLS parameter string resulting from the current session parameters and the provided NLS parameter string. For example:

    ```
    ALTER SESSION SET NLS_NUMERIC_CHARACTERS = 'dg';
    SELECT TO_CHAR(i, '999.999U', 'NLS_DUAL_CURRENCY = E') FROM t;

       ----- is translated
    ```

```
to -----

SELECT ORA.TO_CHAR_DECIMAL(CAST (I AS CHAR(40)), '999.999U',
'NLS_ISO_CURRENCY=America NLS_CURRENCY=$ NLS_DUAL_CURRENCY=$
NLS_NUMERIC_CHARACTERS=dg' FROM T!
```

– If the parameter string is a complex expression, then it cannot be analyzed during conversion. In this case the converter will produce an NLS parameter string with the provided expression. The parameter string results from the concatenation of a generated parameter string that holds the current session parameters. The resulting string might specify the value of a given session parameter twice, but because the value provided by the source code is located at the end of the string, the UDF will know that this value is the one that should be used. For example:

```
ALTER SESSION SET NOS_NUMERIC_CHARACTERS = 'dg';
SELECT TO_CHAR(i, '999.999U', UPPER('NLS_DUAL_CURRENCY = E')) FROM t;

   ----- is translated
to -----

SELECT ORA.TO_CHAR_DECIMAL(CAST (I AS CHAR(40)), '999.999U',
'NLS_ISO_CURRENCY=America NLS_CURRENCY=$ NLS_DUAL_CURRENCY=$
NLS_NUMERIC_CHARACTERS=dg' || UPPER('NLS_DUAL_CURRENCY = E')) FROM T!
```

Translation limitations

The NLS session parameters are supported in only the following functions:
- TO_CHAR(num, ...)
- TO_CHAR(date, ...)
- TO_DATE(char, ...)
- TO_NUM(char, ...)

The NLS session parameters that are supported are:
- NLS_DATE_FORMAT
- NLS_DATE_LANGUAGE (*)
- NLS_CURRENCY
- NLS_ISO_CURRENCY (*)
- NLS_DUAL_CURRENCY
- NLS_NUMERIC_CHARACTERS

(*)The states of these parameters are tracked and passed to the UDFs by the converter, but the UDFs do not take them into account.

Oracle PL/SQL has several special NLS session parameters that will change the value of other session parameters when they are set (for example, NLS_LANG, NLS_TERRITORY, NLS_LANGUAGE). These session parameters are not supported, and setting them will have no effect on the other session parameters.

List of functions that simulate Oracle built-in functions

This topic contains descriptions of functions simulate the behavior of certain Oracle built-in functions that do not have DB2 equivalents, and in some cases, that do not have Informix Dynamic Server equivalents. (MTK does not convert Oracle analytic functions into Informix Dynamic Server.)

See the table in “Built-in-functions” on page 159 for a complete list of Oracle functions and the corresponding DB2 UDB and Informix Dynamic Server functions.

The descriptions below explain the differences between Oracle and target server built-in functions, highlight any restrictions that apply to conversion, and show common usage examples where applicable. The words ″DB2″ and ″IDS″ in parenthesis after the function name indicate which converted database might contain the function.

ORA.ADD_MONTHS (DB2 and IDS)
: Simulates the Oracle function ADD_MONTHS. This function handles the last day of a month in the same way as Oracle. (Whereas, if add_month(date,num) were converted to date + num MONTHS, the result would be incorrect.) For example, ORA.ADD_MONTHS(’28-FEB-99’,1) returns 31-MAR-99. The function also subtracts months, as with ORA.ADD_MONTHS(’28-FEB-99’,-1)

ORA.ATAN2 (DB2)
: Simulates the Oracle ATAN2 function. Compared to DB2 UDB ATAN2 built-in function, arguments are passed in reverse order.

ORA.ASCII (IDS)
: Simulates the Oracle ASCII function. The ASCII function returns the NUMBER code that represents the specified character. For example, ORA.ASCII(’t’) returns 116.

ORA.CEIL (IDS)
: Simulates the Oracle CEIL function. The CEIL function returns the smallest integer value that is greater than or equal to a number. For example ORA.CEIL(32.65) returns 33.

ORA.CHR (DB2)
: Provides a simple simulation of the Oracle CHR function. Oracle CHR(num) is equivalent to DB2 UDB chr(mod(num,256)), and ORA.CHR(num) only works for common numbers. Returns NULL when num < 0 . The USING clause is not supported.

ORA.CONCAT (DB2)
: Simulates the Oracle CONCAT function. This UDF is needed because the DB2 UDB built-in function CONCAT differs from Oracle’s for NULL and empty strings.

ORA.COSH (DB2 and IDS)
: Simulates the Oracle COSH function. The COSH function returns the hyperbolic cosine of a number. For example, ORA.COSH(0.2) returns 1.020066755619.

ORA.FLOOR (IDS)
: Simulates the Oracle FLOOR function. The floor function returns the largest integer value that is equal to or less than a number. For example, ORA.FLOOR(5.9) returns 5.

ORA.INITCAP (DB2)
: Simulates the Oracle built-in function INITCAP, which capitalizes the first letter of each word — word being defined as any alphabetical string after a non-alphabetical character (space, number, punctuation characters, special characters) .

ORA.INSTR (DB2)
: Simulates the Oracle built-in function INSTR. Translated using the DB2

UDB function LOCATE. Added support for parameters start_search_at, and occurrence. Supports backward search when start_search_at is negative.

ORA.LAST_DAY (DB2 and IDS)
Simulates the Oracle built-in function LAST_DAY. Translated using target server Date arithmetic.

ORA.LOG(x,y) (DB2 and IDS)
Equivalent to LOG(y)/LOG(x)

ORA.LPAD (DB2)
Specifies a string to be used for padding. This function only pads with blanks.

ORA.LTRIM (DB2 and IDS)
Can provide a set of chars to be trimmed.

ORA.MOD (DB2)
In Oracle PL/SQL, FLOAT values can be used for parameters.

ORA.MONTHS_BETWEEN (DB2 and IDS)
Simulates the Oracle built-in function MONTHS_BETWEEN. Translated using DB2 UDB date arithmetic. MONTHS_BETWEEN returns number of months between dates d1 and d2 as a float. If d1 is later than d2, the result is positive; if earlier, negative. If d1 and d2 are either the same days of the month or both are last days of months, the result is always an integer. Otherwise, Oracle calculates the fractional portion of the result based on a 31-day month and considers the difference in time components of d1 and d2.

ORA.NEXT_DAY (DB2 and IDS)
Simulates the Oracle built-in function NEXT_DAY. Currently, only US day names are supported.

ORA.NO_PAD (DB2)
Selecting the **enforce non-blank-padding**option will invoke the ORA.NO_PAD utility function provided by MTK and make DB2 UDB blank-padding ineffective. The use of this option can cause a significant loss in performance during the execution of a query and should be avoided if you do not require trailing blanks to be significant in string comparisons.

ORA.NO_PAD works by concatenating a character at the end of the sting passed as an argument in order to render DB2 UDB blank-padding ineffective. For example:

```
CREATE TABLE T1 (a CHAR (10));
CREATE TABLE T2 (b VARCHAR (10));

SELECT * FROM T1, T2 WHERE T1.a = T2.b;

   ----- is translated
to -----

SELECT *
FROM T1,T2
WHERE ORA.NO_PAD (T1.A) = ORA.NO_PAD(T2.B) !
```

ORA.REPLACE (DB2)
Must protect DB2 UDB REPLACE equivalent to support NULL strings.

ORA.ROUND (date function) (DB2 and IDS)
DB2 UDB has no equivalent for Date/Time. Currently, the strings that specify the round-level (such as century, month, year) must be a final literal string, not the result of a query of an expression to evaluate.

ORA.RPAD (DB2)
: Simulates the Oracle built-in function RPAD. The DB2 UDB built-in function differs in that you cannot specify what string to use for padding. Using this UDF you can.

ORA.RTRIM (DB2 and IDS)
: Simulates the Oracle built-in function RTRIM. The target built-in function differs in that you cannot specify a set of char to trim. Using this UDF you can.

ORA.SIGN (DB2 and IDS)
: Simulates the Oracle built-in function SIGN.

ORA.SINH (DB2 and IDS)
: Simulates the Oracle built-in function SINH.

ORA.SUBSTR (DB2)
: Simulates the Oracle built-in function SUBSTR. The DB2 UDB built-in function differs in that a start parameter cannot be negative to search backward. This UDF can be used for a backward search.

ORA.TANH (DB2 and IDS)
: Simulates the Oracle built-in function TANH.

ORA.TO_CHAR (IDS)
: All of the format units that are supported by DB2 are also supported by Informix Dynamic Server. In addition, Informix Dynamic Server also supports the following formats which are not supported by DB2:

- DAY
- EE

ORA.TO_CHAR (date, ...) (DB2)
: The following format units are not supported by this function:

- DAY, DY, D
- EE, E (era)
- YEAR, SYEAR (year spelled out)
- IYYY, IYY, IY, I (ISO year)
- WW, W, IW (week)
- CC, SCC (century)
- Q (quarter)

ORA.TO_CHAR (number, ...) (DB2)
: Calls to the TO_CHAR built-in function are converted to a call to the ORA.TO_CHAR or ORA.TO_CHAR_DECIMAL UDF in DB2 UDB:

- If the function is called using an INTEGER or a FLOAT value, the ORA.TO_CHAR UDF is used.
- If the function is called using a DECIMAL value, the ORA.TO_CHAR_DECIMAL UDF is used. This UDF expects a string containing a DB2 UDB formatted number as the first argument. This argument is provided by casting the DECIMAL value to a CHAR(40).

These functions can be called using one or two parameters:

- The first parameter is a FLOAT or CHAR(40) (depending on which UDF is used) representing the number.
- The second parameter (optional) corresponds to the format that should be used for the conversion, and should comply with the Oracle number format models.

ORA.TO_DATE (DB2 and IDS)

Protected single-quote conversions will result in an error (ex: TO_DATE('2002'' 01 01' , 'YYYY'' DD MM')). The last part of a date cannot be skipped (02-DEC-2002 will not match DD-MON-YYYY HH24:MI:SS).

The following format units are not supported by the DB2 converter:

- AD, A.D., BC, B.C., SYYYY
- DAY, DDD, D, DY, J
- EE, E (era)
- SSSSS (seconds past midnight)
- Y,YYY (ex: 1,999)

The following format units are not supported by the Informix Dynamic Server converter:

- SYYYY
- DDD, D, DY, J
- E (era)
- SSSSS (seconds past midnight)
- Y,YYY (ex: 1,999)

ORA.TO_NUMBER (DB2 and IDS)

Calls to the TO_NUMBER built-in function are converted to a call to the ORA.TO_NUMBER or ORA.TO_NUMBER_DECIMAL UDF in DB2 UDB:

- If the function is called in a context where an INTEGER or a FLOAT value is expected, the ORA.TO_NUMBER UDF is used.
- If the function is called in a context where a DECIMAL value is expected, the ORA.TO_NUMBER_DECIMAL UDF is used. This UDF returns a string containing the DB2 UDB formatted number, and an explicit cast is added around the function call to convert the string to a number.

These functions can be called using one or two parameters:

- The first parameter is a CHAR or VARCHAR2 representing the number.
- The second parameter (optional) corresponds to the format that should be used for the conversion, and should comply with the Oracle number format models.

The following number format units are not supported:

- B, FM, TM (formatting options)
- EEEE (scientific notation)
- RN, rn (Roman numerals)
- U (Euro symbol)
- X (hexadecimal value)

ORA.TO_TIMESTAMP (DB2 and IDS)

Protected single-quote conversions will result in an error (ex: TO_DATE('2002'' 01 01' , 'YYYY'' DD MM')). The last part of a date cannot be skipped (02-DEC-2002 will not match DD-MON-YYYY HH24:MI:SS).

The following format units are not supported by the DB2 converter:

- AD, A.D., BC, B.C., SYYYY
- DAY, DDD, D, DY, J
- EE, E (era)
- SSSSS (seconds past midnight)

- Y,YYY (ex: 1,999)

The following format units are not supported by the Informix Dynamic Server converter:

- SYYYY
- DDD, D, DY, J
- E (era)
- SSSSS (seconds past midnight)
- Y,YYY (ex: 1,999)

ORA.TRANSLATE (DB2)
Oracle and DB2 UDB do not have the same parameter order. Protect DB2 UDB TRANSLATE function for NULL strings.

ORA.TRIM_BOTH (DB2)
Simulates Oracle's TRIM(BOTH) function. This function trims the leading and trailing trim_string specified.

ORA.TRUNC (number function) (DB2)
Simulates the Oracle built-in function TRUNC for handling numbers.

ORA.TRUNC (date function) (DB2 and IDS)
Simulates the Oracle built-in function TRUNC for handling dates. DB2 has no equivalent TRUNC function for Date/Time, but this UDF does.

Related reference

"Blank and non-blank padding" on page 167
Oracle blank and non-blank padded semantics are mimicked in translations to IBM Informix Dynamic Server. String comparison is handled differently in DB2 UDB and Oracle, in particular when the string is of type VARCHAR.

DB2 UDB supported features

The following tables list the Oracle features that the converter supports in conversions to DB2 UDB.

Table 10. Oracle SQL

Feature	Status	Comments
Data types:		
RowId and URowId	Supported	
Number	Partial support	Negative scale not supported
Others	Supported	
Expressions:		
Built-in Functions	Most are supported	See"Built-in-functions" on page 159 for complete list
PseudoColumns:		
Level	Not supported	(hierarchical queries)

Table 10. Oracle SQL (continued)

Feature	Status	Comments
RowID	Not supported	
RowNum	Partial support	Using DB2 UDB OLAP row_number function, and fetch first clause when possible
Sequence: currval and nextval	Supported	
User defined operators	Not supported	
@dblink	Not supported	remote database reference
Object type constructor, method call, and attribute	Not supported	(objects)
CAST of multiset or sub-query	Not supported	(collection types)
CURSOR (sub-query)	Not supported	
Others	Supported	
Conditions	Supported	
Column Definitions:		
Default expressions	Partial support	Non-constant expressions not supported
Constraints:		
Names	Partial support	Null/Not Null names are dropped
SCOPE IS and WITH ROWID	Not supported	(objects)
ON DELETE {CASCADE|SET NULL}	Not supported	
Foreign key over object tables	Not supported	
Constraint state	Not supported	
Others	Supported	
Statements (except DML):		
ALTER X	Partial support	Only support TABLE and SESSION

Table 10. Oracle SQL (continued)

Feature	Status	Comments
ALTER SESSION	Partial support	NLS_DATE_FORMAT, CURRENT_SCHEMA, NLS_CURRENCY, NLS_ISO_CURRENCY, NLS_NUMERIC_ CHARACTERS, NLS_DATE_LANGUAGE, NLS_DUAL_CURRENCY, INITIAL_NLS_DATE_LANGUAGE
ALTER TABLE	Partial support	ADD CONSTRAINT, ADD COLUMN, and MODIFY column type and default
ANALYZE	Not supported	
ASSOCIATE STATISTICS	Not supported	
AUDIT	Not supported	
CALL	Partial support	Procedures only
COMMENT	Supported	
COMMIT	Supported	
CREATE (OR REPLACE) X	Partial support	Replace not supported Only the following statements
CREATE TYPE and TYPE BODY	Not supported	
CREATE TABLE:		
Object tables	Not supported	
Temporary tables	Not supported	
Physical/Table properties	Not supported	Partitions, etc.
AS sub-query	Supported	
Tablespace clause	Supported	Option to copy these
CREATE INDEX	Supported	Basic create index and other options are supported.
cluster index	Not supported	
function-based index	Not supported	
Global/local index clause, index attributes, domain index clause	Not supported	

Table 10. Oracle SQL (continued)

Feature	Status	Comments
CREATE SEQUENCE	Supported	
CREATE SYNONYM	Supported	Tables and views only
Reference synonym before it is created	Not supported	
CREATE VIEW		
FORCE, object type view	Not supported	
With check option	Partial support	
CREATE TRIGGER		
DDL and Database event, BEFORE without FOR EACH ROW	Not supported	
INSTEAD OF	Supported	Supported for DB2 UDB Version 8 only
Nested table, referencing Parent as	Not supported	
CREATE PROCEDURE and FUNCTION		For more information, see the Table 11 on page 248 table.
Wrapped bodies	Not supported	Encoded bodies
Call spec	Not supported	External subprograms
NOCOPY, invokers rights	Not supported	
CREATE FUNCTION		For more information, see the Table 11 on page 248 table.
PARALLEL_ENABLE	Not supported	
CREATE PACKAGE and PACKAGE BODY		For more information, see the Table 11 on page 248 table.
DISASSOCIATE STATISTICS	Not supported	
DROP X	Supported	Supported only for objects where CREATE X is supported.
EXPLAIN PLAN	Not supported	
GRANT	Not supported	
LOCK TABLE		

Table 10. Oracle SQL (continued)

Feature	Status	Comments
PARTITION, SUBPARTITION, @dblink	Not supported	
row share, row exclusive, share update, share row exclusive, NOWAIT	Not supported	
NOAUDIT	Not supported	
RENAME	Not supported	
REVOKE	Not supported	
ROLLBACK	Partial support	FORCE 'text' not supported
SAVEPOINT	Partial support	multiple savepoint warning
SET CONSTRAINTS	Not supported	
SET TRANSACTION	Not supported	
SET ROLE	Not supported	
TRUNCATE	Not supported	
DML Statements:		
hints	Not recognized	
PARTITION, SUBPARTITION, @dblink, sample	Not supported	
Table collection	Not supported	
SELECT		
with clause in SUBQUERY	Not supported	
Order by in SUBQUERY	Supported	Supported for DB2 UDB Version 8 only
Hierarchical query	Not supported	
NULLS FIRST/LAST in order by	Not supported	
FOR UPDATE	Partial support	Specific columns ignored
INSERT		
Subquery in table expression	Not supported	

Table 10. Oracle SQL (continued)

Feature	Status	Comments
Returning clause	Not supported	
Order by in inserting sub-query	Supported	Supported for DB2 UDB Version 8 only
DELETE		
Subquery in table expression	Not supported	
Returning clause	Not supported	
UPDATE		
Subquery in table expression	Not supported	
Returning clause	Not supported	
VALUE set clause	Not supported	(object tables)

Table 11. Oracle PL/SQL

Feature	Status	Comments
Data types:		
Boolean	Supported	
Records	Supported	Split into field variables, no parameter or return value
Varrays	Supported	Supported for DB2 Viper II only
Nested tables	Not supported	
Object types (with REF)	Not supported	
REF cursor	Not supported	(cursor variables)
Subtypes	Supported	Substitutes base type for subtype name
%Type, %Rowtype	Partial support	No collection or cursor variable as object
Expressions:		
Collection methods	Not supported	Predefined

Table 11. Oracle PL/SQL (continued)

Feature	Status	Comments
Cursor attributes:		
%FOUND, %NOTFOUND	Supported	
%ISOPEN, %ROWCOUNT	Not supported	
SQL cursor attributes		implicit cursor
%FOUND, %NOTFOUND, %ROWCOUNT	Supported	
%ISOPEN, %BULKROWCOUNT	Not supported	
Boolean expressions	Supported	
Labelled variable+function reference	Supported	
SQLCODE + SQLERRM	Partial support	SQLERRM using DB2 UDB messages
Exponentiation (**)	Supported	
Others	Supported	
Declarations:		
Types:		
Record	Supported	Info used to make split variables later
Subtype	Supported	Info used to inline base type
REF cursor, Nested table	Not supported	
Varray	Supported	Supported in migrations to DB2 Viper II
Variables:		
Null/Not Null constraint	Not supported	Not enforced
Non-constant default value	Supported	Assigned at beginning of block
Records	Supported	One variable per field
Cursor variables	Not supported	

Table 11. Oracle PL/SQL (continued)

Feature	Status	Comments
Objects	Not supported	
Constants	Supported	Translated to variables
Cursors	Supported	Not in triggers, functions, top level blocks
Local procedures and functions	Not supported	
Exceptions	Supported	Not in triggers, functions, top level blocks
Pragmas:		
AUTONOMOUS_TRANSACTION EXCEPTION_INIT RESTRICT_REFERENCES	Not supported	
Statements:		
Assignment	Supported	(not for collection(index), cursor variable, objects and object attributes)
SELECT INTO	Supported	BULKCOLLECT is only supported for VARRAY data types in migrations to DB2 Viper II
OPEN, CLOSE, FETCH	Supported	Single Fetch with BULKCOLLECT is supported for varray variables in migrations to DB2 Viper II No cursor variables, no BULKCOLLECT Not in triggers, functions, top level blocks
EXECUTE IMMEDIATE	Not supported	
EXIT	Supported	
FORALL	Supported	FORALL statement is supported for insert of varray variables in migrations to DB2 Viper II Collections are not supported

Table 11. Oracle PL/SQL (continued)

Feature	Status	Comments
GOTO	Supported	Not in triggers, functions, top level blocks
IF	Supported	
LOOP	Supported	
FOR (cursor)	Supported	
FOR (range)	Supported	
NULL	Supported	
OPEN FOR	Not supported	(cursor variables)
NESTED BLOCK	Supported	Not in triggers, functions, top level blocks
RAISE	Supported	
RETURN	Supported	Not in Blocks
WHILE	Supported	
COMMIT, ROLLBACK, SAVEPOINT	Supported	Same as SQL Not in triggers, functions, top level blocks
LOCK TABLE	Supported	Same as SQL Not in triggers, functions, top level blocks
SET TRANSACTION	Not supported	
INSERT, DELETE, UPDATE	Supported	Same as SQL, with WHERE CURRENT OF
Exceptions:		
Predefined	Partial support	Where there is good correspondence
User defined	Supported	
Program Units:		
<<labels>>	Partial support	Not in triggers, functions, top level blocks (except with loops)

Table 11. Oracle PL/SQL (continued)

Feature	Status	Comments
Top Level Block	Partial support	No label, see individual statements+declarations
Function:		Not local, see individual statemenst+declarations
OUT parameters	Supported	Simulate using a procedure with an extra parameter for the result value.
DML	Not supported	
Procedure	Supported	Not local
Packages:		Requires workarounds
Subtypes	Supported	Substitutes base type for subtype name
Variables	Not supported	
Constants	Supported	Translated to UDF
Exception	Not supported	
Cursor	Not supported	
Function + Procedure	Supported	Translated to a top level schema
Call spec	Not supported	(external definition)
Initialization block	Supported	
Pragma: SERIALLY_REUSABLE	Not supported	
Reference to item before its definition	Partial support	Package item definitions are reordered, where possible, so that definitions precede any references to them within the package body.
Library Packages	Not supported	

Table 11. Oracle PL/SQL (continued)

Feature	Status	Comments
DBMS_OUTPUT	Partial support	Provide procedures and functions with default definitions that user may modify
Others	Not supported	
UTL_FILE	Partial support	Provide the JAVA UDFs to simulate the UTL_FILE package behavior

Informix Dynamic Server supported features

These tables list the Oracle features that the converter supports in conversions to Informix Dynamic Server.

Table 12. Oracle SQL

Feature	Status	Comments
Data types:		
ROWID	Supported	Integer
UROWID	Supported	
Number	Partial support	Negative scale not supported.
INTERVAL	Not supported	
Others	Supported	For more information, see the Data types mapping table.
Expressions:		
Built-in Functions	Most are supported	For complete support information, see “Built-in-functions” on page 159.
PseudoColumns:		
Level	Not supported	(hierarchical queries)
RowID	Supported	Informix Dynamic Server has its own implementation of the RowID pseudocolumn.
RowNum	Not supported	
Sequence: curval and nextval	Supported	
PRIOR expression	Not supported	(hierarchical queries)

Table 12. Oracle SQL (continued)

Feature	Status	Comments
User defined operators	Not supported	
@dblink	Not supported	Remote database reference
Object type constructor, method call, and attribute	Not supported	(objects)
CAST of multiset or sub-query	Not supported	(collection types)
Interval Expressions	Not supported	
CURSOR (sub-query)	Not supported	
Others	Supported	
Conditions:	Supported	
Column Definitions:		
Default expressions	Not supported	Only default literal values are supported.
Constraints:		
Names	Partial support	Null/Not Null names are dropped.
SCOPE IS and WITH ROWID	Not supported	(objects)
ON DELETE CASCADE	Supported	
ON DELETE SET NULL	Not supported	
Foreign key over object tables	Not supported	
Constraint state	Not supported	
Others	Supported	
Statements (except DML):		
ALTER X	Partial support	Only support TABLE and SESSION
ALTER SESSION	Partial support	NLS_DATE_FORMAT, CURRENT_SCHEMA, NLS_CURRENCY, NLS_ISO_CURRENCY, NLS_NUMERIC_ CHARACTERS, NLS_DATE_LANGUAGE, NLS_DUAL_CURRENCY, INITIAL_NLS_DATE_LANGUAGE

Table 12. Oracle SQL (continued)

Feature	Status	Comments
ALTER TABLE	Partial support	ADD CONSTRAINT, ADD COLUMN, and MODIFY column type and default
ANALYZE	Not supported	
ASSOCIATE STATISTICS	Not supported	
AUDIT	Not supported	
CALL	Partial support	Procedures only
COMMENT	Not supported	
COMMIT	Supported	
CREATE (OR REPLACE) X	Partial support	Replace not supported Only the following statements
CREATE TYPE and TYPE BODY	Not supported	
CREATE TABLE:		
Object tables	Not supported	
Temporary tables	Not supported	
Physical/Table properties	Not supported	Partitions, etc.
AS sub-query	Not supported	
Tablespace clause	Not supported	
CREATE INDEX	Supported	Basic create index and other options are supported.
cluster index	Not supported	
function-based index	Not supported	
Global/local index clause, index attributes, domain index clause	Not supported	
CREATE SEQUENCE	Supported	
CREATE SYNONYM	Supported	Tables and views only.
Reference synonym before it is created	Not supported	
CREATE VIEW	Supported	Basic create view and other options are supported.

Table 12. Oracle SQL (continued)

Feature	Status	Comments
FORCE, object type view	Not supported	
With check option	Partial support	
CREATE TRIGGER	Supported	Basic create view and other options are supported.
DDL and Database event, BEFORE without FOR EACH ROW	Not supported	
INSTEAD OF	Not supported	
Nested table, referencing Parent as	Not supported	
CREATE PROCEDURE and FUNCTION		For more information, see the Table 13 on page 258 table.
Wrapped bodies	Not supported	Encoded bodies
Call spec	Not supported	External subprograms
NOCOPY, invokers rights	Not supported	
CREATE FUNCTION		For more information, see the Table 13 on page 258 table.
PARALLEL_ENABLE	Not supported	
CREATE PACKAGE and PACKAGE BODY		For more information, see the Table 13 on page 258 table.
DISASSOCIATE STATISTICS	Not supported	
DROP X	Supported	Supported only for objects where CREATE X is supported.
EXPLAIN PLAN	Not supported	
GRANT	Not supported	
LOCK TABLE	Not supported	
PARTITION, SUBPARTITION, @dblink	Not supported	
row share, row exclusive, share update, share row exclusive, NOWAIT	Not supported	
NOAUDIT	Not supported	
RENAME	Not supported	

Table 12. Oracle SQL (continued)

Feature	Status	Comments
REVOKE	Not supported	
ROLLBACK	Not supported	
SAVEPOINT	Not supported	
SET CONSTRAINTS	Not supported	
SET TRANSACTION	Not supported	
SET ROLE	Not supported	
TRUNCATE	Not supported	
DML Statements:		
hints	Not recognized	
PARTITION, SUBPARTITION, @dblink, sample	Not supported	
Table collection	Not supported	
SELECT	Partial support	Basic SELECT and other options are supported.
with clause in SUBQUERY	Not supported	
Order by in SUBQUERY	Not supported	
Hierarchical query	Not supported	
NULLS FIRST/LAST in order by	Not supported	
FOR UPDATE	Not supported	
INSERT	Partial support	Basic SELECT and other options are supported.
Subquery in table expression	Not supported	
Returning clause	Not supported	
Order by in inserting sub-query	Not supported	
DELETE	Partial support	Basic SELECT and other options are supported.
Subquery in table expression	Not supported	
Returning clause	Not supported	
UPDATE		

Table 12. Oracle SQL (continued)

Feature	Status	Comments
Subquery in table expression	Not supported	
Returning clause	Not supported	
UPDATE	Partial support	Basic SELECT and other options are supported.
Subquery in table expression	Not supported	
Returning clause	Not supported	
VALUE set clause	Not supported	(object tables)

Table 13. Oracle PL/SQL

Feature	Status	Comments
Data types:		
Boolean	Supported	
Records	Supported	Split into field variables, no parameter or return value
Varrays, Nested tables	Not supported	
Object types (with REF)	Not supported	
REF cursor	Not supported	(cursor variables)
Subtypes	Not supported	
%Type, %Rowtype	Partial support	No collection or cursor variable as object
Expressions:		
Collection methods	Not supported	Predefined
Cursor attributes:		
%FOUND, %NOTFOUND	Not supported	
%ISOPEN, %ROWCOUNT	Not supported	

Table 13. Oracle PL/SQL (continued)

Feature	Status	Comments
SQL cursor attributes		implicit cursor
%FOUND, %NOTFOUND, %ROWCOUNT	Not supported	
%ISOPEN, %BULKROWCOUNT	Not supported	
Boolean expressions	Supported	
Labelled variable+function reference	Supported	
SQLCODE + SQLERRM	Not supported	
Exponentiation (**)	Supported	
Others	Supported	
Declarations:		
Types:		
Record	Supported	Info used to make split variables later
Subtype	Not supported	
REF cursor, Nested table, Varray	Not supported	
Variables:		
Null/Not Null constraint	Not supported	Not enforced
Non-constant default value	Supported	Assigned at beginning of block
Records	Supported	One variable per field
Cursor variables	Not supported	
Objects	Not supported	
Constants	Supported	Translated to variables
Cursors	Supported	Not in triggers, functions, top level blocks

Table 13. Oracle PL/SQL (continued)

Feature	Status	Comments
Local procedures and functions	Not supported	
Exceptions	Partial support	Not in triggers, functions, top level blocks
Pragmas:		
AUTONOMOUS_TRANSACTION EXCEPTION_INIT RESTRICT_REFERENCES	Partial support	Only EXCEPTION_INIT is supported
Statements:		
Assignment	Supported	(not for collection(index), cursor variable, objects and object attributes)
CASE	Supported	
COMMIT	Supported	Same as SQL Not in triggers, functions, top level blocks
EXECUTE IMMEDIATE	Not supported	
EXIT	Supported	
FORALL	Not supported	(collections)
FOR (cursor)	Supported	
FOR (range)	Supported	
GOTO	Not supported	
IF	Supported	
INSERT, DELETE, UPDATE	Partial support	Same as SQL, with WHERE CURRENT OF not supported.
LOCK TABLE	Not supported	
LOOP	Supported	
NESTED BLOCK	Not supported	

Table 13. Oracle PL/SQL (continued)

Feature	Status	Comments
NULL	Supported	
OPEN, CLOSE, FETCH	Not supported	
OPEN FOR	Not supported	(cursor variables)
RAISE	Supported	
RETURN	Supported	Not in Blocks
ROLLBACK, SAVEPOINT	Not supported	
SELECT INTO	Supported	No BULKCOLLECT
SET TRANSACTION	Not supported	
WHILE	Supported	
Exceptions:		
Predefined	Partial support	The following predefined exceptions are not supported: ACCESS_INTO_NULL, ASE_NOT_FOUND, COLLECTION_IS_NULL, CURSOR_ALREADY_OPEN, ROWTYPE_MISMATCH, SELF_IS_NULL, SUBSCRIPT_BEYOND_COUNT, SUBSCRIPT_OUTSIDE_LIMIT, SYS_INVALID_ROWID, TIMEOUT_ON_RESSOURCE,
User defined	Supported	
Program Units:		
<<labels>>	Partial support	Not in triggers, functions, top level blocks (except with loops)
Top Level Block	Partial support	No label, see individual statements+declarations
Function:		Not local, see individual statemenst+declarations
OUT parameters	Not supported	

Table 13. Oracle PL/SQL (continued)

Feature	Status	Comments
DML	Not supported	
Procedure	Supported	Not local
Packages:	Not supported	
Library Packages	Not supported	
DBMS_OUTPUT	Not supported	
Others	Not supported	

SQL Server and Sybase converter

The following sections detail the support that the converter provides for translations from Microsoft SQL Server, Sybase Enterprise, and Sybase ASA.

Converter behavior and limitations

The topics in this section describe how each particular feature of the Microsoft SQL Server and Sybase source languages are converted.

These reference topics are not intended to be a comprehensive overview of the features of the source and target SQL languages. The information here lists any translation limitations and provides some tips when you need to provide your own modifications to the SQL.

Identifiers

The converter adjusts identifiers as necessary to compensate for any length limitations.

Specifies the maximum number of bytes that can be used to identify the types of database objects listed. The limits are measured in bytes. Languages with multi-byte characters would thereby be limited to a lesser number of characters for the identifying string.

Object Name	Microsoft SQL Server	Sybase	DB2 UDB	Informix Dynamic Server
Table names or aliases	128	30	128	128
View names or aliases	128	30	128	128
Procedure names	128	30	128	128
Column names	128	30	30	128
Variable names	128	30	64	128
Trigger names	128	30	18	128
Rule names	128	30	n/a	128
Constraint names	128	30	18	128
UDT names	128	30	18	128
Default names	128	30	n/a	128
UDF names	128	n/a	18	128
Index names	128	30	18	128
Schema names	—	n/a	30*	128
Database names	—	—	8	128
User ID	—	—	30	128

Table note

* Truncated to 8 characters because it is used in data type definitions.

Since some SQL Server and Sybase identifiers that are too long are abbreviated in the generated DB2 UDB code, there might be places where a name that appears unqualified in the T-SQL source has to be qualified in the DB2 UDB code to avoid ambiguity.

If the name of the source database user-defined type is the same name as an underlying data type of the target database, then it is renamed to avoid errors. For example, in the following translated statement, *date* becomes *date1*:

```
sp_addtype date, smalldatetime;

    ----- is translated
to -----

CREATE DISTINCT TYPE date1 AS TIMESTAMP WITH COMPARISONS!
```

The converter also generates a new identifier when it detects instances of target database keywords that are used as identifiers in the source.

Only proper T-SQL identifiers can be used as user-defined type names. For example, double quoted strings cannot be used as user-defined type names.

Migration strategies

The converter renames identifiers by retaining the first few characters and appending a number. To help you identify which objects have been renamed, the converter issues a message for each rename. If you are satisfied with the generated names, you can hide the message when refining the translation. If you prefer your own naming scheme, you can use the object renaming features on the Refine page of MTK.

Related concepts

"Object renaming" on page 129
Because of differences in rules for identifiers, some objects must be renamed during the translation. The converter generates new names in such a way that they do not conflict with any preexisting names.

Related tasks

"Changing an object name" on page 41

If an edit icon ( ) exists in the target column, you can change the name of the object.

Related reference

DB2 UDB Version 8 - SQL Limits

DB2 UDB Version 8 - Identifiers

Data types

MTK converts Microsoft SQL Server and Sybase built-in types to DB2 UDB and Informix Dynamic Server built-in types, according to the mapping shown here. Server information in parenthesis (for example, *Sybase SQL Anywhere*) indicates that the data type in the row or in the line just above the server name is applicable only in the identified server.

SQL Server and Sybase	DB2 UDB	Informix Dynamic Server
TINYINT	SMALLINT	SMALLINT
SMALLINT	SMALLINT	SMALLINT*****

SQL Server and Sybase	**DB2 UDB**	**Informix Dynamic Server**
INT INTEGER	INTEGER	INTEGER*****
BIGINT	BIGINT	INT8*****
UNSIGNED TINYINT *(Sybase SQL Anywhere)*	SMALLINT	SMALLINT
UNSIGNED SMALLINT *(Sybase SQL Anywhere)*	INTEGER	INTEGER
Sybase ASA UNSIGNED INT	BIGINT	INT8
Sybase ASA UNSIGNED BIGINT	DECIMAL(20,0)	DECIMAL(20,0)
NUMERIC(p,s)	DECIMAL(p,s)	DECIMAL(p,s)
DECIMAL(p,s)	DECIMAL(p,s)	DECIMAL(p,s)
DECIMAL	DECIMAL(30,4)	DECIMAL(30,4)
FLOAT	DOUBLE	FLOAT
DOUBLE PRECISION	DOUBLE	DOUBLE PRECISION
REAL	REAL	REAL
SMALLMONEY	DECIMAL(10,4)	DECIMAL(10.4)
MONEY	DECIMAL(19,4)	DECIMAL(19,4)****
BIT	SMALLINT	SMALLINT
DATE	DATE	DATE
TIME		DATETIME HOUR TO FRACTION(5)
TIMESTAMP	CHAR(8) FOR BIT DATA (two increment triggers are created)	DATETIME YEAR TO FRACTION(5)
DATETIME	TIMESTAMP*	DATETIME YEAR TO FRACTION(5)
SMALLDATETIME	TIMESTAMP	DATETIME YEAR TO FRACTION(5)

CHAR(n)
CHAR(n) NULL
	CHAR(n)
	VARCHAR(n)
		CHAR(n)
		VARCHAR(n)

(Sybase SQL Anywhere)
CHAR(n)
 	
CHARACTER (n)
n < 256
else
	VARCHAR(n)
	
	VARCHAR(n)
		VARCHAR(n)
		
		VARCHAR(n)
		LVARCHAR(n)

VARCHAR(n)
CHAR VARYING *(Sybase SQL Anywhere)*
	VARCHAR(n)
	VARCHAR(1)
		VARCHAR(n)
		VARCHAR

NVARCHAR(n)	VARGRAPHIC(n)**	(not translated)

SQL Server and Sybase	DB2 UDB	Informix Dynamic Server

NCHAR(n)
NCHAR(n) NULL
LONG VARCHAR
(Sybase SQL Anywhere)
	GRAPHIC(n)**
	VARGRAPHIC(n)
	CLOB(2G)
		(not translated)
		(not translated)
		TEXT

BINARY(n) BINARY(n) NULL	CHAR(n) FOR BIT DATA VARCHAR(n) FOR BIT DATA	BYTE
Sybase ASA LONG BINARY	BLOB(2G)	BLOB
VARBINARY(n)	VARCHAR(n) FOR BIT DATA	BYTE
IMAGE	BLOB(2G)	BLOB
TEXT	CLOB(2G)	TEXT
NTEXT** *(SQL Server)*	DBCLOB***	(not translated)

SYSNAME	VARCHAR(30)
	(Sybase)
	VARCHAR(128)
	(SQL Server)
		(not translated)

SQL_VARIANT	VARCHAR(800)	(not translated)
UNIQUE IDENTIFIER	CHAR(16) FOR BIT DATA	
UNIQUEIDENTIFIERSTR *(Sybase SQL Anywhere)*	CHAR(36)	CHAR(36)
XML *(Sybase SQL Anywhere)*	(not translated)	TEXT

Some Sybase SQL Anywhere to Dynamic Server Data Type Conversion Examples

Table notes

- * DB2 UDB stores full TIMESTAMP values while Sybase rounds DATETIME values to 1/300 of a millisecond. During conversion to DB2 UDB, DATETIME values will be converted to the same value in TIMESTAMP. All new TIMESTAMP value entries made after migration will not be rounded.
- **MTK can convert NTEXT to DB2 UDB from MS SQL. MTK cannot convert NTEXT to DB2 UDB from Sybase SQL Anywhere.
- *** If the NCHAR, NVARCHAR, or NTEXT data types are directly deployed to DB2 UDB, the following message appears:

  ```
  DB2 UDB1034E  The command was processed as an SQL statement
  because it was not a   valid Command Line Processor command.
  During SQL processing it returned:  SQL1216N  Graphic data
  and graphic functions are not supported for this database.
  SQLSTATE=5603
  ```

  The DB2 UDB graphic data type requires that specific graphic support be enabled in the target DB2 UDB database before it can be used. You can enable graphic support when you create the target DB2 UDB database by specifying that the UTF-8 code set be used:

  ```
  create database databasename codeset utf-8 territory us
  ```

You can also change the Global Type Mapping in MTK to translate NCHAR and NVARCHAR to CHAR and VARCHAR respectively.

- **** The Informix Dynamic Server DECIMAL data type is used instead of the MONEY data type because the size and precision is different than that of Sybase SQL.
- ***** When Sybase SQL Anywhere SMALLINT, INTGEGER, and BIGINT data types are converted to Informix Dynamic Server, the default mapping cannot accommodate the least of the values of these data types. This occurs because the lowest bounds for Sybase are lower by one (1) than what Informix Dynamic Server data types can handle. See the paragraphs below this table for examples of default mappings of these data types.

  If you want to use the least of the values of these data types, you can change the mapping to the next higher data type. For example, you can make these changes:
  - SMALLINT -> INTEGER
  - INTEGER -> BIGINT
  - BIGINT -> DECIMAL(20,0)

The following examples show default mappings from Sybase SQL Anywhere to Informix Dynamic Server:

```
SMALLINT  (-2^15  to 2^15 -1)
   ----- is translated
to -----
SMALLINT   -(2^15-1)  to 2^15 -1

INTEGER   (-2^31 to 2^31-1)
   ----- is translated
to -----
INTEGER      -(2^31-1) to 2^31-1

BIGINT      (-2^63 to 2^63 -1)
   ----- is translated
to -----
INT8          -(2^63-1)    to 2^63 -1
```

Related reference

DB2 UDB Version 8 - Data types

System procedures

Some of the Sybase system procedures are translated as described here.

SP_ADDTYPE

The converter processes all but the third argument. Translations to Informix Dynamic Server use the CREATE DISTINCT TYPE statement.

SP_PRIMARYKEY and SP_FOREIGNKEY

The ALTER TABLE ADD constraint is used to translate SP_PRIMARYKEY and SP_FOREIGNKEY if the appropriate option is selected in the MTK interface before conversion.

SP_BINDEFAULT and SP_BINDRULE

The converter's SP_BINDRULE processing ignores the FUTUREONLY specification.

The converter gives a warning message for SP_BINDRULE and SP_BINDEFAULT in the presence of the wrong number of arguments or in the case of multiple bindings (that is, rebinding to the same object).

Since DB2 UDB has no statement similar to either SP_BINDRULE or SP_BINDEFAULT, the translator encodes the effect of these statements into the definition (that is, the CREATE TABLE statement) of the tables that are affected by these calls. Thus, although no corresponding DB2 UDB statement is produced for these statements, the translator does correctly account for them.

Translation limitations

When converted output is derived from multiple distinct input statements (e.g., statements ″connected together″ using SP_BIND), the source information copied (in comments) to the converted output may not include all of the source from which the output was derived.

System procedures that are not listed above are not yet supported.

Related reference

“CREATE DOMAIN” on page 285
The Sybase SQL Anywhere CREATE DOMAIN statement is translated to Informix Dynamic Server as a CREATE DISTINCT TYPE statement.

Built-in functions

Microsoft SQL Server and Sybase functions are mapped directly to the DB2 UDB equivalent where available; and Sybase SQL Anywhere functions are mapped to the IBM Informix Dynamic Server equivalent where available.

Some calls are mapped to user-defined functions (UDFs) provided with MTK. These UDFs are called compatibility functions and are prefixed with SYB or MSSQL.

The table below shows Microsoft SQL Server and Sybase functions with the equivalent DB2 UDB and Informix Dynamic Server implementation. Examples showing how some particular Sybase ASA functions convert to Dynamic Server functions are included in this topic, below the table.

Table 14. Function equivalents.. This table shows the Microsoft SQL Server and Sybase functions that have equivalent DB2 UDB and Informix Dynamic Server functions. If an equivalent function is not listed, the target database does not support an equivalent function.

Function	Type	DB2 UDB equivalent	Informix Dynamic Server equivalent
abs	math	abs	abs
acos	math	acos	acos
argn	miscellaneous		
ascii	string	ascii	
asin	math	asin	asin
atan	math	atan	atan
atn2	math	atan2	atan2
avg	math	SYB or MSSQL.avg	avg
binary_checksum (SQL Server)			
cast	conversion	cast	
ceiling	math	ceiling or cast	

Table 14. Function equivalents. (continued). This table shows the Microsoft SQL Server and Sybase functions that have equivalent DB2 UDB and Informix Dynamic Server functions. If an equivalent function is not listed, the target database does not support an equivalent function.

Function	Type	DB2 UDB equivalent	Informix Dynamic Server equivalent
char	string	chr	
charindex	string	SYB or MSSQL.charindex	
char_length	string	length	char_length
checksum (SQL Server)			
checksum_agg (SQL Server)			
coalesce	miscellaneous	coalesce	
col_length	system	(not converted)	
col_name	system	(not converted)	
compare			
convert	conversion	cast	
concat	str	SYB or MSSQL.concat	
cos	math	cos	cos
cot	math	cot	
count (Sybase SQL Anywhere)	math		count
count_big (SQL Server)			
current_server (SQL Server)	system	user	
curunreservedpgs	system	(not converted)	
data_pgs	system	(not converted)	
datalength	system	length	
date (Sybase SQL Anywhere)	date		date
dateadd	date	date arithmetic	
datediff	date	SYB or MSSQL.datediff	
datename	date	SYB, syb.datename, or MSSQL.datename	
datepart	date	DB2 UDB date functions	

Table 14. Function equivalents. (continued). This table shows the Microsoft SQL Server and Sybase functions that have equivalent DB2 UDB and Informix Dynamic Server functions. If an equivalent function is not listed, the target database does not support an equivalent function.

Function	Type	DB2 UDB equivalent	Informix Dynamic Server equivalent
datetime	date	DB2 UDB date functions	
day	date	day	day
dayname	date	DB2 UDB date functions	
days	date	SYB.days()	days
db_id	system	(not converted)	
db_name	system	(not converted)	
degrees	math	degrees or cast	
difference	string	difference	
dow	date	DB2 UDB date functions	
exp	math	exp	
exprtype	miscellaneous		
floor	math	floor	
getdate	date	current timestamp	
getutcdate (SQL Server)	date	MSSQL.getutcdate	
greater	miscellaneous	SYB.greater	
grouping (SQL Server)			
hextoint (Sybase)	conversion	SYB.hextoint	
host_id	system	(not converted)	
host_name	system	(not converted)	
hour	time	hour	
hours	time	MSSQL.datediff	
identity	miscellaneous	identity	
ifnull	miscellaneous		
index_col (Sybase)	system	(not converted)	
index_colorder			
insertstr	string		
inttohex (Sybase)	conversion	SYB.inttohex	

Table 14. Function equivalents. (continued). This table shows the Microsoft SQL Server and Sybase functions that have equivalent DB2 UDB and Informix Dynamic Server functions. If an equivalent function is not listed, the target database does not support an equivalent function.

Function	Type	DB2 UDB equivalent	Informix Dynamic Server equivalent
isdate (SQL Server)	date	MSSQL.isdate	
isnull	system	coalesce	
isnumeric (SQL Server)	date	MSSQL.isnumeric	
is_sec_service_on	security	(not converted)	
lcase	string	lcase	
lct_admin	system	(not converted)	
left	string	left	
len (SQL Server)	string	MSSQL.len	
length (Sybase SQL Anywhere)	aggregate		length
lesser	miscellaneous	SYB.lesser	
license_enabled			
lockscheme			
log	math	log or ln	
log10	math	log10	log10
lower	string	lcase (or lower)	lower
ltrim	string	SYB.empty_to_null or ltrim	trim (leading from x)
max	math	SYB or MSSQL.maxv	max
min	math	SYB or MSSQL.minv	min
minute	time	minute	
minutes	time	SYB or MSSQL.datediff	
mod (Sybase SQL Anywhere)	math		mod
month (Sybase SQL Anywhere)	date	month	month
monthname	date	monthname	
months	date	SYB or MSSQL.datediff	
mut_excl_roles	system	(not converted)	
nchar			

Table 14. Function equivalents. (continued). This table shows the Microsoft SQL Server and Sybase functions that have equivalent DB2 UDB and Informix Dynamic Server functions. If an equivalent function is not listed, the target database does not support an equivalent function.

Function	Type	DB2 UDB equivalent	Informix Dynamic Server equivalent
newid (SQL Server)	system	MSSQL.newid	
now (Sybase SQL Anywhere)	date	current timestamp	current date
nullif	miscellaneous	nullif	
number	miscellaneous		
object_id	system	(not converted)	
object_name	system	(not converted)	
patindex	string	partially converted	
pagesize			
pi	math	SYB or MSSQL.pi	
power	math	power	
proc_role	system	(not converted)	
ptn_data_pgs	system	(not converted)	
quarter	time	quarter	
quotename (SQL Server)	string	MSSQL.quotename	
radians	math	radians or cast	
rand	math	rand	
remainder	math		
replicate	string	SYB or MSSQL.replicate	
replace (SQL Server)	string	MSSQL.replace or replace	
reserved_pgs	system	(not converted)	
reverse	string	SYB or MSSQL.reverse	
right	string	SYB or MSSQL.right	
role_contain	system	(not converted)	
role_id	system	(not converted)	
role_name	system	(not converted)	
round	math	SYB or MSSQL.round	round
rowcnt	system	(not converted)	
rtrim	string	SYB.empty_to_null or rtrim	trim (leading from x)

Table 14. Function equivalents. (continued). This table shows the Microsoft SQL Server and Sybase functions that have equivalent DB2 UDB and Informix Dynamic Server functions. If an equivalent function is not listed, the target database does not support an equivalent function.

Function	Type	DB2 UDB equivalent	Informix Dynamic Server equivalent
second	time	second	
seconds	time	SYB or MSSQL.datediff	
session_user (SQL Server)	system	user	
show_role	system	(not converted)	
show_sec_services	security	(not converted)	
sign	math	sign	
similar	string		
sin	math	sin	sin
sortkey	string		
soundex	string	soundex	
space	string	SYB.empty_to_null or space	
sqrt	math	sqrt	sqrt
square (SQL Server)	math	MSSQL.square	
stddev_samp (Sybase SQL Anywhere)	aggregate		
str	string	SYB.STR, SYB or MSSQL.str	
string	string		Important: STRING does not have a direct mapping to Informix Dynamic Server.
stuff	string	cast, SYB or MSSQL.stuff	
substring	string	cast, SYB, MSSQL.substring (or SUBSTR where possible)	substr
sum	math		sum
suser_id	system	(not converted)	
suser_name	system	current user	
suser_snare (SQL Server)	system	user	

Table 14. Function equivalents. (continued). This table shows the Microsoft SQL Server and Sybase functions that have equivalent DB2 UDB and Informix Dynamic Server functions. If an equivalent function is not listed, the target database does not support an equivalent function.

Function	Type	DB2 UDB equivalent	Informix Dynamic Server equivalent
syb_sendmsg (Sybase)			
system_user (SQL Server)	system	user	
tan	math	tan	tan
textptr	text	(not converted)	
textvalid	text	(not converted)	
to_unichar			
today	date	current date	today
trim (Sybase SQL Anywhere)	string		trim
truncate	math	truncate	trunc
truncum	math		
tsequal	system	(not converted)	
ucase	string		
uhighsurr			
ulowsurr			
unicode (SQL Server)	string	MSSQL.unicode	
upper	string	ucase (or upper)	upper
used_pgs	system	(not converted)	
user	system	current user	
user_id	system	(not converted)	
user_name	system	current user	
var_samp (Sybase SQL Anywhere)	aggregate		variance
valid_name	system	(not converted)	
valid_user	system	(not converted)	
weeks	date	SYB or MSSQL.datediff	
year	date	year	year
years	date	SYB or MSSQL.datediff	
ymd	date		

Some Sybase SQL Anywhere to Dynamic Server Conversion Examples

The Sybase SQL Anywhere SQL statement

```
SELECT * FROM circles WHERE (3.1416 * POWER(10,2)) < 1000
   ----- is translated
to -----

SELECT * FROM circles WHERE (3.1416 * POW(10,2)) < 1000
```

The Sybase SQL Anywhere SQL statement

```
Select var_samp(quantity) from sales_order_items
   ----- is translated
to -----

Select variance(quantity) from sales_order_items
```

The Sybase SQL Anywhere SQL statement

```
Select stddev_samp(quantity) from sales_order_items
   ----- is translated
to -----

Select stdev(quantity) from sales_order_items
```

The Sybase SQL Anywhere SQL statement

```
SELECT YEAR(NOW()) from orders
   ----- is translated
to -----

SELECT YEAR(CURRENT) FROM orders
```

Some Sybase SQL Anywhere to DB2 conversion examples

Some Sybase SQL Anywhere SELECT statements differ from related DB2 UDB statements, as demonstrated in the following examples.

Table 15.

Sample Sybase SQL Anywhere Statement	Corresponding DB2 UDB Statement
`SELECT CONVERT(DATE, '2000-10-31', 100);`	`SELECT CAST (DATE('2000-10-31') AS DATE) FROM (VALUES 1) temp_table`
`SELECT CONVERT (TIMESTAMP, '12:34');`	`SELECT CAST (TIMESTAMP ('2005-06-13-12.34.00') AS TIMESTAMP) FROM (VALUES 1) temp_table` The date specified is the current date.
`SELECT CONVERT (TIME, '12:34', 114);`	`SELECT CAST (TIME('12.34.00') AS TIME) FROM (VALUES 1) temp_table`
`SELECT DATENAME( MONTH , '1987/05/02' );`	`SELECT SYB.DATENAME('month', TIMESTAMP('1987-05-02-00.00.00')) FROM (VALUES 1) temp_table`
`SELECT DATEPART( MONTH , '1987/05/02' );`	`SELECT MONTH(TIMESTAMP ('1987-05-02-00.00.00')) FROM (VALUES 1) temp_table`

Related concepts

"Data format recommendations" on page 38
When converting a database with DBCS data, you must give careful

consideration to functions that handle character data. Where Sybase or SQL Server count characters, DB2 UDB counts bytes in functions such as left() or length().

Related reference

“Compatibility library (MSSQL and SYB functions)” on page 302
The following functions simulate the behavior of certain Sybase and Microsoft SQL Server built-in functions that do not have equivalents in the target database server. If a function is used in the translated script, then it is bound to the database during its deployment to DB2 UDB or Informix Dynamic Server.

Expressions and operators

All the basic expressions and operators are supported by the converter. Some expressions require special handling, as described here.

Bit operators
Not handled.

LIKE and NOT LIKE predicates
The converter recognizes these constructs and uses a compatibility function to handle the conversion.

The following require more detailed explanation:

Literals

Character, hexadecimal, and string literals are mapped to the equivalent form in the target database.

Some forms of datetime literals are also converted according to your specification of their format. Literals whose translations are unclear are left as is, and marked with an error message.

Examples

The string provides a good example of a type of literal that must be mapped. In Sybase, the empty string literal (denoted as ’’ in a SQL expression) is equivalent to the single space string (denoted as ’ ’ in a SQL expression). So the condition (’’ = ’ ’) evaluates to true in Sybase, and (’abc’ + ’’ + ’def’) evaluates to ’abc def’. In order to preserve this behavior in DB2 UDB, MTK translates the empty string literal to a string literal containing a single space.

Comparisons of NULL values

Because NULL is interpreted differently in Microsoft SQL Server, Sybase, and the target database servers, comparisons involving NULL can result in some complex translations to ensure the original logic is maintained.

A straightforward translation is possible for equality comparisons that involve the NULL literal.

```
X = NULL

   ----- is translated
to -----

X IS NULL
```

Comparisons of variables that could contain a NULL value, however, require a more complex translation. Further, some translations differ depending upon the setting of the ANSI NULL option, which SQL Server sets by default and Sybase does not.

For example, if both *someColumn* and *@someVar* are NULL, the expression would evaluate to true if ANSINULL were off in the source SQL language. The expression will always evaluate to NULL in DB2 UDB. Therefore, the following translation is used to ensure validity when ANSINULL is off.

```
someColumn = @someVar

   ----- is translated
to -----

((someVar IS NULL) AND (someColumn IS NULL) OR
(someVar IS NOT NULL) AND (someColumn = someVar))
```

Migration strategies

The above code is not easy to maintain, so the translator will produce simpler code when it can verify that it is safe to do so. For cases when you know that the relevant variable will never contain a NULL value, you can simplify the translated expression after the function or procedure is deployed. If you *know* that all expressions in the source SQL script use the ANSI standard for NULL comparisons, you can force the standard on for a more straightforward translation.

```
set ansinull on;
someColumn = @someVar

   ----- is translated
to -----

someVar = v_someVar
```

Comparisons of strings

The translation of string comparisons is straightforward; however, it can be unclear why a particular translation is chosen without understanding the differences in how each database manager handles null values with respect to strings.

To better understand the different treatment of strings between Sybase, SQL Server, and DB2 UDB, suppose a table is defined with three character columns: one column with the special length of one and two with other lengths, one of which does not allow null values. Using the Sybase SQL language, such a table would be defined as follows. The DB2 UDB translation is also shown.

```
create table t ( cOne      char(1) null,
                 cNull     char(5) null,
                 cNotnull  char(5) );

   ----- is translated
to -----

CREATE TABLE t ( cOne     CHAR(1),
                 cNull    VARCHAR(5),
                 cNotnull CHAR(5) NOT NULL)!
```

Except for the special case of character types of length one (CHAR(1)), nullable character types are converted to variable length character types, since in Sybase the behavior of VARCHAR is the same as a nullable CHAR; that is, extra space padding is truncated and NULL values are allowed. Nullable CHAR columns in DB2 UDB remain fixed length for all values except NULL. Sybase CHAR(1) columns are not converted to VARCHAR(1) because strings in Sybase are never of a length less than 1. Finally, in DB2 UDB the default is that columns are nullable rather than non-nullable.

In the SQL Server language, the following definition would produce the same table defined above.

```
crate table  t ( cOne     CHAR(1),
                 cNull    CHAR(5),
                 cNotnull CHAR(5) not null);
```

The behavior of SQL Server has similarities to both Sybase and DB2 UDB. Like Sybase, CHAR columns are variable length, yet like DB2 UDB, the default is that the column is nullable unless otherwise specified.

Table 16. Lengths of various character values. Shows the resulting length when the values shown are inserted into the tables defined above.

Inserted data values	Value in Sybase for char_length(cOne) char_length(cNull) char_length (cNotnull)	Value in Microsoft SQL Server for len(cOne) len(cNull) len(cNotnull)	Value in DB2 UDB for length(cOne) length(cNull) length(cNotnull)
insert into t values('x ', 'x ', 'x ')	1 1 5	1 1 1	1 5 5
insert into t values(' ', ' ', ' ')	1 1 5	0 0 0	1 5 5
insert into t values ('', '', '')	1 1 5	0 0 0	1 0 5
insert into t values (NULL, NULL, NULL)	NULL NULL -error-	NULL NULL -error-	NULL NULL -error-

Related reference

"Data types" on page 264
MTK converts Microsoft SQL Server and Sybase built-in types to DB2 UDB and Informix Dynamic Server built-in types, according to the mapping shown here. Server information in parenthesis (for example, *Sybase SQL Anywhere*) indicates that the data type in the row or in the line just above the server name is applicable only in the identified server.

Implicit type conversions

The differences in the implicit type conversions of T-SQL and DB2 UDB can result in some complexities in the translation of some DML statements.

In the following example, the T-SQL INSERT statement is valid, because DATETIME values are implicitly converted into VARCHAR. DB2 UDB also implicitly converts TIMESTAMP values to VARCHAR, but the format of the date value is different. The translator thus applies explicit transformations (library UDFs and casts) where necessary to simulate the T-SQL results. First the * is expanded into the underlying columns (only c1 in this case), then the transformation is applied to the appropriate columns, in this case by the SYB.date_to_char function.

```
create table t1(c1 datetime)
create table t2(c1 varchar(26))
insert into t2 select * from t1

   ----- is translated
to -----

INSERT INTO t2 SELECT SYB.date_to_char0(c1) FROM t1
```

Similarly, implicit conversions can occur in a FETCH statement. To handle this case, the translator uses a temporary variable of the correct type to hold the value of the fetch and then assigns this value to the variable @x (v_x) using an explicit cast.

```
create procedure p1 as
begin
declare @x varchar(26)
declare c1 cursor for select * from t1
open c1
fetch c1 into @x
close c1
end

   ----- is translated
to -----
...
FETCH FROM c1 INTO v_tempvar;
SET v_x = SYB.date_to_char0(v_tempvar, 0);
...
```

Queries

The converter provides good support for standard-conforming queries and partial support for the T-SQL extensions in Sybase and Microsoft SQL Server.

In T-SQL, the result of a query can be assigned to a variable. If the number of rows returned is more than one, the result of the first row is assigned to the variable. As this can lead to undesired results, DB2 UDB does not allow such an assumption to be made. For migration purposes, the converter achieves the same functionality by performing an explicit row fetch.

```
select @x = c1+1
from t
where c2 = 0

   ----- is translated
to -----

SELECT c2 INTO v_x
FROM t
WHERE c2 = 0
FETCH FIRST 1 ROWS ONLY;
```

In DB2 UDB, all null values are displayed at the end of the list when using ORDER BY and GROUP BY commands, while in SQL Server and Sybase all null values are displayed at the beginning of the list. If the target database is specified to be Version 8 of DB2 UDB, then the ORDER BY clause will be translated in all sub-queries.

The TOP *n* clause in a SELECT list is translated to DB2 UDB using the FETCH FIRST clause. The TOP *n percent* clause is not translated.

Column name change

You can use the IBM Migration Toolkit (MTK) to change the names of the generated columns in tables. The converter qualifies column names where necessary to avoid name conflicts.

For example, the original source DDL specifies the following.

```
create table sales (salesID int)
create table mktg (mktgID int)
select salesID from sales,mktg where salesID = mktgID
```

Then, suppose you decide that it is more manageable to use 'ID' as the name for all primary keys. If you use MTK to change the column names, the converter can then reference the old names and ensure all queries using the column name are specified with no ambiguity, since `SELECT ID FROM sales,mktg WHERE ID = ID` is invalid. The correct translation produced by the converter following the name change is as follows:

```
SELECT sales.ID FROM sales,mktg WHERE sales.ID = mktg.ID
```

Non-intuitive T-SQL extensions

Some T-SQL-extensions to SQL, which are non-intuitive and difficult to understand, might not be translated.

For example, consider the following query:

```
select c1, max(c2) as max from t1 where c1 > 10
```

This query is not a valid query according to the SQL standard (because in the SELECT expression, c1 is not a part of the group by clause). However T-SQL treats this as a valid query. For example, assume the table t1 contains the following:

```
     c1      c2
   ----    ----
      5      25
     15      20
     25      10
```

The result of the T-SQL-query is:

```
     c1     max
   ----    ----
      5      20
     15      20
     25      20
```

This result is non-intuitive because the row containing c1 = 5 is selected even though the where clause specifies that c1 > 10. Whenever the converter detects such queries being used, a message is issued and no translation is done.

Complex outer-joins

Sybase and Microsoft SQL Server provide the capability of performing outer-joins in queries via outer-join operators *= and =*, which can be used in the where clause of queries.

However, in some cases, these operators can be used in a way where the semantics of the query are not clear. For example, the following query seems ambiguous:

```
select * from t1, t2, t3, t4
where t1.c1 *= t2.c1 or t3.c1 *= t4.c1
```

The problem in this query is caused by the OR in the where clause. If the OR was instead an AND, then the semantics of the query are clear and it can be translated into:

```
select *
from (t1 left outer join t2 on t1.c1 = t2.c1),
     (t3 left outer join t4 on t3.c1 = t4.c1)
```

Whenever the converter comes across complex uses of outer join operators that make it difficult to understand the intended semantics of the query, the statement is not translated and a message is issued.

Join and cross join

In conversions for Sybase ASA to DB2 UDB, MTK translates JOIN and CROSS JOIN as INNER JOIN, as shown in the following examples.

The Sybase ASA SQL statement

```
SELECT * from tab_test2 CROSS JOIN tab_test3
   ----- is translated
to -----

SELECT * FROM tab_test2 INNER JOIN tab_test3 ON 1 = 1
```

The Sybase ASA SQL statement

```
SELECT * from tab_test2 JOIN tab_test3 on tab_test2.col1 = tab_test3.col1
   ----- is translated
to -----

SELECT * FROM tab_test2 INNER JOIN tab_test3 ON tab_test2.col1 =
tab_test3.col1
```

Statements

Other than the statements listed in the table, most statements are translated.

The following Sybase or Microsoft SQL Server statements are not supported for DB2 UDB and Informix Dynamic Server deployments:

Table 17. Unsupported statements

ALTER DATABASE ALTER ROLE BACKUP BULK INSERT CREATE DATABASE CREATE ROLE CREATE SCHEMA CREATE STAT DBCC DENY DROP DATABASE	DROP DBSPACE DROP DEFAULT DROP EVENT DROP MESSAGE DROP ROLE DROP RULE DROP STAT DUMP KILL MESSAGE PRINT READ	READ TEXT RECONFIGURE RESTORE REVOKE SHUTDOWN UPDATE STAT UPDATE TEXT WRITE TEXT WRITETEXT Some SET Statements Global variables

Some statement translations are not straightforward and are further described in the following topics:

Limitations during the conversion of T-SQL DROP INDEX statements

In Sybase and SQL Server, you can create identical index names on different tables in the same schema, whereas in DB2 UDB, index names must be unique across the schema. Therefore, the T-SQL DROP INDEX statement is supported only in the form DROP INDEX *tablename.indexname*. The DROP INDEX *indexname*statement is not supported. The *tablename* must be in the statement in order to resolve any ambiguities that might arise when identical indexes have been created on different tables in the same schema.

ALTER

For DB2 UDB deployments, ALTER statements (on tables and views) are converted to DROP and CREATE statements. For Informix Dynamic Server deployments, ALTER statements are translated except as noted. If you are converting multiple script files, ensure there are no dependencies on the altered object.

Translation limitations

For ALTER TABLE translations to DB2 UDB SQL, only ADD *column definition* and ADD *constraint* are supported on the ALTER TABLE.

For Informix Dynamic Server deployments, Sybase SQL Anywhere ALTER TABLE statements involving the renaming of a column are converted to RENAME COLUMN statements. For example:

```
ALTER TABLE test_tab2_1 RENAME col1 TO col1_1;
   ----- is translated
to -----
RENAME COLUMN test_tab2_1.col1 to col1_1
```

For ALTER INDEX translations to Informix Dynamic Server, only the clustering clause is supported. Sybase SQL Anywhere ALTER INDEX statements that rename an index or a foreign key are not supported. For example:

```
alter index idx_name on prod_name clustered;

   ----- is translated
to -----
ALTER INDEX idx_name TO CLUSTER;
```

▶Sybase ASA The ALTER DOMAIN statement is not supported.

For Informix Dynamic Server translations, all ALTER TABLE statements are supported except those with the following:

- CHECK clause
- SET NULL option for ON DELETE of the REFERENCE clause
- COMPUTE clause
- REPLICATE ON
- RENAME CONSTRAINT
- The following DELETE options:
  - FOREIGN KEY
  - PRIMARY KEY
  - UNIQUE
  - CHECK
  - column_name
- Those of the form: ALTER TABLE tabname ALTER CONSTRAINT constrname CHECK condition
- DROP COMPUTE
- DROP DEFAULT

- SET clause
- Those of the form: ALTER colname ADD CONSTRAINT constrname CHECK condition
- ADD PCTFREE
- REFERENCES clause
- COMPUTE condition

Related concepts

"Restrictions for migrating scripts" on page 129
If you are importing scripts that have been created by the source DBMS command interpreter, some restrictions apply.

CREATE TABLE

Using the data type mapping information, a straightforward translation is usually possible.

Translation limitations

Non-literal default expressions

In DB2 UDB, only constants can be used as defaults. If a Sybase or SQL Server default expression translates to a non-constant expression, the convertor generates a trigger to assign the default value to the column.

Sybase system procedures

The translation of a table creation statement also reflects the effect of various sp_bindrule and sp_bindefault statements. In particular, the CREATE TABLE statement generated by the translator might contain constraints and default value specifications not contained in the CREATE TABLE statement in the T-SQL source, but which are implied by various sp_bindrule and sp_bindefault statements in the T-SQL source.

NULL constraints

- There are cases when a nullable column must be converted to a NOT NULL column. For example, in DB2 UDB SQL, nullable columns cannot be used for a unique index or a primary key.
- Nullable CHAR, NCHAR, and BINARY columns are converted to VARCHAR, VARGRAPHIC, and VARCHAR FOR BIT DATA in DB2 UDB since the truncation behavior is the same.

Column check constraints

These are converted into table constraints. In DB2 UDB, column check constraints might not refer to other columns, but table constraints can refer to any columns.

Sybase ASA columns with AUTOINCREMENT clauses

Only one column in a table that is translated to Informix Dynamic Server can have the AUTOINCREMENT clause because Dynamic Server supports only one SERIAL column. If a table contains multiple columns with DEFAULT AUTOINCREMENT, MTK changes one column to SERIAL and displays a message telling you that ″Only one column in Informix can be of type serial.″

For example:

```
CREATE TABLE june_default1
(col1 INTEGER DEFAULT AUTOINCREMENT);
   ----- is translated
to -----
CREATE TABLE june_default1 (col1 SERIAL)
```

Unsupported Sybase ASA options

- AT
- ON COMMIT
- NOT TRANSACTIONAL
- Some DEFAULT options:
  - CURRENT DATABASE
  - CURRENT REMOTE USER
  - CURRENT UTC TIMESTAMP
  - CURRENT PUBLISHER
  - GLOBAL AUTOINCREMENT
  - UTC TIMESTAMP AND LAST USER
- CLUSTERED keyword in column and table constraints
- COMPUTE constraint
- CHECK ON COMMIT foreign key constraint
- Location string
- Percent free

Other Sybase ASA conversion issues

If converting from Sybase ASA, be aware of the following conversion restrictions:

- In general, Sybase ASA CREATE TABLE statements are translated to Dynamic Server CREATE TABLE statements. However, MTK does not support the translation of the DEFAULT key word in the INSERT statement in instances when the DEFAULT keyword is used to cause the default column value to be inserted into a table.
- MTK does not support the ON UPDATE CASCADE clause in conversions to either DB2 or Informix Dynamic Server from a Sybase ASA source database. During conversion, MTK removes the clause and translates it to a table with a foreign-key referential constraint.
- MTK does not support CREATE TABLE statements containing ON UPDATE SET NULL and ON UPDATE SET DEFAULT clauses in conversions to either DB2 or Informix Dynamic Server from a Sybase ASA source database.

TRUNCATE TABLE

TRUNCATE statements are converted into DELETE statements inside of procedures. At the top level they are translated to ALTER TABLE *t* ACTIVATE NOT LOGGED INITIALLY WITH EMPTY TABLE, which simulates the behavior of TRUNCATE.

Translation limitations

Depending upon the size of the table or the log file, the DELETE statement might fail when it is executed. A message is issued during translation indicating this.

CREATE INDEX

Sybase and SQL Server allow index names to be the same as long as they belong to different tables. DB2 UDB requires that index names are unique across the schema, and Informix Dynamic Server requires that indexes are unique for a particular user. The generated index names are changed as necessary.

A CREATE INDEX statement is, in some cases, converted into an ALTER TABLE statement. This constraint is necessary in DB2 UDB when there are foreign key constraints over the index columns.

Translation limitations

Sybase ASA For translations from Sybase SQL Anywhere to Informix Dynamic Server, the following options are not supported:

- VIRTUAL keyword
- *function-name* parameter
- IN or ON *database-name* clause

Examples

In the following example, two Sybase SQL Anywhere tables belonging to the same user each have an index with the same name. Since Informix Dynamic Server requires that indexes belonging to the same user have unique names, they are renamed. Sybase ASA

```
create index tab_index on mary.sales(id);
create index tab_index on mary.customers(id);
```

----- is translated to ----- IDS

```
CREATE INDEX tab_index ON sales(id);
CREATE INDEX tab_index1 ON customers(id);
```

Related concepts

"Object renaming" on page 129
Because of differences in rules for identifiers, some objects must be renamed during the translation. The converter generates new names in such a way that they do not conflict with any preexisting names.

Related tasks

"Changing an object name" on page 41

If an edit icon ( ) exists in the target column, you can change the name of the object.

CREATE VIEW

In DB2 UDB SQL there are more limitations on updatable views than in T-SQL.

For example, a view defined on more than one underlying table cannot be updated in DB2 UDB, whereas under certain conditions it can be updated in T-SQL. Such restrictions prevent the translation of some CREATE VIEW statements.

For conversions of Sybase ASA CREATE VIEW statements to IBM Informix Dynamic Server CREATE VIEW statements, the limitations of the SELECT statement apply to the SELECT statements used for the views.

CREATE DOMAIN

The Sybase SQL Anywhere CREATE DOMAIN statement is translated to Informix Dynamic Server as a CREATE DISTINCT TYPE statement.

Translation limitations

The following clauses of the Sybase SQL Anywhere CREATE DOMAIN syntax are not supported:

- NOT NULL
- DEFAULT
- CHECK

Related reference

"System procedures" on page 267
Some of the Sybase system procedures are translated as described here.

INSERT

Most types of INSERT statements are translated. Most inserted values in SELECT statements are also translated.

The ″INSERT table EXEC proc″ statement is converted for stored procedure calls, specific columns, and multiple result sets.

Translation limitations

Certain insert statements cannot be directly translated because of the presence of identity columns. For example, consider the following statements, which is valid in T-SQL.

```
create table t1(c1 int, c2 int identity)
insert into t1 values(1)
```

The number of columns in t1 does not match the number of columns specified in the VALUES clause. A correct translation requires that the columns for which the values are intended are explicitly stated. The following statement is correct for DB2 UDB translation.

```
insert into t1(c1) values (1)
```

Related reference

DB2 UDB Version 8 - DECLARE CURSOR statement

"INSERT INTO *table* EXECUTE *procedure*" on page 290
This insert-execute statement is translated using a cursor and a locator variable for each of the result sets generated by the procedure, which iterates over the cursor using a FETCH and INSERT statement to insert each row from the result sets into the table.

UPDATE and DELETE

In T-SQL, you can specify more than one table in the from clause of UPDATE and DELETE statements (called joined updates and deletes respectively). Some Sybase ASA DELETE clauses are not supported when DELETE statements are converted to Informix Dynamic Server

The conversion of T-SQL UPDATE and DELETE statements

In T-SQL, you can specify more than one table in the from clause of UPDATE and DELETE statements (called ″joined updates″ and ″joined deletes″ respectively).

For example, the following is a valid delete statement in T-SQL but not in DB2 UDB.

```
delete t1 from t1, t2 where t1.c1 = t2.c1
```

The semantics of this statement are to delete all rows from t1 whose c1 value matches the c1 value of any row in t2. Currently, the DELETE statement is translated as follows:

```
DELETE t1 WHERE EXISTS (SELECT * FROM t2 WHERE t1.c1 = t2.c1)
```

However, the translation of joined updates is not so straightforward. This is because of an inherent non-determinism in the definition of joined updates. For example, consider the following statement:

```
update t1 set c1 = t2.c1 from t1, t2 where t1.c2 = t2.c2
```

The statement above specifies to update the c1 column of every row in t1 whose c2 value matches the c2 value of at least one row in t2. The non-determinism arises when more than one matching row exists in t2. In this case, any of the c1 values of these rows is selected for the update. The only way to correctly simulate this behavior is by using an inefficient cursor-based implementation. Such non-determinism is often unintentional and indicative of a bug in the program. Therefore, the above statements are translated as follows:

```
UPDATE t1 set c1 = c1 +
      (SELECT t2.c1 FROM t2 WHERE t1.c2 = t2.c2)
WHERE EXISTS (SELECT * FROM t2 WHERE t1.c2 = t2.c2)

UPDATE t1
      SET c1 = (SELECT DISTINCT t2.c1
                FROM t2
                WHERE t1.c2 = t2.c2)
      WHERE EXISTS(SELECT *
                   FROM t2
                   WHERE t1.c2 = t2.c2)!
```

The above code works correctly except when non-deterministic, that is, when at most one row in t2 matches any row in t1 for update. If DB2 UDB Version 8 or later is the target, a FETCH FIRST 1 ROW ONLY clause can be used in the subselect of the set clause to eliminate the non-determinism. This is not guaranteed to return the same value for t2.c1 as T-SQL, so this should be used only if any value of t2.c2 would be considered valid in the update.

If the right-hand sides of the SET clauses do not refer to columns from any table other than the one being updated, the sub-query is not required in the SET clause. The SET clause is translated as is, and there are no non-determinism issues.

Note in the above examples that the table being updated or deleted is removed from the FROM clause in the DB2 UDB translation in order to preserve the original semantics. If the table being updated or deleted is involved in an outer join or an operation inside of parentheses in the DB2 UDB result, then the table cannot be removed and an error message is issued.

The conversion of Sybase ASA UPDATE and DELETE statements

The conversion of Sybase ASA DELETE statements to Informix Dynamic Server DELETE statements has the following limitations:

- The TOP|FIRST clause is not supported.
- The DELETE statement supports only one FROM clause.

MTK translates Sybase ASA UPDATE statements to Informix Dynamic Server UPDATE statements. However, the following clauses are not be supported:

- FIRST | Top n

- ORDER BY clause

  Related reference

  "@@ROWCOUNT" on page 296
  The global variable @@ROWCOUNT of T-SQL indicates the number of rows that were affected by the last query. The converter provides support for some uses of @@ROWCOUNT.

  "Cursors" on page 299
  The converter is able to partially translate updatable cursors.

SELECT

Top-level select statements are translated into simple select statements. The select statements inside a procedure body correspond to result sets and are translated into a cursor declaration on the select statement, followed by an open cursor statement.

Some variations in T-SQL conversions to DB2

The T-SQL syntax for extracting a value from a select statement into a variable is different than DB2 UDB syntax.

```
select @x=max(c1) from t1;

   ----- is translated
to -----

SELECT MAX(c1) INTO v_x FROM t1;
```

The T-SQL select-into statement is equivalent to a DB2 UDB table declaration followed by an insert statement.

```
select * into tt3 from tt1

   ----- is translated
to -----

CREATE SUMMARY TABLE tt3(c1) AS
(SELECT tt1.* FROM tt1 ) DEFINITION ONLY!

INSERT INTO tt3 SELECT tt1.* FROM tt1!
```

FIRST or TOP keywords in Sybase ASA SELECT statements

FIRST or TOP keywords in Sybase ASA SELECT statements are converted to equivalent keywords in DB2 and Informix Dynamic Server SELECT statements.

Unsupported keyword and clauses in Sybase ASA conversions to IBM Informix Dynamic Server

MTK translates Sybase ASA SELECT statements to Informix Dynamic Server SELECT statements. However, the DEFAULT keyword (which causes a default value to be inserted into a column) and the following clauses are not be supported:

- [ON EXISTING { ERROR | SKIP | UPDATE }]
- [WITH AUTO NAME]
-
- [WITH temporary-views]
- select-list : *| window-function OVER { window-name | window-spec }
- row limitation : START AT m

- INTO hostvar- list
- INTO variable-list

  Note: Note: This will be supported when the conversion of procedures and triggers is supported.
- The following sub clauses of the FROM CLAUSE:
  - table: [[AS] correlation-name]
  - table: [WITH (table-hint | INDEX (index-name)) | FORCE INDEX (index-name)]
  - view : [[AS] correlation-name]
  - view: [WITH (table-hint)]
  - procedure

    Note: Note: This will be supported in a future release.
  - derived table
  - lateral derived table:
- FOR { UPDATE [cursor-concurrency] }
- [WINDOW window-name AS window-spec
- [, window-name AS window-spec ...]]
- [FOR XML xml-mode]

In addition, a Sybase ASA joined table subclause of the FROM CLAUSE is supported only to the extent that Informix Dynamic Server supports joins.

GRANT

Except for execute permission on procedures, the GRANT statement is supported when used to grant permission to access database objects (GRANT *permission* ON).

Translation limitations

GRANT EXECUTE ON STORED PROCEDURE is not supported.

The GRANT statement is not translated when used to grant roles (GRANT ROLE *role* TO) or to grant permission to execute commands that do not require the ON syntax (GRANT *command* TO).

EXECUTE *string*

The EXECUTE statement followed by a string argument is translated only when it occurs inside of a procedure.

If the EXECUTE *string* statement occurs inside of a procedure then it will be translated to a DB2 UDB EXECUTE IMMEDIATE statement, but the contents of the string will not be translated. The contents must be translated by hand to assure its correctness. An appropriate message is given.

```
create procedure p1 @t varchar(25)
as
execute ('insert into '+@t+' values (3)')

   ----- is translated
to -----

CREATE PROCEDURE p1 (v_t VARCHAR(25) )
LANGUAGE SQL
```

```
BEGIN
    DECLARE SQLSTATE CHAR(5) DEFAULT '00000';
    DECLARE l_error CHAR(5) DEFAULT '00000';
    DECLARE execStr VARCHAR(4000);
    DECLARE CONTINUE HANDLER FOR NOT FOUND
        SET l_error = '00000';
    DECLARE CONTINUE HANDLER FOR SQLEXCEPTION, SQLWARNING
        BEGIN
            SET l_error = SQLSTATE;
            IF SUBSTR(l_error, 1, 1) >= '5'
               AND SUBSTR(l_error, 1, 1) <= '9' THEN
                RESIGNAL;
            END IF;
        END;

--* [600225] Unable to validate syntax of dynamic SQL in EXECUTE <string>
    SET execStr = SYB.CONCAT(SYB.CONCAT('insert into ', v_t), ' values (3)');
    EXECUTE IMMEDIATE execStr;
END!
```

INSERT INTO *table* EXECUTE *procedure*

This insert-execute statement is translated using a cursor and a locator variable for each of the result sets generated by the procedure, which iterates over the cursor using a FETCH and INSERT statement to insert each row from the result sets into the table.

A separate cursor and locator variable is used for each result set in the procedure. In certain cases the converter has difficulty determining the number of result sets generated by the procedure. In such cases the converter assumes there is only one result set. For insert-execute statements where the procedure returns more than one result set, the DB2 UDB script should be checked to ensure the statement is translated with the appropriate number of cursors.

Translation limitations

If a procedure is used in a nested procedure call, T-SQL behavior is to return the result-sets of the procedure to the procedure at the next higher nesting level until the result-set is used in a insert-exec statement. If none of the intermediate procedures use the result-set, then the result-set is returned to any client application that might have invoked this procedure.

DB2 UDB procedures are more strict about this behavior. The result set can be returned to either the calling procedure or the client application. Because the translator does not know the context in which a procedure can be used, it must assume that the result set is to be returned to the client application, and thus the insert-exec statement has no result set returned to it and is therefore not translated.

```
CREATE PROCEDURE proc1
as SELECT * from t1
go

insert into table1 execute proc1
go

   ----- is translated
to -----

CREATE PROCEDURE proc1()
LANGUAGE SQL
BEGIN

    DECLARE SQLSTATE CHAR(5) DEFAULT '00000';
    DECLARE l_sqlstatus INTEGER 0;
    DECLARE l_error CHAR(5) DEFAULT '00000';

    DECLARE temp_cursor CURSOR WITH HOLD WITH RETURN TO CLIENT
         FOR SELECT * FROM t1;

    DECLARE CONTINUE HANDLER FOR NOT FOUND
    SET l_sqlstatus = -1;

    DECLARE CONTINUE HANDLER FOR SQLEXCEPTION, SQLWARNING
       BEGIN
     SET l_error = SQLSTATE;
          SET l_sqlstatus = -1;
            IF SUBSTR(l_error, 1, 1) >= '5'
               AND SUBSTR(l_error, 1, 1) <= '9' THEN
                RESIGNAL;
            END IF;
        END;

    OPEN temp_cursor;

END!

CALL proc1();
ASSOCIATE LOCATORS (loc) WITH PROCEDURE proc1;
ALLOCATE temp_cursor CURSOR FOR RESULT SET loc;

IF ((l_error = '00000') OR SUBSTR(l_error, 1, 2) = '01') THEN
  proc1_result:
  LOOP
    FETCH FROM temp_cursor INTO temp_var;

    IF ((l_sqlstatus=-1  and substr(l_error,1,2) != '01')) THEN
      LEAVE proc1_result;
    END IF;

    INSERT INTO table1 VALUES (temp_var);

  END LOOP proc1_result;
END IF;
```

If you know that the procedure is only used for the insert and will not be nested, then you can change the RETURN TO CLIENT option to RETURN TO CALLER and manually translate the insert-exec statement. But, if you do this and the procedure is used in a nested procedure call, then the result-set will not be returned from the procedure to any application that might use the result-sets.

Related reference

"INSERT" on page 286
Most types of INSERT statements are translated. Most inserted values in SELECT statements are also translated.

IF

IF statements within procedures are converted and some top-level IF statements are converted.

Translation limitations

Top-level IF statements are converted only if the following conditions are satisfied:

- There is no ELSE part.
- The condition is of the form:

```
if exists (select <...> from sysobjects <...>)
    drop procedure proc_name
```

Or,

```
if exists (select <...> from sysdatabases <...>)
    drop table table_name
```

- All the statements in the body of the top-level IF statement are of one of the following types:
  - (CREATE|ALTER|DROP) (TABLE|VIEW|INDEX|DATABASE|ROLE)
  - (CREATE|DROP) (RULE|DEFAULT)
  - DROP (PROCEDURE|TRIGGER)
  - A system procedure call

  If only one of the statements inside the IF condition is not one of the above types, the entire IF statement will not be translated.

COMMENT ON

The statement is supported in most cases for translations to DB2 UDB. It is not supported for translations to Informix Dynamic Server.

Translation limitations

The COMMENT ON FOREIGN KEY statement and those with the IS NULL clause are not supported for DB2 UDB deployments.

RAISERROR

A RAISERROR statement is equivalent to a PRINT statement (to display a message to the user) followed by an assignment to @@ERROR.

Support for RAISERROR is as follows:

- The converter converts a RAISERROR statement that is immediately followed by a RETURN into DB2 UDB's SIGNAL SQLSTATE statement.
- The converter also handles the commonly used sequence RAISERROR; ROLLBACK; RETURN as follows. The sequence RAISERROR; ROLLBACK is not completely equivalent to the sequence ROLLBACK; RAISERROR. The latter sequence results in @@ERROR being set to the value specified in the RAISERROR statement, while the former sequence results in @@ERROR being reset to 0 (by the ROLLBACK statement). Since it is more likely that desired behavior is for @@ERROR to be set to the specified value, the converter treats the sequence RAISERROR; ROLLBACK; RETURN as if it were ROLLBACK; RAISERROR; RETURN.

Translation limitations

- The translator ignores a RAISERROR statement that is followed by a RETURN statement that returns a value, and it ignores a RAISERROR statement that is not followed by a simple RETURN statement.
- Another possible misuse of the RAISERROR statement is to use the sequence RAISERROR; RETURN (some-special=value). This sequence causes @@ERROR to be reset to zero before returning from the procedure. Hence, the converter ignores the RAISERROR statement in such a context. (Translating the statement into a SIGNAL SQLSTATE will prevent the procedure from returning the (some-special-value).)
- Any other use of RAISERROR statement effectively functions as a PRINT statement. Since DB2 UDB provides no function equivalent to a PRINT statement, the converter does not generate any DB2 UDB code for other uses of RAISERROR statements, but produces a corresponding warning message.

Related reference

"@@ERROR"
The converter attempts to preserve the T-SQL style error handling by generating a default exception handler in every procedure that catches all exceptions, and copies the error status value SQLCODE into a local variable. The exception handler is declared to be a CONTINUE exception handler, which indicates that the execution should continue on with the statement following the statement that produced the exception.

"@@SQLSTATUS" on page 296
The converter converts uses of @@SQLSTATUS (and @@FETCHSTATUS for SQL Server into a variable *l_sqlstatus* whose value is set in the CONTINUE handlers to be the same as it would have been in the T-SQL source.

@@ERROR

The converter attempts to preserve the T-SQL style error handling by generating a default exception handler in every procedure that catches all exceptions, and copies the error status value SQLCODE into a local variable. The exception handler is declared to be a CONTINUE exception handler, which indicates that the execution should continue on with the statement following the statement that produced the exception.

The basis for this approach is that in T-SQL, the execution of every statement as well as the evaluation of any IF or WHILE condition has the effect of setting the global variable @@ERROR to a suitable value indicating if the execution was successful or, if not, the problem encountered. The execution, in either case, continues on to the next statement. It is the programmer's job to check if the execution of any statement failed and to take appropriate action in case of failure. In DB2 UDB, by contrast, an error during the execution of any SQL statement has the effect of raising an exception, which will terminate the execution of the application, unless the exception is caught by a specially designated exception handler. The particular error status can be retrieved from either of the system variables SQLCODE and SQLSTATE.

By default, the converter uses SQLSTATE values when translating @@ERROR. Both SQLSATE and SQLCODE are set by DB2 UDB when an error occurs while processing a statement, but SQLSTATE has more specific error code values for specific errors. So tracking errors with SQLSTATE is more precise; however, SQLSTATE has type CHAR(5) which makes it more difficult to use when translating @@ERROR, which has type INTEGER.

When @@ERROR occurs in a comparison to a constant literal, it is translated using a local variable *l_error*, which captures the SQLSTATE value in the CONTINUE handlers. The constant literal is translated to the corresponding 5 character SQLSTATE code value. When @@ERROR occurs in any other construct, it is translated using a local variable *l_sqlcode*, which captures the SQLCODE value in the CONTINUE handlers.

For each procedure, the converter generates continue handlers to capture the SQLSTATE value and to preserve the control flow behavior of the T-SQL source code in the case of a NOT FOUND, SQLWARNING, or SQLEXCEPTION condition. For SQLWARNING and SQLEXCEPTION, the SQLSTATE value is copied to *l_error*. Since NOT FOUND conditions do not set @@ERROR to a non-zero number, the NOT FOUND continue handler does not set the *l_error* value to SQLSTATE. The converter determines when the value of *l_error* will be required after a given statement and sets *l_error* to the default '00000' value before that statement. When the converter determines that the *l_sqlcode* variable will be needed for a given procedure, then the continue handler will also set *l_sqlcode* to the SQLCODE value in the SQLWARNING and SQLEXCEPTION handlers, and it will set *l_sqlcode* to zero before the statement where the @@ERROR value is required.

Note that the handler for SQLEXCEPTION also includes the following code, which is included to prevent the procedure from continuing in the case of a serious error (SQLSTATE beginning with 5,6,7,8,or 9). In these cases, the procedure will stop and send the error to the caller.

```
IF SUBSTR(l_error, 1, 1) >= '5'
   AND SUBSTR(l_error, 1, 1) <= '9' THEN
       RESIGNAL;
END IF;
```

Translation limitations

Errors occurring during the evaluation of an IF-THEN-ELSE condition lead to different behaviors in T-SQL and the DB2 UDB SQL Procedures language. T-SQL interprets the condition as evaluating to false and executes the ELSE branch. In the SQL Procedures language an exception is raised, and the default exception handler produced by the converter will cause the execution to continue with the statement following the IF-THEN-ELSE statement. The converter does not currently handle this difference in behavior between the two languages.

Another problem with conditions is the need to reset the variable *l_error* to '00000' immediately before a condition whose @@ERROR value is required later. This is not always possible. For example, consider the condition of an ELSE IF in the statement IF *cond1* THEN *stmt1* ELSE IF *cond2 stmt2* END IF. It is impossible to insert an assignment statement to set the value of *l_error* after *cond1* but before *cond2* without completely restructuring the IF statement. Because of this the converter issues an error statement whenever the @@ERROR value of a condition is required.

Migration strategies

Testing @@ERROR after a procedure call

When @@ERROR is tested after a procedure call and the error value corresponds to a system generated error value, the SQL exception raised by the most recent statement in DB2 UDB should be propagated to the calling statement, but instead is caught by the local continue exception handler. A solution to this problem requires changing the DB2 UDB code after it is generated.

Restriction: The following solution cannot be applied to migrations to DB2 UDB for z/OS.

Move the most recently executed statement outside of the block containing the continue handler. Use a nested compound statement containing all the statements following the DECLARE SQLCODE INT statement, but excluding the last statement executed. If that statement is not the final statement in the procedure body, replace it with a GOTO and move it (with the appropriate label) to the end of the procedure (outside of the block with the continue handler).

For example:

```
create table t1 (x int primary key)
go
create table t2 (s varchar(50))
go
```

```
insert into t1 values(1)
go
create procedure p
as
insert into t1 values(2)
insert into t1 values(1)
go
create procedure p1
as
exec p
IF @@ERROR !=0
   insert into t2 values('A primary key constraint violation occurred')
go
exec p1
```

If you convert the above source SQL, you should modify the DB2 UDB output as follows (changes made to the converted output are indicated with comments):

```
CREATE TABLE t1 (
  x INT NOT NULL  PRIMARY KEY)!

CREATE TABLE t2 (
  s VARCHAR(50) NOT NULL)!

INSERT INTO t1
VALUES (1)!

CREATE PROCEDURE p()
LANGUAGE SQL
BEGIN
  DECLARE SQLSTATE CHAR(5) DEFAULT '00000';
  DECLARE l_error CHAR(5) DEFAULT '00000';
  begin                 -- ADD this "begin" here
    DECLARE CONTINUE HANDLER FOR NOT FOUND
      SET l_error = '00000';
    DECLARE CONTINUE HANDLER FOR SQLEXCEPTION, SQLWARNING
    BEGIN
      SET l_error = SQLSTATE;
      IF SUBSTR(l_error, 1, 1) >= '5'
         AND SUBSTR(l_error, 1, 1) <= '9' THEN
           RESIGNAL;
      END IF;
    END;
    INSERT INTO t1 VALUES (2);
    COMMIT;
  end;            -- ADD this "end" here
  INSERT INTO t1 VALUES (1);
  COMMIT;
END!

CREATE PROCEDURE p1()
LANGUAGE SQL
BEGIN
  DECLARE SQLSTATE CHAR(5) DEFAULT '00000'
  DECLARE l_error CHAR(5) DEFAULT '00000';
  DECLARE CONTINUE HANDLER FOR NOT FOUND
    SET l_error = '00000';
  DECLARE CONTINUE HANDLER FOR SQLEXCEPTION, SQLWARNING
    BEGIN
      SET l_error = SQLSTATE;
      IF SUBSTR(l_error, 1, 1) >= '5'
         AND SUBSTR(l_error, 1, 1) <= '9' THEN
           RESIGNAL;
      END IF;
    END;
  SET l_error = '00000';
  CALL p();
  IF l_error <> '00000' THEN
    INSERT INTO t2 VALUES (
      'A primary key constraint violation occurred'
    );
    COMMIT;
  END IF;
END!

CALL p1()!
```

Related reference

"RAISERROR" on page 292
A RAISERROR statement is equivalent to a PRINT statement (to display a message to the user) followed by an assignment to @@ERROR.

"@@SQLSTATUS"
The converter converts uses of @@SQLSTATUS (and @@FETCHSTATUS for SQL Server into a variable *l_sqlstatus* whose value is set in the CONTINUE handlers to be the same as it would have been in the T-SQL source.

@@SQLSTATUS

The converter converts uses of @@SQLSTATUS (and @@FETCHSTATUS for SQL Server into a variable *l_sqlstatus* whose value is set in the CONTINUE handlers to be the same as it would have been in the T-SQL source.

The value of @@SQLSTATUS indicates if a FETCH operation was successful, or if there was no more data, or if there was some other error. In each of these cases, one of the CONTINUE handlers will be triggered. When the value of *l_sqlstatus* is required, the converter adds assignments to the CONTINUE handlers to set the appropriate value of *l_sqlstatus*.

Related reference

"RAISERROR" on page 292
A RAISERROR statement is equivalent to a PRINT statement (to display a message to the user) followed by an assignment to @@ERROR.

"@@ERROR" on page 293
The converter attempts to preserve the T-SQL style error handling by generating a default exception handler in every procedure that catches all exceptions, and copies the error status value SQLCODE into a local variable. The exception handler is declared to be a CONTINUE exception handler, which indicates that the execution should continue on with the statement following the statement that produced the exception.

@@ROWCOUNT

The global variable @@ROWCOUNT of T-SQL indicates the number of rows that were affected by the last query. The converter provides support for some uses of @@ROWCOUNT.

The translator analyzes the use of @@ROWCOUNT and uses this information to generate the code that computes the value of @@ROWCOUNT.

In T-SQL, the SET ROWCOUNT statement can be used to limit the number of rows that are affected by subsequent queries. This particularly makes the handling of joined updates and deletes a little complex as it requires the use of cursors to simulate the behavior of these statements. The converter attempts to analyze the converted procedure to identify updates and queries that are affected by such SET ROWCOUNT statements and generates appropriate DB2 UDB code to simulate this effect.

Translation limitations

In some complex cases, the converter may be unable to generate the appropriate code for computing the value of @@ROWCOUNT. In such cases, an appropriate message is produced.

The row count analysis assumes that the value of the ROWCOUNT limit is zero at the beginning of any procedure. In particular, the converter will not correctly handle cases where one procedure sets the ROWCOUNT and calls another procedure. However, if it identifies such code, it issues a message for further investigation.

Transactions

In T-SQL, updates and changes are immediately committed unless a transaction is explicitly initiated with a BEGIN TRANSACTION statement. By contrast, a transaction is always in effect for DB2 UDB and Informix Dynamic Server when in ANSI mode. Changes are committed only when an explicit COMMIT statement is executed (or when the application terminates).

In DB2 UDB, a transaction is initiated when the application starts, and a new transaction is initiated whenever a COMMIT or ROLLBACK is executed. (There is no statement in DB2 UDB to explicitly initiate a transaction). The same is true for Informix Dynamic Server, when the database is in ANSI mode, and MTK currently supports only ANSI mode for Informix Dynamic Server targets.

When a transaction is in effect in T-SQL, the execution of a COMMIT statement does not necessarily cause the current transaction to be committed. T-SQL maintains an integer variable @@TRANCOUNT that is initially zero and is incremented by one every time a BEGIN TRANSACTION statement is executed. The execution of a COMMIT statement has no effect if the value of @@TRANCOUNT is zero at that point. If the value of @@TRANCOUNT is greater than zero when the COMMIT statement is executed, then COMMIT statement decrements the value of @@TRANCOUNT by one and if the resulting value is zero, then it actually commits the transaction.

The effect of a ROLLBACK statement is to rollback the current transaction and to reset the value of @@TRANCOUNT to be zero.

By default, the converter translates procedure bodies under the assumption that the procedure is not called within a transaction (in other words, the value of @@trancount would be 0).

If there are no transaction statements (BEGIN TRANSACTION, COMMIT, ROLLBACK) in the body of the procedure, it is assumed that all statements occur outside of any transactions. In this situation, the converter inserts a COMMIT after each statement that could make changes to the database (UPDATE, INSERT, DELETE). This simulates the T-SQL semantics under these conditions.

On the other hand, if the body of the procedure contains a transaction statement, a local variable called v_trancount is declared and initialized to zero to keep track of the transaction nesting level during the execution. The transaction statements are translated as follows in order to maintain the appropriate value in v_trancount:

T-SQL Statement	**is translated to:**
`BEGIN TRANSACTION`	`SET v_trancount = v_trancount+1;`

T-SQL Statement	is translated to:
`COMMIT`	`IF (v_trancount = 1) THEN COMMIT; END IF; IF (v_trancount>0) THEN SET v_trancount = v_trancount -1; END IF;`
`ROLLBACK`	`ROLLBACK; SET v_trancount = 0;`

Furthermore, the translations of UPDATE, INSERT, and DELETE are each succeeded by the following:

```
IF (v_trancount=0)
  THEN COMMIT; END IF;
```

In this way, a commit occurs only outside of a transaction context, or when a COMMIT statement occurs at the end of the outermost nested transaction (when @@trancount becomes zero).

Translation limitations

It is possible that the default assumption (that a procedure is not called within a transaction) might be false. The procedure might be called from within a transaction in another stored procedure or an application routine. The converter is unable to determine the context in which a procedure is called.

Migration strategies

You can, however, solve the limitation by indicating to the converter which procedures may be called from within a transaction:

1. After an initial conversion, go to the Refine page of MTK and click the Source page (SQL Server or Sybase).
2. Select a procedure that can be called from with a transaction and select the **Add Transaction** check box in the DB2 column of the Properties table. Do this for each procedure to which this characteristic applies.
3. Go back to the Convert page and click **Convert** again.

When the **Add Transaction** option is set for a procedure, the converter will translate the procedure with an extra parameter in DB2 UDB to store the value of @@trancount (the current transaction nesting level) as follows:

```
CREATE PROCEDURE procName @a int
as INSERT INTO t1 VALUES(5)
go

   ----- is translated
to -----

CREATE PROCEDURE procName (IN v_a INTEGER,
   INOUT v_trancount INTEGER)
LANGUAGE SQL
BEGIN
   INSERT INTO t1 VALUES(5)
   IF (v_trancount=0) THEN
 COMMIT;
   ENDIF;
END
```

The converter will also add an appropriate argument to all the calls to this procedure from any other stored procedure it converts. If the call occurs inside of a routine, it will use v_trancount as the argument. Otherwise, it will use zero.

```
EXECUTE procName 5;

   ----- is translated
to -----

CALL procName(5,v_trancount);  or CALL procName(5,0);
```

All calls to this procedure from application programs will need to be adjusted manually.

Cursors

The converter is able to partially translate updatable cursors.

DB2 UDB has more restrictions on updatable cursors than T-SQL. For example, in T-SQL, an updatable cursor may be defined using a select statement which uses more than one table in the from clause as long as the columns to be updated all belong to the same table. An update on such a cursor is very similar to the joined update statement, and DB2 UDB does not support either.

DB2 UDB automatically closes all open cursors whenever there is a ROLLBACK or GOTO statement, Sybase does not.

To manipulate the result set returned by a stored procedure, the converter attempts to introduce a cursor.

Translation limitations

The converter does not support cursor FETCH statements that do not fetch into a variable.

Scrollable options for FETCH statements are accepted but not converted (DB2 UDB does not have scrollable cursors). NEXT converts correctly; other options generate a warning message.

Related reference

"UPDATE and DELETE" on page 286
In T-SQL, you can specify more than one table in the from clause of UPDATE and DELETE statements (called joined updates and deletes respectively). Some Sybase ASA DELETE clauses are not supported when DELETE statements are converted to Informix Dynamic Server

Triggers

The converter provides limited support for triggers.

A T-SQL trigger might be associated with more than one type of INSERT, UPDATE, or DELETE event. In DB2 UDB, however, a trigger can be associated with only one type of event. The converter translates T-SQL triggers that are associated with multiple events by creating a separate trigger definition for each associated event.

In T-SQL, you can reference the *inserted* table in a DELETE trigger and the *deleted* table in an INSERT trigger. (These tables will be empty when the corresponding triggers are activated.) In DB2 UDB, however, a DELETE trigger cannot reference the new table and an INSERT trigger cannot reference the old table. The converter translates references to the *inserted* table in a DELETE trigger and the *deleted* table in an INSERT trigger into a query that evaluates to an empty table (with the correct set of columns). In some cases, the statement containing references to these tables might be redundant and you might be able simplify the trigger by deleting the statement.

Translation limitations

In particular, cursors or transaction statements such as ROLLBACK or COMMIT cannot be used in DB2 UDB triggers, and the converter will not convert any of these types of constructs.

One exception is that the converter converts the sequence ROLLBACK; RAISERROR; RETURN into a SIGNAL SQLSTATE statement which provides similar functionality. Such a translation is not complete, however, because the SIGNAL SQLSTATE has the effect of only undoing the effects of the trigger as well as the original operation that caused the trigger to fire. Therefore, the code that contains the original operation must then rollback the transaction if desired. Such modification must be done manually.

The update() condition is not supported. Unlike in DB2 UDB, multiple update() conditions can be made in triggers on tables in Sybase. The update() conditions represented in Sybase can be split into separate DB2 UDB triggers, but the trigger logic can easily become very complex and can produce unwanted side effects when certain unexpected conditions exist. To provide equivalent functionality in DB2 UDB, manual evaluation and redesign of the trigger logic is necessary.

T-SQL INSTEAD OF triggers are not supported. DB2 UDB Version 8 INSTEAD OF triggers differ significantly in behavior and a direct translation would not be valid.

Temporary tables

For translations to Informix Dynamic Server, MTK supports the DECLARE LOCAL TEMPORARY TABLE statement, not the DECLARE GLOBAL TEMPORARY TABLE statement.

Translation limitations

For translations to DB2, the DECLARE LOCAL TEMPORARY TABLE statement is not supported. For translations to DB2 UDB, SELECT statements in the DECLARE GLOBAL TEMPORARY TABLE statement cannot contain local variables.

In DB2 UDB, SELECT statements in the DECLARE GLOBAL TEMPORARY TABLE statement cannot contain local variables. Local variables are commonly used in the WHERE and HAVING clauses of SELECT statements. Because these clauses do not add any necessary information for the global temporary table definition, the converter removes them from the definition.

However, if a SELECT element involves a local variable, the translation is erroneous and must be fixed by hand. You can fix the problem manually by replacing all occurrences of variables in the select list with ″CAST(NULL AS var_type)″ where var_type is the data type of the variable begin replaced.

Sybase ASA DECLARE TEMPORARY TABLE statements are translated to Informix Dynamic Server CREATE TEMP TABLE statements.

Related reference

“CREATE TABLE” on page 212
In a few situations, differences can occur in translations from Oracle to DB2 UDB and IBM Informix Dynamic Server.

Table variables

SQL Server allows variables to be declared of type table. The converter translates these table variables to global temporary tables.

```
create table emp(empName varchar(20), empInt int)
go

create procedure P
as
 declare @T TABLE (aName varchar(20))
 insert @T select empName from emp
go

drop procedure P
drop table emp
go

   ----- is translated
to -----

CREATE TABLE emp (
  empName VARCHAR(20) NOT NULL,
  empInt INT NOT NULL)!


CREATE PROCEDURE P()
DYNAMIC RESULT SETS 1
LANGUAGE SQL
BEGIN
  DECLARE SQLCODE INT;
  DECLARE l_sqlcode INT DEFAULT 0;
  DECLARE CONTINUE HANDLER FOR SQLEXCEPTION,SQLWARNING,NOT FOUND
    SET l_sqlcode = SQLCODE;
  DECLARE GLOBAL TEMPORARY TABLE SESSION.v_T
  (
    aName VARCHAR(20) NOT NULL
  ) WITH REPLACE ON COMMIT PRESERVE ROWS NOT LOGGED;
  INSERT INTO SESSION.v_T
  SELECT empName
  FROM emp;
  COMMIT;
END!


DROP PROCEDURE P!
DROP TABLE emp!
```

Stored procedures

The converter accepts top-level procedure calls with constant arguments.

In DB2 UDB, variables cannot be declared to have a UDT type. Therefore, in the declaration the UDT will be replaced by its base type (no casting).

Translation limitations

In T-SQL, calls can be made to stored procedures compiled on remote procedures. For example:

```
exec server1.master.dbo.proc1
```

The statement calls the procedure proc1 in the *master* database on server server1 for the user dbo. In DB2 UDB, it is not possible to call procedures on remote servers. Such statements cannot therefore be translated.

The converter does not support a call to an unknown procedure with named arguments.

The use of wildcard characters in a default parameter is not supported.

The maximum number of parameters that can be passed on a procedure in DB2 UDB is 90; Sybase procedures can pass 255 parameters.

Variable name conflicts can occur when used in different scopes. Since during conversion, variables are moved from different scopes to the same scope in the generated DB2 UDB procedure (all variables are moved to the outermost begin-end block), variable renaming might be necessary to avoid conflicts.

Functions

The converter provides limited support for functions.

T-SQL function	DB2 UDB translation
Scalar functions	Scalar functions
Inline table-valued functions	Table functions
Multi-statement table-valued functions	[not supported]

Translation limitations

Cursors and transaction statements such as ROLLBACK or COMMIT cannot be used in DB2 UDB functions. But the converter translates functions to DB2 UDB procedures where a procedure would otherwise allow such statements.

Compatibility library (MSSQL and SYB functions)

The following functions simulate the behavior of certain Sybase and Microsoft SQL Server built-in functions that do not have equivalents in the target database server. If a function is used in the translated script, then it is bound to the database during its deployment to DB2 UDB or Informix Dynamic Server.

The functions are prefixed with SYB and MSSQL and are written as SQL UDFs whenever possible and reasonable; otherwise, they are written in Java in a separate file. Regardless, these UDFs are documented here and their code prototypes are included in the UDF file in the MTK installation directory. The descriptions address the differences between the Sybase or SQL Server and the DB2 UDB built-in functions, highlight any restrictions that apply to conversion, and show common usage examples where applicable.

Related reference

"Built-in functions" on page 268
Microsoft SQL Server and Sybase functions are mapped directly to the DB2 UDB equivalent where available; and Sybase SQL Anywhere functions are mapped to the IBM Informix Dynamic Server equivalent where available.

Auxiliary functions

These functions simulate the behavior of certain Sybase and Microsoft SQL Server auxiliary functions.

SYB.minv, MSSQL.minv, SYB.maxv, MSSQL.maxv Auxiliary functions that compute minimum and maximum integer values, used by other functions in the compatibility library.

Built-in functions

These functions simulate the behavior of certain Sybase and SQL Server built-in functions. In many cases the differences are due to the source and target database giving different results for null arguments, or returning different values for certain edge case arguments.

SYB.calyearofweek, MSSQL.calyearofweek
Simulates the Sybase and SQL Server function datepart(calyearofweek,date). (The other datepart functions are converted inline by the converter.)

```
--| select datepart(calyearofweek,02/02/2002)

SELECT MSSQL.calyearofweek(MSSQL.double_to_date(02 / 02 / 2002))
FROM TABLE (VALUES 1) AS temp_table!
```

SYB.charindex(e1,e2), MSSQL.charindex(e1,e2)
Simulates the Sybase and SQL Server built-in function charindex. In Sybase and SQL Server, if only one argument is null, charindex returns 0. (in DB2 UDB, if either argument is null, locate returns null). This function simulates the Sybase and SQL Server behavior.

SYB.charindexBin(e1,e2), MSSQL.charindexBin(e1,e2)
Simulates the Sybase and SQL Server built-in function charindexbin. In Sybase and SQL Server, if only one argument is null, charindex returns 0. (in DB2 UDB, if either argument is null, locate returns null). This function simulates the Sybase and SQL Server behavior.

```
--| select charindex(0x70, 0x49424d20436f72706f726174696f6e)

SELECT SYB.charindexBin(X'70', X'49424d20436f72706f726174696f6e')
FROM TABLE (VALUES 1) AS temp_table!
```

SYB.concat(x,y), MSSQL.concat(x,y)
Simulates the Sybase and SQL Server concatenation operator (+). In Sybase and SQL Server, if only one argument is null, concatenation returns the other argument (in DB2 UDB, if either argument is null, concatenation returns null). This function simulates the Sybase and SQL Server behavior.

```
--| select '|' + convert(char, 3) + '|'

SELECT CAST(SYB.concat('|', CAST(3 AS CHAR(30))) AS
   VARCHAR(31)) || '|'
FROM TABLE (VALUES 1) AS temp_table!
```

SYB.datediffday, MSSQL.datediffday
Simulates the Sybase and SQL Server function datediff(day,d1,d2).

```
--| create proc test_dt_datediff_day
--| as
--| select datediff(day,d1,d4), datediff(day,d2,d3)
        from basetbl_date order by id
--| select datediff(dd,d1,d4), datediff(dd,d2,d3)
        from basetbl_date order by id

CREATE PROCEDURE test_dt_datediff_day()
DYNAMIC RESULT SETS 1
LANGUAGE SQL
BEGIN
  DECLARE SQLCODE INT;
  DECLARE l_sqlcode INT DEFAULT 0;
  DECLARE temp_cursor CURSOR WITH HOLD WITH RETURN TO CLIENT FOR
  SELECT SYB.datediffday(d1, d4), SYB.datediffday(d2, d3)
  FROM basetbl_date
  ORDER BY id ;
  DECLARE temp_cursor1 CURSOR WITH HOLD WITH RETURN TO CLIENT FOR
  SELECT SYB.datediffday(d1, d4), SYB.datediffday(d2, d3)
  FROM basetbl_date
  ORDER BY id ;
  DECLARE CONTINUE HANDLER FOR SQLEXCEPTION,SQLWARNING,NOT FOUND
    SET l_sqlcode = SQLCODE;

  OPEN temp_cursor;
  OPEN temp_cursor1;
END!
```

SYB.datediffhour, MSSQL.datediffhour
Simulates the Sybase and SQL Server function datediff(hour,d1,d2).

SYB.datediffmilli, MSSQL.datediffmilli
Simulates the Sybase and SQL Server function datediff(millisecond,d1,d2).

SYB.datediffminute, MSSQL.datediffminute
Simulates the Sybase and SQL Server function datediff(minute,d1,d2).

SYB.datediffmonth, MSSQL.datediffmonth
Simulates the Sybase and SQL Server function datediff(month,d1,d2).

SYB.datediffquarter, MSSQL.datediffquarter
Simulates the Sybase and SQL Server function datediff(quarter,d1,d2).

SYB.datediffsecond, MSSQL.datediffsecond
Simulates the Sybase and SQL Server function datediff(second,d1,d2).

SYB.datediffweek, MSSQL.datediffweek
Simulates the Sybase and SQL Server function datediff(week,d1,d2).

SYB.datediffwkday, MSSQL.datediffwkday
Simulates the Sybase and SQL Server function datediff(wkday,d1,d2).

SYB.datediffyear, MSSQL.datediffyear
Simulates the Sybase and SQL Server function datediff(year,d1,d2).

MSSQL.dateminus
Simulates the SQL Server subtraction of dates.

SYB.datename, MSSQL.datename
Simulates the Sybase and SQL Server built-in function datename using the DB2 UDB built-in functions that correspond to the date part.

```
--| create proc test_dt_datename_month
--| as
--| select datename(month,d1), datename(mm,d2)
      from basetbl_date

CREATE PROCEDURE test_dt_datename_month()
DYNAMIC RESULT SETS 1
LANGUAGE SQL
BEGIN
  DECLARE SQLCODE INT;
  DECLARE l_sqlcode INT DEFAULT 0;
  DECLARE temp_cursor CURSOR WITH HOLD WITH RETURN TO CLIENT FOR
  SELECT SYB.datename('month', d1), SYB.datename('month', d2)
  FROM basetbl_date
  DECLARE CONTINUE HANDLER FOR SQLEXCEPTION,SQLWARNING,NOT FOUND
    SET l_sqlcode = SQLCODE;

  OPEN temp_cursor;
END!
```

MSSQL.dateplus
Simulates the SQL Server addition of dates.

SYB.emptytonull, MSSQL.emptytonull
A wrapper that returns string values. It maps the empty string, which is interpreted as null in Sybase and SQL Server, to a DB2 UDB null.

MSSQL.getutcdate
Obtains the UTC date.

SYB.hex_to_int(s), MSSQL.hex_to_int(s)
Simulates the Sybase and SQL Server built-in function hextoint.

```
--| create proc test_hextoint
--| as
--| select hextoint("0x00000202"), hextoint("0x22000202")

CREATE PROCEDURE test_hextoint()
DYNAMIC RESULT SETS 1
LANGUAGE SQL
BEGIN
  DECLARE SQLCODE INT;
  DECLARE l_sqlcode INT DEFAULT 0;
  DECLARE temp_cursor CURSOR WITH HOLD WITH RETURN TO CLIENT FOR
  SELECT SYB.hex_to_int('0x00000202'), SYB.hex_to_int('0x22000202')
  FROM TABLE (VALUES 1) AS temp_table;
  DECLARE CONTINUE HANDLER FOR SQLEXCEPTION,SQLWARNING,NOT FOUND
    SET l_sqlcode = SQLCODE;

  OPEN temp_cursor;
END!
```

SYB.int_to_hex(s), MSSQL.int_to_hex(s)
Simulates the Sybase and SQL Server built-in function inttohex Implemented as Java UDFs. Nothing similar in DB2 UDB (hex is platform dependent).

```
--| create proc test_inttohex
--| as
--| select si,int, inttohex(si), inttohex(int)  from basetbl_test
        where si is null
--| select si,int, inttohex(si), inttohex(int)  from basetbl_test
        where si is not null

CREATE PROCEDURE test_inttohex()
DYNAMIC RESULT SETS 1
LANGUAGE SQL
BEGIN
  DECLARE SQLCODE INT;
  DECLARE l_sqlcode INT DEFAULT 0;
  DECLARE temp_cursor CURSOR WITH HOLD WITH RETURN TO CLIENT FOR
  SELECT si, int, SYB.int_to_hex(si), SYB.int_to_hex(int)
  FROM basetbl_test
  WHERE si IS NULL;
  DECLARE temp_cursor1 CURSOR WITH HOLD WITH RETURN TO CLIENT FOR
  SELECT si, int, SYB.int_to_hex(si), SYB.int_to_hex(int)
  FROM basetbl_test
  WHERE si IS NOT NULL;
  DECLARE CONTINUE HANDLER FOR SQLEXCEPTION,SQLWARNING,NOT FOUND
    SET l_sqlcode = SQLCODE;

  OPEN temp_cursor;
  OPEN temp_cursor1;
END!
```

MSSQL.isdate
Determine if VARCHAR is a date. Used with datetime conversion functions.

```
--| create procedure sqlspecial_isdate
--| as
--| select isdate('kjhklh'), isdate('11/07/2000 10:29:50')

CREATE PROCEDURE sqlspecial_isdate()
DYNAMIC RESULT SETS 1
LANGUAGE SQL
BEGIN
  DECLARE SQLCODE INT;
  DECLARE l_sqlcode INT DEFAULT 0;
  DECLARE temp_cursor CURSOR WITH HOLD WITH RETURN TO CLIENT FOR
  SELECT MSSQL.isdate('MDY', 'kjhklh'),
         MSSQL.isdate('MDY', '11/07/2000 10:29:50')
  FROM TABLE (VALUES 1) AS temp_table;
  DECLARE CONTINUE HANDLER FOR SQLEXCEPTION,SQLWARNING,NOT FOUND
    SET l_sqlcode = SQLCODE;

  OPEN temp_cursor;
END!
```

SYB.islike, MSSQL.islike

Used for like.

```
--| create procedure db2issue_like_predicate
--| as
--| select distinct au_lname, authors.city from publishers, authors
--| where au_lname like "[ABC]%" and
--| authors.city not in (select city from publishers
--| where publishers.city = authors.city)

CREATE PROCEDURE db2issue_like_predicate()
DYNAMIC RESULT SETS 1
LANGUAGE SQL
BEGIN
  DECLARE SQLCODE INT;
  DECLARE l_sqlcode INT DEFAULT 0;
  DECLARE temp_cursor CURSOR WITH HOLD WITH RETURN TO CLIENT FOR
  SELECT  DISTINCT au_lname, authors.city
  FROM publishers, authors
  WHERE SYB.isLike(au_lname, '[ABC]%', '') = 1 AND
        authors.city NOT IN
        (SELECT city
         FROM publishers
         WHERE publishers.city = authors.city);
  DECLARE CONTINUE HANDLER FOR SQLEXCEPTION,SQLWARNING,NOT FOUND
    SET l_sqlcode = SQLCODE;

  OPEN temp_cursor;
END!
```

MSSQL.isnumeric

Simulates SQL Server isnumeric function.

```
--| create procedure sqlspecial_isnumeric
--| as
--| select isnumeric(null), isnumeric(''), isnumeric('abc')

CREATE PROCEDURE sqlspecial_isnumeric()
DYNAMIC RESULT SETS 1
LANGUAGE SQL
BEGIN
  DECLARE SQLCODE INT;
  DECLARE l_sqlcode INT DEFAULT 0;
  DECLARE temp_cursor CURSOR WITH HOLD WITH RETURN TO CLIENT FOR
  SELECT MSSQL.isnumeric(CAST(NULL AS VARCHAR(1))), MSSQL.isnumeric(''),
         MSSQL.isnumeric('abc')
  FROM TABLE (VALUES 1) AS temp_table;
  DECLARE CONTINUE HANDLER FOR SQLEXCEPTION,SQLWARNING,NOT FOUND
```

```
    SET l_sqlcode = SQLCODE;

  OPEN temp_cursor;
END!
```

MSSQL.len

Simulates SQL Server len function.

```
--| create procedure test_bin_len
--| as
--| select len(0xAABBCC0000)

CREATE PROCEDURE test_bin_len()
DYNAMIC RESULT SETS 1
LANGUAGE SQL
BEGIN
  DECLARE SQLCODE INT;
  DECLARE l_sqlcode INT DEFAULT 0;
  DECLARE temp_cursor CURSOR WITH HOLD WITH RETURN TO CLIENT FOR
  SELECT MSSQL.len(X'AABBCC0000')
  FROM TABLE (VALUES 1) AS temp_table;
  DECLARE CONTINUE HANDLER FOR SQLEXCEPTION,SQLWARNING,NOT FOUND
    SET l_sqlcode = SQLCODE;

  OPEN temp_cursor;
END!
```

MSSQL.nchar

Simulates SQL Server nchar function.

SYB.patindex, MSSQL.patindex

Simulates the Sybase and SQL Server patindex on simple string arg (with no internal wildcard characters) using DB2 UDB's locate. If the second argument of patindex is null, patindex returns 0 (locate would return null).

```
--| create proc test_str_patindex
--| as
--| select patindex("%str%",ch20), patindex("%str%",vc20)
    from basetbl_test

CREATE PROCEDURE test_str_patindex()
DYNAMIC RESULT SETS 1
LANGUAGE SQL
BEGIN
  DECLARE SQLCODE INT;
  DECLARE l_sqlcode INT DEFAULT 0;
  DECLARE temp_cursor CURSOR WITH HOLD WITH RETURN TO CLIENT FOR
  SELECT MSSQL.patindex('%str%', ch20), MSSQL.patindex('%str%', vc20)
  FROM basetbl_test
  DECLARE CONTINUE HANDLER FOR SQLEXCEPTION,SQLWARNING,NOT FOUND
    SET l_sqlcode = SQLCODE;

  OPEN temp_cursor;
END!
```

SYB.pi(), MSSQL.pi()

Simulates the Sybase and SQL Server built-in function (constant) pi.

```
--| create proc test_math_pi
--| as
--| select convert(int,pi()*10e6)

CREATE PROCEDURE test_math_pi()
DYNAMIC RESULT SETS 1
LANGUAGE SQL
BEGIN
  DECLARE SQLCODE INT;
  DECLARE l_sqlcode INT DEFAULT 0;
  DECLARE temp_cursor CURSOR WITH HOLD WITH RETURN TO CLIENT FOR
  SELECT CAST((MSSQL.pi() * 10e6) AS INT)
  FROM TABLE (VALUES 1) AS temp_table;
  DECLARE CONTINUE HANDLER FOR SQLEXCEPTION,SQLWARNING,NOT FOUND
    SET l_sqlcode = SQLCODE;

  OPEN temp_cursor;
END!
```

MSSQL.proc_id

Retrieves the ID from the current procedure in the catalog. Similar to @@proc_id in SQL Server.

```
MSSQL.proc_id!
```

MSSQL.proc_info

Retrieves the information from the current procedure, which is identified by MSSQL.proc_id. The schema, name, specific name, and ID are returned in a table.

```
MSSQL.proc_info!
```

MSSQL.procedure_name

Retrieves the name of the procedure that has the specified ID.

```
MSSQL.procedure_name(237)!
```

MSSQL.quotename

Simulates SQL Server quotename function.

```
--| create procedure sqlspecial_quotename
--| as
--| declare @c char(30)
--| set @c='quotename from variable'
--| select quotename(@c)
--| select quotename('quotename from literal')

CREATE PROCEDURE sqlspecial_quotename()
DYNAMIC RESULT SETS 1
LANGUAGE SQL
BEGIN
  DECLARE SQLCODE INT;
  DECLARE v_c CHAR(30);
  DECLARE l_sqlcode INT DEFAULT 0;
  DECLARE temp_cursor CURSOR WITH HOLD WITH RETURN TO CLIENT FOR
  SELECT MSSQL.quotename(v_c)
  FROM TABLE (VALUES 1) AS temp_table;
  DECLARE temp_cursor1 CURSOR WITH HOLD WITH RETURN TO CLIENT FOR
  SELECT MSSQL.quotename('quotename from literal')
  FROM TABLE (VALUES 1) AS temp_table1;
  DECLARE CONTINUE HANDLER FOR SQLEXCEPTION,SQLWARNING,NOT FOUND
    SET l_sqlcode = SQLCODE;

  SET v_c = 'quotename from variable';
  OPEN temp_cursor;
  OPEN temp_cursor1;
END!
```

MSSQL.remid

Simulates SQL Server remid function.

MSSQL.replace

Simulates SQL Server replace function.

SYB.replicate(x,n), MSSQL.replicate(x,n)

Simulates the Sybase and SQL Server built-in function replicate. In Sybase and SQL Server, if only n is negative, replicate returns null. (in DB2 UDB, repeat gives an error if n is negative). This function simulates the Sybase and SQL Server behavior. This function works only when x is less than the

DB2 UDB page size (default is 4K). If x is between 4K and 8K you can manually modify the resulting code to use the **SYB.replicate8k** or **MSSQL.replicate8k** function, and ensure you change the DB2 UDB page size before deploying the code.

```
--| create proc test_bin_replicate
--| as
--| select replicate(0x41,10)

CREATE PROCEDURE test_bin_replicate()
DYNAMIC RESULT SETS 1
LANGUAGE SQL
BEGIN
  DECLARE SQLCODE INT;
  DECLARE l_sqlcode INT DEFAULT 0;
  DECLARE temp_cursor CURSOR WITH HOLD WITH RETURN TO CLIENT FOR
  SELECT MSSQL.replicate(CAST(MSSQL.bin_to_varchar(X'41', 2)
     AS VARCHAR(2)), 10)
  FROM TABLE (VALUES 1) AS temp_table;
  DECLARE CONTINUE HANDLER FOR SQLEXCEPTION,SQLWARNING,NOT FOUND
    SET l_sqlcode = SQLCODE;

  OPEN temp_cursor;
END!
```

SYB.reverse(s), MSSQL.reverse(s)

Simulates the Sybase and SQL Server built-in function reverse. Implemented in Java.

```
--| create proc test_str_reverse
--| as
--| select reverse(vc20) from basetbl_test

CREATE PROCEDURE test_str_reverse()
DYNAMIC RESULT SETS 1
LANGUAGE SQL
BEGIN
  DECLARE SQLCODE INT;
  DECLARE l_sqlcode INT DEFAULT 0;
  DECLARE temp_cursor CURSOR WITH HOLD WITH RETURN TO CLIENT FOR
  SELECT CAST(SYB.reverse(vc20) AS VARCHAR(20))
  FROM basetbl_test
  DECLARE CONTINUE HANDLER FOR SQLEXCEPTION,SQLWARNING,NOT FOUND
    SET l_sqlcode = SQLCODE;

  OPEN temp_cursor;
END!
```

SYB.right(x,n), MSSQL.right(x,n)

Simulates the Sybase and SQL Server built-in function right. Avoids DB2 UDB's padding of spaces if length(x) is greater than n. Also gives null instead of an error state when n is negative.

```
--| create proc test_str_right
--| as
--| select right(vc20,5), right(ch20,0), right(ch1, -1) from basetbl_test

CREATE PROCEDURE test_str_right()
DYNAMIC RESULT SETS 1
LANGUAGE SQL
BEGIN
  DECLARE SQLCODE INT;
  DECLARE l_sqlcode INT DEFAULT 0;
  DECLARE temp_cursor CURSOR WITH HOLD WITH RETURN TO CLIENT FOR
  SELECT SYB.right(vc20, 5), SYB.right(ch20, 0), SYB.right(ch1, -1)
  FROM basetbl_test
  DECLARE CONTINUE HANDLER FOR SQLEXCEPTION,SQLWARNING,NOT FOUND
```

```
    SET l_sqlcode = SQLCODE;

  OPEN temp_cursor;
END!
```

SYB.round(s,k), MSSQL.round(s,k)

Simulates the Sybase and SQL Server built-in function round. The DB2 UDB built-in round function converts decimals to floating point numbers and returns floating point values as the result (potentially losing precision and accuracy). In Sybase and SQL Server, decimal arguments to round give decimal results. In order to accurately simulate this in DB2 UDB, a Java UDF interprets the decimal as a Java BigDecimal type and performs the round accurately. The interface between DB2 UDB UDFs and their Java implementations do not perform this mapping of DB2 UDB decimals to BigInts directly, so the decimal is converted to a character string in DB2 UDB and passed to the function as a varchar that is mapped to a Java String, which is converted to a BigDecimal. The result is returned as a string, which is cast into the appropriate DB2 UDB decimal type by the code generated by the converter. The floating point and integer instances of round are implemented as Java UDFs (that also use the BigDecimal rounding function) for consistency.

MSSQL.servername

Simulates SQL Server servername function.

```
--| create procedure the_sname
--| as
--| select @@servername

CREATE PROCEDURE the_sname()
DYNAMIC RESULT SETS 1
LANGUAGE SQL
BEGIN
  DECLARE SQLCODE INT;
  DECLARE l_sqlcode INT DEFAULT 0;
  DECLARE l_rowcount INT;
  DECLARE temp_cursor CURSOR WITH HOLD WITH RETURN TO CLIENT FOR
  SELECT MSSQL.servername()
  FROM TABLE (VALUES 1) AS temp_table;
  DECLARE CONTINUE HANDLER FOR SQLEXCEPTION,SQLWARNING,NOT FOUND
    SET l_sqlcode = SQLCODE;

  OPEN temp_cursor;
END!
```

MSSQL.square

Simulates SQL Server square function.

```
--| create proc test_math_square
--| as
--| select square(si) from basetbl_test order by id

CREATE PROCEDURE test_math_square()
DYNAMIC RESULT SETS 1
LANGUAGE SQL
BEGIN
  DECLARE SQLCODE INT;
  DECLARE l_sqlcode INT DEFAULT 0;
  DECLARE temp_cursor CURSOR WITH HOLD WITH RETURN TO CLIENT FOR
  SELECT MSSQL.square(si)
  FROM basetbl_test
  ORDER BY id ;
  DECLARE CONTINUE HANDLER FOR SQLEXCEPTION,SQLWARNING,NOT FOUND
```

```
    SET l_sqlcode = SQLCODE;

  OPEN temp_cursor;
END!
```

SYB.str(x,n,k), MSSQL.str(x,n,k)

Simulates the Sybase and SQL Server built-in function str. The str function raises issues similar to those of round. The function does take integers and decimals, in addition to floats (the documentation only mentions floats) and for integers and decimals no precision is lost in the result (as would be the case if they were converted to floats first). So there are instances for floating points, integers, and decimals (which are passed through varchar arguments as in round). The third argument is optional, and the second is optional when there is not a third, requiring three instances for each of the type groups, for a total of 9 instances for this function.

```
--| create proc test_str_str
--| as
--| select str(int*dec10_5,12,5)  from basetbl_test order by id

CREATE PROCEDURE test_str_str()
DYNAMIC RESULT SETS 1
LANGUAGE SQL
BEGIN
  DECLARE SQLCODE INT;
  DECLARE l_sqlcode INT DEFAULT 0;
  DECLARE temp_cursor CURSOR WITH HOLD WITH RETURN TO CLIENT FOR
  SELECT SYB.str(int * dec10_5, 12, 5)
  FROM basetbl_test
  ORDER BY id ;
  DECLARE CONTINUE HANDLER FOR SQLEXCEPTION,SQLWARNING,NOT FOUND
    SET l_sqlcode = SQLCODE;

  OPEN temp_cursor;
END!
```

SYB.stuff(x1,st,len,x2), MSSQL.stuff(x1,st,len,x2)

Simulates the Sybase and SQL Server built-in function stuff. In Sybase and SQL Server, edge conditions such as st (the start index) <= 0, len (how many characters to delete) < 0 or st > length of x1 return null (in DB2 UDB's insert function these give error states). If x2 is null, Sybase and SQL Server treat this as the empty string, so SYB.stuff replaces x2 with the empty string when it is null.

```
--| create proc test_str_stuff
--| as
--| select stuff(ch20,6,8,"knowledge"), stuff(vc20,0,7,"useless"),
--|        stuff(ch1,2,7,"useless") from basetbl_test order by id

CREATE PROCEDURE test_str_stuff()
DYNAMIC RESULT SETS 1
LANGUAGE SQL
BEGIN
  DECLARE SQLCODE INT;
  DECLARE l_sqlcode INT DEFAULT 0;
  DECLARE temp_cursor CURSOR WITH HOLD WITH RETURN TO CLIENT FOR
  SELECT CAST(SYB.stuff(ch20, 6, 8, 'knowledge') AS VARCHAR(29)),
         CAST(SYB.stuff(vc20, 0, 7, 'useless') AS VARCHAR(27)),
         CAST(SYB.stuff(ch1, 2, 7, 'useless') AS VARCHAR(8))
  FROM basetbl_test
  ORDER BY id ;
  DECLARE CONTINUE HANDLER FOR SQLEXCEPTION,SQLWARNING,NOT FOUND
    SET l_sqlcode = SQLCODE;

  OPEN temp_cursor;
END!
```

SYB.substring(x,a,b), MSSQL.substring(x,a,b)

Simulates the Sybase and SQL Server built-in function substring. In Sybase and SQL Server if a (the starting position) <= 0 or length(x) < a, substring returns null (DB2 UDB's substr gives an error). If b (the >s <...>) number of characters to return) is 0, Sybase and SQL Server return null (its representation of the empty string—DB2 UDB returns the empty string). If a+b > length(x), Sybase and SQL Server return the string starting at a, (DB2 UDB gives an error). This function simulates the Sybase and SQL Server behavior.

```
--| create proc test_str_substring
--| as
--| select substring(ch20,6,8), substring(ch20,0,8),
--|        substring(ch20,-3,8) from basetbl_test order by id

CREATE PROCEDURE test_str_substring()
DYNAMIC RESULT SETS 1
LANGUAGE SQL
BEGIN
  DECLARE SQLCODE INT;
  DECLARE l_sqlcode INT DEFAULT 0;
  DECLARE temp_cursor CURSOR WITH HOLD WITH RETURN TO CLIENT FOR
  SELECT CAST(SYB.substring(ch20, 6, 8) AS VARCHAR(20)),
           CAST(SYB.substring(ch20, 0, 8) AS VARCHAR(20)),
           CAST(SYB.substring(ch20, -3, 8) AS VARCHAR(20))
  FROM basetbl_test
  ORDER BY id ;
  DECLARE CONTINUE HANDLER FOR SQLEXCEPTION,SQLWARNING,NOT FOUND
    SET l_sqlcode = SQLCODE;

  OPEN temp_cursor;
END!
```

SYB.timediff, MSSQL.timediff

An auxiliary function that returns a DB2 UDB timestamp duration equal to difference of the time portions only (ignores the date but includes the microseconds) : hhmmss.nnnnnn It is used in the functions that compute time differences.

SYB.tsequal, MSSQL.tsequal

Simulates the Sybase and SQL Server tsequal function.

```
--| create proc updates @c1 int, @c2 datetime, @ts binary(8)
--| as
--| update temp_tsequal set c2=@c2  where c1=@c1 and tsequal(ts,@ts)
--| if (@@error = 0)
--|  select "updated"
--| else
--|  select "cannot update"

CREATE PROCEDURE updates(
  IN v_c1 INT,
  IN v_c2 TIMESTAMP,
  IN v_ts CHAR(8) FOR BIT DATA)
DYNAMIC RESULT SETS 1
LANGUAGE SQL
BEGIN
  DECLARE SQLCODE INT;
  DECLARE l_sqlcode INT DEFAULT 0;
  DECLARE temp_cursor CURSOR WITH HOLD WITH RETURN TO CLIENT FOR
  SELECT 'updated'
  FROM TABLE (VALUES 1) AS temp_table;
  DECLARE temp_cursor1 CURSOR WITH HOLD WITH RETURN TO CLIENT FOR
  SELECT 'cannot update'
  FROM TABLE (VALUES 1) AS temp_table1;
  DECLARE CONTINUE HANDLER FOR SQLEXCEPTION,SQLWARNING,NOT FOUND
```

```
      SET l_sqlcode = SQLCODE;

    SET l_sqlcode = 0;
    UPDATE temp_tsequal
    SET c2 = v_c2
    WHERE c1 = v_c1 AND (MSSQL.tsequal(ts, v_ts) IS NOT NULL);
    COMMIT;
    IF l_sqlcode = 0 OR l_sqlcode = 100 THEN
      OPEN temp_cursor;
    ELSE
      OPEN temp_cursor1;
    END IF;
  END!
```

MSSQL.unicode
: Simulates SQL Server unicode function.

SYB.waitforfunc, MSSQL.waitforfunc
: Simulates the Sybase and SQL Server WAITFOR statement using a Java UDF.

SYB.waitfor, MSSQL.waitfor
: Simulates the Sybase and SQL Server WAITFOR statement by calling a UDF (as a statement, instead of a function).

```
--| create procedure callwaitfor
--| @timestring varchar(13)
--| as
--| waitfor time @timestring

CREATE PROCEDURE callwaitfor(
  IN v_timestring VARCHAR(13))
DYNAMIC RESULT SETS 1
LANGUAGE SQL
BEGIN
  DECLARE SQLCODE INT;
  DECLARE temp_var VARCHAR(5);
  DECLARE temp_var1 VARCHAR(13);
  DECLARE l_sqlcode INT DEFAULT 0;
  DECLARE CONTINUE HANDLER FOR SQLEXCEPTION,SQLWARNING,NOT FOUND
    SET l_sqlcode = SQLCODE;

  SET temp_var = 'time';
  SET temp_var1 = v_timestring;
  CALL SYB.waitfor(temp_var, temp_var1);
END!
```

Conversion functions

These functions are used to simulate Sybase and Microsoft SQL Server behavior on certain type conversions, both explicit (the convert function) and implicit.

SQL Server In the resulting code, some functions are cast to the length of the known constant. The reason is that these functions in SQL Server return VARCHAR(8000), which depending upon the operation can exceed the page size.

General conversions:

Functions related to general types.

SYB.char_to_decimal, MSSQL.char_to_decimal
: Used in conversion from char to decimal. Interpret scientific notation, arbitrary decimal precision, and blank strings.

```
--| create proc test_conv_char2dec
--| as
--| select convert(decimal(10,5), "505.54321"),
       convert(decimal(10,5), ""),
--|        convert(decimal(10,5), "   "),
       convert(decimal(10,5), null)

CREATE PROCEDURE test_conv_char2dec()
DYNAMIC RESULT SETS 1
LANGUAGE SQL
BEGIN
  DECLARE SQLCODE INT;
  DECLARE l_sqlcode INT DEFAULT 0;
  DECLARE temp_cursor CURSOR WITH HOLD WITH RETURN TO CLIENT FOR
  SELECT CAST(SYB.char_to_decimal('505.54321', 5) AS DECIMAL(10,5)),
         CAST(SYB.char_to_decimal(' ', 5) AS DECIMAL(10,5)),
         CAST(SYB.char_to_decimal('   ', 5) AS DECIMAL(10,5)),
         CAST(NULL AS DECIMAL(10,5))
  FROM TABLE (VALUES 1) AS temp_table;
  DECLARE CONTINUE HANDLER FOR SQLEXCEPTION,SQLWARNING,NOT FOUND
    SET l_sqlcode = SQLCODE;

  OPEN temp_cursor;
END!
```

SYB.char_to_float, MSSQL.char_to_float

Used in conversion from char to float.

SYB.char_to_int, MSSQL.char_to_int

Used in conversion from char to int. Maps the blank string to 0.

SYB.dec_to_char, MSSQL.dec_to_char, SYB.dec_to_char1 (auxiliary), MSSQL.dec_to_char1 (auxiliary)

Used in conversion of decimal to character (given the output of the DB2 UDB conversion to character). Removes leading zeroes generated by DB2 UDB.

```
--| create proc test_conv_dec2ch
--| as
--| select convert(char, dec10_5) from basetbl_test order by id

CREATE PROCEDURE test_conv_dec2ch()
DYNAMIC RESULT SETS 1
LANGUAGE SQL
BEGIN
  DECLARE SQLCODE INT;
  DECLARE l_sqlcode INT DEFAULT 0;
  DECLARE temp_cursor CURSOR WITH HOLD WITH RETURN TO CLIENT FOR
  SELECT CAST(SYB.dec_to_char(char(dec10_5)) AS CHAR(30))
  FROM basetbl_test
  ORDER BY id ;
  DECLARE CONTINUE HANDLER FOR SQLEXCEPTION,SQLWARNING,NOT FOUND
    SET l_sqlcode = SQLCODE;

  OPEN temp_cursor;
END!
```

SYB.float_to_char, MSSQL.float_to_char

Simulate conversion of float values to character In Sybase and SQL Server, if the normalized exponent (from scientific notation) is less than 17 and greater than -5, the float value is expressed in simple decimal notation, otherwise it is expressed in scientific notation (DB2 UDB always uses scientific notation).

SYB.pad_with_00s, MSSQL.pad_with_00s

Pads a string with NULL characters (x'00', instead of spaces: x'20') in conversion from character to binary.

```
CREATE TABLE bintable (
  k1 INT NOT NULL,
  img BLOB(2G) NOT LOGGED,
  bnry31 VARCHAR(255) FOR BIT DATA NOT NULL,
  vbnary51 VARCHAR(255) FOR BIT DATA NOT NULL)!

INSERT INTO bintable
VALUES (1, CAST(NULL AS BLOB(2G)),
        SYB.pad_with_00s(X'49424d20436f72706f726174696f6e', 255),
        X'7265616461626c652062696e6172792064617461120234021255e26')!
```

SYB.round_to_minutes, MSSQL.round_to_minutes

Used to simulate conversion from datetime (or a string representation of datetime) to smalldatetime by rounding up when seconds are at least 30.

```
CREATE TABLE test1 (
  id INT NOT NULL,
  ch20 VARCHAR(20),
  sdt TIMESTAMP,
  dt TIMESTAMP NOT NULL,
  b1 SMALLINT NOT NULL)!

INSERT INTO test1
VALUES (1, 'mystring',
        SYB.round_to_minutes('2001-03-12-00.00.00'),
        '2000-04-23-00.00.00', 1)!
```

SYB.rtrim_on_insert(s), MSSQL.rtrim_on_insert(s)

Simulates Sybase and SQL Server trimming of spaces from the end of character strings before entering into a varchar column. A string containing blanks only maps to a single space.

```
--| create trigger dim_insert on mnp_dim for insert
--| as
--|    declare @x char(5)
--|    declare @y char(7)
--|    if (@@rowcount <= 0)                  return
--|    if (select count(*) from inserted) <=0    return
--|    select @x = c2, @y = c2+'xx' from inserted
--|    insert into mnp_testresult values( "dim_insert", @x, @y)

CREATE TRIGGER dim_insert
AFTER INSERT ON mnp_dim
REFERENCING NEW_TABLE AS inserted
FOR EACH STATEMENT
MODE DB2SQL
TRGR : BEGIN ATOMIC
  DECLARE v_x CHAR(5);
  DECLARE v_y CHAR(7);
  DECLARE l_rowcount INT;
  SET l_rowcount  = (SELECT COUNT(*)
                     FROM inserted) ;
  IF l_rowcount <= 0 THEN
    LEAVE TRGR;
  END IF;
  IF (SELECT COUNT(*)
      FROM inserted) <= 0 THEN
    LEAVE TRGR;
  END IF;
  SET (v_x, v_y)  =
    (SELECT cor_name, cor_name1
     FROM TABLE (SELECT c2 AS cor_name, c2 || 'xx' AS cor_name1,
                          ROW_NUMBER () OVER () AS rn
                 FROM inserted) AS temp_table
     WHERE temp_table.rn = 1) ;
  INSERT INTO mnp_testresult
  VALUES ('dim_insert', SYB.rtrim_on_insert(v_x),
     SYB.rtrim_on_insert(v_y));
END!
```

SYB.rtrim_on_insert_b(s), MSSQL.rtrim_on_insert_b(s)

Simulates Sybase and SQL Server trimming of NULL characters (x'00') from the end of binary strings before entering into a varbinary (mapped to varchar for bit data in DB2 UDB) column. Implemented in Java.

```
--| create proc test_conv_vch2bin
--| as
--| create table #temp(b1 binary(20) null, b2 binary(4) null)
--| insert into #temp select convert(binary(20), vc20),
--|                          convert(binary(4), vc20)
                             from basetbl_test
--| select * from #temp
--| drop table #temp

CREATE PROCEDURE test_conv_vch2bin()
DYNAMIC RESULT SETS 1
LANGUAGE SQL
BEGIN
  DECLARE SQLCODE INT;
  DECLARE l_sqlcode INT DEFAULT 0;
  DECLARE CONTINUE HANDLER FOR SQLEXCEPTION,SQLWARNING,NOT FOUND
    SET l_sqlcode = SQLCODE;

  DECLARE GLOBAL TEMPORARY TABLE SESSION."#temp"
  (
    b1 VARCHAR(20) FOR BIT DATA,
    b2 VARCHAR(4) FOR BIT DATA
  ) WITH REPLACE ON COMMIT PRESERVE ROWS NOT LOGGED;
  BEGIN
    DECLARE temp_cursor CURSOR WITH HOLD WITH RETURN TO CLIENT FOR
    SELECT SESSION."#temp".*
    FROM SESSION."#temp";
    INSERT INTO SESSION."#temp"
    SELECT SYB.rtrim_on_insert_b(CAST(SYB.varchar_to_bin(vc20, 20)
        AS CHAR(20))),
             SYB.rtrim_on_insert_b(CAST(SYB.varchar_to_bin(vc20, 4)
        AS CHAR(4)))

    FROM basetbl_test;
    COMMIT;
    OPEN temp_cursor;
    DROP TABLE SESSION."#temp";

  END;
END!
```

MSSQL.str_uniqueid

Converts a VARCHAR to a unique identifier

SYB.sybDateTodb2Date, MSSQL.MSSQLDateTodb2Date

Takes a Sybase or SQL Server date format specifier (such as 'MDY') and a string in a Sybase or SQL Server date format. It returns a string containing the equivalent DB2 UDB timestamp format for that date. (Note: string literals in date contexts are converted by the MTK converter). Implemented in Java.

```
CREATE TABLE basetbl_basic (pkey INT NOT NULL, chr20 CHAR(20),
                            sdt TIMESTAMP,
                            dt mydatetime)!

INSERT INTO basetbl_basic
```

```
VALUES (1, 'mystring',
        MSSQL.MSSQLDateTodb2Date
         ('MDY', CAST(year(CURRENT TIMESTAMP) AS CHAR(30))),
        CAST(MSSQL.MSSQLDateTodb2Date('MDY',
        CAST(year(CURRENT TIMESTAMP) AS CHAR(30))) AS TIMESTAMP))!
```

SYB.to_bit, MSSQL.to_bit

Converts anything non-zero to 1 and zero to 0, to represent the Sybase and SQL Server bit type as a DB2 UDB smallint.

```
--| create proc test_conv_char2bit
--| as
--| select convert(bit, "5")
--| select convert(bit, "")
--| select convert(bit, "   ")

CREATE PROCEDURE test_conv_char2bit()
DYNAMIC RESULT SETS 1
LANGUAGE SQL
BEGIN
  DECLARE SQLCODE INT;
  DECLARE l_sqlcode INT DEFAULT 0;
  DECLARE temp_cursor CURSOR WITH HOLD WITH RETURN TO CLIENT FOR
  SELECT CAST(SYB.to_bit('5') AS SMALLINT)
  FROM TABLE (VALUES 1) AS temp_table;
  DECLARE temp_cursor1 CURSOR WITH HOLD WITH RETURN TO CLIENT FOR
  SELECT CAST(SYB.to_bit(' ') AS SMALLINT)
  FROM TABLE (VALUES 1) AS temp_table1;
  DECLARE temp_cursor2 CURSOR WITH HOLD WITH RETURN TO CLIENT FOR
  SELECT CAST(SYB.to_bit('   ') AS SMALLINT)
  FROM TABLE (VALUES 1) AS temp_table2;
  DECLARE CONTINUE HANDLER FOR SQLEXCEPTION,SQLWARNING,NOT FOUND
    SET l_sqlcode = SQLCODE;

  OPEN temp_cursor;
  OPEN temp_cursor1;
  OPEN temp_cursor2;
END!
```

MSSQL.uniqueid_str

Converts from a unique identifier to a VARCHAR.

Money to character conversions:

Functions that relate to the money type.

SYB.calc_len, MSSQL.calc_len

Calculates how many commas will be necessary for style 1.

SYB.comma_test, MSSQL.comma_test

Returns comma if money value is bigger than 10 to the exp power (else empty string) (for style 1).

SYB.money_comma, MSSQL.money_comma

Inserts commas where necessary for style 1.

```
--| create proc test_conv_mn2ch
--| as
--| select convert(char(30),mn), convert(char(30),smn)
       from basetbl_test order by id
--| select convert(char(30),mn,1), convert(char(30),smn,1)
       from basetbl_test order by id

CREATE PROCEDURE test_conv_mn2ch()
DYNAMIC RESULT SETS 1
LANGUAGE SQL
BEGIN
```

```
  DECLARE SQLCODE INT;
  DECLARE l_sqlcode INT DEFAULT 0;
  DECLARE temp_cursor CURSOR WITH HOLD WITH RETURN TO CLIENT FOR
  SELECT CAST(MSSQL.str(mn, 30, 2) AS CHAR(30)),
           CAST(MSSQL.str(smn, 30, 2) AS CHAR(30))
  FROM basetbl_test
  ORDER BY id ;
  DECLARE temp_cursor1 CURSOR WITH HOLD WITH RETURN TO CLIENT FOR
  SELECT CAST(MSSQL.money_comma(mn, 30) AS CHAR(30)),
           CAST(MSSQL.money_comma(smn, 30) AS CHAR(30))
  FROM basetbl_test
  ORDER BY id ;
  DECLARE CONTINUE HANDLER FOR SQLEXCEPTION,SQLWARNING,NOT FOUND
    SET l_sqlcode = SQLCODE;

  OPEN temp_cursor;
  OPEN temp_cursor1;
END!
```

SYB.money_to_char, MSSQL.money_to_char
Calls SYB.money_comma (style=1) or SYB.str (default, style=0) to convert money to char.

Datetime to character conversions:

Functions relating to date and time.

SYB.char3, MSSQL.char3
Right justifies 3 digit int, adding leading zeros if needed.

SYB.char2, MSSQL.char2
Right justifies 2 digit int, adding leading zero or space if needed.

SYB.convert_hour, MSSQL.convert_hour
Converts 24 hour clock to 12 hour clock.

SYB.date_to_char and MSSQL.date_to_char
Converts datetime to type character. And, for the various styles of datetime, a function of the form **SYB.date_to_char*N*** or **MSSQL.date_to_char*N*** is used, where *N* matches the style options: *N* or 100 + *N*. For example, MSSQL.date_to_char7 is used for style option 7 or 107.

The range of supported styles for *N* is 1–12 for Sybase and for SQL Server, 1–14, 20, 21, 26, 30, and 31.

```
--| create proc test_conv_dt2ch
--| as
--| select convert(char(30),d1,9), convert(char(30),d2,109)
--|   from basetbl_date order by id

CREATE PROCEDURE test_conv_dt2ch()
DYNAMIC RESULT SETS 1
LANGUAGE SQL
BEGIN
  DECLARE SQLCODE INT;
  DECLARE l_sqlcode INT DEFAULT 0;
  DECLARE temp_cursor CURSOR WITH HOLD WITH RETURN TO CLIENT FOR
  SELECT CAST(SYB.date_to_char9(d1, 0) AS CHAR(30)),
           CAST(SYB.date_to_char9(d2, 1) AS CHAR(30))
  FROM basetbl_date
  ORDER BY id ;
  DECLARE CONTINUE HANDLER FOR SQLEXCEPTION,SQLWARNING,NOT FOUND
    SET l_sqlcode = SQLCODE;

  OPEN temp_cursor;
END!
```

MSSQL.double_to_date

Converts an SQL Server double type to a date.

```
--| create proc test_conv_flt2dt
--| as
--| select convert(datetime, flt), convert(smalldatetime, flt)
--|   from basetbl_test order by id

CREATE PROCEDURE test_conv_flt2dt()
DYNAMIC RESULT SETS 1
LANGUAGE SQL
BEGIN
  DECLARE SQLCODE INT;
  DECLARE l_sqlcode INT DEFAULT 0;
  DECLARE temp_cursor CURSOR WITH HOLD WITH RETURN TO CLIENT FOR
  SELECT CAST(MSSQL.double_to_date(flt) AS TIMESTAMP),
           CAST(MSSQL.double_to_date(flt) AS TIMESTAMP)
  FROM basetbl_test
  ORDER BY id ;
  DECLARE CONTINUE HANDLER FOR SQLEXCEPTION,SQLWARNING,NOT FOUND
    SET l_sqlcode = SQLCODE;

  OPEN temp_cursor;
END!
```

SYB.year, MSSQL.year

Converts integer year value to 4 or 2 digit char value.

Informix Dynamic Server converter messages

The Informix converter might return the following messages.

MTKI0000

Translation Information: MTK Informix Converter. Version: <mtk version>

Description

Specifies the version of the Informix converter.

MTKI0001

Translation Error: This statement is not translated.

Description

This statement is not translated.

MTKI0002

Input Script Error: Unexpected syntax - not translated.

Description

The translator encountered some text that it did not understand.

MTKI0003

Fatal Internal Error: Error walking the AST.

Description

The translator encountered some text that it did not understand.

MTKI0004

Input Script Error: Unrecognized character - skipped.

Description

The translator does not understand this character.

MTKI0005

Translation Error: This feature is not translated.

Description

A feature of the data source code was encountered and recognized as a feature not translated in this version.

MTKI0006

Translation Error: Untranslated expression: <cause>

Description

The expression is not translated for the given reason.

MTKI0010

Translation Error: Unable to open file ″<filename>″ for output.

Description

Unable to open the specified file for output. Ensure the file exists in the specified directory and is not locked or open by another application.

MTKI0011

Input Script Error: Reference to unknown <object>: <object name>

Description

The translator is not aware of the definition of this object.

Ensure the definition exists in the file being translated or one of the files used as context for this translation. Also, ensure the schema name is specified as indicated.

Related tasks

"Referring to previously converted metadata" on page 34
If you are converting metadata that refers to other metadata that has already been converted, you still must include all of the files in the conversion. However, your translated data might be in a state where you would not want to translate it again. You can specify that the IBM Migration Toolkit use certain metadata *in-context*.

MTKI0012

Input Script Error: Referenced table does not have a primary key.

Description

The table being referenced in this foreign key constraint does not have a primary key. Assign a primary key in the original source and re-convert.

MTKI0013

Input Script Error: No matching unique or primary key for this column list.

Description

The list of columns in this foreign key constraint has no matching unique or primary key constraint in the table it is referencing. Assign a primary key in the original source and re-convert.

MTKI0014

Input Script Error: Unknown command, rest of line ignored.

Description

The translator does not recognize this command, so the entire line is skipped.

MTKI0015

Input Script Error: Unknown statement, ignored.

Description

The translator does not recognize this statement, so the entire statement is skipped.

MTKI0016

Input Script Error: Duplicate definition of <object name>

Description

This object has already been defined. The previous definition will be used by the translator.

MTKI0017

Translation Error: This ALTER TABLE clause is not supported.

Description

Only the add constraint clause is translated in the ALTER TABLE statement. All other ALTER TABLE clauses are ignored.

Related reference

"SQL statements" on page 140

MTKI0018

Input Script Error: Possibly ambiguous column reference: <column name>

Description

This column occurs in more than one table in the from clause (or more than once as a column heading in the select list for an order by clause). The translation might cause an error in DB2 UDB.

Related reference

"SQL statements" on page 140

MTKI0019

Input Script Error: Call to unknown <subprogram type>: <subprogram type>

Description

The translator could not resolve the name in this function/procedure call. It was not recognized as an Informix built-in function name or the name of a previously defined user defined function (UDF) or procedure.

Related information

"Built-in functions" on page 137

MTKI0020

Translation Warning: Object name has been changed to <new name>.

Description

Object names that are too long for DB2 UDB are truncated. Names that are DB2 UDB reserved words are enclosed in double quotes. Names that conflict with other names in DB2 UDB (because the name is already in use) are renamed.

MTKI0021

Translation Error: Call to <subprogram type> <subprogram name> is not supported.

Description

This Informix function or procedure call is not translated to DB2 UDB.

Related information

"Built-in functions" on page 137

MTKI0022

Input Script Error: No matching instance of <subprogram type> <subprogram name> with such arguments.

Description

The translator expects a different number of arguments when calling this function or procedure, or the argument types do not match any of the instances of this function or procedure.

Related information

"Built-in functions" on page 137

MTKI0023

Fatal Internal Error: Translator runtime error.

Description

The translator experienced an unexpected internal error. Report this error to IBM Migration Toolkit support.

MTKI0024

Translation Warning: NOT NULL constraint is added because of a PRIMARY KEY or UNIQUE constraint.

Description

In DB2 UDB, a column with a UNIQUE or PRIMARY KEY constraint must also have a NOT NULL constraint.

MTKI0025

Translation Error: Insert/Delete/Update on subquery or table collection expression not translated.

Description

This insert, delete, or update operates over a sub-query or a table collection expression. This operation is not supported in DB2 UDB and is not translated.

MTKI0026

Translation Error: Create Synonym for object other than table or view not translated.

Description

DB2 UDB synonyms are supported for tables and views only. Synonyms for other objects are not translated.

MTKI0027

Translation Error: RESCINDED. Translation of default expression results in a non-constant expression.

Description

RESCINDED.

In DB2 UDB, default expressions may be constants, datetime special registers, USER, NULL, or certain forms of cast expressions only.

MTKI0028

Translation Error: Incompatible Nulls First or Nulls Last clause: <conflicting clauses>

Description

DB2 UDB does not support NULLS LAST in a DESC order or NULLS FIRST in an ASC (the default) order.

MTKI0029

Translation Error: Subquery in From clause: not yet translated.

Description

SELECT statements that have sub-queries in the FROM clause are not yet translated.

MTKI0030

Input Script Error: Invalid date format

Description

The date format is invalid or the input data (a date value) does not match the date format.

MTKI0031

Translation Error: VALUE set clause of update statement not translated.

Description

DB2 UDB does not have an equivalent set clause in the update statement.

MTKI0032

Translation Warning: Order by clause in INSERT, SELECT INTO, VIEW or derived table subquery not translated.

Description

DB2 UDB does not allow the order by clause in a sub-query used to insert values in an INSERT statement or to define a VIEW.

MTKI0033

Input Script Error: Inserted data error

Description

The number of inserted values does not match the number of columns specified for the table.

MTKI0034

Translation Warning: No DB2 UDB translation available, but statement has been taken into account

Description

There is no DB2 UDB translation available, but the information in the statement will be used by the converter in translating the statements that follow.

MTKI0035

Input Script Error: All columns of the subquery of a view must be named.

Description

The columns of the view must be named by either giving an explicit list of names in the definition, or by using column aliases in the sub-query. In this case, there is no list of names and at least one sub-query column has no name.

MTKI0036

Input Script Error: Number of columns of subquery must match names given in view definition.

Description

In a view, if a list of column names is given in the definition, this must match the number of columns in the result set of the sub-query.

MTKI0037

Translation Error: Invalid or unsupported outer-join query

Description

The following restrictions apply to the translation of outer-join queries:
- Cyclic outer-joins in the WHERE clause are not translated (invalid input)
- The (+) operator must not follow a complex expression, it can follow a column reference only
- Only the equality (=) operator is supported

MTKI0038

Input Script Error: Type mismatch: expression has an unexpected data type.

Description

The expression has inappropriate data type in this context.

MTKI0039

Input Script Error: Unexpected size of expression-list or result-set

Description

The size of this expression-list or result-set (generated by a sub-query) does not match the size expected by the context.

This error is likely to happen in:
- INSERT statements if the VALUES clause or the result-set generated by the sub-query does not match the table size
- Comparisons, if the sizes of the elements on each side of the comparison operator do not match

MTKI0040

Translation Error: Conversion from type <SourceType> to type <TargetType> is not supported by MTK.

Description

An implicit conversion from a value of the source type to the target type in Informix is not directly supported in DB2 UDB.

MTKI0041

Translation Warning: Unable to infer DB2 UDB data type for the expression. Using Varchar(1).

Description

A data type is needed in this context for use in a declaration or cast expression.

MTKI0042

Input Script Error: Unrecognized data type: <datatype>

Description

This data type is not recognized by the translator. This could be because:
- It is not a valid Informix data type
- It is a user-defined type, not yet handled by the translator

MTKI0043

Input Script Error: Length required for type <datatype>

Description

Length must be specified for this data type. For instance, when declaring a column of type VARCHAR, VARCHAR2 or RAW, length must be explicitly defined, as in: VARCHAR(10), RAW(100)

MTKI0044

Translation Warning: Warning: Negative scale and scale greater then the precision are not supported in DB2 UDB.

Description

The DB2 UDB DECIMAL data type does not support negative scales or scales greater than the precision. The translator adjusts the precision and the scale in order to avoid loss of data but the DB2 UDB results might differ.

For example:

```
CREATE TABLE T(X NUMBER(4,-2))
```

is translated into:

```
CREATE TABLE T(X DECIMAL(6,0))!
```

But the result of an INSERT VALUES(123456) will be 123400 in Informix and 123456 in DB2 UDB.

MTKI0045

Translation Error: Untranslated data type: <datatype>

Description

The Informix data type has no DB2 UDB equivalent and could not be translated. This can happen when the translation of a NUMBER(p,s) is not able to capture any digit of the source data type (when s-p>31 or s<-31).

MTKI0046

Translation Warning: Non blank-padded comparison cannot be enforced here

Description

This happens when the non-blank-padded comparisons enforcement option is set, in membership comparisons involving expression-lists where CHAR and VARCHAR are being compared.

For example, if a and b are CHAR, c is VARCHAR and xn is any other data:

```
(a, x1) IN ((b, x2),(c,x3))
```

cannot be translated accurately, because a vs. b should be a non-blank-padded comparison whereas a vs. c is blank-padded. The only way to enforce blank-padding rules here would be to explode this membership comparison into several comparison, which the translator does not handle. In the above example this be converted to:

```
(a = b AND x1 = x2) OR (ORA8.NO_PAD(a) = ORA8.NO_PAD(c) AND x1 = x3)
```

MTKI0047

Translation Warning: RESCINDED. BEFORE translated to NO CASCADE BEFORE

Description

RESCINDED. Informix triggers using the BEFORE event type are translated in DB2 UDB by NO CASCADE BEFORE triggers. This type of trigger does not allow other triggers to fire from the trigger body, which might lead to incorrect behavior.

MTKI0048

Translation Error: DDL and Database event triggers are not translated

Description

Triggers using Data Definition Language (DDL) events and database events are not translated.

MTKI0049

Translation Error: INSTEAD OF triggers are not translated.

Description

Triggers using the INSTEAD OF event type are not translated.

MTKI0050

Translation Warning: This statement is not supported in a DB2 UDB Dynamic Compound Statement

Description

This statement is not supported in a DB2 UDB dynamic compound statement. Dynamic compound statements are used as bodies for a top-level anonymous blocks, user-defined functions, and triggers. Procedures use DB2 UDB compound statements as bodies that are less restrictive. Examples of restrictions for dynamic compound statements are:

- Nested blocks are not supported.
- CASE expressions are not supported
- GOTO is not supported

MTKI0051

Translation Error: BEFORE triggers without FOR EACH ROW are not supported

Description

In DB2 UDB, FOR EACH STATEMENT must not be specified for BEFORE triggers. For this reason, BEFORE triggers that do not specify FOR EACH ROW cannot be translated.

MTKI0052

Input Script Error: Table name <table> not allowed in this context.

Description

In this context an expression is expected. A table name that does not qualify a column name or a star (*) is not allowed.

MTKI0053

Input Script Error: No column <column name> in table.

Description

The given table has been found in a surrounding from clause, but it does not contain the indicated column.

MTKI0054

Input Script Error: Parameters in named notation cannot be followed by parameters in positional notation.

Description

The actual parameters in this procedure/function call use bad mixed notation. Mixed parameter notation is only allowed if named parameters follow positional parameters.

MTKI0055

Translation Error: Cursor RETURN clauses are not translated.

Description

Due to Informix Dynamic Server limitations, all FETCH statements from the same cursor must fetch into the same variables. Please see the Informix Dynamic Server manuals for more information about the FOREACH statement and SPL.

MTKI0056

Input Script Error: An INTO clause is expected in this select statement

Description

In a PLSQL context, a select statement must have an INTO clause.

MTKI0057

Translation Error: Unable to translate subquery with set operations to DB2 UDB Select Into.

Description

In SPL a SELECT INTO statement can be united with other selects. This is not allowed in DB2 UDB.

MTKI0058

Translation Error: This ALTER SESSION clause is not supported.

Description

The following ALTER SESSION clauses are currently simulated (but not translated). The remaining clauses are ignored.

- SET NLS_DATE_FORMAT
- SET CURRENT_SCHEMA
- SET NLS_CURRENCY
- SET NLS_ISO_CURRENCY
- SET NLS_NUMERIC_CHARACTERS

MTKI0059

Translation Warning: Ignored input - not translated.

Description

This input is ignored. It is not supported in DB2 UDB. This omission should not cause the DB2 UDB code to produce different results from the corresponding Informix code.

MTKI0060

Translation Error: References to remote database objects are not translated

Description

References to objects in remote databases (indicated by ″@dblink″) are not supported.

MTKI0061

Translation Warning: Parameter defaults are not supported in DB2 UDB procedure definitions. Calls to the procedure are adjusted accordingly.

Description

In procedure and function declarations, the optional DEFAULT value of a parameter is not translated, but the translator will use the value as necessary through the remainder of the translation.

Related reference

"SQL statements" on page 140

MTKI0062

Translation Warning: Labels are not allowed in DB2 UDB dynamic compound statements.

Description

Labels are not allowed in DB2 UDB dynamic compound statements, except for loops. The translator will not translate the label for this statement.

Related reference

"SQL statements" on page 140

MTKI0063

Translation Warning: Labels are not supported for top-level blocks in DB2 UDB

Description

Labels are not supported for top-level blocks in DB2 UDB. The translator omits these labels.

MTKI0064

Input Script Error: PUBLIC synonym cannot have schema qualifier.

Description

In the definition of a public synonym, the name of the synonym is not allowed to be qualified by a schema name.

MTKI0065

Translation Warning: This savepoint might not be allowed by DB2 UDB.

Description

If another savepoint is active when this savepoint is encountered, it will be rejected by DB2 UDB. DB2 UDB allows only one active savepoint at a time. The converter is unable to determine at compile time the number of savepoints that might be active when this statement is encountered.

MTKI0066

Translation Error: Autonomous transactions are not supported by DB2 UDB.

Description

DB2 UDB does not support autonomous transactions. The transaction statements COMMIT, ROLLBACK, and SAVEPOINT in this block and the enclosing dynamic context might give different behavior in DB2 UDB.

MTKI0067

Translation Warning: NOWAIT is not supported by DB2 UDB Lock Table statement.

Description

NOWAIT not supported by in the DB2 UDB Lock Table statement.

MTKI0068

Translation Error: This lock mode is not supported by the DB2 UDB Lock Table statement.

Description

The DB2 UDB Lock Table statement supports SHARE and EXCLUSIVE lock modes only.

MTKI0069

Translation Warning: Locks for partitions, subpartitions, and remote tables are not supported by DB2 UDB.

Description

These clauses are not translated to DB2 UDB.

MTKI0070

Translation Error: OUT and IN OUT parameters are not supported in DB2 UDB user-defined functions.

Description

In user-defined functions, DB2 UDB only allows input parameters. The translator cannot properly translate a function having OUT or IN OUT parameters to an equivalent construct in DB2 UDB. Evaluate the code to determine if you can use a procedure instead.

MTKI0071

Translation Warning: Reference to OLD or NEW column translated to <NULL or variable>

Description

DB2 UDB does not accept references to OLD from an inserting trigger or references to NEW from a deleting trigger. In the WHEN clause of the trigger, these references are translated to NULL. In the body of the trigger, they are translated to a variable generated for this purpose.

MTKI0072

Translation Error: DB2 UDB only allows constants as arguments to a top-level procedure call.

MTKI0073

Translation Error: This statement is not supported in a DB2 UDB UDF.

Description

This statement is not supported in a DB2 UDB UDF. The following statements are not supported in this context: INSERT, DELETE and UPDATE.

MTKI0074

Translation Error: Nested exception handlers are not supported.

Description

Nested exception handlers are not supported because DB2 UDB does not allow the declaration of a handler inside another handler.

MTKI0075

Translation Error: This statement is not supported in a DB2 UDB Before Trigger

Description

This statement is not supported in a DB2 UDB Before Trigger. The following statements are not supported in this context: INSERT, DELETE and UPDATE.

MTKI0076

Translation Error: This statement is not supported in a DB2 UDB After Trigger

Description

This statement is not supported in a DB2 UDB After Trigger. The following statement is not allowed in this context: a SET transition-variable statement (when FOR EACH ROW is specified).

MTKI0077

Translation Error: This form of the SQL Plus Execute command is not supported.

Description

Only procedure calls and begin-end blocks are supported in the SQL Plus Execute command. Other PLSQL statements in an Execute command are not translated.

MTKI0078

Translation Warning: Assignment of non-constant default expression moved to block.

Description

DB2 UDB does not allow a non-constant expression to be the default value of a variable declaration. A statement assigning the expression to the variable has been moved to the beginning of the block.

MTKI0079

Translation Warning: An abritrary size of 255 KB was picked for the LOB type.

Description

In SPL context, certain Informix data types are translated to LOB(255K) or CLOB(255K). The converter picks 255 KB as an arbitrary size for those Large Object types. However, depending on your needs, the size can be modified (increased or decreased), allowing better performance.

Considering that DB2 UDB allocates memory for variables and parameters of these types, try to limit the size of these data types as much as possible.

MTKI0080

Translation Error: This package item is not translated.

Description

Only the following items are supported inside a package: function specifications, functions, procedure specifications, procedures, and constants. Variable declarations, cursor declarations and type definitions are not be translated.

MTKI0081

Translation Warning: The generated SQLSTATE might be incorrect.

Description

When translating calls to raise_application_error, the translator uses the error code to generate an SQLSTATE. See the *DB2 UDB Message Reference* for rules on specifying the necessary SQLSTATE.

MTKI0082

Translation Error: Untranslated reference.

Description

This reference was not translated because the syntax is incorrect or the feature is not supported.

MTKI0083

Translation Warning: The called function was not translated successfully.

Description

The function definition of the function being called was not translated successfully. The function definition must be corrected.

MTKI0084

Translation Error: AS Expressions are not yet translated.

Description

AS Expressions are not yet translated

MTKI0085

Translation Error: Procedure or function call not translated: the procedure or function must be defined first.

Description

Because DB2 UDB does not support function or procedure specifications, functions or procedures can only be referenced once they have been fully defined.

If possible, replace the function or procedure specification with its body.

MTKI0086

Translation Warning: No BITMAP index in DB2 UDB, ignored the BITMAP clause.

Description

DB2 UDB only has a UNIQUE index; it does not have a BITMAP index. The translator ignores the BITMAP clause.

MTKI0087

Input Script Error: This specification item has no definition in the input file

Description

The function, procedure, or cursor specification does not have a corresponding definition in the input file. The specification does not have a DB2 UDB translation so this item will not appear in the output file.

MTKI0088

Translation Error: Top-level procedure-calls with CLOB parameters are not supported by DB2 UDB.

Description

Procedures with CLOB parameters cannot be called from the top-level in DB2 UDB. Top-level procedure-calls require that all arguments being passed must be literals, but DB2 UDB does not support these kinds of literals in this context. Calls would end up with error: ″DB2 UDB1036E The CALL command failed.″

MTKI0089

Translation Error: This declaration is not translated

Description

This declaration is not translated

MTKI0090

Translation Error: EXCEPTION_INIT is not translated. The exception declaration needs to be modified.

Description

The EXCEPTION_INIT pragma is not translated because it has no equivalent in DB2 UDB.

To obtain the same behavior in DB2 UDB, modify the exception declaration by changing the unreserved SQLSTATE to a reserved SQLSTATE that corresponds to the Informix error number.

MTKI0091

Translation Error: This ALTER TABLE statement is not translated.

Description

This ALTER TABLE statement is not translated because none of its clauses are supported.

MTKI0092

Translation Error: Update or Delete with reference to rownum pseudocolumn not translated.

Description

Either the statement occurs at the top level and thus cannot be translated using a cursor, or the ROWNUM condition is too complicated to translate to a ″FETCH FIRST n ROWS ONLY″ clause.

MTKI0093

Translation Error: Reference to rownum pseudocolumn in set clause of UPDATE statement not translated.

Description

Reference to the row number in a set clause of an UPDATE statement is not supported in DB2 UDB.

MTKI0094

Translation Warning: This UDF is translated to DB2 UDB as a procedure.

Description

This Informix user-defined function is translated to DB2 UDB as a procedure. This happens with functions with parameters in OUT mode. Since this feature is not available in DB2 UDB, the translator uses a DB2 UDB procedure instead. The calls to the Informix function will be translated into procedure-calls.

MTKI0095

Input Script Error: Calls to functions with an OUT parameter are not allowed in an SQL statement.

Description

A call to a function defined with parameters of type OUT is not allowed in a SQL statement (INSERT, UPDATE, DELETE, SELECT...)

MTKI0096

Translation Error: Call to a function with sn OUT parameter id too complex to be translated.

Description

The function being called has OUT parameters and the translator does not support such translation in this context. This happens because functions with OUT parameters are translated to procedures in DB2 UDB. Therefore, calls to such functions must be translated to procedure-calls. In certain contexts (ELSIF, WHILE, etc..), this would require a complex workaround that the translator does not handle.

For example, consider the function `foo(a OUT INT)` returning an int in Informix:

```
IF (...)
   ...
 ELSIF (foo(x)=0)
   statements
 END IF;
```

It would have to be translated to:

```
IF (...)
   ...
 ELSE
   CALL foo(x, return_val);
   IF (return_val=0)
     statements
   END IF;
 END IF;
```

MTKI0097

Translation Error: ROLLUP and CUBE are not translated.

Description

The ROLLUP and CUBE extensions of the GROUP BY clause are not yet translated.

MTKI0098

Translation Warning: A column definition has been changed because of an ALTER TABLE statement.

Description

A column definition has been added or modified because of a subsequent ALTER TABLE statement.

MTKI0099

Translation Warning: This ALTER TABLE clause has been taken into account by modifying the column definition.

Description

This ALTER TABLE is not directly supported in DB2 UDB, but it has been taken into account through the modification of the corresponding column definition

MTKI0100

Translation Error: This match-expression or pattern-expression is not allowed in the DB2 UDB LIKE predicate.

Description

This match-expression or pattern-expression is not allowed in the DB2 UDB LIKE predicate. Please refer to the LIKE predicate section in the DB2 UDB documentation for more details.

MTKI0101

Translation Warning: Temporary variable <variable name> used to avoid name clash with column name <column name> in INSERT.

Description

In a DB2 UDB INSERT statement, variable names must not be the same as the names of the columns in the insert table. A temporary variable with a distinct name has been used instead of the original variable that had the same name as a column.

MTKI0102

Translation Error: This cursor for loop cannot be translated because the cursor is invalid.

Description

This cursor for loop cannot be translated because the cursor is invalid.

MTKI0103

Translation Error: Reference to <object> not translated because the declaration failed.

Description

This reference was not translated because the declaration of the corresponding object failed.

MTKI0104

Translation Warning: Reference to <object> not translated because the object is not supported.

Description

This reference was not translated because the declaration of the corresponding object is not supported.

MTKI0105

Translation Warning: CREATE TABLESPACE generated with minimal default parameters.

Description

The ″TABLESPACE″ option of the translator is turned on. For each TABLESPACE clause found, a ″CREATE TABLESPACE″ statement is generated at the beginning of the output file. This warning indicates that minimal default parameters have been used and must therefore be changed or enhanced depending on the physical properties wanted.

MTKI0108

Translation Information: Translation Ratio: <percentage>% (<absolute ratio> statements were translated successfully)

Description

This provides an assessment of the translation by giving the ratio of Informix statements translated without producing any error messages compared to the total number of statements. This number provides a general indication regarding the success of the translation and does not intend to give an exact and accurate measure.

″Statement″ here designates Informix SQL and SPL statements. For instance, in a CREATE PROCEDURE statement, the whole SQL statement is counted as 1 (one) and each SPL statement inside the body of the procedure is also counted as one.

MTKI0109

Translation Error: Procedure call in trigger body is not translated.

Description

The translator does not support the execute procedure statements as a trigger action.

DB2 UDB trigger bodies do not support procedure calls.

A possible solution is to copy the procedure code into the trigger block, if this is possible.

MTKI0113

Input Script Error: This reference does not resolve to <object-name>: <reference>

Description

The given reference is to an object with a type that is unexpected in this context. For instance, in a procedure call, the given name is in fact a function.

MTKI0114

Translation Warning: The CREATE SYNONYM statement is not translated. But any object referred to by the synonym is referred to directly.

Description

In DB2 UDB it is possible to create an alias on a table, a view or another alias, but not on a function, procedure or sequence. Thus, there is no direct translation for a CREATE SYNONYM on these objects.

To produce a correct translation, the converter does not translate the CREATE SYNONYM statement, but replaces all references to the synonym by a reference to the object it designates.

MTKI0115

Input Script Error: This is not a table or view: null

Description

In this context, the reference is to an object that is not a table or view.

MTKI0119

Translation Warning: This database has not yet been created in this script. The converter will assume it is not ANSI.

Description

The statement refers to a database that has not yet been defined.

The converter will assume the database was created using the default options (non-ANSI database).

MTKI0120

Translation Error: The CREATE SEQUENCE statement contains a value that is out of range.

Description

The CREATE SEQUENCE statement contains a MINVALUE, MAXVALUE or START VALUE clause with a value that is outside the range that can be handled by the converter.

MTKI0121

Translation Error: The index matches a unique constraint of the table.

Description

DB2 UDB automatically generates an index for each unique table constraint (or primary key) and, therefore, rejects the creation of indexes matching the constraints.

The translation of this CREATE INDEX statement was successful but will probably cause an error in DB2 UDB.

MTKI0122

Translation Warning: Source type for Create Distinct Type is another Distinct type.

Description

The source type for Create Distinct Type is restricted to built-in types in DB2 UDB. MTK does not currently translate the source type to its base type.

MTKI0123

Translation Error: <expression>.* not translated when expression is not a table.

Description

Only table.* is currently translated.

MTKI0124

Translation Error: Only the first trigger action is converted.

Description

This trigger contains multiple action lists, or it contains an action list with multiple items. Only the first action is translated to DB2 UDB; the others are discarded.

MTKI0125

Translation Error: DISABLED triggers are not supported.

Description

This problem can be solved by simply removing the trigger definition.

MTKI0126

Translation Error: BEFORE triggers are not supported.

Description

In Informix, BEFORE triggers are executed for each statement. These triggers cannot be successfully translated because DB2 UDB does not allow you to specify FOR EACH STATEMENT on a NO CASCADE BEFORE trigger.

A solution for successfully translating this trigger is to simply replace the FOR EACH STATEMENT with a FOR EACH ROW statement, if this is appropriate.

For example, if it is known that rows are inserted, updated, or deleted one at a time, then having a FOR EACH STATEMENT trigger or a FOR EACH ROW trigger are equivalent.

MTKI0127

Translation Error: WITH RESUME in RETURN statement is not supported.

Description

The behavior of the WITH RESUME clause of the RETURN statement is not easily simulated in DB2 UDB. In DB2 UDB, functions and procedures have a single entry point.

MTKI0128

Translation Error: Translation of WITH RESUME exception handler might be inaccurate.

Description

Informix WITH RESUME exception handlers always resume the execution in the top-level block (the one where the handler is declared), whereas DB2 UDB CONTINUE handlers resume the execution just after the statement that caused the exception.

If the exception that the handler is handling can be caused by a statement that is in a compound statement (nested block, loop ...), the translated code behaves differently in the case where the exception is raised.

MTKI0129

Translation Error: This statement is not supported in a DB2 UDB Procedure, Function, or Trigger

Description

Some statements, especially DDL statements such as ALTER TABLE or CREATE objects other than tables, indexes, or views, are not allowed inside DB2 UDB procedures, functions, and triggers. In the context of a procedure, such statements can be executed using dynamic SQL.

MTKI0130

Translation Error: Informix error number not translated.

Description

This Informix Dynamic Server error number is not translated.

MTKI0131

Translation Error: Unable to interpret this connect statement.

Description

MTK takes the database name and user name from the connect statement to keep track of the current database and current owner name. MTK uses these values to resolve the names of database objects, such as tables and procedures. If the connect statement contains DEFAULT or host variables, MTK is unable to determine the current database or current owner from the connect statement.

MTKI0132

Input Script Error: Group By column number is too big (or zero).

Description

A number in the group by clause corresponds to an item from the select list. In this case, either the number was zero, or it was greater than the number of items generated by the select list.

MTKI0133

Translation Error: Translation of SYSTEM command might not work as is.

Description

INFX.SYSTEM expects its argument to have a specific format. See the documentation for details.

MTKI0134

Translation Error: Trigger actions with several procedure calls might result in colliding declarations.

Description

In Informix, trigger actions can contain procedure calls. Since this is not allowed in DB2 UDB, each procedure call is inlined by copying the body of the procedure directly inside the trigger.

If several procedure calls are made, and if the procedures have variables or parameters with the same names, the resulting declarations will collide in the resulting DB2 UDB trigger.

This problem can be solved by editing the resulting trigger and renaming the duplicate variables or parameters.

MTKI0135

Translation Warning: BEFORE trigger was translated to AFTER trigger.

Description

BEFORE triggers are usually translated to NO CASCADE BEFORE triggers in DB2 UDB. Unfortunately this kind of trigger does not allow DML statements (INSERT, UPDATE and DELETE) and the FOR EACH STATEMENT clause.

If the source trigger does use one of these items, the translator will try to translate the BEFORE trigger to an AFTER trigger, which does not have those restrictions.

This special translation is not possible if the body of the trigger refers to the table as the object of the trigger or if a column of the NEW table is assigned.

MTKI0136

Translation Error: Collection types are not supported in this context.

Description

Collection types are supported in local variable declarations and formal parameters only.

MTKI0137

Translation Error: Complex column default on a nullable column.

Description

This complex column default has been translated inside a special trigger. The column is nullable, so the trigger cannot distinguish between a null value and an unspecified value and, thus, will change the value using the default.

You must review the code to check that null values are not inserted.

If this is not the case, a solution is to use a special value (a value that is not likely to be inserted) as the default for the column. Then test the inserted value against this special value inside the trigger instead of using null.

MTKI0138

Translation Warning: Trigger generated for this table's complex defaults and check constraints

Description

Complex column defaults and check constraints cannot be translated directly into the DB2 UDB CREATE TABLE statement. An extra trigger is generated for this table. This trigger will perform the necessary operations to emulate the defaults and check constraints.

MTKI0139

Translation Error: NULL constraint was removed because of the complex DEFAULT.

Description

This column definition contains a complex default that was translated as a trigger. This special translation requires removal of the NOT NULL constraint so that NULL values can be used to represent an unspecified value.

A manual code review is necessary.

MTKI0140

Input Script Error: Invalid number format

Description

The number format is invalid or the input data (a number value) does not match the number format.

MTKI0141

Translation Warning: RETURN statements are not allowed in DB2 UDB excepetion handlers.

Description

In DB2 UDB, RETURN statements are not allowed in condition handlers. In some situations the RETURN statement can be removed; it is necessary to perform a manual code review to ensure that the control flow of the handler body is correct.

MTKI0142

Translation Error: Unable to parse the literal collection.

Description

MTK is unable to parse the literal collection contained in the string. This information is not in a format understood by MTK.

MTKI0143

Translation Error: Routine overloading not allowed or not supported

Description

Function or procedure overloading is either not allowed or not supported in this context. The object is translated, but calls to this function or procedure will not resolve correctly.

MTKI0144

Translation Error: Translation of cursor function might not be correct

Description

If the cursor function and the loop where it is called depend on each other (for example, if the cursor function the changes global variables or tables that the loop uses or vice versa), the DB2 UDB translation might give different results from the original. In this case manual translation is required.

MTKI0153

Translation Warning: TRUNCATE is translated using DELETE inside procedures.

Description

The semantics of the TRUNCATE statement are to delete all rows in the table without keeping a log of the operation. This statement is typically used for very large tables. DB2 UDB does not have equivalent functionality inside of a procedure body. Currently, TRUNCATE statements inside procedures are translated to DELETE statements for which a log is kept. If the size of the log file grows too large, a failure might occur.

MTKI0160

Translation Warning: FOR EACH ROW trigger was translated to NO CASCADE BEFORE trigger

Description

FOR EACH ROW triggers are usually translated to AFTER triggers in DB2 UDB. This kind of trigger does not allow changing a column value.

If the source trigger uses this features, the converter tries to translate the FOR EACH ROW trigger to a NO CASCADE BEFORE trigger, which does not have this restriction.

This special translation is not possible if the body of the trigger contains DML statements.

MTKI0161

Translation Error: Translation requires a FETCH FIRST clause, which is not supported in subqueries in DB2 UDB V7.

Description

Translation requires a FETCH FIRST clause, which is not supported in sub-queries in DB2 UDB V7.

MTKI0163

Translation Error: Unable to translate subquery assignment in joined update in this context.

Description

MTK is unable to translate sub-query assignment in joined update when some assignments refer to tables other than the one being updated.

MTKI0164

Translation Error: Table <table name> is involved in an outer join or a parenthesized operation.

Description

MTK is unable to generate a correct translation of a joined update or delete, because the table being modified is involved in an outer join or parenthesized operation. It cannot be removed from the from clause, which is required for a valid translation.

MTKI0191

Translation Error: Column <column name> is not in the group by clause but is used in a select, order by or having clause.

Description

This sub-query contains a group-by clause and the select clause, order-by clause, or having clause of this sub-query contains a reference to a column that is not in the group-by clause and is not inside of a call to an aggregate function. This is an extension to the SQL standard that is not supported by MTK or DB2 UDB.

MTKI0195

Translation Error: Nonliteral error numbers are not supported

Description

MTK is capable of translating literal error numbers only. Manual translation is required for this non-literal expression.

MTKI0196

Translation Error: ISAM error numbers are not supported.

Description

ISAM error numbers are not translated to DB2 UDB because there is no equivalent in DB2 UDB.

MTKI0212

Translation Error: This view cannot be updated in DB2. The check option is omitted.

Description

The check option is not allowed in DB2 UDB in the definition of read-only views.

MTKI0213

Translation Error: This view has been declared as read-only.

MTKI0214

Translation Error: The base view or table-expression used to define this view is read-only.

MTKI0215

Translation Error: This view is read-only because it selects only distinct elements.

MTKI0216

Translation Error: This view might not be updatable because it selects complex non-column values.

MTKI0217

Translation Error: This view is read-only because it has no FROM clause.

MTKI0218

Translation Error: This view is read-only in DB2 because it has multiple base tables.

Description

This view has multiple base tables. Deletes, updates, or inserts to such a view are not standard and are similar to joined deletes and joined updates.

MTKI0219

Translation Error: This view is read-only because it has a GROUP BY clause.

MTKI0220

Translation Error: This view is read-only because it has a HAVING clause.

MTKI0221

Translation Error: This view might be read-only because of the use of UNION.

MTKI0226

Translation Error: Cannot return a FULLSELECT in this context.

Description

The following are the restrictions for using FULLSELECT in a RETURN statement:
- If the FULLSELECT is returned from scalar function, the FULLSELECT must return one column, and at most, one row.
- A FULLSELECT cannot be specified for a RETURN from a procedure.
- If the FULLSELECT is returned from a row function, it must return, at most, one row.

MTKI0227

Translation Error: No column-list specified in the RETURN clause of the CREATE FUNCTION statement.

Description

The translator is unable to determine the columns returned by the CREATE TABLE FUNCTION.

MTKI0244

Translation Warning: Label <label name> associated with the removed nested compound statement has been moved to the first contained statement.

Description

The associated label, along with the contents of the compound statement, have been moved to the first contained statement. You must review all references to the label. Manual translation of some references might be required.

MTKI0245

Translation Warning: The translator has removed the nested DB2 UDB compound statement by moving the contents to a higher block.

Description

The translator has removed the nested DB2 UDB compound statement. The contents of the compound statement have been moved to a higher block.

MTKI0246

Translation Warning: Nested compound statements are not supported inside an exception handler. Manual translation of the statement is required.

Description

This version of DB2 UDB does not support nested compound statements. Consider replacing the compound statement with a stored procedure call.

MTKI0247

Translation Warning: Nested compound statements containing an exception handler are not supported. Manual translation of the statement is required.

Description

This version of DB2 UDB does not support nested compound statements. Consider replacing the compound statement with a stored procedure call.

Oracle converter messages

The Oracle converter might return the following messages.

MTKO0000

Translation Information: MTK Oracle Converter. Version: <mtk version>

Description

Specifies the version of the Oracle converter.

MTKO0001

Translation Error: This statement is not translated.

Description

This statement is not translated.

MTKO0002

Input Script Error: Unexpected syntax - not translated.

Description

The translator encountered some text that it could not understand.

MTKO0003

Fatal Internal Error: Error walking the AST.

Description

The translator encountered some text that it could not understand.

MTKO0004

Input Script Error: Unrecognized character - skipped.

Description

The translator does not understand this character.

MTKO0005

Translation Error: This feature is not translated.

Description

A feature of the source code was encountered and recognized as a feature that is not translated in this version of MTK.

MTKO0006

Translation Error: Untranslated Expression: <cause>

Description

The expression is not translated for the given reason.

MTKO0010

Translation Error: Unable to open file ″<filename>″ for output.

Description

MTK was unable to open the specified file for output. Ensure the file exists in the specified directory and is not locked or open by another application.

MTKO0011

Input Script Error: Reference to unknown <object>: <object name>

Description

The translator is not aware of the definition of this object.

Ensure that the definition exists in the file being translated or in one of the files used as context for this translation. Also, ensure that the schema name is specified as indicated.

Related tasks

"Referring to previously converted metadata" on page 34
If you are converting metadata that refers to other metadata that has already been converted, you still must include all of the files in the conversion. However, your translated data might be in a state where you would not want to translate it again. You can specify that the IBM Migration Toolkit use certain metadata *in-context*.

MTKO0012

Input Script Error: Referenced table does not have a primary key.

Description

The table being referenced in this foreign key constraint does not have a primary key. Assign a primary key in the original source and re-convert.

MTKO0013

Input Script Error: No matching unique or primary key for this column list.

Description

The list of columns in this foreign key constraint has no matching unique or primary key constraint in the table it is referencing. Assign a primary key in the original source and re-convert.

MTKO0014

Input Script Error: Unknown command, rest of line ignored.

Description

The translator does not recognize this command, so the entire line is skipped.

MTKO0015

Input Script Error: Unknown statement, ignored.

Description

The translator does not recognize this statement, so the entire statement is skipped.

MTKO0016

Input Script Error: Duplicate definition of <object name>

Description

This object has already been defined. The translator will use the previous definition.

MTKO0017

Translation Error: This ALTER TABLE clause is not supported.

Description

Only the add constraint clause is translated in the ALTER TABLE statement. All other ALTER TABLE clauses are ignored.

Related reference

"Statements" on page 206
The converter accepts the top-level statements and clauses shown in the table below.

MTKO0018

Input Script Error: Possibly ambiguous column reference: <column name>

Description

This column occurs in more than one table in the from clause (or it occurs more than once as a column heading in the select list for an order by clause). The translation might cause an error in the target server.

Related reference

"Statements" on page 206
The converter accepts the top-level statements and clauses shown in the table below.

MTKO0019

Input Script Error: Call to unknown <subprogram type>: <subprogram type>

Description

The translator could not resolve the name in this function/procedure call. The name was not recognized as an Oracle built-in function name or as the name of a previously defined user-defined function (UDF) or procedure.

Related reference

"Built-in-functions" on page 159
There are three possible scenarios when converting Oracle built-in functions to the target IBM database (either DB2 UDB or Informix Dynamic Server).

MTKO0020

Translation Warning: Object name has been changed to <new name>.

Description

Object names that are too long for the target server are truncated. Names that are reserved words in the target server are enclosed in double quotes. Names that conflict with other names in the target server (because the name is already in use) are renamed.

MTKO0021

Translation Error: Call to <subprogram type> <subprogram name> is not supported.

Description

This Oracle function or procedure call is not translated to the target server.

Related reference

"Built-in-functions" on page 159
There are three possible scenarios when converting Oracle built-in functions to the target IBM database (either DB2 UDB or Informix Dynamic Server).

MTKO0022

Input Script Error: No matching instance of <subprogram type> <subprogram name> with such arguments.

Description

The translator expects a different number of arguments when calling this function or procedure, or the argument types do not match any of the instances of this function or procedure.

Related reference

"Built-in-functions" on page 159
There are three possible scenarios when converting Oracle built-in functions to the target IBM database (either DB2 UDB or Informix Dynamic Server).

MTKO0023

Fatal Internal Error: Translator runtime error.

Description

The translator experienced an unexpected internal error. Report this error to IBM Migration Toolkit support.

MTKO0024

Translation Warning: NOT NULL constraint is added because of a PRIMARY KEY or UNIQUE constraint.

Description

In the target server, a column with a UNIQUE or PRIMARY KEY constraint must also have a NOT NULL constraint.

MTKO0025

Translation Error: Insert/Delete/Update on subquery or table collection expression not translated.

Description

This insert, delete, or update operates over a sub-query or a table collection expression. This operation is not supported in the target database and is not translated.

MTKO0026

Translation Error: RESCINDED. Create Synonym for object other than table, view or synonym not translated.

Description

RESCINDED.

DB2 UDB synonyms are supported for tables, views and synonyms only. Synonyms for other objects are not translated.

MTKO0027

Translation Error: RESCINDED. Translation of default expression results in a non-constant expression.

Description

RESCINDED.

In DB2 UDB, default expressions can be constants, datetime special registers, USER, NULL, or certain forms of cast expressions only.

MTKO0028

Translation Error: Incompatible Nulls First or Nulls Last clause: <conflicting clauses>

Description

DB2 UDB does not support NULLS LAST in a DESC order or NULLS FIRST in an ASC (the default) order.

MTKO0029

Translation Error: Subquery in FROM clause: not yet translated.

Description

SELECT statements that have sub-queries in the FROM clause are not yet translated.

MTKO0030

Input Script Error: Invalid date format

Description

The date format is invalid or the input data (a date value) does not match the date format.

MTKO0031

Translation Error: VALUE set clause of update statement not translated.

Description

The target server does not have an equivalent set clause in the update statement.

MTKO0032

Translation Warning: Order by clause in INSERT, SELECT INTO, VIEW or derived table subquery not translated.

Description

The target server does not allow the order by clause in a sub-query to insert values in an INSERT statement or to define a VIEW.

MTKO0033

Input Script Error: Inserted data error

Description

The number of inserted values does not match the number of columns specified for the table.

MTKO0034

Translation Information: No translation available, but the statement has been taken into account

Description

There is no DB2 UDB translation available, but the converter will use the information in the statement in translating the statements that follow.

MTKO0035

Input Script Error: All columns of the subquery of a view must be named.

Description

The columns of the view must be named by either giving an explicit list of names in the definition, or by using column aliases in the sub-query. In this case, there is no list of names and at least one sub-query column has no name.

MTKO0036

Input Script Error: Number of columns of subquery must match names given in view definition.

Description

In a view, if a list of column names is given in the definition, this must match the number of columns in the result set of the sub-query.

MTKO0037

Translation Error: Invalid or unsupported outer-join query

Description

The following restrictions apply to the translation of outer-join queries:
- Cyclic outer-joins in the WHERE clause are not translated (invalid input).
- The (+) operator must not follow a complex expression; it can follow a column reference only.
- Only the equality (=) operator is supported.

MTKO0038

Input Script Error: Type mismatch: expression has unexpected data type

Description

The expression has an inappropriate data type in this context.

MTKO0039

Input Script Error: Unexpected size of expression-list or result-set

Description

The size of this expression-list or result-set (generated by a sub-query) does not match the size expected by the context.

This error is likely to happen in:

- INSERT statements if the VALUES clause or the result-set generated by the sub-query does not match the table size
- Comparisons, if the sizes of the elements on each side of the comparison operator do not match

MTKO0040

Translation Error: Conversion from type <SourceType> to type <TargetType> is not supported by MTK

Description

An implicit conversion from a value of the source type to the target type is not directly supported in the target database server.

MTKO0041

Translation Warning: Unable to infer the target server data type for the expression. Using Varchar(1).

Description

A data type is needed in this context for use in a declaration or cast expression.

MTKO0042

Input Script Error: Unrecognized data type: <data type>

Description

This data type is not recognized by the translator. The reason might be that:
- It is not a valid Oracle data type.
- It is a user-defined type, not yet handled by the translator (for example, collections and object types).

If the data type is encountered as a parameter to a procedure, a suggested translation is provided, but it is commented out because it will probably require manual editing.

MTKO0043

Input Script Error: Length required for type <data type>

Description

Length must be specified for this data type. For instance, when declaring a column of type VARCHAR, VARCHAR2 or RAW, length must be explicitly defined, as in: VARCHAR(10), RAW(100)

MTKO0044

Translation Warning: Warning: Negative scale and scale greater than the precision are not supported in the target server.

Description

The target server DECIMAL data type does not support negative scales or scales greater than the precision. The translator adjusts the precision and the scale in order to avoid loss of data, but the target server results might differ.

For example:

```
CREATE TABLE T(X NUMBER(4,-2))
```

is translated into:

```
CREATE TABLE T(X DECIMAL(6,0))!
```

But the result of an INSERT VALUES(123456) will be `123400` in Oracle and 123456 in DB2 UDB if DB2 UDB is the target server.

MTKO0045

Translation Error: Untranslated data type: <data type>

Description

The Oracle data type does not have an equivalent in the target server and could not be translated. This can happen when the translation of a NUMBER(p,s) is not able to capture any digit of the source data type (when s-p>31 or s<-31).

MTKO0046

Translation Warning: Non blank-padded comparison cannot be enforced here

Description

This happens when the non-blank-padded comparisons enforcement option is set, in membership comparisons involving expression-lists where CHAR and VARCHAR are being compared.

For example, if `a` and `b` are CHAR, `c` is VARCHAR and `xn` is any other data:

```
(a, x1) IN ((b, x2),(c,x3))
```

cannot be translated accurately, because `a` versus `b` should be a non-blank-padded comparison, whereas `a` versus `c` is blank-padded. The only way to enforce blank-padding rules here is to explode this membership comparison into several comparisons, which the translator does not handle. In the above example, this would be converted to:

```
(a = b AND x1 = x2) OR (ORA.NO_PAD(a) = ORA.NO_PAD(c) AND x1 = x3)
```

MTKO0047

Translation Warning: RESCINDED. BEFORE translated to NO CASCADE BEFORE

Description

Oracle triggers using the BEFORE event type are translated to NO CASCADE BEFORE triggers in target database server. This type of trigger does not allow other triggers to fire from the trigger body, a situation that might lead to incorrect behavior.

MTKO0048

Translation Error: DDL and database event triggers are not translated.

Description

Triggers using Data Definition Language (DDL) events and database events are not translated.

MTKO0049

Translation Error: INSTEAD OF triggers are not supported in this version of the target server.

Description

DB2 UDB Version 7.2 does not support INSTEAD OF triggers. The version 8 translation is given here.

MTKO0050

Translation Error: This statement is not supported in the target server dynamic compound statement.

Description

This statement is not supported in the target server dynamic compound statement. Dynamic compound statements are used as bodies for top-level anonymous blocks, user-defined functions, and triggers. Procedures use the target server compound statements as bodies that are less restrictive. Some statements that are not allowed inside the target server dynamic compound statements are:

- Nested blocks
- Statements containing CASE expressions
- GOTO
- Procedure calls
- Cursors
- COMMIT
- Exception handlers

If the compound statement is found in an Oracle function, depending on the use of the function, changing it to a procedure might be a better approach.

Related reference

"Statements" on page 206
The converter accepts the top-level statements and clauses shown in the table below.

MTKO0051

Translation Error: BEFORE triggers without FOR EACH ROW are not supported.

Description

In the target server, FOR EACH STATEMENT must not be specified for BEFORE triggers. For this reason, BEFORE triggers that do not specify FOR EACH ROW cannot be translated.

MTKO0052

Input Script Error: Table name <table> not allowed in this context.

Description

In this context an expression is expected. A table name that does not qualify a column name or a star (*) is not allowed.

MTKO0053

Input Script Error: No column <column name> in table.

Description

The given table has been found in a surrounding from clause, but it does not contain the indicated column.

MTKO0054

Input Script Error: Parameters in named notation cannot be followed by parameters in positional notation.

Description

The actual parameters in this procedure/function call use bad mixed notation. Mixed parameter notation is only allowed if named parameters follow positional parameters.

MTKO0055

Translation Error: All FETCH statements from the same cursor must be into the same variables.

Description

The target server has no equivalent clause to declare the return type of a cursor. The behavior of the cursor should not be affected.

MTKO0056

Input Script Error: An INTO clause is expected in this select statement

Description

In a PL/SQL context, a select statement must have an INTO clause.

MTKO0057

Translation Error: Unable to translate subquery with set operations to the target server Select Into.

Description

In PL/SQL a SELECT INTO might be united with other selects. This is not allowed in DB2 UDB.

MTKO0058

Translation Error: This ALTER SESSION clause is not supported.

Description

The following ALTER SESSION clauses are currently simulated (but not translated). The remaining clauses are ignored.
- SET NLS_DATE_FORMAT
- SET CURRENT_SCHEMA
- SET NLS_CURRENCY
- SET NLS_ISO_CURRENCY
- SET NLS_NUMERIC_CHARACTERS

MTKO0059

Translation Warning: Ignored input - not translated.

Description

This input is ignored, since it is not supported in the target server. This omission should not cause the target server code to produce different results from the corresponding Oracle code.

MTKO0060

Translation Error: References to remote database objects are not translated

Description

References to objects in remote databases (indicated by ″@dblink″) are not supported.

MTKO0061

Translation Warning: Parameter defaults are not supported in the target server procedure definitions. Calls to the procedure are adjusted accordingly.

Description

In procedure and function declarations, the optional DEFAULT value of a parameter is not translated, but the translator will use the value as necessary through the remainder of the translation.

Related reference

"Statements" on page 206
The converter accepts the top-level statements and clauses shown in the table below.

MTKO0062

Translation Warning: Labels are not allowed in the target server dynamic compound statements.

Description

Labels are not allowed in target server dynamic compound statements, except for loops. The translator will not translate the label for the statement.

Related reference
"Statements" on page 206
The converter accepts the top-level statements and clauses shown in the table below.

MTKO0063

Translation Warning: Labels are not supported for top-level blocks in the target server.

Description

Labels are not supported for top-level blocks in DB2 UDB. The translator omits these labels.

MTKO0064

Input Script Error: PUBLIC synonym cannot have schema qualifier.

Description

In the definition of a public synonym, the name of the synonym is not allowed to be qualified by a schema name.

MTKO0065

Translation Warning: This save point might not be allowed by the target server.

Description

If another save point is active when this save point is encountered, the target server will reject this save point. The target server allows only one active save point at a time. The converter is unable to determine at compile time the number of save points that can be active when this statement is encountered.

MTKO0066

Translation Error: Autonomous transactions are not supported by the target server.

Description

The target server does not support autonomous transactions. The transaction statements COMMIT, ROLLBACK, and SAVEPOINT in this block and the enclosing dynamic context can result in different behavior in the target server.

MTKO0067

Translation Warning: NOWAIT is not supported by the target server Lock Table statement.

Description

NOWAIT is not supported by the target server Lock Table statement.

MTKO0068

Translation Error: This lock mode is not supported by the target server Lock Table statement.

Description

The target server Lock Table statement supports SHARE and EXCLUSIVE lock modes only.

MTKO0069

Translation Warning: Locks for partitions, subpartitions, and remote tables are not supported by the target server.

Description

These clauses are not translated to the target server.

MTKO0070

Translation Error: OUT and IN OUT parameters are not supported in the target server user-defined functions.

Description

In user-defined functions, the target server only allows input parameters. The translator cannot properly translate a function with OUT or IN OUT parameters to an equivalent construct in the target server. Evaluate the code to determine if you can use a procedure instead.

MTKO0071

Translation Warning: Reference to OLD or NEW column translated to <NULL or variable>

Description

The target server does not accept references to OLD from an inserting trigger or references to NEW from a deleting trigger. In the WHEN clause of the trigger, these references are translated to NULL. In the body of the trigger, they are translated to a variable generated for this purpose.

MTKO0072

Translation Error: The target server only allows constants as arguments to a top-level procedure call.

Description

The target server only allows constants as arguments to a top-level procedure call.

MTKO0073

Translation Error: This statement is not supported in the target server UDF.

Description

This statement is not supported in a target server UDF. DML statements, cursors, and exception blocks are examples of items that cannot be in target server functions. Depending on the use of the Oracle function, changing the function to a procedure might be a better approach.

Related reference

"Statements" on page 206
The converter accepts the top-level statements and clauses shown in the table below.

MTKO0074

Translation Error: Nested exception handlers are not supported.

Description

Nested exception handlers are not supported because the target server does not allow the declaration of a handler inside another handler.

MTKO0075

Translation Error: This statement is not supported in the target server Before Trigger.

Description

This statement is not supported in a target server Before trigger. The following statements are not supported in this context:

- INSERT
- DELETE
- UPDATE

MTKO0076

Translation Error: This statement is not supported in a target server After Trigger.

Description

This statement is not supported in a target server After trigger. The following statement is not allowed in this context:

- a SET transition-variable statement (when FOR EACH ROW is specified)

MTKO0077

Translation Error: This form of the SQL Plus Execute command is not supported.

Description

Only procedure calls and begin-end blocks are supported in the SQL Plus Execute command. Other PL/SQL statements in an Execute command are not translated.

MTKO0078

Translation Warning: Assignment of non-constant default expression moved to block.

Description

The target server does not allow a non-constant expression to be the default value of a variable declaration. A statement assigning the expression to the variable has been moved to the beginning of the block.

MTKO0079

Translation Warning: An arbitrary size of 255KB was picked for the LOB type.

Description

In PL/SQL context, certain Oracle data types are translated to LOB(255K) or CLOB(255K). The converter picks 255 KB as an arbitrary size for those Large Object types. However, depending on your needs, the size can be modified (increased or decreased) for better performance.

Considering that DB2 UDB allocates memory for variables and parameters of these types, try to limit the size of these data types as much as possible.

MTKO0080

Translation Error: This package item is not translated.

Description

Only the following items are supported inside a package: function specifications, functions, procedure specifications, procedures, and constants. Variable declarations, cursor declarations and type definitions are not translated.

MTKO0081

Translation Warning: The generated SQLSTATE might be incorrect.

Description

When translating calls to raise_application_error, the translator uses the error code to generate an SQLSTATE. Consult the target server *Message Reference* for rules on specifying the necessary SQLSTATE.

MTKO0082

Translation Error: Untranslated reference.

Description

This reference was not translated because the syntax is incorrect or the feature is not supported.

MTKO0083

Translation Warning: The called function was not translated successfully.

Description

The function definition of the function being called was not translated successfully. The function definition must be corrected.

MTKO0084

Translation Error: AS Expressions are not yet translated.

Description

AS Expressions are not yet translated.

MTKO0085

Translation Error: Procedure or function call not translated: the procedure or function must be defined first.

Description

Because the target server does not support function or procedure specifications, functions or procedures can only be referenced after they have been fully defined.

If possible, replace the function or procedure specification with its body.

MTKO0086

Translation Warning: No BITMAP index in DB2 UDB, ignored the BITMAP clause.

Description

DB2 UDB has only a UNIQUE index, not a BITMAP index. The translator ignores the BITMAP clause.

MTKO0087

Input Script Error: This specification item has no definition in the input file

Description

The function, procedure, or cursor specification has no corresponding definition in the input file. The specification does not have a target server translation, so this item will not appear in the output file.

MTKO0088

Translation Error: Top-level procedure-calls with CLOB parameters are not supported by the target server.

Description

Procedures with CLOB parameters cannot be called from the top-level in DB2 UDB. Top-level procedure calls require all arguments being passed to be literals, but the target server does not support these kinds of literals in this context. If DB2 UDB is the target, calls result in this error: ″DB2 UDB1036E The CALL command failed.″

MTKO0089

Translation Error: This declaration is not translated.

Description

This declaration is not translated

MTKO0090

Translation Error: EXCEPTION_INIT is not translated. The exception declaration must be modified.

Description

The EXCEPTION_INIT pragma is not translated because it does not have an equivalent in the target server.

To obtain correct behavior in the target server, you must modify the exception declaration by changing the unreserved SQLSTATE to a reserved SQLSTATE that corresponds to the Oracle error number.

MTKO0091

Translation Error: This ALTER TABLE statement is not translated.

Description

This ALTER TABLE statement is not translated because none of its clauses are supported.

MTKO0092

Translation Error: Update or Delete with reference to rownum pseudocolumn not translated.

Description

Either the statement occurs at the top level and thus cannot be translated using a cursor, or the rownum condition is too complicated to translate to a ″FETCH FIRST n ROWS ONLY″ clause.

MTKO0093

Translation Error: Reference to rownum pseudocolumn in set clause of UPDATE statement not translated.

Description

The reference to the row number in a set clause of an UPDATE statement is not supported in the target server.

MTKO0094

Translation Warning: The function <function_name> is translated to the target server as a procedure.

Description

This Oracle user-defined function is translated to DB2 UDB as a procedure. This happens with functions with parameters in OUT mode. Since this feature is not available in the target server, the translator uses a target server procedure instead. The calls to the Oracle function will be translated into procedure calls.

MTKO0095

Input Script Error: Calls to functions with OUT parameter are not allowed in an SQL statement.

Description

A call to a function defined with parameters of type OUT is not allowed in an SQL statement (INSERT, UPDATE, DELETE, SELECT).

MTKO0096

Translation Error: A call to a function translated as procedure is not translated in this context.

Description

In certain cases, Oracle these functions are translated to procedures in the target server:

- Functions using OUT parameters
- Functions containing statements that are not allowed in dynamic compound statements in the target server

Calls to such functions in Oracle must be translated to procedure calls in the target server, which is not always possible.

There are two different kinds of context in which the translation of the function call fails:

- In a target server dynamic compound statement (function/trigger/anonymous block bodies), because procedure calls are not allowed in them.
- In branching language constructs (ELSIF, WHILE, etc..), because these constructs require a complex workaround that the translator does not handle.

  For example, consider the function `foo(a OUT INT)` returning an `int` in Oracle:

  ```
  IF (...)
    ...
  ELSIF (foo(x)=0)
    statements
  ENDIF;
  ```

  This would need to be translated to:

  ```
  IF (...)
    ...
  ELSE
    CALL foo(x return_val);
    IF (return_val=0)
      statements
    END IF;
  END IF;
  ```

MTKO0097

Translation Error: ROLLUP and CUBE are not translated.

Description

The ROLLUP and CUBE extensions of the GROUP BY clause are not yet translated.

MTKO0098

Translation Warning: A column definition has been changed because of an ALTER TABLE statement.

Description

A column definition has been added or modified because of a subsequent ALTER TABLE statement.

MTKO0099

Translation Warning: This ALTER TABLE clause has been taken into account by modifying the column definition.

Description

This ALTER TABLE is not directly supported in the target server, but the converter took it into account by modifying the corresponding column definition.

MTKO0100

Translation Error: This match-expression or pattern-expression is not allowed in the target server LIKE predicate.

Description

This match-expression or pattern-expression is not allowed in the target server LIKE predicate. Please refer to the LIKE predicate section in the target server documentation for more details.

MTKO0101

Translation Warning: Temporary variable <variable name> used to avoid name clash with column name <column name> in INSERT.

Description

In a target server INSERT statement, variable names must not be the same as the names of the columns in the insert table. A temporary variable with a distinct name has been used in place of the original variable that had the same name as a column.

MTKO0102

Translation Error: This cursor for loop cannot be translated because the cursor is invalid.

Description

This cursor for loop cannot be translated because the cursor is invalid.

MTKO0103

Translation Error: Reference to <object> was not translated because the declaration failed.

Description

This reference was not translated because the declaration of the corresponding object failed.

MTKO0104

Translation Warning: Reference to <object> was not translated because the object is not supported.

Description

This reference was not translated because the declaration of the corresponding object is not supported.

MTKO0105

Translation Warning: CREATE TABLESPACE generated with minimal default parameters.

Description

The ″TABLESPACE″ option of the translator is turned on. For each TABLESPACE clause found, a ″CREATE TABLESPACE″ statement is generated at the beginning of the output file. This warning indicates that minimal default parameters have been used and, therefore, you must change or enhance these, depending on the physical properties that are required.

MTKO0106

Translation Error: Only one INSTEAD OF trigger per view and per event is allowed in the target server.

Description

In the target server, only one INSTEAD OF trigger is allowed for each kind of operation on a given subject view. Try merging the conflicting triggers in a single target server trigger.

MTKO0107

Translation Warning: SQLCODE or SQLERRM was translated directly using the target server SQLCODE or the target server message text.

Description

References to the Oracle SQLCODE are directly translated to references to the DB2 UDB SQLCODE.

References to the Oracle SQLERRM are translated using the DB2 UDB GET STATISTICS statement to retrieve the error message text.

Most of the time, the error codes and messages do not match in Oracle and DB2 UDB, so it might be necessary to modify the code.

For example, if the SQLCODE is compared to a literal value, it is necessary to change this value to the corresponding SQLCODE in DB2 UDB.

MTKO0108

Translation Information: Translation Ratio: <percentage>% (<absolute ratio> statements were translated successfully)

Description

This provides an assessment of the provided translation by giving the ratio of Oracle statements translated without producing any error message out of the total number of statements. This number provides a general indication regarding the success of the automated translation and does not intend to give an exact and accurate measure.

Statement here designates Oracle SQL and PL/SQL statements. For instance, in a CREATE PROCEDURE, the whole SQL statement is counted as 1 (one) and each PL/SQL statement inside the body of the procedure is also counted as one.

MTKO0109

Translation Error: A procedure call in the trigger body is not translated.

Description

In Oracle the body of a trigger can be either a PL/SQL block or a procedure call statement. The translator does not support call statements for trigger bodies.

The target server trigger bodies do not support procedure calls, so embedding the procedure call in a block will not solve the problem.

A solution is to copy the procedure code into the trigger block, if this is possible.

MTKO0110

Translation Error: Arguments having a structured type are not supported: <structured-type-name>

Description

The converter does not support the translation of arguments having structured types (%ROWTYPE and record types). In this context, DB2 UDB does not support such kinds of data types.

One possible workaround is to explode this structured argument into several arguments having scalar data types. However, this would modify the signature of the routine which means the calls to the function/procedure should be modified accordingly, including calls located in external applications.

MTKO0111

Input Script Error: This is not a procedure: <object-name>

Description

In this procedure call, the references resolve to an object that is not a procedure.

MTKO0112

Translation Error: The following objects are involved in a circular dependency: <objects>

Description

The specified objects are involved in a circular dependency.

Since the target server does not support forward declarations (through function specifications, cursor specificationss), there is no way to declare these objects in the target server.

The specified objects are translated, but they contain unresolved references.

MTKO0113

Input Script Error: This reference does not resolve to <a specific object type>: <reference>

Description

The given reference is made to an object with a type that is unexpected in this context. For instance, in a procedure call, the given name is a function.

MTKO0114

Translation Warning: The CREATE SYNONYM statement is not translated. But any object referred to by the synonym is referred to directly.

Description

In the target server, it is possible to create an alias on a table, a view or another alias, but not on a function, procedure, sequence or schema (which is used to translate packages). There is therefore no direct translation for a CREATE SYNONYM on these objects.

To produce a correct translation, the converter does not translate the CREATE SYNONYM statement, but replaces all references to the synonym by a reference to the object it designates.

MTKO0115

Input Script Error: This is not a table or view: <object-name>

Description

In this context, the reference resolves to an object that is not a table or view.

MTKO0116

Translation Error: Oracle Rowid values are ignored.

Description

The Oracle Rowid type is translated to Integer in DB2 UDB. The rowid values that identify rows in the database are generated by DB2 UDB. Only these DB2 UDB-generated values can be stored in the DB2 UDB database.

MTKO0117

Translation Warning: DECIMAL type for arguments is translated with arbitrary precision (31) and scale (0)

Description

For arguments to functions, procedures, or parameterized cursors, Oracle requires that a precision, scale, or length for the data type not be specified. Oracle automatically specifies them based on the actual arguments used in calls.

The target server, however, does require that those parameters be specified. Since the translator cannot infer the precision and scale to pick when translating to the target server, for the DECIMAL type, it picks arbitrary values: `31` for precision and `0` for scale. Type conversions (ROUND...) are also applied accordingly.

MTKO0118

Translation Error: Support for Rowid column not enabled.

Description

This reference to ROWID will not be valid in DB2 UDB unless the translation is performed with the ″rowid columns″ option enabled. This option adds an identity column called ROWID to each base table in DB2 UDB.

MTKO0120

Translation Error: The CREATE SEQUENCE statement contains a value that is out of range.

Description

The CREATE SEQUENCE statement contains a MINVALUE, MAXVALUE or START VALUE clause with a value that is outside the range that the converter can handle.

MTKO0121

Translation Error: The index matches a unique constraint of the table.

Description

The target server automatically generates an index for each unique constraint (or primary key) of the tables. It will therefore reject the creation of indexes matching these constraints.

The translation of this CREATE INDEX statement was successful, but it will probably cause an error in the target server.

MTKO0129

Translation Error: This statement is not supported in a target server procedure, function, or trigger.

Description

Some statements, especially DDL statements such as ALTER TABLE or CREATE objects other than tables, indexes, or views, are not allowed inside the target server procedures, functions, and triggers. In the context of a procedure, such statements can be executed using dynamic SQL.

MTKO0135

Translation Warning: BEFORE trigger was translated to AFTER trigger.

Description

BEFORE triggers are usually translated to NO CASCADE BEFORE triggers in DB2 UDB. Unfortunately this kind of trigger does not allow DML statements (INSERT, UPDATE and DELETE) and the FOR EACH STATEMENT clause.

If the source trigger does use one of the features, the translator will try to translate the BEFORE trigger to an AFTER trigger, which does not have those restrictions.

This special translation is not possible if the body of the trigger refers to the table as the object of the trigger, or if a column of the NEW table is assigned.

MTKO0136

Translation Error: Collection types are not supported in this context.

Description

Collection types are supported in local variable declarations and formal parameters only.

MTKO0137

Translation Warning: Complex column default on a nullable column

Description

This complex column default has been translated inside a special trigger. The column is nullable, so the trigger cannot distinguish between a null value and an unspecified value. The converter will change the value using the default.

You must review the code to check that null values are not inserted.

If this is not the case, a solution is to use a special value (that is not likely to be inserted) as the default for the column, and to test the inserted value against this special value inside the trigger instead of using null.

MTKO0138

Translation Information: Trigger generated for this table's complex defaults and check constraints

Description

Complex column defaults and check constraints cannot be translated directly into the target CREATE TABLE statement. An extra trigger is generated for this table. This trigger will perform the necessary operations to emulate the defaults and check constraints.

MTKO0139

Translation Warning: NOT NULL constraint was removed because of the complex DEFAULT.

Description

This column definition contains a complex default that was translated as a trigger. This special translation requires the removal of the NOT NULL constraint so that NULL values can be used to represent an unspecified value.

A manual code review is necessary.

MTKO0140

Input Script Error: Invalid number format

Description

The number format is invalid or the input data (a number value) does not match the number format.

MTKO0141

Translation Error: RETURN statements are not allowed in the target server exception handlers.

Description

In the target server, RETURN statements are not allowed in condition handlers. In some situations the RETURN statement can be removed, but you must perform a manual code review to ensure that the control flow of the handler body is correct.

MTKO0143

Translation Error: Routine overloading not allowed or not supported.

Description

Function or procedure overloading is either not allowed or not supported in this context. The object is translated, but calls to this function or procedure will not resolve correctly.

MTKO0145

Translation Warning: The translation of TIMESTAMP(p) might cause the loss of significant digits.

Description

The Oracle TIMESTAMP(p) type is translated to TIMESTAMP in the target server regardless of the provided precision. If the precision is greater than 6 for DB2 UDB as a target or greater than 5 for Informix Dynamic Server as a target, the target server will loose some significant digits in the fractional part of the timestamp.

MTKO0191

Translation Error: Column <column name> is not in the group by clause, but is used in a select, order by or having clause.

Description

This sub-query contains a group-by clause; and the select clause, order-by clause, or having clause of this sub-query contains a reference to a column that is not in the group-by clause and is not inside of a call to an aggregate function. This is an extension to the SQL standard that neither MTK or the target server supports.

MTKO0194

Translation Error: Unsupported predefined exception: <exception name>

Description

This Oracle predefined exception does not clearly match a DB2 UDB SQL state. Manual translation is required.

MTKO0211

“Messages” on page 131: Column <column name> occurs with an aggregate function in a select or order by clause without a group by clause.

Description

This subquery contains an aggregate function without a GROUP BY clause and the SELECT clause or ORDER BY clause of this subquery contains a reference to a column that is not inside a call to an aggregate function. This is a PL/SQL feature that neither MTK or the target server supports.

MTKO0212

Translation Error: This view cannot be updated in DB2. The check option is omitted.

Description

The check option is not allowed in the definition of read-only views in the target server.

MTKO0213

Translation Error: This view has been declared as read-only.

MTKO0214

Translation Error: The base view or table-expression used to define this view is read-only.

MTKO0215

Translation Error: This view is read-only because it selects only distinct elements.

MTKO0216

Translation Error: This view might not be updatable because it selects complex non-column values.

MTKO0217

Translation Error: This view is read-only because it has no FROM clause.

MTKO0218

Translation Error: This view is read-only in DB2 because it has multiple base tables.

Description

This view has multiple base tables. Deletes, updates, or inserts to such a view are not standard and are similar to joined deletes and joined updates.

MTKO0219

Translation Error: This view is read-only because it has a GROUP BY clause.

MTKO0220

Translation Error: This view is read-only because it has a HAVING clause.

MTKO0221

Translation Error: This view might be read-only because of the use of UNION.

MTKO0225

Translation Warning: Unable to validate syntax of dynamic SQL in EXECUTE IMMEDIATE.

Description

The dynamic string argument to the EXECUTE IMMEDIATE statement cannot be translated, because it might contain elements that are not known until runtime. Inspect the string and translate the string manually if necessary.

MTKO0226

Translation Error: Cannot return a FULLSELECT in this context.

Description

The following are the restrictions with using FULLSELECT in a RETURN statement:

- If the FULLSELECT is returned from scalar function, then the FULLSELECT must return one column, and at most, one row.
- A FULLSELECT cannot be specified for a RETURN from a procedure.
- If the FULLSELECT is returned from a row function, it must return, at most, one row.

MTKO0227

Translation Error: No column-list specified in the RETURN clause of the CREATE FUNCTION statement.

Description

The translator is unable to determine the columns returned by the CREATE TABLE FUNCTION.

MTKO0244

Translation Warning: Label <label name> associated with the removed nested compound statement has been moved to the first contained statement.

Description

This version of the target does not support nested compound statements. Consider replacing the compound statement with a stored procedure call.

MTKO0245

Translation Warning: The translator has removed the nested target server compound statement by moving the contents to a higher block.

Description

The translator has removed the nested target server compound statement. The contents of the compound statement have been moved to a higher block.

MTKO0246

Translation Warning: Nested compound statements are not supported inside an exception handler. Manual translation of the statement is required.

Description

This version of the target server does not support nested compound statements. Consider replacing the compound statement with a stored procedure call.

MTKO0247

Translation Warning: Nested compound statements containing an exception handler are not supported. Manual translation of the statement is required.

Description

This version of the target server does not support nested compound statements. Consider replacing the compound statement with a stored procedure call.

MTKO0251

Translation Warning: Ignored the ON DELETE SET NULL clause, since IDS does not support it with a foreign key constraint.

Description

This version of Informix Dynamic Server does not support using the ON DELETE SET NULL clause with a foreign key constraint.

MTKO0252

Translation Warning: No BITMAP index in Informix Dynamic Server; ignored the BITMAP clause.

Description

This version of Informix Dynamic Server does not support BITMAP indexes.

MTKO0253

Translation Error: Informix Dynamic Server does not support the COMMENT statement.

Description

This version of Informix Dynamic Server does not support the COMMENT statement.

MTKO0254

Translation Error: Informix Dynamic Server does not support a reference to rownum pseudocolumn.

Description

The rownum pseudocolumn is not supported in the Informix Dynamic Server target; the translator will ignore the rownum contexts and generate remaining statements into the output file.

MTKO0255

Translation Error: IDS supports only one element in the expression list of the IN predicate.

Description

Informix Dynamic Server supports only one element in the expression list of the IN predicate.

MTKO0256

Translation Error: IDS does not support the use of lists with the ALL operator.

Description

Informix Dynamic Server does not support the use of lists with the ALL operator.

MTKO0257

Translation Error: IDS does not support aggregate functions in the SET clause of UPDATE.

Description

The Informix Dynamic Server UPDATE statement does not support aggregate functions in the SET clause.

MTKO0258

Translation Error: IDS does not support the following in a subselect clause: ORDER BY, UNION, FIRST and INTO TEMP.

Description

The subselect clause of the Informix Dynamic Server SQL grammar might not contain FIRST, INTO TEMP, ORDER BY, or UNION.

MTKO0259

Translation Warning: Unable to infer the IDS data type for the expression using Varchar(1).

Description

A data type is needed in this context for use in a declaration or cast expression.

MTKO0260

Translation Warning: MTK truncates the TIMESTAMP fraction digits that exceed the target limit.

Description

Informix Dynamic Server supports up to five DATETIME fractional digits. MTK truncates the TIMESTAMP fraction digits that exceed this limit.

MTKO0261

Translation Error: The target does not allow this constraint on columns of type <typeName>

Description

The target database server does not allow this constraint on columns of type {0}.

MTKO0262

Translation Error: The target does not allow indexes on columns of type <typeName>

Description

The target database server does not allow indexes on columns of type {0}.

MTKO0263

Translation Warning: This statement cannot be translated because the select list has more than one column.

Description

Informix Dynamic Server does not support WHERE <*column*> IN with more than one column in the select list.

MTKO0264

Translation Warning: This statement is translated but please note that if the columns contain NULL values this translation is incorrect, as the values returned by source and target will not be the same.

Description

MTK maps MINUS in Oracle to WHERE <column> NOT IN for Informix Dynamic Server, if the column contains NULL values the output returned by Informix Dynamic Server will not be the same as Oracle.

MTKO0266

Translation Warning: Assignment of default value moved to code block.

Description

Informix Dynamic Server does not support default values for local variables. The assignment of the default value was moved to the code block.

MTKO0267

Translation Warning: SPL variables that are not default or constant variables are set to NULL.

Description

Informix Dynamic Server does not initialize SPL variables. An SPL variable was set to NULL in the code block.

MTKO0269

Translation Error: Informix Dynamic Server does not support the GOTO statement.

Description

This version of Informix Dynamic Server does not support the GOTO statement in stored procedure language.

MTKO0270

Translation Error: Informix Dynamic Server does not support labels.

Description

This version of Informix Dynamic Server does not support labels in stored procedure language.

MTKO0272

Translation Warning: The value of ROUND''s precision argument has been modified to meet Informix Dynamic Server's requirements.

Description

Informix Dynamic Server''s ROUND precision must be within 32 and -32. The value of precision in this ROUND call has been modified.

MTKO0273

Translation Error: Oracle and Informix Dynamic Server database error numbers are different, check the documentation and manually correct the error number.

Description

The Oracle pragma EXCEPTION_INIT assigned an Oracle error number to the Informix Dynamic Server ON EXCEPTION statement. However, because the Oracle and Informix Dynamic Server error numbers are different you have to manually update the error number in the ON EXCEPTION statement and exceptional variable assignment value. See the Informix Dynamic Server documentation for more information on Informix Dynamic Server errors.

MTKO0274

Translation Error: Informix Dynamic Server does not support SQLCODE and SQLERRM in the Stored Procedure Language.

Description

The target server Informix Dynamic Server does not support SQLCODE and SQLERRM.

MTKO0275

Translation Error: An AFTER trigger with the REFERENCING clause is not supported by Informix Dynamic Server

Description

Informix Dynamic Server does not support an AFTER trigger with the REFERENCING clause.

MTKO0276

Translation Warning: Informix Dynamic Server does not support the Oracle PL/SQL DETERMINISTIC option.

Description

Informix Dynamic Server does not support the Oracle PL/SQL DETERMINISTIC option the in the CREATE FUNCTION statement.

MTKO0277

Translation Error: An AFTER trigger with the FOR EACH ROW clause is not supported by Informix Dynamic Server

Description

Informix Dynamic Server does not support an AFTER trigger with the FOR EACH ROW clause.

MTKO0278

Translation Warning: In some cases Oracle predefined exceptions and Informix Dynamic Server error numbers do not match, check the exception handling manually.

Description

In some cases Informix Dynamic Server generates different exceptions that do not match the Oracle predefined exceptions. Please verify the logic at run time and make manual corrections if necessary.

MTKO0279

Translation Error: Due to Informix Dynamic Server limitations, at most one cursor declaration is supported.

Description

Due to Informix Dynamic Server limitations, MTK translates at most one cursor declaration. Please see the Informix Dynamic Server manuals for more information about the FOREACH statement and SPL.

MTKO0280

Translation Error: FETCH statements within looping statements (e.g. WHILE, FOR LOOP) are not supported.

Description

Due to Informix Dynamic Server limitations, FETCH statements within looping statements (e.g. WHILE, FOR LOOP) are not supported. Please see the Informix Dynamic Server manuals for more information about the FOREACH statement and SPL.

MTKO0281

Translation Error: MTK currently only supports the translation of a single FETCH statement from an open cursor.

Description

When an Oracle cursor is explicitly opened and fetched from MTK, MTK only translates the first FETCH statement.

MTK does offer a more complete translation of the cursor FOR loop statement. Please see the Informix Dynamic Server manuals for more information about the FOREACH statement and SPL.

MTKO0293

Translation Warning: The target does not support the NOT NULL constraint on the ARRAY type.

Description

The target database does not support the NOT NULL constraint on the ARRAY type.

MTKO0294

Translation Warning: Type scope of the type declaration has been changed to global.

Description

The target database does not support type declarations in PL/SQL. Because of this, it has been moved outside of the PL/SQL scope.

MTKO0295

Translation Error: An ARRAY cannot be a data type, the parameters or return type of a user-defined function, or a global variable.

Description

The target database does not support the ARRAY data type because it is not a valid table column type. The ARRAY cannot be the parameters or return type of a user-defined function or a global variable.

SQL Server and Sybase converter messages

The T-SQL converter might return the following messages.

MTKS0000

Translation Information: MTK T-SQL Converter. Version: <mtk version>

Description

Specifies the version of the T-SQL converter.

MTKS0001

Translation Error: This statement is not translated.

Description

This statement is not translated.

MTKS0002

Input Script Error: Unexpected syntax - not translated.

Description

The translator encountered some text that it did not understand.

MTKS0003

Fatal Internal Error: Error walking the AST.

Description

The translator encountered some text that it did not understand.

MTKS0004

Input Script Error: Unrecognized character - skipped.

Description

The translator does not understand this character.

MTKS0005

Translation Error: This feature is not translated.

Description

A feature of the data source code was encountered and recognized as a feature that is not translated in this version of MTK.

MTKS0006

Translation Error: Untranslated Expression: <cause>

Description

The expression is not translated for the given reason.

MTKS0010

Translation Error: Unable to open file ″<filename>″ for output.

Description

MTK was not able to open the specified file for output. Ensure that the file exists in the specified directory and is not locked or open by another application.

MTKS0011

Input Script Error: Reference to unknown <object>: <object name>

Description

The translator is not aware of the definition of this object.

Ensure that the definition exists in the file being translated or in one of the files used as context for this translation. Also, ensure that the schema name is specified as indicated.

Related tasks

"Referring to previously converted metadata" on page 34
If you are converting metadata that refers to other metadata that has already been converted, you still must include all of the files in the conversion. However, your translated data might be in a state where you would not want to translate it again. You can specify that the IBM Migration Toolkit use certain metadata *in-context*.

MTKS0012

Input Script Error: Referenced table does not have a primary key.

Description

The table being referenced in this foreign key constraint does not have a primary key. Assign a primary key in the original source and reconvert.

MTKS0013

Input Script Error: No matching unique or primary key for this column list.

Description

The list of columns in this foreign key constraint has no matching unique or primary key constraint in the referenced table. Assign a primary key in the original source and reconvert.

MTKS0014

Input Script Error: Unknown command; rest of line ignored.

Description

The translator does not recognize this command, so the entire line is skipped.

MTKS0015

Input Script Error: Unknown statement ignored.

Description

The translator does not recognize this statement, so the entire statement is skipped.

MTKS0016

Input Script Error: Duplicate definition of <object name>

Description

This object has already been defined. The translator will use the previous definition.

MTKS0017

Translation Error: This ALTER TABLE clause is not supported.

Description

Only the add constraint clause is translated in the ALTER TABLE statement. All other ALTER TABLE clauses are ignored.

Related reference

"Statements" on page 281
Other than the statements listed in the table, most statements are translated.

MTKS0018

Input Script Error: Possibly ambiguous column reference: <column name>

Description

This column occurs in more than one table in the from clause (or more than once as a column heading in the SELECT list for an ORDER BY clause). The translation might cause an error in the target server.

Related reference

"Statements" on page 281
Other than the statements listed in the table, most statements are translated.

MTKS0019

Input Script Error: Call to unknown <subprogram type>: <subprogram type>

Description

The translator could not resolve the name in this function/procedure call. The name was not recognized as an Oracle built-in function name or as the name of a previously defined user defined function (UDF) or procedure.

Related reference

"Built-in functions" on page 268
Microsoft SQL Server and Sybase functions are mapped directly to the DB2 UDB equivalent where available; and Sybase SQL Anywhere functions are mapped to the IBM Informix Dynamic Server equivalent where available.

MTKS0020

Translation Warning: Object name has been changed to <new name>.

Description

Object names that are too long for the target server are truncated. Names that are reserved words in the target server are enclosed in double quotes. Names that conflict with other names in the target server (because the name is already in use) are renamed.

MTKS0021

Translation Error: Call to <subprogram type> <subprogram name> is not supported.

Description

This function or procedure call is not translated to the target server.

Related reference

"Built-in functions" on page 268
Microsoft SQL Server and Sybase functions are mapped directly to the DB2 UDB equivalent where available; and Sybase SQL Anywhere functions are mapped to the IBM Informix Dynamic Server equivalent where available.

MTKS0022

Input Script Error: No matching instance of <subprogram type> <subprogram name> with such arguments.

Description

The translator expects a different number of arguments when calling this function or procedure, or the argument types do not match any of the instances of this function or procedure.

Related reference

"Built-in functions" on page 268
Microsoft SQL Server and Sybase functions are mapped directly to the DB2 UDB equivalent where available; and Sybase SQL Anywhere functions are mapped to the IBM Informix Dynamic Server equivalent where available.

MTKS0023

Fatal Internal Error: Translator runtime error.

Description

The translator experienced an unexpected internal error. Report this error to IBM Migration Toolkit support.

MTKS0024

Translation Warning: NOT NULL constraint is added because of a PRIMARY KEY or UNIQUE constraint.

Description

In DB2, a column with a UNIQUE or PRIMARY KEY constraint must also have a NOT NULL constraint.

MTKS0025

Translation Error: Insert/Delete/Update on sub-query or table collection expression not translated.

Description

This insert, delete, or update operates over a subquery or a table collection expression. This operation is not supported in the target server and is not translated.

MTKS0026

Translation Error: RESCINDED. Create Synonym for object other than table, view or synonym not translated.

Description

DB2 UDB synonyms are supported for tables, views, and synonyms only. Synonyms for other objects are not translated.

MTKS0027

Translation Error: Translation of default expression results in a non-constant expression.

Description

In DB2 UDB, default expressions can be constants, datetime special registers, USER, NULL, or certain forms of cast expressions only.

MTKS0028

Translation Error: Incompatible Nulls First or Nulls Last clause: <conflicting clauses>

Description

The target server does not support NULLS LAST in a DESC order or NULLS FIRST in an ASC (the default) order.

MTKS0029

Translation Error: Sub-query in FROM clause: not yet translated.

Description

SELECT statements that have sub-queries in the FROM clause are not yet translated.

MTKS0030

Input Script Error: Invalid date gormat

Description

The input data (a date value) does not match the current date format. You might need to change the initial date format and reconvert.

MTKS0031

Translation Error: VALUE set clause of update statement not translated.

Description

The target server does not have an equivalent SET clause in the UPDATE statement.

MTKS0032

Translation Warning: Order by clause in INSERT, SELECT INTO, VIEW or derived table sub-query not translated.

Description

The target server does not allow the ORDER BY clause in a sub-query used to insert values in an INSERT statement or to define a view.

MTKS0033

Input Script Error: Inserted data error

Description

The number of inserted values does not match the number of columns specified for the table.

MTKS0034

Translation Warning: No DB2 UDB translation available, but statement has been taken into account

Description

A translation to the target server is not available; however, the converter will use the information in the statement when translating the statements that follow.

MTKS0035

Input Script Error: All columns of the sub-query of a view must be named.

Description

The columns of the view must be named through either an explicit list of names in the definition or through the use of column aliases in the sub-query. In this view, there is no list of names and at least one sub-query column has no name.

MTKS0036

Input Script Error: Number of columns of sub-query must match names given in view definition.

Description

In a view, if a list of column names is given in the definition, this list must match the number of columns in the result set of the sub-query.

MTKS0037

Translation Error: Invalid or unsupported outer-join query

Description

The following restrictions apply to the translation of outer-join queries:
- Cyclic outer-joins in the WHERE clause are not translated (invalid input).
- The (+) operator must not follow a complex expression; it can only follow a column reference.
- Only the equality (=) operator is supported.

MTKS0038

Input Script Error: Type mismatch: expression has unexpected data type.

Description

This expression has an inappropriate data type in this context.

MTKS0039

Input Script Error: Unexpected size of expression-list or result-set.

Description

The size of this expression-list or result-set (generated by a sub-query) does not match the size expected by the context.

This error is likely to happen:
- In INSERT statements, if the VALUES clause or the result-set generated by the sub-query does not match the table size.
- In comparisons, if the sizes of the elements on each side of the comparison operator do not match.

MTKS0040

Translation Error: Conversion from type <SourceType> to type <TargetType> is not supported by MTK.

Description

An implicit conversion from a value of the source type to the target type in T-SQL is not directly supported by DB2 UDB.

MTKS0041

Translation Warning: Unable to infer DB2 UDB data type for the expression. Using Varchar(1).

Description

A data type is needed in this context for use in a declaration or cast expression.

MTKS0042

Input Script Error: Unrecognized data type: <data type>

Description

This data type is not recognized by the translator. The reason might be that:

- The data type is not a valid T-SQL data type.
- The data type is a user-defined type, not yet handled by the translator.

If the data type is encountered as a parameter to a procedure, a suggested translation is provided, but the translation is commented out because it probably requires manual editing.

MTKS0043

Input Script Error: Length required for type <data type>

Description

Length must be specified for this data type. For instance, when declaring a column of type VARCHAR, VARCHAR2 or RAW, the length must be explicitly defined, as in: VARCHAR(10), or RAW(100).

MTKS0044

Translation Warning: Warning: Negative scale and scale greater then the precision are not supported in DB2 UDB.

Description

The DB2 UDB DECIMAL data type does not support negative scales or scales greater than the precision. The translator adjusts the precision and the scale in order to avoid loss of data, but results in the target server might differ.

For example, in a conversion to DB2 UDB:

```
CREATE TABLE T(X NUMBER(4,-2))

   ----- is translated
to -----

CREATE TABLE T(X DECIMAL(6,0))!
```

However, the result of an INSERT VALUES(123456) will be different between T-SQL and DB2 UDB.

MTKS0045

Translation Error: Untranslated data type: <data type>

Description

The data type has no target server equivalent and could not be translated. This can happen when the translation of a NUMBER(p,s) is not able to capture any digit of the source data type (when s-p>31 or s<-31).

MTKS0046

Translation Warning: Non blank-padded comparison cannot be enforced here.

Description

This happens when the non-blank-padded comparisons enforcement option is set, in membership comparisons involving expression-lists where CHAR and VARCHAR, are being compared.

For example, if `a` and `b` are CHAR, `c` is VARCHAR, and `xn` is any other data, then the expression `(a, x1) IN ((b, x2),(c,x3))` cannot be translated accurately, because `a` versus `b` should be a non-blank-padded comparison whereas `a` versus `c` is blank-padded. The only way to enforce blank-padding rules here is to explode this membership comparison into several comparisons, which the translator does not handle. In the above example, MTK would convert the expression to: `(a = b AND x1 = x2) OR (SYB.NO_PAD(a) = SYB.NO_PAD(c) AND x1 = x3)`.

MTKS0047

Translation Warning: BEFORE translated to NO CASCADE BEFORE

Description

Triggers using the BEFORE event type are translated as NO CASCADE BEFORE triggers in the target server. This type of trigger does not allow other triggers to fire from the trigger body, because this might lead to incorrect behavior.

MTKS0048

Translation Error: DDL and database event triggers are not translated.

Description

Triggers using Data Definition Language (DDL) events and database events are not translated.

MTKS0049

Translation Error: INSTEAD OF triggers are not supported in this version of DB2 UDB.

Description

DB2 UDB Version 7.2 does not support INSTEAD OF triggers. The version 8 translation is given here.

MTKS0050

Translation Warning: This statement is not supported in a DB2 UDB dynamic compound statement.

Description

This statement is not supported in a dynamic compound statement in the target server.

In DB2 UDB, dynamic compound statements are used as bodies for top-level anonymous blocks, user-defined functions, and triggers. Procedures use the compound statements as bodies that are less restrictive. Some statements that are not allowed inside a DB2 UDB dynamic compound statements are:

- Nested blocks
- Statements containing CASE expressions
- GOTO
- Procedure calls
- Cursors
- COMMIT
- Exception handlers

If the compound statement is found in a T-SQL function, depending on the use of the function, changing it to a procedure might be a better approach.

MTKS0051

Translation Error: BEFORE triggers without FOR EACH ROW are not supported.

Description

In DB2 UDB, a FOR EACH STATEMENT cannot be specified for BEFORE triggers. For this reason, BEFORE triggers that do not specify FOR EACH ROW cannot be translated.

MTKS0052

Input Script Error: Table name <table> not allowed in this context.

Description

In this context an expression is expected. A table name that does not qualify a column name or an asterisk/star (*) is not allowed.

MTKS0053

Input Script Error: No column <column name> in table.

Description

MTK found the given table in a surrounding from clause, but the table does not contain the indicated column.

MTKS0054

Input Script Error: Parameters in named notation cannot be followed by parameters in positional notation.

Description

The actual parameters in this procedure/function call use mixed notation. Mixed parameter notation is only allowed if the named parameters follow positional parameters.

MTKS0055

Translation Error: Cursor RETURN clauses are not translated.

Description

The target server does not have an equivalent clause for declaring the return type of a cursor. The behavior of the cursor should not be affected.

MTKS0056

Input Script Error: An INTO clause is expected in this select statement.

Description

In a T-SQL context, a select statement must have an INTO clause.

MTKS0057

Translation Error: Unable to translate sub-query with set operations to DB2 UDB Select Into.

Description

In T-SQL, a SELECT INTO statement can be a union joined with other select statements. This is not permitted in the target server.

MTKS0058

Translation Error: This ALTER SESSION clause is not supported.

Description

The following ALTER SESSION clauses are currently simulated (but not translated). The remaining clauses are ignored.

- SET NLS_DATE_FORMAT
- SET CURRENT_SCHEMA
- SET NLS_CURRENCY
- SET NLS_ISO_CURRENCY
- SET NLS_NUMERIC_CHARACTERS

MTKS0059

Translation Warning: Ignored input - not translated.

Description

This input is ignored. It is not supported in the target server. For translations from T-SQL to DB2 UDB, this omission should not cause the DB2 UDB code to produce different results from the corresponding T-SQL code.

MTKS0060

Translation Error: References to remote database objects are not translated

Description

References to objects in remote databases (indicated by ″@dblink″) are not supported.

MTKS0061

Translation Warning: Parameter defaults are not supported in DB2 UDB procedure definitions. Calls to the procedure are adjusted accordingly.

Description

In procedure and function declarations, the optional DEFAULT value of a parameter is not translated, but the translator will use the value as necessary through the remainder of the translation.

Related reference

“Stored procedures” on page 301
The converter accepts top-level procedure calls with constant arguments.

MTKS0062

Translation Warning: Labels are not allowed in DB2 UDB dynamic compound statements.

Description

Labels, except for loops, are not allowed in dynamic compound statements in DB2 UDB. The translator will not translate the label for this statement.

MTKS0063

Translation Warning: Labels are not supported for top-level blocks in DB2 UDB.

Description

Labels are not supported for top-level blocks in DB2 UDB. The translator omits these labels.

MTKS0064

Input Script Error: A PUBLIC synonym cannot have schema qualifier.

Description

In the definition of a public synonym, the name of the synonym cannot be qualified by a schema name.

MTKS0065

Translation Warning: This savepoint might not be allowed by DB2 UDB.

Description

If another save point is active when this save point is encountered, this save point will be rejected. DB2 UDB allows only one active save point at a time. The converter is unable to determine at compile time the number of save points that might be active when this statement is encountered.

MTKS0066

Translation Error: Autonomous transactions are not supported by DB2 UDB.

Description

DB2 UDB does not support autonomous transactions. The transaction statements COMMIT, ROLLBACK, and SAVEPOINT in this block and the enclosing dynamic context might operate differently in the target server.

MTKS0067

Translation Warning: NOWAIT is not supported by DB2 UDB Lock Table statement.

Description

DB2 UDB does not support NOWAIT in Lock Table statement.

MTKS0068

Translation Error: This lock mode is not supported by the DB2 UDB Lock Table statement.

Description

The DB2 Lock Table Statement supports SHARE and EXCLUSIVE lock modes only.

MTKS0069

Translation Warning: Locks for partitions, subpartitions, and remote tables are not supported by DB2 UDB.

Description

These clauses are not translated to the target server.

MTKS0070

Translation Error: OUT and IN OUT parameters are not supported in DB2 UDB user-defined functions.

Description

In user-defined functions, DB2 UDB only allows input parameters. The translator cannot properly translate a function having OUT or IN OUT parameters to an equivalent construct. Evaluate the code to determine if you can use a procedure instead.

MTKS0071

Translation Warning: Reference to OLD or NEW column translated to <NULL or variable>

Description

DB2 UDB does not accept references to OLD from an inserting trigger or references to NEW from a deleting trigger. In the WHEN clause of the trigger, these references are translated to NULL. In the body of the trigger, they are translated to a variable generated for this purpose.

MTKS0072

Translation Error: DB2 UDB only allows constants as arguments to a top-level procedure call.

MTKS0073

Translation Error: This statement is not supported in a DB2 UDB UDF.

Description

This statement is not supported in a DB2 UDB UDF. DML statements, cursors, and exception blocks are examples of what cannot be in DB2 UDB functions. Depending on the use of the Oracle function, changing the function to a procedure might be a better approach.

MTKS0074

Translation Error: Nested exception handlers are not supported.

Description

Nested exception handlers are not supported, because DB2 UDB does not allow the declaration of a handler inside another handler.

MTKS0075

Translation Error: This statement is not supported in a DB2 UDB Before Trigger.

Description

This statement is not supported in a target Before Trigger. The following statements are not supported in this context in DB2 UDB: INSERT, DELETE and UPDATE.

MTKS0076

Translation Error: This statement is not supported in a DB2 UDB After Trigger.

Description

This statement is not supported in a target After Trigger. The following statement is not allowed in this context in DB2 UDB: a SET transition-variable statement (when FOR EACH ROW is specified).

MTKS0077

Translation Error: This form of the SQL Plus Execute command is not supported.

Description

Only procedure calls and begin-end blocks are supported in the SQL Plus Execute command. Other statements in an Execute command are not translated.

MTKS0078

Translation Warning: Assignment of non-constant default expression moved to block.

Description

The target server does not allow a non-constant expression to be the default value of a variable declaration. A statement assigning the expression to the variable has been moved to the beginning of the block.

MTKS0079

Translation Warning: An abritrary size of 255 KB was picked for the LOB type.

Description

In T-SQL context, certain data types are translated to LOB(255K) or CLOB(255K). The converter picks 255 KB as an arbitrary size for those Large Object types. However, depending on your needs, the size can be modified (increased or decreased) for better performance.

Considering that DB2 UDB allocates memory for variables and parameters of these types, try to limit the size of these data types as much as possible.

MTKS0080

Translation Error: This package item is not translated.

Description

Only the following items are supported inside a package: function specifications, functions, procedure specifications, procedures, and constants. Variable declarations, cursor declarations, and type definitions are not translated.

MTKS0081

Translation Warning: The generated SQLSTATE might be incorrect.

Description

When translating calls to raise_application_error, the translator uses the error code to generate an SQLSTATE. Consult the *DB2 UDB Message Reference* for rules on specifying the necessary SQLSTATE.

MTKS0082

Translation Error: Untranslated reference.

Description

This reference was not translated because the syntax is incorrect or the feature is not supported.

MTKS0083

Translation Warning: The called function was not translated successfully.

Description

The definition of the function being called was not translated successfully. The function definition must be corrected.

MTKS0084

Translation Error: AS Expressions are not yet translated.

MTKS0085

Translation Error: Procedure or function call not translated: the procedure or function must be defined first.

Description

Because target server does not support function or procedure specifications, functions or procedures can only be referenced once they have been fully defined.

If possible, replace the function or procedure specification with its body.

MTKS0086

Translation Warning: No BITMAP index in DB2 UDB, ignored the BITMAP clause.

Description

DB2 UDB only has UNIQUE available as an index. The translator ignores the BITMAP clause.

MTKS0087

Input Script Error: This specification item has no definition in the input file.

Description

The function, procedure, or cursor specification does not have a corresponding definition in the input file. The specification has no translation, so this item will not appear in the output file.

MTKS0088

Translation Error: Top-level procedure-calls with CLOB parameters are not supported by DB2 UDB.

Description

Procedures with CLOB parameters cannot be called from the top-level in DB2 UDB. Top-level procedure-calls require that all arguments that are being passed must be literals, but DB2 UDB does not support these kinds of literals in this context, and CALL commands have this error: ″DB2 UDB1036E The CALL command failed.″

MTKS0089

Translation Error: This declaration is not translated.

MTKS0090

Translation Error: EXCEPTION_INIT is not translated. The exception declaration must be modified.

Description

The EXCEPTION_INIT pragma is not translated because it does not have an equivalent in the target server.

To obtain correct behavior in DB2 UDB, the exception declaration must be modified by changing the unreserved SQLSTATE to a reserved SQLSTATE that corresponds to the T-SQL error number.

MTKS0091

Translation Error: This ALTER TABLE statement is not translated.

Description

This particular ALTER TABLE statement is not translated because none of its clauses are supported.

MTKS0092

Translation Error: Update or Delete with reference to rownum pseudocolumn not translated.

Description

Either the statement occurs at the top level and cannot be translated using a cursor, or the rownum condition is too complicated to translate to a FETCH FIRST *n* ROWS ONLY clause.

MTKS0093

Translation Error: Reference to rownum pseudocolumn in set clause of UPDATE statement not translated.

Description

Reference to the row number in a SET clause of an UPDATE statement is not supported in the target server.

MTKS0094

Translation Warning: The function <function_name> is translated to DB2 UDB as a procedure.

Description

This user-defined function is translated as a procedure.

For translations to DB2 UDB, this type of translation occurs with functions that have parameters in OUT mode. Since OUT parameters are not allowed in functions in DB2 UDB, the translator uses a DB2 UDB procedure instead.

The calls to the function will be translated accordingly; they will become procedure calls.

MTKS0095

Input Script Error: Calls to functions with OUT parameter are not allowed in SQL statements.

Description

A call to a function defined with parameters of type OUT is not allowed in a SQL statement (INSERT, UPDATE, DELETE, or SELECT).

MTKS0096

Translation Error: A call to function translated as procedure is not translated in this context.

Description

The following SQL functions are translated to procedures in DB2 UDB:

- Functions using OUT parameters
- Functions containing statements that are not allowed in DB2 UDB dynamic compound statements

Calls to such functions in T-SQL must be translated to procedure-calls in DB2 UDB, which is not always possible.

There are two different kinds of contexts in which the translation of the function call fails:

- In a DB2 UDB dynamic compound statement (function, trigger, or anonymous block bodies), because procedure calls are not allowed.
- In branching language constructs (ELSIF, WHILE, etc..), which would require a complex workaround that the translator does not handle.

  For example, consider the function: `foo(a OUT INT)`

  It returns an int in T-SQL. Consider the following block:

  ```
  IF (...)
    ...
  ELSIF
    (foo(x)=0)
    statements
  END IF;
  ```

  It would need to be translated to:

  ```
  IF (...)
    ...
  ELSE
    CALL foo(x, return_val);
    IF (return_val=0)
      statements
    END IF;
  END IF;
  ```

MTKS0097

Translation Error: ROLLUP and CUBE are not translated

Description

The ROLLUP and CUBE extensions of the GROUP BY clause are not yet translated.

MTKS0098

Translation Warning: A column definition has been changed because of another statement

Description

A column definition has been added or modified because of a subsequent ALTER TABLE statement or call to SP_BINDEFAULT or SP_BINDRULE.

MTKS0099

Translation Warning: This ALTER TABLE clause has been taken into account by modifying the column definition

Description

This ALTER TABLE is not directly supported in the target server, but it has been taken into account by modifying the corresponding column definition

MTKS0100

Translation Error: This match-expression or pattern-expression is not allowed in the DB2 UDB LIKE predicate.

Description

This match-expression or pattern-expression is not allowed in the DB2 UDB LIKE predicate. Please refer to the LIKE predicate section in the DB2 UDB documentation for more details.

MTKS0101

Translation Warning: Temporary variable <variable name> used to avoid name clash with column name <column name> in INSERT.

Description

In a DB2 UDB INSERT statement, variable names must not be the same as the names of the columns in the insert table. A temporary variable with a distinct name has been used in place of the original variable which had the same name as a column.

MTKS0102

Translation Error: This cursor for loop cannot be translated because the cursor is invalid.

MTKS0103

Translation Error: Reference to <object> not translated because the declaration failed.

Description

This reference was not translated because the declaration of the corresponding object failed.

MTKS0104

Translation Warning: Reference to <object> not translated because the object is not supported.

Description

This reference was not translated because the declaration of the corresponding object is not supported.

MTKS0105

Translation Warning: CREATE TABLESPACE generated with minimal default parameters.

Description

The TABLESPACE option of the translator is turned on. For each TABLESPACE clause found, a CREATE TABLESPACE statement is generated at the beginning of the output file. This warning indicates that the minimal default parameters have been used during migration, and you must change or enhance them depending on the physical properties desired.

MTKS0106

Translation Error: Only one INSTEAD OF trigger per view and per event is allowed in DB2 UDB.

Description

In DB2 UDB, only one INSTEAD OF trigger is allowed for each kind of operation on a given subject view. Try merging the conflicting triggers in a single DB2 UDB trigger.

MTKS0107

Translation Warning: SQLCODE or SQLERRM was translated directly using the DB2 UDB SQLCODE or DB2 UDB message text.

Description

References to the T-SQL SQLCODE are directly translated to references to the DB2 UDB SQLCODE.

References to the T-SQL SQLERRM are translated using the DB2 UDB GET STATISTICS statement to retrieve the error message text.

Most of the time, the error codes and messages between T-SQL and DB2 UDB do not match. It might be necessary to perform some modification to the code.

For example, if the SQLCODE is compared to a literal value, it is necessary to change this value to the corresponding SQLCODE in DB2 UDB.

MTKS0108

Translation Information: Translation Ratio: <percentage>% (<absolute ratio> statements were translated successfully).

Description

This information provides an assessment of the provided translation by giving the ratio of T-SQL statements translated without producing any error messages to the total number of statements. This ratio provides a general indication about the success of the automated translation and does not intend to give an exact and accurate measure.

The term statement here refers to Sybase and SQL Server SQL and T-SQL statements. For example, in a CREATE PROCEDURE statement, the procedure statement is counted as one and each T-SQL statement inside the body of the procedure is also each counted as one.

MTKS0109

Translation Error: A procedure call in a trigger body is not translated

Description

In T-SQL, the body of a trigger can be either a block or a procedure call statement. The translator does not support call statements for trigger bodies.

DB2 UDB trigger bodies do not support procedure calls; therefore, you cannot embed the procedure call in a block. A solution is to copy the procedure code into the trigger block, if possible.

MTKS0110

Translation Error: Arguments having a structured type are not supported: <structured-type-name>

Description

The converter does not support the translation of arguments having structured types (%ROWTYPE and record types). In this context, DB2 UDB does not support these kinds of data types.

One possible workaround is to explode this structured argument into several arguments having scalar data types. However, this would modify the signature of the routine, which means the calls to the function/procedure , including calls located in external applications, should be modified accordingly.

MTKS0111

Input Script Error: This is not a procedure: <object-name>

Description

In this procedure call, the references resolve to an object that is not a procedure.

MTKS0112

Translation Error: The following objects are involved in a circular dependency: <objects>

Description

The specified objects are involved in a circular dependency. Since the target server does not support forward declarations (through, for example, function specifications or cursor specifications), there is no way to declare these objects in the target server. The specified objects are translated but contain unresolved references.

MTKS0113

Input Script Error: This reference does not resolve to <a specific object type>: <reference>

Description

The given reference is made to an object that type is unexpected in this context. For instance, in a procedure call, the given name is in fact a function.

MTKS0114

Translation Warning: The CREATE SYNONYM statement is not translated. But any object referred to by the synonym is referred to directly.

Description

In DB2 UDB, it is possible to create an alias on a table, a view or another alias, but not on a function, procedure, sequence or schema (which is used to translate packages). Therefore, there is no direct translation for a CREATE SYNONYM on these objects.

To produce a correct translation, the converter does not translate the CREATE SYNONYM statement, but replaces all references to the synonym by a reference to the object it designates.

MTKS0115

Input Script Error: This is not a table or view: <object-name>

Description

In this context, the reference resolves to an object that is not a table or view.

MTKS0116

Translation Error: Oracle Rowid values are ignored.

Description

The ROWID type is translated to INTEGER in the target server.

For translations to DB2 UDB, the ROWID values that identify rows in the database are generated by DB2 UDB. Only these DB2 UDB-generated values can be stored in the DB2 UDB database.

MTKS0117

Translation Warning: DECIMAL type for arguments is translated with arbitrary precision (31) and scale (0)

Description

For arguments to functions, procedures or parameterized cursors, Oracle does not require you to specify precision, a scale, or length for the data type, and then Oracle automatically picks the best one depending on the actual arguments used in calls.

However, DB2 UDB requires that you specify parameters. Since the translator cannot determine the precision and scale to pick when translating to DB2 UDB, for the DECIMAL type, it picks arbitrary values: `31` for precision and `0` for scale. Type conversions (ROUND...) are also applied accordingly.

MTKS0118

Translation Error: Support for Rowid column not enabled.

Description

This reference to ROWID will not be valid in DB2 UDB unless the translation is performed with the ″rowid columns″ option enabled. This option adds an identity column called ROWID to each base table in DB2 UDB.

MTKS0120

Translation Error: The CREATE SEQUENCE statement contains a value that is out of range.

Description

The CREATE SEQUENCE statement contains a MINVALUE, MAXVALUE or START VALUE clause with a value that is outside the range that the converter can handle.

MTKS0121

Translation Error: The index matches a unique constraint of the table.

Description

DB2 UDB automatically generates an index for each unique constraint (or primary key) of the tables. Therefore, it will reject the creation of indexes matching these constraints. The translation of this CREATE INDEX statement was successful, but will probably cause an error in DB2 UDB.

MTKS0122

Translation Warning: The source type for Create Distinct Type is another Distinct type.

Description

The source type for Create Distinct Type is restricted to built-in types in DB2 UDB. The converter does not currently translate the source type to its base type.

MTKS0129

Translation Error: This statement is not supported in a DB2 UDB Procedure, Function, or Trigger.

Description

Some statements, especially DDL statements such as ALTER TABLE or CREATE objects other than tables, indexes, or views, are not allowed inside DB2 UDB procedures, functions, and triggers. In the context of a procedure, such statements can be executed using dynamic SQL.

MTKS0130

Translation Error: This expression could not be converted to a DB2 UDB SQLSTATE value.

Description

It has been determined that this expression is being used as a T-SQL error number because it is being compared to @@ERROR. An equivalent translation of this T-SQL error number to a DB2 UDB SQLSTATE value is not available.

MTKS0137

Translation Warning: Complex column default on a nullable column

Description

This complex column default has been translated inside a special trigger. The column is nullable, so the trigger cannot make the distinction between a null value and an unspecified value. It will therefore change the value using the default.

It is necessary to review the code to check that null values are not inserted. If this is not the case, you might use a special value (one that is not likely to be inserted) as the default for the column and then test the inserted value against this special value inside the trigger instead of using null.

MTKS0138

Translation Warning: Triggers generated for the TIMESTAMP column in this table.

Description

Triggers are generated in DB2 UDB to produce similar behavior as the TIMESTAMP column in T-SQL.

MTKS0139

Translation Information: NULL constraint was removed because of the complex DEFAULT.

Description

This column definition contains a complex default that was translated as a trigger. This special translation requires the removal of the NOT NULL constraint so NULL values can be used to represent an unspecified value. Therefore, a manual code review is necessary.

MTKS0143

Translation Warning: Routine overloading not allowed or not supported.

Description

Function or procedure overloading is either not allowed or not supported in this context. The object is translated but calls to this function or procedure will not resolve correctly.

MTKS0146

Translation Error: The GRANT statement is not translated in this case.

Description

In DB2 UDB, the user, authority, and privilege mechanisms differ considerably from those of T-SQL. Therefore, manual translation of this GRANT statement is necessary.

MTKS0147

Translation Error: The DROP DATABASE statement is not supported.

Description

Dropping databases within scripts or stored procedures is not supported. Databases can be removed using the DB2 UDB command processor or control center.

MTKS0148

Translation Error: The DROP ROLE statement is not supported.

Description

Explicit GRANT and REVOKE statements are used in DB2 UDB.

MTKS0149

Translation Error: The DROP RULE statement is not supported.

Description

Support in the translator for dynamic object deletion in scripts or stored procedures is limited to certain frequently-used idioms.

MTKS0150

Translation Error: The DROP DEFAULT statement is not supported.

Description

Support in the translator for dynamic object deletion in scripts or stored procedures is limited to certain frequently-used idioms.

MTKS0151

Translation Error: This SET statement is not supported.

DB2 UDB provides no equivalent functionality.

MTKS0152

Translation Error: The ON UPDATE CASCADE clause is not supported.

You can use a trigger to update the slave table.

MTKS0153

Translation Information: TRUNCATE is translated using DELETE inside procedures.

Description

The semantics of the TRUNCATE statement are to delete all rows in the table without keeping a log of the operation. This statement is typically used for very large tables. DB2 UDB does not have equivalent functionality inside a procedure body. Currently, TRUNCATE statements inside procedures are translated to DELETE statements, which a log tracks. If the size of the log file grows too large, a failure might occur.

MTKS0154

Translation Error: The translation of type TIMESTAMP is approximate.

Description

The T-SQL TIMESTAMP type is being translated to the target TIMESTAMP type. This is not an accurate translation, and manual translation might be necessary.

MTKS0155

Translation Error: The ROWGUIDCOL clause is not supported in DB2 UDB.

MTKS0156

Translation Error: The COLLATE clause is not supported in DB2 UDB.

MTKS0157

Translation Warning: There is a primary key already defined on table <tableName>

Description

There is a primary key already defined on this table, and the table cannot be altered to add another primary key. Sometimes the system procedure SP_PRIMARYKEY, in addition to the ALTER TABLE statement, is left in the source SQL for documentation purposes. If this is the case, and if you have specified the SP_KEYS option (an advanced option on the Convert page of the MTK interface), then the first statement encountered is converted to DB2 UDB and the second causes the message--which can be ignored.

If you do not want SP_PRIMARYKEY procedures to be converted, clear the option in the interface and reconvert.

MTKS0158

Translation Warning: The ALTER VIEW statement has been translated to DROP and CREATE VIEW statements.

Description

ALTER VIEW has been translated to DROP and CREATE VIEW statements. Errors will occur if there are dependencies on this view.

MTKS0159

Translation Error: Unexpected argument to system procedure call.

Description

The system procedure call has encountered an unexpected argument.

MTKS0160

Translation Warning: A column definition has been changed because of a SYSTEM PROCEDURE CALL.

MTKS0161

Translation Error: Translation requires a FETCH FIRST clause, which is not supported in sub-queries in DB2 UDBV7.

Description

Translation requires a FETCH FIRST clause, which is not supported in sub-queries in DB2 UDB Version 7.2.

MTKS0162

Translation Information: This function could only return 4 KB.

Description

This function could only return 4 KB. If you want to handle up to 8 KB, set the table size and use the 8 KB version of the UDF.

Related reference

"Compatibility library (MSSQL and SYB functions)" on page 302
The following functions simulate the behavior of certain Sybase and Microsoft SQL Server built-in functions that do not have equivalents in the target database server. If a function is used in the translated script, then it is bound to the database during its deployment to DB2 UDB or Informix Dynamic Server.

MTKS0163

Translation Error: Unable to translate sub-query assignment in joined update in this context.

Description

MTK is not able to translate a sub-query assignment in a joined update when one or more assignments refer to tables other than the one being updated.

MTKS0164

Translation Error: Unable to translate joined update or delete: Table <table name> occurs in an outer join or a parenthesized operation.

Description

MTK is not able to generate a correct translation of the joined update or delete because the table being modified is involved in an outer join or parenthesized operation. It cannot be removed from the from clause, which is required for a valid translation.

MTKS0165

Translation Error: Cannot translate cursor FETCH statements that do not fetch into a variable.

MTKS0166

Translation Error: Scrolling fetch is not supported.

Description

The following FETCH options require scrollable cursors, which are not yet supported in DB2 UDB:

- PRIOR
- FIRST
- LAST
- ABSOLUTE
- RELATIVE.

MTKS0167

Input Script Error: Syntax Error:

MTKS0168

Translation Error: Syntax Error: <character> Unexpected character ignored.

MTKS0169

Translation Error: Syntax Error: parsing terminated at this token.

MTKS0170

Translation Error: Syntax Error: <token> inserted before this token.

Parsing can continue if the designated token is inserted before the token indicated in the message.

MTKS0171

Translation Error: Syntax Error: <token> inserted after this token.

Parsing can continue if the designated token is inserted after the token indicated in the message.

MTKS0172

Translation Error: Syntax Error: unexpected input discarded.

For parsing to continue, the input tokens indicated in the message are discarded.

MTKS0173

Translation Error: Syntax Error: <token> expected instead of this token.

For parsing to continue this token is substituted for the input token indicated in the message.

MTKS0174

Translation Error: Syntax Error: <token> unexpected token(s) ignored.

For parsing to continue the indicated tokens are skipped.

MTKS0175

Translation Error: Syntax Error: <token> replaces one or more tokens by merging.

Description

This token is formed by merging the indicated error tokens into a single token.

MTKS0176

Translation Error: Syntax Error: <construct> misplaced construct(s)

Description

The indicated syntactic construct has been deleted so that parsing can continue.

MTKS0177

Translation Error: Syntax Error: <token(s)> inserted to complete scope.

Description

One or more tokens, such as END, have been inserted to close a syntactic construct, such as BEGIN ... END.

MTKS0178

Translation Error: Syntax Error: <tokens> unexpected input ignored.

Description

For parsing to continue, the indicated tokens have been discarded.

MTKS0179

Translation Error: Syntax Error: <token> reached after this token.

Description

End of file reached before the input is completely parsed.

MTKS0180

Translation Error: Syntax Error: <token> is invalid.

MTKS0181

Translation Error: Top-level IF statements are not translated.

Description

Translation of this construct is only supported in stored procedures or for certain top-level idioms that contain object-creation statements.

Related reference

"IF" on page 291
IF statements within procedures are converted and some top-level IF statements are converted.

MTKS0182

Translation Warning: The condition in the top-level IF statement has been ignored.

Description

DB2 UDB does not support control flow statements outside of stored procedures. However, certain top-level object creation idioms are partially supported through the translation of only the object creation statements.

Related reference

"IF" on page 291
IF statements within procedures are converted and some top-level IF statements are converted.

MTKS0183

Translation Error: The bit MOD, AND, OR, XOR, and bit negation operators are not supported.

The target server does not provide equivalent functionality.

MTKS0184

Translation Warning: @@ROWCOUNT is reset by the evaluation of IF or WHILE conditions. The translation might be incomplete.

Description

In T-SQL, the value of @@ROWCOUNT is reset by the evaluation of IF or WHILE conditions. The value of @@ROWCOUNT used at this point might have been reset by the evaluation of an IF or WHILE condition. Manual modification might be necessary.

Related reference

"@@ROWCOUNT" on page 296
The global variable @@ROWCOUNT of T-SQL indicates the number of rows that were affected by the last query. The converter provides support for some uses of @@ROWCOUNT.

MTKS0185

Translation Warning: The translation of @@ROWCOUNT is incomplete (cannot compute ROWCOUNT for this statement).

Description

The current statement sets the @@ROWCOUNT in a way that cannot be translated. The subsequent use of @@ROWCOUNT will not be handled correctly. Manual correction is required to generate the appropriate value of @@ROWCOUNT and to assign this value to the variable V_ROWCOUNT.

Related reference

"@@ROWCOUNT" on page 296
The global variable @@ROWCOUNT of T-SQL indicates the number of rows that were affected by the last query. The converter provides support for some uses of @@ROWCOUNT.

MTKS0186

Translation Error: @@ERROR is reset by the evaluation of IF or WHILE conditions. The translation might be incomplete.

Description

In T-SQL, the value of @@ERROR is reset by the evaluation of IF or WHILE conditions. The value of @@ERROR used at this point might have been reset by the evaluation of an IF or WHILE condition. Manual modification might be necessary.

MTKS0187

Translation Error: @@ERROR translation is incomplete.

Description

The value of @@ERROR used at this point might have been set in the procedure calling the current procedure. The translator is currently unable to handle such situations correctly. Manual modification might be necessary.

Related reference

"@@ERROR" on page 293
The converter attempts to preserve the T-SQL style error handling by generating a default exception handler in every procedure that catches all exceptions, and copies the error status value SQLCODE into a local variable. The exception handler is declared to be a CONTINUE exception handler, which indicates that the execution should continue on with the statement following the statement that produced the exception.

MTKS0188

Translation Warning: Translation of IN(sub-query) or =ANY(sub-query) might be incorrect.

Description

When null values are involved, the result of this comparison in the target server might not give the same results as in the source server.

MTKS0189

Translation Error: The translation of <global-variable> is incomplete. The value might have been set by a calling procedure.

Description

The value of the global variable used at this point might have been set by the procedure calling the current procedure. The translator is currently unable to handle such situations. Manual verification and possible modification of the DB2 UDB script is necessary.

MTKS0190

Translation Error: The use of <global-variable> inside triggers or functions is not translated.

Description

The use of global variables @@ERROR, @@SQLSTATUS and @@FETCH_STATUS inside triggers or functions is not supported.

MTKS0191

Translation Warning: Column <column name> is not in the GROUP BY clause, MTK added missing select/order/having columns to the GROUP BY clause. Since conversion of this SYBASE query to DB2 might yield different results, it is highly recommended that an analyst review this query.

Description

This sub-query contains a GROUP BY clause and the SELECT clause, ORDER BY clause, or HAVING clause of this sub-query contains a reference to a column that is not in the GROUP BY clause and is not inside a call to an aggregate function. This is a T-SQL-extension that is not supported by MTK or the target server.

MTKS0192

Translation Information: The global variable <global variable> is not supported.

MTKS0193

Translation Error: Unable to translate the global variable <global variable> in this context.

Description

In some cases MTK is unable to translate the global variable correctly. Some contexts where this is not possible include at the top level and after a procedure call. The global-variable @@ERROR is translated only when it is used in a equality comparison.

MTKS0197

Translation Error: SET ROWCOUNT effect over procedure calls is not implemented.

Description

A subsequent procedure call statement is under the influence of this SET ROWCOUNT statement, but MTK does not take this into account when translating the body of the called procedure. Manual translation might be required.

MTKS0198

Translation Error: Triggers cannot return a value.

Related reference

"Triggers" on page 299
The converter provides limited support for triggers.

MTKS0199

Translation Error: A RAISERROR inside a trigger is translated only if accompanied by a ROLLBACK and RETURN statement.

Description

The translator attempts to translate uses of RAISERROR to a corresponding use of the DB2 UDB construct SIGNAL SQLSTATE. Inside triggers, SIGNAL SQLSTATE has the effect of aborting the execution of the trigger and undoing the changes made by the trigger as well as the event that initiated it. Consequently, the translation is appropriate only for occurrences of RAISERROR that are accompanied by ROLLBACK and RETURN statements.

Related reference

"Triggers" on page 299
The converter provides limited support for triggers.

MTKS0200

Translation Error: Signalling an exception using RAISERROR and returning a value simultaneously is not possible. The RAISERROR statement is not translated.

Description

The RAISERROR statement is equivalent to a PRINT statement followed by an assignment to the system variable @@ERROR. The translator simulates this behavior using the DB2 UDB SIGNAL statement. However, the @@ERROR value is available in the calling procedure only if a RAISERROR statement is followed by a simple return from the procedure. A RETURN statement that returns a value has the effect of resetting the value of @@ERROR. Consequently, the translator ignores RAISERROR statements that are followed by a RETURN statement that returns a value.

Fix the generated code, if necessary, depending on the needs of the calling procedure.

Related reference

"RAISERROR" on page 292
A RAISERROR statement is equivalent to a PRINT statement (to display a message to the user) followed by an assignment to @@ERROR.

MTKS0201

Translation Error: The RAISERROR statement is not followed by a RETURN statement. The RAISERROR statement is not translated.

Description

The RAISERROR statement is equivalent to a PRINT statement followed by an assignment to the system variable @@ERROR. The translator simulates this behavior using the DB2 UDB SIGNAL statement. However, the @@ERROR value is available in the calling procedure only if the RAISERROR statement is followed by a simple RETURN from the procedure. Consequently, the translator ignores RAISERROR statements that are not followed by a simple RETURN.

Fix the generated code as necessary.

MTKS0202

Translation Error: The error number in the RAISERROR statement cannot be used as a valid SQLSTATE value and must be changed to a suitable SQLSTATE value.

Description

The error number in the RAISERROR statement does not translate to a valid DB2 UDB SQLSTATE value. You must manually change the SQLSTATE value in the DB2 UDB script.

SQLSTATE values are required to be strings of length 5, consisting only of digits and upper case letters, where the first two characters are referred to as the class and the last three characters are referred to as the subclass. The value specified in a SIGNAL SQLSTATE statement in a PSM procedure must have a non-zero class. And, the value specified in a SIGNAL SQLSTATE statement in a trigger must conform to the following restrictions:

- The class cannot be 00, 01, or 02.
- If the class starts with a character between 0 and 6 or A through H, then the subclass must start with a character in the range I through Z.

MTKS0203

Translation Error: The RAISERROR error number is stored in a local variable. The DB2 UDB SIGNAL statement allows only constants.

Description

This RAISERROR statement obtains the error number through a local integer variable.

In DB2 UDB, the error number, passed as an argument to the SIGNAL statement, must be a string constant (a literal constant or a symbolic constant declared as a condition).

MTKS0204

Translation Information: This statement is taken into consideration while translating the RAISERROR statement occuring after it.

Description

The translation of the RAISERROR statement occurring after the current statement, takes the current statement into consideration. Hence, no further translation of this statement is necessary.

MTKS0205

Translation Information: This statement is taken into consideration while translating the RAISERROR statement occuring prior to it.

Description

The translation of the RAISERROR statement occurring prior to the current statement takes the current statement into consideration. Hence, no further translation of this statement is necessary.

MTKS0206

Translation Error: RAISERROR format arguments are not translated.

Description

The translator currently ignores the extra parameters of the formatting string in RAISERROR statements.

MTKS0207

Translation Error: A rollback to a savepoint or a named BEGIN TRANSACTION statement is not supported.

Description

A rollback to a save point or a named BEGIN TRANSACTION statement is not currently supported by the translator.

MTKS0208

Translation Error: The translator does not support conversion from other types to STRING types, for the specified style of date format.

Description

The translator currently does not handle conversion from T-SQL money data types or datetime data types to string types for this style of date format.

MTKS0209

Translation Error: Translation requires a FETCH FIRST clause, which is not supported in fullselects in DB2 UDBV7.

Description

Translation requires a FETCH FIRST clause, which is not supported in full selects in DB2 UDB Version 7.2.

MTKS0210

Translation Warning: The @@IDENTITY translation might return incorrect values in some cases.

Description

If the most recent INSERT or SELECT INTO statement before the reference to the @@IDENTITY performed an insert into a table without an identity column, then Sybase returns a `0`, SQL Server returns `NULL`, and the translation in DB2 UDB returns the most recent identity value. Manually inspect the code to ensure the desired behavior will occur.

MTKS0211

Translation Error: Column <column name> occurs with an aggregate function in a select or order by clause without a group by clause.

Description

This sub-query contains an aggregate function without a GROUP BY clause and the SELECT or ORDER BY clause of this sub-query contains a reference to a column that is not inside a call to an aggregate function. This is a T-SQL-extension that is not supported by MTK or the target server.

MTKS0212

Translation Error: This view cannot be updated in DB2 UDB. The check option is omitted.

Description

The check option is not allowed in DB2 UDB in the definition of read-only views.

MTKS0213

Translation Error: This view has been declared as read-only.

MTKS0214

Translation Error: The base view or table-expression used to define this view is read-only.

MTKS0215

Translation Error: This view is read-only because it selects only distinct elements.

MTKS0216

Translation Error: This view might not be updatable because it selects complex non-column values.

MTKS0217

Translation Error: This view is read-only because it has no FROM clause.

MTKS0218

Translation Error: This view is read-only in DB2 UDB because it has multiple base tables.

Description

This view has multiple base tables. Deletes, updates, or inserts to such a view are not standard and are similar to joined deletes and joined updates.

MTKS0219

Translation Error: This view is read-only because it has a GROUP BY clause.

MTKS0220

Translation Error: This view is read-only because it has a HAVING clause.

MTKS0221

Translation Error: This view might be read-only because of the use of UNION.

MTKS0222

Translation Error: The assignment of a top-level procedure call return value to a variable is not translated.

MTKS0223

Translation Information: The index has been translated to a constraint on a table to enable foreign key specifications.

Description

DB2 UDB requires for any foreign key specification that the foreign table have a corresponding UNIQUE or PRIMARY KEY constraint on the corresponding columns, while T-SQL requires only a unique index. This index should be declared as a uniqueness constraint on the table to enable a different foreign key specification.

Related reference

"CREATE TABLE" on page 283
Using the data type mapping information, a straightforward translation is usually possible.

MTKS0224

Translation Error: In DB2 UDB, cursors cannot be used as output parameters.

MTKS0225

Translation Warning: Unable to validate the syntax of dynamic SQL in EXECUTE <string>

Description

The dynamic string argument to the EXECUTE <string> statement cannot be translated, because it could contain elements that are not known until runtime. Inspect the string and translate manually if necessary.

MTKS0226

Translation Error: Cannot return a FULLSELECT in this context.

Description

The following are the restrictions with using FULLSELECT in a RETURN statement:

- If the FULLSELECT is returned from scalar function, then the FULLSELECT must return one column, and at most, one row.
- A FULLSELECT cannot be specified for a RETURN from a procedure.
- If the FULLSELECT is returned from a row function, it must return, at most, one row.

MTKS0227

Translation Error: No column-list specified in the RETURN clause of the CREATE FUNCTION statement.

Description

The translator is unable to determine the columns returned by the CREATE TABLE FUNCTION.

MTKS0228

Translation Error: The CLUSTER clause in the Unique/Primary-Key constraint has been ignored.

Description

Correct translation of the unique or primary key constraint that is CLUSTERED involves defining a CLUSTERED INDEX on the key columns. But an index has already been defined on these columns either explicitly through a CREATE INDEX statement, or implicitly by defining a unique constraint on these columns.

MTKS0229

Translation Error: The Insert Exec-String statement is not translated.

MTKS0230

Translation Error: Verify that the number of result sets is correct.

Description

The translator is unable to determine the exact number of result sets returned by the EXEC statement. The translator always assumes at least one.

MTKS0231

Translation Error: Top-level INSERT-EXEC statements are not translated.

Description

Translation of this construct is only supported in stored procedures.

MTKS0232

Translation Warning: The temporary table <temp table name> is considered to have a local scope.

Description

The global scoping option cannot be applied to the temporary table <temp table name> because it was declared using the SELECT ... INTO syntax.

If global scoping is necessary, a workaround is to manually edit the temporary table declaration, splitting it into a CREATE TABLE statement followed by an INSERT INTO... SELECT statement. Then retranslate.

For example, replace:

```
SELECT * INTO #TT FROM T
```

with:

```
CREATE TABLE #TT (I INT)<br>INSERT INTO #TT SELECT * FROM T
```

MTKS0233

Translation Error: Need to change the WITH RETURN option in the cursor declarations for procedure <proc-name> from RETURN TO CLIENT to RETURN TO CALLER.

Description

The translator always declares the DB2 UDB cursor to return the result set to the client. The reason for doing this is that the translator does not know all the contexts in which the procedure is being called (for example, from applications or other stored-procedures that are not yet migrated).

Assuming that result sets are mostly used by client applications, the RETURN TO CLIENT option is used. Decide if this assumption is appropriate in the context of your system.

Related reference

"INSERT INTO *table* EXECUTE *procedure*" on page 290
This insert-execute statement is translated using a cursor and a locator variable for each of the result sets generated by the procedure, which iterates over the cursor using a FETCH and INSERT statement to insert each row from the result sets into the table.

MTKS0234

Translation Error: Top-level COMMIT statements are not translated.

MTKS0235

Translation Error: Top-level BEGIN TRANSACTION statements are not translated.

MTKS0236

Translation Error: Top-level ROLLBACK statements are not translated.

MTKS0237

Translation Warning: The BEGIN TRANSACTION statement is ignored. DB2 begins transactions implicitly.

MTKS0238

Translation Error: The event specified in the WAITFOR statement is not supported.

Description

Only the following constructs of a WAITFOR statement are supported:
- WAITFOR TIME *time*
- WAITFOR DELAY *time*

Other events in similar context, such as ERROREXIT, PROCESSEXIT, or MIRROREXIT, are not supported because the target server does not provide equivalent functionality. Evaluate the use of these statements and translate manually.

MTKS0239

Translation Error: The value of <trancount-variable> might be incorrect inside trigger/function bodies.

Description

Transaction statements (such as ROLLBACK and COMMIT) are not supported inside a DB2 dynamic compound statement. Therefore, transaction statements in the source are not translated to DB2 UDB and the value of v_trancount is updated in the DB2 UDB translation.

Related reference

"Transactions" on page 297
In T-SQL, updates and changes are immediately committed unless a transaction is explicitly initiated with a BEGIN TRANSACTION statement. By contrast, a transaction is always in effect for DB2 UDB and Informix Dynamic Server when in ANSI mode. Changes are committed only when an explicit COMMIT statement is executed (or when the application terminates).

MTKS0240

Translation Error: @@error is translated using DB2 SQLCODE in this context (outside of an equality comparison).

Description

References to @@error, when outside of an equality comparison, are directly translated to references to SQLCODE in DB2 UDB .

Most of the time the error code values of T-SQL do not match DB2 UDB SQLCODE values, so it might be necessary to perform some modification to the code. For example, if the SQLCODE is compared to a literal value, you must change this value to the corresponding SQLCODE.

When @@error occurs inside an equality comparison, it is translated using SQLSTATE and the compared value is translated to the corresponding DB2 UDB SQLSTATE value.

MTKS0241

Translation Error: This specific DEFAULT constraint clause is not supported.

Description

The constraint clauses DEFAULT GLOBAL AUTOINCREMENT and DEFAULT CURRENT DATABASE are not supported for the SYBASE SQL Anywhere source. MTK ignores these constraints and generates the rest of the SQL statement.

Related reference

"CREATE TABLE" on page 283
Using the data type mapping information, a straightforward translation is usually possible.

MTKS0242

Translation Error: The DECLARE LOCAL TEMPORARY TABLE statement is not supported.

Description

The DECLARE LOCAL TEMPORARY TABLE statement is not supported for any target, so MTK will ignore this statement.

Related reference

"CREATE TABLE" on page 283
Using the data type mapping information, a straightforward translation is usually possible.

MTKS0243

Translation Error: This specific COMMENT ON statement is not supported.

Description

The COMMENT ON statement, when on a FOREIGN KEY or if character string is NULL, is not supported.

Related reference

"COMMENT ON" on page 292
The statement is supported in most cases for translations to DB2 UDB. It is not supported for translations to Informix Dynamic Server.

MTKS0244

Translation Warning: Label <label name> associated with the removed nested compound statement has been moved to the first contained statement.

Description

The associated label along with the contents of the compound statement have been moved to first contained statement. You must review all references to the label and manually translate them as necessary.

MTKS0245

Translation Warning: The translator has removed the nested DB2 UDB compound statement by moving the contents to a higher block.

Description

The translator has removed the nested DB2 UDB compound statement. The contents of the compound statement have been moved to a higher block.

MTKS0246

Translation Warning: Nested compound statements are not supported inside an exception handler. Manual translation of the statement is required.

Description

This version of DB2 UDB does not support nested compound statements. Consider replacing the compound statement with a stored procedure call.

MTKS0247

Translation Warning: Nested compound statements containing an exception handler are not supported. Manual translation of the statement is required.

Description

This version of DB2 UDB does not support nested compound statements. Consider replacing the compound statement with a stored procedure call.

MTKS0248

Translation Warning: The RENAME COLUMN is not supported in the DB2 target

Description

The ALTER TABLE RENAME column and the RENAME column are not supported in the DB2 target.

MTKS0249

Translation Warning: Constraints are not supported for temporary tables.

Description

Informix Dynamic Server does not support constraints in a temporary table.

MTKS0250

Translation Warning: Only one column in Informix Dynamic Server can be of type serial.

Description

Informix Dynamic Server does not support more than one column with a serial data type.

MTKS0255

Translation Error: IDS supports only one element in the expressioon list of the IN predicate.

Description

Informix Dynamic Server supports only one element in the expression list of the IN predicate.

MTKS0256

Translation Error: IDS does not support the use of lists with the ALL operator.

Description

Informix Dynamic Server does not support the use of lists with the ALL operator.

MTKS0261

Translation Error: The target does not allow this constraint on columns of type <typeName>

Description

The target server does not allow this constraint on columns of type {0}.

MTKS0262

Translation Error: The target does not allow indexes on columns of type <typeName>

Description

The target server does not allow indexes on columns of type {0}.

MTKS0265

Translation Information: To enable foreign-key specifications, MTK added a unique constraint in the reference of a unique index on a table.

Description

DB2 Universal Database requires that all foreign key specifications in a foreign table have a corresponding UNIQUE or PRIMARY KEY constraint on the corresponding columns. T-SQL requires only a unique index.

This index should be declared as a uniqueness constraint on the table to enable a different foreign key specification.

MTKS0266

Translation Warning: Assignment of default value moved to code block.

Description

Informix Dynamic Server does not support default values for local variables. The assignment of the default value was moved to the code block.

MTKS0267

Translation Warning: SPL variables that are not default or constant variables are set to NULL.

Description

Informix Dynamic Server does not initialize SPL variables. An SPL variable was set to NULL in the code block.

MTKS0268

Translation Warning: The Sybase CLUSTERED clause is not supported by DB2 UDB on a temporary table.

Description

In translations from Sybase to DB2 Universal Database™, MTK ignores the CLUSTERED clause and instead uses the CREATE INDEX statement.

MTKS0269

Translation Error: Informix Dynamic Server does not support the GOTO statement.

Description

This version of Informix Dynamic Server does not support the GOTO statement in stored procedure language.

MTKS0270

Translation Error: Informix Dynamic Server does not support labels.

Description

This version of Informix Dynamic Server does not support labels in stored procedure language.

MTKS0271

Translation Warning: Schema name in a Create Distinct Type statement cannot be more than 8 bytes.

Description

DB2 does not support more than 8 bytes for a schema name in a Create Distinct Type statement.

Action

To get around this message:

1. Place all sp_add_type statements in a separate SQL source file.
2. Translate this file then refine the translation to a schema of 8 bytes or less.

Translate any additional ASE SQL files in the same manner.

MTKS0282

Translator Limitation: Variable reference {0} is not supported in this context.

Description

The variable reference is not supported in the data definition language statements contained in the generated target statements.

MySQL converter messages

The MySQL converter might return the following messages.

MTKM0000

Translation Information: MTK MySQL Converter. Version: <mtk version>

Description

Specifies the version of the Oracle converter.

MTKM0001

Translation Error: This statement is not translated.

Description

This statement is not translated.

MTKM0002

Input Script Error: Unexpected Syntax - not translated.

Description

The translator encountered some text it could not understand.

MTKM0003

Fatal Internal Error: Error walking the AST.

Description

The translator encountered some text it could not understand.

MTKM0004

Input Script Error: Unrecognized Character - skipped.

Description

The translator does not understand this character.

MTKM0005

Translation Error: This feature is not translated.

Description

A feature of the data source code was encountered and recognized as a feature not translated in this version.

MTKM0006

Translation Error: Untranslated Expression: <cause>

Description

The expression is not translated for the given reason.

MTKM0010

Translation Error: Unable to open file ″<filename>″ for output

Description

Unable to open the specified file for output. Ensure the file exists in the specified directory and is not locked or open by another application.

MTKM0011

Input Script Error: Reference to unknown <object>: <object name>

Description

The translator is not aware of the definition of this object.

Ensure the definition exists in the data source file. Also, ensure the schema name is specified as indicated.

MTKM0012

Input Script Error: Referenced table does not have a primary key.

Description

The table being referenced in this foreign key constraint does not have a primary key. Assign a primary key in the original source and re-convert.

MTKM0013

Input Script Error: No matching unique or primary key for this column list.

Description

The list of columns in this foreign key constraint has no matching unique or primary key constraint in the table it is referencing. Assign a primary key in the original source and re-convert.

MTKM0014

Input Script Error: Unknown command, rest of line ignored.

Description

The translator does not recognize this command, so the entire line is skipped.

MTKM0015

Input Script Error: Unknown statement, ignored.

Description

The translator does not recognize this statement, so the entire statement is skipped.

MTKM0016

Input Script Error: Duplicate definition of <object name>

Description

This object has already been defined. The previous definition will be used by the translator.

MTKM0017

Translation Error: This ALTER TABLE clause is not supported.

Description

Only the add constraint clause is translated in the ALTER TABLE statement. All other ALTER TABLE clauses are ignored.

Related concepts

"Statements" on page 153
MTK provides limited support for translating MySQL statements CREATE TABLE, CREATE INDEX, and INSERT.

MTKM0018

Input Script Error: Possibly ambiguous column reference: <column name>

Description

This column occurs in more than one table in the from clause (or more than once as a column heading in the select list for an order by clause). The translation might cause an error in the target server.

Related concepts

"Statements" on page 153
MTK provides limited support for translating MySQL statements CREATE TABLE, CREATE INDEX, and INSERT.

MTKM0019

Input Script Error: Call to unknown <subprogram type>: <subprogram type>

Description

The translator could not resolve the name in this function/procedure call. The name was not recognized as an Oracle built-in function name or the name of a previously defined user-defined function (UDF) or procedure.

Related concepts

"Statements" on page 153
MTK provides limited support for translating MySQL statements CREATE TABLE, CREATE INDEX, and INSERT.

MTKM0020

Translation Warning: Object name has been changed to <new name>.

Description

Object names that are too long for the target server are truncated. Names that are target server reserved words are enclosed in double quotes. Names that conflict with other names in the target server (because the name is already in use) are renamed.

MTKM0021

Translation Error: Call to <subprogram type> <subprogram name> is not supported.

Description

This Oracle function or procedure call is not translated to the target server.

Related concepts

"Statements" on page 153
MTK provides limited support for translating MySQL statements CREATE TABLE, CREATE INDEX, and INSERT.

MTKM0022

Input Script Error: No matching instance of <subprogram type> <subprogram name> with such arguments.

Description

The translator expects a different number of arguments when calling this function or procedure, or the argument types do not match any of the instances of this function or procedure.

Related concepts

"Statements" on page 153
MTK provides limited support for translating MySQL statements CREATE TABLE, CREATE INDEX, and INSERT.

MTKM0023

Fatal Internal Error: Translator runtime error.

Description

The translator experienced an unexpected internal error. Report this error to IBM Migration Toolkit support.

MTKM0024

Translation Warning: NOT NULL constraint is added because of a PRIMARY KEY or UNIQUE constraint.

Description

In the target server, a column with a UNIQUE or PRIMARY KEY constraint must also have a NOT NULL constraint.

MTKM0025

Translation Error: Insert/Delete/Update on subquery or table collection expression not translated.

Description

This insert, delete, or update operates over a subquery or a table collection expression. This operation is not supported in the target database and is not translated.

MTKM0026

Translation Error: RESCINDED. Create Synonym for object other than table, view or synonym not translated.

Description

In the target server, synonyms are supported for tables, views and synonyms only. Synonyms for other objects are not translated.

MTKM0027

Translation Error: RESCINDED. Translation of default expression results in a non-constant expression.

Description

In the target server, default expressions can only be constants, datetime special registers, USER, NULL, or certain forms of cast expressions.

MTKM0028

Translation Error: Incompatible Nulls First or Nulls Last clause: <conflicting clauses>

Description

The target server does not support NULLS LAST in a DESC order or NULLS FIRST in an ASC (the default) order.

MTKM0029

Translation Error: Subquery in From clause: not yet translated.

Description

SELECT statements that have subqueries in the FROM clause are not yet translated.

MTKM0030

Input Script Error: Invalid date format

Description

The date format is invalid or the input data (a date value) does not match the date Format.

MTKM0031

Translation Error: VALUE set clause of update statement not translated.

Description

The target server does not have an equivalent set clause in the update statement.

MTKM0032

Translation Warning: Order by clause in INSERT, SELECT INTO, VIEW or derived table subquery not translated.

Description

The target server does not allow the order by clause in a subquery to insert values in an INSERT statement or to define a VIEW.

MTKM0033

Input Script Error: Inserted data error

Description

The number of inserted values does not match the number of columns specified for the table.

MTKM0034

Translation Warning: No translation available, but the statement has been taken into account

Description

There is no translation available, but the information in the statement will be used by the converter in translating the statements that follow.

MTKM0035

Input Script Error: All columns of the subquery of a view must be named.

Description

The columns of the view must be named by either giving an explicit list of names in the definition, or by using column aliases in the subquery. In this case, there is no list of names and at least one subquery column has no name.

MTKM0036

Input Script Error: Number of columns of subquery must match names given in view definition.

Description

In a view, if a list of column names is given in the definition, this must match the number of columns in the result set of the subquery.

MTKM0037

Translation Error: Invalid or unsupported outer-join query

Description

The following restrictions apply to the translation of outer-join queries:
- Cyclic outer-joins in the WHERE clause are not translated (invalid input).
- The (+) operator must not follow a complex expression; it can only follow a column reference.
- Only the equality (=) operator is supported.

MTKM0038

Input Script Error: Type mismatch: expression has unexpected data type

Description

The expression has an inappropriate data type in this context.

MTKM0039

Input Script Error: Unexpected size of expression-list or result-set

Description

The size of this expression-list or result-set (generated by a subquery) does not match the size expected by the context.

This error is likely to happen:
- In INSERT statements if the VALUES clause or the result-set generated by the subquery does not match the table size.
- In comparisons, if the sizes of the elements on each side of the comparison operator do not match.

MTKM0040

Translation Error: Conversion from type <SourceType> to type <TargetType> is not supported by MTK

Description

An implicit conversion from a value of the source type to the target type in Oracle is not directly supported in the target server.

MTKM0041

Translation Warning: Unable to infer the target server data type for the expression. Using Varchar(1).

Description

A data type is needed in this context for use in a declaration or cast expression.

MTKM0042

Input Script Error: Unrecognized data type: <data type>

Description

This data type is not recognized by the translator. The reason might be that:

- It is not a valid Oracle data type.
- It is a user-defined type, not yet handled by the translator (for example, collections and object types).
-

If the data type is encountered as a parameter to a procedure, a suggested translation is provided, but is commented out because it probably requires manual editing.

MTKM0043

Input Script Error: Length required for type <data type>

Description

Length must be specified for this data type. For instance, when declaring a column of type VARCHAR, VARCHAR2 or RAW, length must be explicitly defined, as in: VARCHAR(10), RAW(100)

MTKM0044

Translation Warning: Warning: Negative scale and scale greater then the precision are not supported in the target server.

Description

The target server DECIMAL data type does not support negative scales or scales greater than the precision. The translator adjusts the precision and the scale in order to avoid loss of data but the target server results might differ.

For example:

```
CREATE TABLE T(X NUMBER(4,-2))
   ----- is translated
to -----
CREATE TABLE T(X DECIMAL(6,0))!
```

But the result of an INSERT VALUES(123456) will be 123400 in Oracle and 123456 in DB2 UDB, if DB2 UDB is the target server.

MTKM0045

Translation Error: Untranslated data type: <data type>

Description

The Oracle data type has no target server equivalent and could not be translated. This can happen when the translation of a NUMBER(p,s) is not able to capture any digit of the source data type (when s-p>31 or s<-31).

MTKM0046

Translation Warning: Non blank-padded comparison cannot be enforced here

Description

This happens when the non-blank-padded comparisons enforcement option is set, in membership comparisons involving expression-lists where CHAR and VARCHAR are being compared.

For example, if a and b are CHAR, c is VARCHAR and x*n* is any other data, `(a, x1) IN ((b, x2),(c,x3))` cannot be translated accurately, because a versus b should be a non-blank-padded comparison whereas a versus c is blank-padded. The only way to enforce blank-padding rules here would be to explode this membership comparison into several comparisons, which the translator does not handle. In the above example, this would be converted to `(a = b AND x1 = x2) OR (ORA8.NO_PAD(a) = ORA8.NO_PAD(c) AND x1 = x3)`.

MTKM0047

Translation Warning: RESCINDED. BEFORE translated to NO CASCADE BEFORE

Description

RESCINDED. Oracle triggers using the BEFORE event type are translated in the target server to NO CASCADE BEFORE triggers. This type of trigger does not allow other triggers to fire from the trigger body, a situation that might lead to incorrect behavior.

MTKM0048

Translation Error: DDL and Database event triggers are not translated

Description

Triggers using Data Definition Language (DDL) events and database events are not translated.

MTKM0049

Translation Error: INSTEAD OF triggers are not supported in this version of the target server.

Description

DB2 UDB Version 7.2 does not support INSTEAD OF triggers. The version 8 translation is given here.

MTKM0050

Translation Warning: This statement is not supported in the target server dynamic compound statement

Description

This statement is not supported in the target server dynamic compound statement. Dynamic compound statements are used as bodies for top-level anonymous blocks, user-defined functions, and triggers. Procedures use the target server compound statements as bodies that are less restrictive. Some statements that are not allowed inside the target server dynamic compound statements are:

- Nested blocks
- Statements containing CASE expressions
- GOTO
- Procedure calls
- Cursors
- COMMIT
- Exception handlers

If the compound statement is found in an Oracle function, depending on the use of the function, changing it to a procedure might be a better approach.

Related concepts

"Statements" on page 153
MTK provides limited support for translating MySQL statements CREATE TABLE, CREATE INDEX, and INSERT.

MTKM0051

Translation Error: BEFORE triggers without FOR EACH ROW are not supported

Description

In the target server, FOR EACH STATEMENT must not be specified for BEFORE triggers. For this reason, BEFORE triggers that do not specify FOR EACH ROW cannot be translated.

MTKM0052

Input Script Error: Table name <table> not allowed in this context.

Description

In this context an expression is expected. A table name that does not qualify a column name or a star (*) is not allowed.

MTKM0053

Input Script Error: No column <column name> in table.

Description

The given table has been found in a surrounding from clause, but it does not contain the indicated column.

MTKM0054

Input Script Error: Parameters in named notation cannot be followed by parameters in positional notation.

Description

The actual parameters in this procedure/function call use bad mixed notation. Mixed parameter notation is only allowed if named parameters follow positional parameters.

MTKM0056

Input Script Error: An INTO clause is expected in this select statement

Description

In a PL/SQL context, a select statement must have an INTO clause.

MTKM0057

Translation Error: Unable to translate subquery with set operations to the target server Select Into.

Description

In PL/SQL a SELECT INTO might be united with other selects. This is not allowed in the target server.

MTKM0058

Translation Error: This ALTER SESSION clause is not supported.

Description

The following ALTER SESSION clauses are currently simulated by the tool (but not translated). The remaining clauses are ignored.

- SET NLS_DATE_FORMAT
- SET CURRENT_SCHEMA
- SET NLS_CURRENCY
- SET NLS_ISO_CURRENCY
- SET NLS_NUMERIC_CHARACTERS

MTKM0059

Translation Warning: Ignored input - not translated.

Description

This input is ignored. It is not supported in the target server. This omission should not cause the target server code to produce different results from the corresponding Oracle code.

MTKM0060

Translation Error: References to remote database objects are not translated

Description

References to objects in remote databases (indicated by ″@dblink″) are not supported.

MTKM0061

Translation Warning: Parameter defaults are not supported in the target server procedure definitions. Calls to the procedure are adjusted accordingly.

Description

In procedure and function declarations, the optional DEFAULT value of a parameter is not translated, but the translator will use the value as necessary through the remainder of the translation.

Related concepts

“Statements” on page 153
MTK provides limited support for translating MySQL statements CREATE TABLE, CREATE INDEX, and INSERT.

MTKM0062

Translation Warning: Labels are not allowed in the target server dynamic compound statements.

Description

Labels are not allowed in the target server dynamic compound statements, except for loops. The translator will not translate the label for this statement.

Related concepts

“Statements” on page 153
MTK provides limited support for translating MySQL statements CREATE TABLE, CREATE INDEX, and INSERT.

MTKM0063

Translation Warning: Labels are not supported for top-level blocks in the target server.

Description

Labels are not supported for top-level blocks in the target server. The translator omits these labels.

MTKM0064

Input Script Error: PUBLIC synonym cannot have schema qualifier.

Description

In the definition of a public synonym, the name of the synonym is not allowed to be qualified by a schema name.

MTKM0065

Translation Warning: This save point may not be allowed by the target server.

Description

If another save point is active when this save point is encountered, the target server will reject the save point. The target server allows only one active save point at a time. The converter is unable to determine at compile time the number of save points that can be active when this statement is encountered.

MTKM0066

Translation Error: Autonomous transactions are not supported by the target server.

Description

The target server does not support autonomous transactions. The transaction statements COMMIT, ROLLBACK, and SAVEPOINT in this block and the enclosing dynamic context can result in a different behavior in the target server.

MTKM0067

Translation Warning: NOWAIT is not supported by the target server Lock Table Statement.

Description

NOWAIT is not supported by the target server Lock Table Statement.

MTKM0068

Translation Error: This lock mode is not supported by the target server Lock Table Statement.

Description

The target server Lock Table Statement supports SHARE and EXCLUSIVE lock modes only.

MTKM0069

Translation Warning: Locks for partitions, subpartitions, and remote tables are not supported by the target server.

Description

These clauses are not translated to the target server.

MTKM0070

Translation Error: OUT and IN OUT parameters are not supported in the target server user-defined functions

Description

In user-defined functions, the target server only allows input parameters. The translator cannot properly translate a function with OUT or IN OUT parameters to an equivalent construct in the target server. Evaluate the code to determine if you can use a procedure instead.

MTKM0071

Translation Warning: Reference to OLD or NEW column translated to <NULL or variable>

Description

The target server does not accept references to OLD from an inserting trigger or references to NEW from a deleting trigger. In the WHEN clause of the trigger, these references are translated to NULL. In the body of the trigger, they are translated to a variable generated for this purpose.

MTKM0072

Translation Error: The target server only allows constants as arguments to a top-level procedure call.

Description

The target server only allows constants as arguments to a top-level procedure call.

MTKM0073

Translation Error: This statement is not supported in the target server UDF.

Description

This statement is not supported in the target server UDF. DML statements, cursors, and exception blocks are examples of items that cannot be in target server functions. Depending on the use of the Oracle function, changing the function to a procedure might be a better approach.

Related concepts

“Statements” on page 153
MTK provides limited support for translating MySQL statements CREATE TABLE, CREATE INDEX, and INSERT.

MTKM0074

Translation Error: Nested exception handlers are not supported.

Description

Nested exception handlers are not supported because the target server does not allow the declaration of a handler inside another handler.

MTKM0075

Translation Error: This statement is not supported in the target server Before Trigger

Description

This statement is not supported in the target server Before Trigger. The following statements are not supported in this context : INSERT, DELETE and UPDATE.

MTKM0076

Translation Error: This statement is not supported in a target server After Trigger

Description

This statement is not supported in a target server After Trigger. The following statement is not allowed in this context: a SET transition-variable statement (when FOR EACH ROW is specified).

MTKM0077

Translation Error: This form of the SQL Plus Execute command is not supported.

Description

Only procedure calls and begin-end blocks are supported in the SQL Plus Execute command. Other PL/SQL statements in an Execute command are not translated.

MTKM0078

Translation Warning: Assignment of non-constant default expression moved to block.

Description

The target server does not allow a non-constant expression to be the default value of a variable declaration. A statement assigning the expression to the variable has been moved to the beginning of the block.

MTKM0079

Translation Warning: An arbitrary size of 255KB was picked for the LOB type.

Description

In PL/SQL context, certain Oracle data types are translated to LOB(255K) or CLOB(255K). The converter picks 255KB as an arbitrary size for those Large Object types. However, depending on your needs, the size can be modified (increased or decreased), allowing better performance.

Considering that the target server allocates memory for variables and parameters of these types, try to limit the size of these data types as much as possible.

MTKM0080

Translation Error: This package item is not translated.

Description

Only the following items are supported inside a package: function specifications, functions, procedure specifications, procedures, and constants. Variable declarations, cursor declarations and type definitions are not translated.

MTKM0081

Translation Warning: The generated SQLSTATE might be incorrect.

Description

When translating calls to raise_application_error, the translator uses the error code to generate an SQLSTATE. Consult the target server message reference for rules on specifying the necessary SQLSTATE.

MTKM0082

Translation Error: Untranslated reference.

Description

This reference was not translated because the syntax is incorrect or the feature is not supported.

MTKM0083

Translation Warning: The called function was not translated successfully.

Description

The function definition of the function being called was not translated successfully. The function definition must be corrected.

MTKM0084

Translation Error: AS Expressions are not yet translated

Description

AS Expressions are not yet translated

MTKM0085

Translation Error: Procedure or function call not translated: the procedure or function must be defined first.

Description

Because the target server does not support function or procedure specifications, functions or procedures can only be referenced once they have been fully defined.

If possible, replace the function or procedure specification with its body.

MTKM0086

Translation Warning: No BITMAP index in DB2 UDB, ignored the BITMAP clause.

Description

DB2 UDB only has a UNIQUE index, not a BITMAP index. The translator ignores the BITMAP clause.

MTKM0087

Input Script Error: This specification item has no definition in the input file

Description

The function, procedure, or cursor specification has no corresponding definition in the input file. The specification has no target server translation, so this item will not appear in the output file.

MTKM0088

Translation Error: Top-level procedure-calls with CLOB parameters are not supported by the target server.

Description

Procedures with CLOB parameters cannot be called from the top-level in the target server. Top-level procedure calls require all arguments being passed to be literals, but the target server does not support these kinds of literals in this context. If DB2 UDB is the target, these calls result in error: ″DB2 UDB1036E The CALL command failed.″

MTKM0089

Translation Error: This declaration is not translated

Description

This declaration is not translated

MTKM0090

Translation Error: EXCEPTION_INIT is not translated. The exception declaration needs to be modified.

Description

The EXCEPTION_INIT pragma is not translated because it has no equivalent in the target server.

In order to get a correct behavior in the target server, you must modify the exception declaration by changing the unreserved SQLSTATE to a reserved SQLSTATE that corresponds to the Oracle error number.

MTKM0091

Translation Error: This ALTER TABLE statement is not translated.

Description

This ALTER TABLE statement is not translated because none of its clauses are supported.

MTKM0092

Translation Error: Update or Delete with reference to rownum pseudocolumn not translated.

Description

Either the statement occurs at the top level and thus cannot be translated using a cursor, or the rownum condition is too complicated to translate to a ″FETCH FIRST n ROWS ONLY″ clause.

MTKM0093

Translation Error: Reference to rownum pseudocolumn in set clause of UPDATE statement not translated.

Description

The reference to the row number in a set clause of an UPDATE statement is not supported in the target server.

MTKM0094

Translation Warning: The function <function_name> is translated to the target server as a Procedure

Description

This Oracle user-defined function is translated to the target server as a procedure. This happens with functions with parameters in OUT mode. Since this feature is not available in the target server, the translator uses a target server procedure instead. The calls to the Oracle function will be translated into procedure calls.

MTKM0095

Input Script Error: Calls to functions with OUT parameter are not allowed in a SQL statement

Description

A call to a function defined with parameters of type OUT is not allowed in an SQL statement (INSERT, UPDATE, DELETE, SELECT...)

MTKM0096

Translation Error: Call to function translated as procedure not translated in this context

Description

In certain cases, Oracle functions are translated to procedures in the target server:
- Functions using OUT parameters
- Functions containing statements that are not allowed in the target server dynamic compound statements

Calls to such functions in Oracle must be translated to procedure calls in the target server, which is not always possible.

There are two different kinds of context in which the translation of the function call fails:
- In a target server dynamic compound statement (function/trigger/anonymous block bodies), because procedure calls are not allowed there
- In branching language constructs (ELSIF, WHILE, etc..), because these constructs require a complex workaround that the translator does not handle.

  For example, consider the function `foo(a OUT INT)` returning an int in Oracle:

  ```
  IF (...)
  ...
  ELSIF (foo(x)=0)
  statements
  END IF;
  ```

  would have to be translated to:

  ```
  IF (...)
  ...
  ELSE
  CALL foo(x, return_val);
  IF (return_val=0)
  statements
  END IF;
  ```

MTKM0097

Translation Error: ROLLUP and CUBE are not translated

Description

The ROLLUP and CUBE extensions of the GROUP BY clause are not yet translated.

MTKM0098

Translation Warning: A column definition has been changed because of an ALTER TABLE statement

Description

A column definition has been added or modified because of a subsequent ALTER TABLE statement.

MTKM0099

Translation Warning: This ALTER TABLE clause has been taken into account by modifying the column definition

Description

This ALTER TABLE is not directly supported in the target server, but the translator took it into account by modifying the corresponding column definition.

MTKM0100

Translation Error: This match-expression or pattern-expression is not allowed in the target server LIKE predicate

Description

This match-expression or pattern-expression is not allowed in the target server LIKE predicate. Please refer to the LIKE predicate section in the target server documentation for more details.

MTKM0101

Translation Warning: Temporary variable <variable name> used to avoid name clash with column name <column name> in INSERT.

Description

In a target server INSERT statement, variable names must not be the same as the names of the columns in the insert table. A temporary variable with a distinct name has been used in place of the original variable which had the same name as a column.

MTKM0102

Translation Error: This cursor for loop cannot be translated because the cursor is invalid

Description

This cursor for loop cannot be translated because the cursor is invalid

MTKM0103

Translation Error: Reference to <object> not translated because the declaration failed

Description

This reference was not translated because the declaration of the corresponding object failed.

MTKM0104

Translation Warning: Reference to <object> not translated because the object is not supported

Description

This reference was not translated because the declaration of the corresponding object is not supported.

MTKM0105

Translation Warning: CREATE TABLESPACE generated with minimal default parameters

Description

The ″TABLESPACE″ option of the translator is turned on. For each TABLESPACE clause found, a ″CREATE TABLESPACE″ statement is generated at the beginning of the output file. This warning indicates that minimal default parameters have been used and, therefore, you must change/enhance these, depending on the physical properties wanted.

MTKM0106

Translation Error: Only one INSTEAD OF trigger per view and per event is allowed in the target server.

Description

In the target server, only one INSTEAD OF trigger is allowed for each kind of operation on a given subject view. Try merging the conflicting triggers in a single target server trigger.

MTKM0107

Translation Warning: SQLCODE or SQLERRM was translated directly using the target server SQLCODE or the target server message text.

Description

References to the Oracle SQLCODE are directly translated to references to the target server SQLCODE.

References to the Oracle SQLERRM are translated using the target server GET STATISTICS statement to retrieve the error message text.

Most of the time the error codes and messages do not match in Oracle and the target server, so it might be necessary to perform some modification to the code.

For example if the SQLCODE is compared to a literal value, it is necessary to change this value to the corresponding SQLCODE in the target server.

MTKM0108

Translation Information: Translation Ratio: <percentage>% (<absolute ratio> statements were translated successfully)

Description

This provides an assessment of the provided translation by giving the ratio of Oracle statements translated without producing any error message out of the total number of statements. This number provides a general indication regarding the success of the automated translation and does not intend to give an exact and accurate measure.

Statement here designates Oracle SQL and PL/SQL statements. For instance, in a CREATE PROCEDURE, the whole SQL statement is counted as 1 and each PL/SQL statement inside the body of the procedure are also counted as one.

MTKM0109

Translation Error: Procedure call in trigger body is not translated

Description

In Oracle the body of a trigger can be either a PL/SQL block or a procedure call statement. The translator does not support call statements for trigger bodies.

The target server trigger bodies do not support procedure calls, so embedding the procedure call in a block will not solve the problem.

A possible solution is to copy the procedure code into the trigger block, if this is possible.

MTKM0110

Translation Error: Arguments having a structured type are not supported: <structured-type-name>

Description

The converter does not support the translation of arguments having structured types (%ROWTYPE and record types). In this context, the target server does not support such kinds of data types.

One possible workaround is to explode this structured argument into several arguments having scalar data types. However, this would modify the signature of the routine, which means the calls to the function/procedure, including calls located in external applications, should be modified accordingly.

MTKM0111

Input Script Error: This is not a procedure: <object-name>

Description

In this procedure call, the references resolve to an object that is not a procedure.

MTKM0112

Translation Error: The following objects are involved in a circular dependency: <objects>

Description

The specified objects are involved in a circular dependency.

Since the target server does not support forward declarations (through function specifications, cursor specifications...), there is no way to declare these objects in the target server.

The specified objects are translated, but they contain unresolved references.

MTKM0113

Input Script Error: This reference does not resolve to <a specific object type>: <reference>

Description

The given reference is made to an object with a type that is unexpected in this context. For instance, in a procedure call, the given name is in fact a function.

MTKM0114

Translation Warning: The CREATE SYNONYM statement is not translated. But any object referred to by the synonym is referred to directly.

Description

In the target server, it is possible to create an alias on a table, a view or another alias, but not on a function, procedure, sequence or schema (which is used to translate packages). There is therefore no direct translation for a CREATE SYNONYM on these objects.

To produce a correct translation, the converter does not translate the CREATE SYNONYM statement, but replaces all references to the synonym by a reference to the object it designates.

MTKM0115

Input Script Error: This is not a table or view: <object-name>

Description

In this context, the reference resolves to an object that is not a table or view.

MTKM0116

Translation Error: Oracle Rowid values are ignored.

Description

The Oracle Rowid type is translated to Integer in DB2 UDB. The rowid values that identify rows in the database are generated by DB2 UDB. Only these DB2 UDB-generated values can be stored in the DB2 UDB database.

MTKM0117

Translation Warning: DECIMAL type for arguments is translated with arbitrary precision (31) and scale (0)

Description

For arguments to functions, procedures, or parameterized cursors, Oracle requires that a precision, scale, or length for the data type not be specified. Oracle automatically specifies them based on the actual arguments used in calls.

The target server, however, does require that those parameters be specified. Since the translator cannot infer the precision and scale to pick when translating to the target server, for the DECIMAL type, it picks arbitrary values: 31 for precision and 0 for scale. Type conversions (ROUND...) will also be applied accordingly.

MTKM0118

Translation Error: Support for Rowid column not enabled.

Description

This reference to ROWID will not be valid in DB2 UDB unless the translation is performed with the ″rowid columns″ option enabled. This option adds an identity column called ROWID to each base table in DB2 UDB.

MTKM0120

Translation Error: The CREATE SEQUENCE statement contains a value that is out of range

Description

The CREATE SEQUENCE statement contains a MINVALUE, MAXVALUE or START VALUE clause with a value that is outside the range that the converter can handle.

MTKM0121

Translation Error: The Index matches a unique constraint of the table

Description

The target server automatically generates an index for each unique constraint (or primary key) of the tables. It will therefore reject the creation of indexes matching these constraints.

The translation of this CREATE INDEX statement was successful, but it will probably cause an error in the target server.

MTKM0129

Translation Error: This statement is not supported in a target server Procedure, Function, or Trigger

Description

Some statements, especially DDL statements such as ALTER TABLE or CREATE objects other than tables, indexes, or views, are not allowed inside the target server procedures, functions, and triggers. In the context of a procedure, such statements can be executed using dynamic SQL.

MTKM0135

Translation Warning: BEFORE trigger was translated to AFTER trigger

Description

BEFORE triggers are usually translated to NO CASCADE BEFORE triggers in the target server. Unfortunately this kind of trigger does not allow DML statements (INSERT, UPDATE and DELETE) and the FOR EACH STATEMENT clause.

If the source trigger does use one of the features, the translator will try to translate the BEFORE trigger to an AFTER trigger, which does not have those restrictions.

This special translation is not possible if the body of the trigger refers to the table being the object of the trigger, of if a column of the NEW table is assigned.

MTKM0136

Translation Error: Collection types are not supported in this context.

Description

Collection types are supported in local variable declarations and formal parameters only.

MTKM0137

Translation Warning: Complex column default on a nullable column

Description

This complex column default has been translated inside a special trigger. The column is nullable, so the trigger cannot make the distinction between a null value and an unspecified value, so the translator will change the value using the default.

It is therefore necessary to review the code to check that null values are not inserted.

If this is not the case, a solution is to use a special value as the default for the column (value that is not likely to be inserted), and to test the inserted value against this special value inside the trigger instead of using null.

MTKM0138

Translation Information: Trigger generated for this table’s complex defaults and check constraints

Description

Complex column defaults and check constraints cannot be translated directly into the target server CREATE TABLE statement. An extra trigger is generated for this table. This trigger will perform the necessary operations to emulate the defaults and check constraints.

MTKM0139

Translation Warning: NOT NULL constraint was removed because of the complex DEFAULT

Description

This column definition contains a complex default that was translated as a trigger. This special translation requires the removal of the NOT NULL constraint so that NULL values can be used to represent an unspecified value.

A manual code review is therefore necessary.

MTKM0140

Input Script Error: Invalid number format

Description

The number format is invalid or the input data (a number value) doesn’t match the number format.

MTKM0141

Translation Error: RETURN statements are not allowed in the target server exception handlers

Description

In the target server, RETURN statements are not allowed in condition handlers. In some situations the RETURN statement can simply be removed, but it is necessary to perform a manual code review to ensure that the control flow of the handler body is correct.

MTKM0143

Translation Error: Routine overloading not allowed or not supported

Description

Function or procedure overloading is either not allowed or not supported in this context. The object is translated, but calls to this function or procedure will not resolve correctly.

MTKM0145

Translation Warning: The translation of TIMESTAMP(p) might cause the loss of significant digits

Description

The Oracle TIMESTAMP(p) type is translated to TIMESTAMP in the target server regardless of the provided precision. If the precision is greater than 6 (for DB2 UDB as target) or greater than 5 (for IDS as target), DB2 UDB or IDS will loose some significant digits in the fractional part of the timestamp.

MTKM0191

Translation Error: Column <column name> is not in the group by clause, but is used in a select, order by or having clause.

Description

This subquery contains a group by clause and the select clause, order by clause, or having clause of this subquery contains a reference to a column that is not in the group by clause and is not inside of a call to an aggregate function. This is an extension to the SQL Standard that is not supported by MTK or the target server.

MTKM0194

Translation Error: Unsupported predefined exception: <exception name>

Description

This Oracle predefined exception does not clearly match a target server SQL State. Manual translation required.

MTKM0211

“Messages” on page 131: Column <column name> occurs with an aggregate function in a select or order by clause without a group by clause.

Description

This subquery contains an aggregate function without a group by clause and the select clause or order by clause of this subquery contains a reference to a column that is not inside of a call to an aggregate function. This is a TSQL-extension that is not supported by MTK or DB2.

MTKM0212

Translation Error: This view cannot be updated in DB2. The check option is omitted.

Description

The check option is not allowed in DB2 in the definition of read-only views.

MTKM0213

Translation Error: This view has been declared as read-only.

MTKM0214

Translation Error: The base view or table-expression used to define this view is read-only.

MTKM0215

Translation Error: This view is read-only because it selects only distinct elements.

MTKM0216

Translation Error: This view might not be updatable because it selects complex non-column values.

MTKM0217

Translation Error: This view is read-only because it has no FROM clause.

MTKM0218

Translation Error: This view is read-only in DB2 because it has multiple base tables.

Description

This view has multiple base tables. Deletes, updates, or inserts to such a view are not standard and are similar to joined deletes and joined updates.

MTKM0219

Translation Error: This view is read-only because it has a GROUP BY clause.

MTKM0220

Translation Error: This view is read-only because it has a HAVING clause.

MTKM0221

Translation Error: This view might be read-only because of the use of UNION.

MTKM0225

Translation Warning: Unable to validate syntax of dynamic SQL in EXECUTE IMMEDIATE.

Description

MTK is unable to translate the dynamic string argument to the EXECUTE IMMEDIATE statement since it might contain elements that are not known until runtime. You should inspect this string and translate it by hand if necessary.

MTKM0226

Translation Error: Cannot return a FULLSELECT in this context.

Description

The following are the restrictions with using FULLSELECT in a RETURN statement:
- A FULLSELECT cannot be specified for a RETURN from a procedure.
- If the FULLSELECT is returned from scalar function, then the FULLSELECT must return one column, and at most, one row.
- If the fullselect is returned from a row function, it must return, at most, one row.

MTKM0227

Translation Error: No column-list specified in the RETURN clause of the CREATE FUNCTION statement.

Description

The translator is unable to determine the columns returned by the CREATE TABLE FUNCTION.

MTKM0244

Translation Warning: Label <label name> associated with the removed nested compound statement has been moved to the first contained statement.

Description

The associated label along with the contents of the compound statement have been moved to the first contained statement. A review of all references to the label is required. Manual translation of some references might be required.

MTKM0245

Translation Warning: The translator has removed the nested target server compound statement by moving the contents to a higher block.

Description

The translator has removed the nested target server compound statement. The contents of the compound statement have been moved to a higher block.

MTKM0246

Translation Warning: Nested compound statements are not supported inside an exception handler. Manual translation of the statement is required.

Description

This version of the target server does not support nested compound statements. Consider replacing the compound statement with a stored procedure call.

MTKM0247

Translation Warning: Nested compound statements containing an exception handler are not supported. Manual translation of the statement is required.

Description

This version of the target server does not support nested compound statements. Consider replacing the compound statement with a stored procedure call.

MTKM0251

Translation Warning: Ignored the ON DELETE SET NULL clause since Informix Dynamic Server does not support it with a foreign key constraint.

Description

This version of Informix Dynamic Server does not support using the ON DELETE SET NULL clause with a foreign key constraint.

MTKM0252

Translation Warning: No BITMAP index in Informix Dynamic Server, ignored the BITMAP clause.

Description

This version of Informix Dynamic Server does not support BITMAP indexes.

MTKM0253

Translation Error: Informix Dynamic Server does not support the COMMENT statement.

Description

This version of Informix Dynamic Server does not support the COMMENT statement.

MTKM0254

Translation Error: Informix Dynamic Server does not support a reference to rownum pseudocolumn.

Description

The rownum pseudocolumn is not supported in the Informix Dynamic Server target; the translator will ignore the rownum contexts and generate remaining statements into the output file.

MTKM0255

Translation Error: IDS supports only one element in the expression list of the IN predicate.

Description

Informix Dynamic Server supports only one element in the expression list of the IN predicate.

MTKM0256

Translation Error: IDS does not support the use of lists with the ALL operator.

Description

Informix Dynamic Server does not support the use of lists with the ALL operator.

MTKM0257

Translation Error: IDS does not support aggregate functions in the SET clause of UPDATE.

Description

The Informix Dynamic Server UPDATE statement does not support aggregate functions in the SET clause.

MTKM0258

Translation Error: IDS does not support the following in a subselect clause: ORDER BY, UNION, FIRST and INTO TEMP.

Description

The subselect clause of the Informix Dynamic Server SQL grammar might not contain FIRST, INTO TEMP, ORDER BY or UNION.

MTKM0259

Translation Warning: Unable to infer Informix Dynamic Server data type for the expression. Using Varchar(1).

Description

A data type is needed in this context for use in a declaration or cast expression.

MTKM0283

Translation Error: Cursor RETURN clauses are not translated.

Description

The target server has no equivalent clause to declare the return type of a cursor. The behavior of the cursor should not be affected.

MTKM0284

Translation Warning: Ignored the ON DELETE RESTRICT clause because Informix Dynamic Server does not support it with a foreign key constraint.

Description

This version of Informix Dynamic Server does not support using the ON DELETE RESTRICT clause with a foreign key constraint.

MTKM0285

Translation Warning: Ignored the ON DELETE NO ACTION clause beacause Informix Dynamic Server does not support it with a foreign key constraint.

Description

This version of Informix Dynamic Server does not support using the ON DELETE NO ACTION clause with a foreign key constraint.

MTKM0286

Translation Warning: Ignored the MATCH FULL/MATCH PARTIAL/MATCH SIMPLE clause because the target does not support it with a foreign key constraint.

Description

This version of target Server does not support using the MATCH FULL/MATCH PARTIAL/MATCH SIMPLE clause with a foreign key constraint.

MTKM0287

Translation Warning: Ignored the ON UPDATE SET NULL clause because DB2 LUW server does not support it with a foreign key constraint.

Description

This version of DB2 LUW Server does not support using the ON UPDATE SET NULL clause with a foreign key constraint.

MTKM0288

Translation Warning: Ignored the ON UPDATE CASCADE clause because DB2 LUW Server does not support it with a foreign key constraint.

Description

This version of DB2 LUW Server does not support using the ON UPDATE CASCADE clause with a foreign key constraint.

MTKM0289

Translation Warning: Ignored the ON UPDATE clause because Informix Dynamic Server does not support it with a foreign key constraint.

Description

This version of Informix Dynamic Server does not support using the ON UPDATE clause with a foreign key constraint.

MTKM0290

Translation Warning: Ignored the LIKE clause because Informix Dynamic Server does not support it with a CREATE TABLE statement.

Description

This version of Informix Dynamic Server does not support using the LIKE clause with a create table statement.

MTKM0291

Translation Warning: Ignored the CHECK constraint because MySQL only parses the CHECK constraints but does not enforce them.

Description

The MySQL server does not enforce CHECK constraints however DB2 and IDS do. To avoid data truncation the constraints are parsed and ignored.

MTKM0292

Translation Warning: Ignored the multiple INSERT statement because Informix Dynamic Server does not support it.

Description

This version of Informix Dynamic Server does not support multiple INSERT statements.

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in all countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION ″AS IS″; WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Trademarks

Index

Special characters

@@ERROR 292
@@ROWCOUNT 296
@@SQLSTATUS 296
%ROWTYPE attribute 178

A

accessibility 8
adding schema names to generated file
  example 33
alter statements
  translations from Sybase 282
ALTER TABLE statements
  translations from Oracle 216
  translations from Sybase 282
application log 112
archive 12

B

backing up a project 15
blank and non-blank padding
  translations from Oracle 167
Boolean expressions
  DB2 166
  Informix Dynamic Server 166
built-in data types
  translations from Microsoft SQL Server 264
  translations from Oracle 172
  translations from Sybase 264
built-in functions
  translations from Microsoft SQL Server 268, 302, 303
  translations from Oracle 159, 237, 239
  translations from Sybase 268, 302, 303
  translations to DB2 from Informix Dynamic Server 137

C

cast expressions 186
Changes report 38, 42, 112
clause, TABLESPACE 215
collection types 232
columns
  changing names 41, 280
command line

    example 100
  arguments 76
  configuration file 75, 77
  convert source SQL
    example 102
  deploy
    example 104

command line *(continued)*
  deploying data with advanced options
    example 105
  extract from source
    example 101
  full migration
    example 106
  generate data transfer scripts
    example 103
  import source SQL file
    example 100
  Linux 74
  process 73
  projects 99
  running 73, 74
  saving projects 100
  UNIX 74
  Windows 73
COMMENT ON statements 292
concatenation 194
conditions
  translations from Oracle 166
  translations to DB2 from Informix Dynamic Server 140
configuration file
  elements
    CONVERSION 78
    CONVERSIONS 78
    CONVERT 79
    DATABASE 84
    DEPLOY_TO_TARGET 85
    EXTRACT 86
    GENERATE_DATA_TRANSFER_SCRIPTS 88
    GLOBAL_SETTINGS 89
    GLOBALCONVERT 89
    IMPORT 94
    JDBC_CONNECTION 94
    ODBC_CONNECTION 95
    PROJECT 96
    SCHEMA 98
    SPECIFY_SOURCE 98
  sample 75
connect statement 207
connecting to database 17
  Informix Dynamic Server 18
  Microsoft SQL Server 21
  MySQL 20
  Oracle 17
  Sybase 22
conversion
  analyzing and refining results 40
  behavior and limitations
    translations from Informix Dynamic Server to DB2 133
    translations from Microsoft SQL Server 263
    translations from Oracle 159
    translations from Sybase 263
  deleting 34

conversion *(continued)*
how it works 127
messages 131
starting a new project 12
conversion action
removing 38, 42
viewing 112
conversion functions
translations from Microsoft SQL Server 313
translations from Sybase 313
conversion summary report 111
conversions
Convert page 34
implicit 278
reloading files 34
Convert page 28, 34
converting source 28
CREATE DOMAIN statement
converted from Sybase 286
CREATE INDEX
MySQL statement 154
CREATE INDEX statements
translations from Microsoft SQL Server 285
translations from Sybase 285
Create New Project window 12
CREATE TABLE
MySQL statement 153
Oracle statement 212
SQL Server statement 283
Sybase statement 283
CREATE TYPE statements
translations from Oracle 219
cross joins 281
cursors 169
translations from Oracle to DB2 UDB 169
translations from Oracle to Informix Dynamic Server 171

D

Data
manually transferring 72
manually transferring to a remote DB2 database server 71
manually transferring to DB2 UDB 68
data compression
DB2 support 214
data extraction script 70
Data Type Editor window 36
data types
mapping 35
translations from Microsoft SQL Server 264
translations from MySQL 151
translations from Oracle 172
translations from Sybase 264
translations to DB2 from Informix Dynamic Server 133
database
connecting to Informix Dynamic Server 18
connecting to Microsoft SQL Server 21
connecting to MySQL 20
database *(continued)*
connecting to Oracle 17
connecting to Sybase 22
extracting objects from 24
date arithmetic limitations
translations to DB2 from Informix Dynamic Server 135
date literals
translations from Oracle 186
datetime conversions
translations from Microsoft SQL Server 318
translations from Sybase 318
DB2 UDB supported SQL features for MySQL
translations from MySQL 154
DDL statement conversion 211
delete statements 286
DELETE statements
translations from Oracle 222
deleting
a project 14
conversions 34
source files 27
Deploy to Target page
DB2 for z/OS 63
DB2 UDB 60
Informix Dynamic Server 65
using 59
deployment log 111
division 194
drivers
Oracle 17
drop index statements 282
DROP statements
translations from Oracle 219
dual tables 213
dynamic compound statements 235

E

estimate of table size report 111
example
adding schema names to generated file 33
exceptions
translations to DB2 from Informix Dynamic Server 147
EXECUTE statements 289
EXIT statements
translations from Oracle 223
explicit and implicit data type conversion
translations from Oracle 175
expressions
translations from Microsoft SQL Server 276
translations from Oracle 185
translations from Sybase 276
translations to DB2 from Informix Dynamic Server 139
Extract window 24
extracting source objects 24

F

files
  project 11
FOREACH statements
  translations to DB2 from Informix Dynamic Server 148
functions
  built-in 237, 239, 268, 303
  that do not have equivalents 237, 302
  that simulate auxiliary functions 303
  translations from Microsoft SQL Server 268
  translations from Oracle 159
  translations from Sybase 268
  translations to DB2 from Informix Dynamic Server 137
  user-defined 228

G

generate data transfer scripts
  deploy 109
  wizard 109
Generate Data Transfer Scripts page 69, 72
Generate Scripts page
  using 44
global type mapping 35
GOTO statement
  Translations to Informix Dynamic Server 222
GRANT statements 289

I

identifiers 192
  translations from Microsoft SQL Server 263, 264
  translations from Sybase 263, 264
IF statements 291
implicit and explicit data type conversion
  support 175
  translations from Oracle 175
implicit commit conversion 211
implicit type conversion
  translations from Oracle 188
in-context metadata 34
indexes
  translations from Oracle 217
Informix Dynamic Server supported SQL features for MySQL
  translations from MySQL 156
INFX functions
  for translations to DB2 149
INSERT
  MySQL statement 154
INSERT statements 286
  translations from Oracle 220
insert-execute statements 290
installing MTK 4
  UNIX and Linux 5
  Windows 4
  Windows silent install 5
interfaces
  command line 9
  graphical user interface (GUI) 9

J

joined update and delete statements 286
joins 281

K

Keyboard shortcuts 8

L

labels
  translations from Oracle 224
large statement warnings report 111
LIKE clause
  translations from Oracle 169
literals 276

M

mapping data types 35
messages
  from conversions from Informix Dynamic Server 321
  from conversions from Microsoft SQL Server 389
  from conversions from MySQL 437
  from conversions from Oracle 353
  from conversions from Sybase 389
  generated during conversion 131
metadata
  converting 28, 39
  previously converted 34
  refining during conversion 39
migration process 17
migration projects 11
MTK
  basics 3
  conversion
    five steps for using MTK 23
  interface 23
  preferences 7
  projects 11
mtkdbmspack.* file 204
multiple source files 34
MySQL
  data types 151

N

naming convention
  Oracle PL/SQL to Informix Dynamic Server 193
national language support (NLS) 237
null literals
  translations from Oracle 187
null values and empty strings
  translations from Oracle 188

O

objects
  changing names 41
ON EXCEPTION statements
  translations to DB2 from Informix Dynamic Server 147
operators 193
  translations from Microsoft SQL Server 276
  translations from Sybase 276
ORA schema 237
Oracle drivers
  JDBC 17
  ODBC 17
outer joins
  translations from Oracle 206
outer-joins 280

P

package
  variables
    conversions to DB2 Viper II 201
    translations from Oracle 201
packages
  translations from Oracle 197, 204
  translations from Oracle to DB2 UDB 197
  UTL_FILE 204
PL/SQL conversion 222
procedures 228, 229
project
  archive 12
  backing up 15
  closing 14
  creating 12
  deleting 14
  deploying 59
  files 11
  general information 11
  modifying description of 14
  opening 13
  removing conversions 34
  restoring 15
  saving 13
  user preference options 7
  user preferences 16
  working collaboratively 34
PROJECT element
  example 96

Q

queries
  translations from Microsoft SQL Server 279
  translations from Oracle 205
  translations from Sybase 279

R

RAISE EXCEPTION statements
  translations to DB2 from Informix Dynamic Server 148
RAISERROR statement 292
referred-to metadata 34
Refine page 40, 41, 42
  Target page 41
  using 39
renaming
  columns 280
  objects 41, 130
  requirements 130
reports
  application log 112
  Changes 38, 42, 112
  conversion summary 111
  deployment log 111
  estimate of table size 111
  large statement warnings 111
  translation error message 111
  verification 111
  viewing 111
Restoring
  a project 15
ROWID pseudocolumn 175
ROWNUM 191

S

saving
  a project 13
schemas 197
scripts
  generating 44
  generating for DB2 for z/OS 57
  generating for DB2 UDB 44
  generating for Informix Dynamic Server 58
select statements 288
short-circuit evaluations
  Informix Dynamic Server support 191
source database support 1
source files
  deleting 27
  importing 24
  reloading 34
  reordering 27
  specifying for migration 24
  SQL script 127, 129
source metadata
  converting 28
special registers
  translations from Oracle 187
Specify source page 27
Specify Source page 24
  using 24
SPL statements
  translations to DB2 from Informix Dynamic Server 146

SQL statements
  in source SQL script 127, 129
  testing the translation of 43
  translations to DB2 from Informix Dynamic Server 140
SQLCODE 191
Start MTK
  Linux 6
  Microsoft Windows 6
  UNIX 6
statements
  behavior and limitations
    translations from Microsoft SQL Server 281
    translations from Sybase 281
  not supported
    in translations from Microsoft SQL Server 281
    in translations from Sybase 281
  translations from MySQL 153
  translations from Oracle 206
strings 276, 277
supported SQL features for DB2 UDB
  translations from Oracle 243
supported SQL features for Informix Dynamic Server
  translations from Oracle 253
synonyms 217
system catalogs
  translations to DB2 from Informix Dynamic Server 133
system procedures
  translations from Sybase 267
system requirements 3
  hardware 3
  software 3

T

tables
  CREATE TABLE statement 215
  CREATE TABLE statements 283
  temporary 300
  translations from Oracle 212
  variables 301
target database support 1
temporary tables 300
timestamp literals
  translations from Oracle 186
transaction statements 235
  translations to DB2 from Informix Dynamic Server 149
transactions 297
translating source 28
translation error message reports 111
trigger events
  translations from Oracle 227
triggers
  procedure calls from within 228
  translations from Microsoft SQL Server 299
  translations from Oracle to DB2 UDB 226
  translations from Oracle to Informix Dynamic Server 226
  translations from Sybase 299
truncate statements 284

U

update statements 221, 286
UPDATE statements
  translations from Oracle 221
user interfaces
  graphical user interface (GUI) 9
user-defined functions 228

V

VARCHAR
  ODBC and JDBC restrictions 17
VARRAY
  function calls 181
  limitations 179
verification report 111
views 218
  CREATE VIEW statements 285

W

wizard
  deploy 109
  generate data transfer scripts 109
  global type mapping 108
  migration summary 110
  process 107
  project information 108
  specify source database 108