The next step in big data starts with IBM.

Extreme speed. Increased flexibility. Enhanced performance.

IBM

Fonctions Applicatives de DB2 V10 et V11

Cécile Benhamou
Technical Sales DB2 z/OS et Tools DB2
cecile_benhamou@fr.ibm.com

DB2 X for z/OS

Réponse à des objectifs de l'entreprise

Fin de support
30 Septembre 2017

Application Enablement	• Améliorations pureXML • Requêtes bi-temporelles • Last Committed reads • Timestamp avec timezone • Améliorations SQL pour simplifier le portage
RAS, Performance, Scalability, Security	• Nombreuses améliorations de performance • Plus de modifications "online schema" • Restructuration du Catalogue pour une meilleure concurrence • Accès aux données "Hash" • Contrôle d'accès plus fin • Nouveaux privilèges DBA avec plus de granularité
Simplification, Productivity	• Full 64-bit SQL runtime • Auto stats • Compression de données "on the fly" • Stabilité des requêtes et améliorations de gestion • Besoin réduit de REORG • Améliorations des Utilitaires
Data Warehousing	• Moving sum, moving average • Nombreuses améliorations d'optimisation de requêtes • Améliorations sur parallélisme de requêtes • Advanced query acceleration

DB2 11: Thèmes Principaux

Annonce
2013, Oct. 1

Disponibilité(GA)
2013, Oct. 25

- **Economies CPU "Out-of-the-box"**
  - Amélioration de coût sans changement des applications
  - Jusqu'à 10% pour des requêtes OLTP complexes
  - Jusqu'à 10% pour des mises à jour batch intensives
  - Jusqu'à 40% pour les requêtes
  - Améliorations de performance additionelles avec l'utilisation des nouvelles fonctionnalités
- **Amélioration de la Résilience et de la Disponibilité**
  - Améliorations autonomiques: réduction des coûts et de la disponibilité
  - Plus de changements online sans affecter les applications
  - Améliorations REORG Online: interruption réduite
  - DROP COLUMN, Modification online des clés limites de partition
  - Extended log record addressing capacity - 1 yottabyte (ou 1M petabytes)
  - BIND/REBIND, DDL – interruption forcée dans les threads persistants
- **Business analytics**
  - Capacités étendures du SQL analytique et du XML
  - Améliorations sur les tables Temporelles et le SQLPL
  - Archivage transparent
  - Intégration Hadoop, Support NoSQL et JSON

- **Migration DB2 plus simple et rapide**
  - Plus aucune modification nécessaire pour les applications lors de migrations de version
  - Améliorations sur la stabilité des Chemins d'accès
  - Meilleure stabilité/qualité

DB2 V10: Données temporelles

- Business Time (Effective Dates, Valid Time)
  - Chaque ligne a une paire de Timestamp mis par l'application
    - Start time: quand les métiers jugent que la ligne est valide
    - End Time: quand les métiers jugent que la ligne n'est plus valide
    - Colonnes définies en TIMESTAMP(6) NOT NULL ou DATE NOT NULL

- System Time (Assertion Dates, Knowledge Dates, Transaction Time)
  - Chaque ligne a une paire de Timestamp mis par le SGBD
    - Start time: quand la ligne est insérée par le SGBD
    - End Time: quand la ligne est modifiée/supprimée
    - Le 'start time' des lignes modifiées est le moment de la modification
    - Colonnes définies en TIMESTAMP(12) NOT NULL GENERATED ALWAYS

- Bi-temporel
  - Inclusion des 2 Temps (System Time et Business Time) dans la ligne

DB2 V10: Données temporelles

```
CREATE TABLE POLICY
   EMPL    VARCHAR(4)    NOT NULL,
   TYPE    VARCHAR(4),
   PLCY    VARCHAR(4)    NOT NULL,
   COPAY   VARCHAR(4),

   SYS_BEG TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW BEGIN,
   SYS_END TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW END,
   CRT_ID  TIMESTAMP(12) GENERATED ALWAYS AS TRANSACTION START ID NOT NULL,
   PERIOD  SYSTEM_TIME (SYS_BEG, SYS_END),

   EFF_BEG DATE          NOT NULL,
   EFF_END DATE          NOT NULL,
   PERIOD  BUSINESS_TIME (EFF_BEG, EFF_END),
   PRIMARY KEY (EMPL,PLCY, BUSINESS_TIME WITHOUT OVERLAPS) );
```

```
CREATE TABLE POLICYHISTORY LIKE POLICY;
```

```
ALTER TABLE POLICY ADD VERSIONING
                   USE HISTORY TABLE POLICYHISTORY;
```

Uniqement nécessaire pour les données temporelles SYSTEME

DB2 V10: Données temporelles: exemples

Step 1 – **9/21/2010** Employee C054 chooses HMO policy with $10 copay effective **1/1/2004**

```
INSERT INTO POLICY
 (EMPL,TYPE,PLCY,COPAY,EFF_BEG,EFF_END)
  VALUES ('C054','HMO', 'P667', '$10', '1/1/2004', '12/31/9999');
```

Policy Table

EMPL	TYPE	PLCY	COPAY	EFF_BEG	EFF_END	SYS_BEG	SYS_END
CO54	HMO	P667	$10	**2004-01-01**	9999-12-31	**2010-09-21**-21.50.14.745082721000	9999-12-30-00.00.00.000000000000
				Business Time start	Business Time end	SYSTEM Time start	SYSTEM Time end

POLICYHISTORY table is empty at this point

DB2 V10: Données temporelles: exemples

Step 1 – 09/21/2010 Employee C054 chooses HMO policy with $10 copay effective 1/1/2004
Step 2 – **09/24/2010** Update all P667 policies to a copay of $15 beginning **01/01/2011**

Original Row

EMPL	TYPE	PLCY	COPAY	EFF_BEG	EFF_END	SYS_BEG	SYS_END
CO54	HMO	P667	$10	2004-01-01	9999-12-31	2010-09-21-21.50.14.745082721000	9999-12-30-00.00.00.000000000000

```
UPDATE POLICY FOR PORTION OF BUSINESS_TIME
FROM '01/01/2011' TO '12/31/9999'
SET COPAY='$15'
WHERE PLCY='P667';
```

Policy Table

EMPL	TYPE	PLCY	COPAY	EFF_BEG	EFF_END	SYS_BEG	SYS_END
CO54	HMO	P667	$10	2004-01-01	2011-01-01	2010-09-24-17.33.22.50672497000	9999-12-30-00.00.00.000000000000
C054	HMO	P667	$15	2011-01-01	9999-12-31	2010-09-24-17.33.22.50672497000	9999-12-30-20000.00.000000000000

Policy History table

EMPL	TYPE	PLCY	COPAY	EFF_BEG	EFF_END	SYS_BEG	SYS_END
CO54	HMO	P667	$10	2004-01-01	9999-12-31	2010-09-21-21.50.14.745082721000	2010-09-24-17.33.22.50672497000

DB2 V10: Données temporelles: exemples


```
SELECT * FROM POLICY FOR SYSTEM_TIME AS OF
    '2010-09-22-00.00.00.000000000000';
```

EMPL	TYPE	PLCY	COPAY	EFF_BEG	EFF_END	SYS_BEG	SYS_END	Which Table
CO54	HMO	P667	$10	2004-01-01	9999-12-31	2010-09-21-21.50.14	2010-09-24-17.33.22	HISTORY
CO54	HMO	P667	$10	2004-01-01	2011-01-01	2010-09-24-17.33.22	9999-12-30-00.00.00	BASE
C054	HMO	P667	$15	2011-01-01	9999-12-31	2010-09-24-17.33.22	9999-12-30-00.00.00	BASE

DB2 V10: Données temporelles: exemples

```
SELECT * FROM POLICY FOR SYSTEM_TIME AS OF
    '2010-09-22-00.00.00.000000000000';
```

EMPL	TYPE	PLCY	COPAY	Z	EFF_END	SYS_BEG	SYS_END	Which Table
CO54	HMO	P667	$10	2004-01-01	9999-12-31	2010-09-21-21.50.14	2010-09-24-17.33.22	HISTORY
CO54	HMO	P667	$10	2004-01-01	2011-01-01	2010-09-24-17.33.22	9999-12-30-00.00.00	BASE
C054	HMO	P667	$15	2011-01-01	9999-12-31	2010-09-24-17.33.22	9999-12-30-00.00.00	BASE

As of 09-22-2010 , the only row that qualifies is the row from the history table, because on 09-24-2010 we updated the rows, and both rows in the current table begin on 09-24-2010.

As of 09-24-2010-17.33 and after, rows from the current table would be returned

Only the POLICY appears in the SELECT statement. POLICYHISTORY is automatically accessed.

Results only come from the history table

EMPL	TYPE	PLCY	COPAY	EFF_BEG	EFF_END	SYS_BEG	SYS_END	Which Table
CO54	HMO	P667	$10	2004-01-01	9999-12-31	2010-09-21-21.50.14	2010-09-24-17.33.22	HISTORY

DB2 V10: Données temporelles: exemples

Step 1 - 09/21/2010 Employee C054 chooses HMO policy with $10 copay effective 1/1/2011
Step 2 - 09/24/2010 Update all policies P667 to a copay of $15 beginning 01/01/2011
Step 3 - **09/24/2010** Later on the same day(19.44) of 09/24, the customer cancelled the policy

EMPL	TYPE	PLCY	COPAY	EFF_BEG	EFF_END	SYS_BEG	SYS_END	Which Table
CO54	HMO	P667	$10	2004-01-01	9999-12-31	2010-09-21-21.50.14	2010-09-24-17.33.22	HISTORY
CO54	HMO	P667	$10	2004-01-01	2011-01-01	2010-09-24-17.33.22	9999-12-30-00.00.00	BASE
C054	HMO	P667	$15	2011-01-01	9999-12-31	2010-09-24-17.33.22	9999-12-30-00.00.00	BASE

```
DELETE FROM POLICY FOR PORTION OF BUSINESS_TIME
FROM CURRENT DATE TO '12/31/9999'
WHERE EMPL='C054' AND PLCY='P667';
```

DB2 V10: Données temporelles: exemples

Step 1 - 09/21/2010 Employee C054 chooses HMO policy with $10 copay effective 1/1/2011
Step 2 - 09/24/2010 Update all policies P667 to a copay of $15 beginning 01/01/2011
Step 3 - **09/24/2010** Later on the same day(19.44) of 09/24, the customer cancelled the policy

EMPL	TYPE	PLCY	COPAY	EFF_BEG	EFF_END	SYS_BEG	SYS_END	Which Table
CO54	HMO	P667	$10	2004-01-01	9999-12-31	2010-09-21-21.50.14	2010-09-24-17.33.22	HISTORY
CO54	HMO	P667	$10	2004-01-01	2011-01-01	2010-09-24-17.33.22	9999-12-30-00.00.00	BASE
C054	HMO	P667	$15	2011-01-01	9999-12-31	2010-09-24-17.33.22	9999-12-30-00.00.00	BASE

```
DELETE FROM POLICY FOR PORTION OF BUSINESS_TIME
FROM CURRENT DATE TO '12/31/9999'
WHERE EMPL='C054' AND PLCY='P667';
```

EMPL	TYPE	PLCY	COPAY	EFF_BEG	EFF_END	SYS_BEG	SYS_END	Which Table
CO54	HMO	P667	$10	2004-01-01	9999-12-31	2010-09-21-21.50.14	2010-09-24-17.33.22	HISTORY
CO54	HMO	P667	$10	2004-01-01	2011-01-01	2010-09-24-17.33.22	**2010-09-24-19.44.47**	HISTORY
C054	HMO	P667	$15	2011-01-01	9999-12-31	2010-09-24-17.33.22	**2010-09-24-19.44.47**	HISTORY
CO54	HMO	P667	$10	2004-01-01	**2010-09-24**	**2010-09-24**-19.44.47	9999-12-30-00.00.00	BASE

DB2 V10: Données temporelles: exemples

EMPL	TYPE	PLCY	COPAY	EFF_BEG	EFF_END	SYS_BEG	SYS_END	Which Table
CO54	HMO	P667	$10	2004-01-01	9999-12-31	2010-09-21-21.50.14	2010-09-24-17.33.22	HISTORY
CO54	HMO	P667	$10	2004-01-01	2011-01-01	2010-09-24-17.33.22	2010-09-24-19.44.47	HISTORY
C054	HMO	P667	$15	2011-01-01	9999-12-31	2010-09-24-17.33.22	2010-09-24-19.44.47	HISTORY
CO54	HMO	P667	$10	2004-01-01	2010-09-24	2010-09-24-19.44.47	9999-12-30-00.00.00	BASE

```
SELECT * FROM POLICY  FOR BUSINESS_TIME AS OF
    '2011-09-25' ORDER BY EFF_BEG;
```

No rows returned.
Business time only looks at base table, not the history table

DB2 V11: Données temporelles

- Support pour les vues
- Support de registres spéciaux pour retrouver les données de tables temporelles sans modifier le SQL
  - Possibilité d'exécuter la même requête pour différentes tranches de temps en changeant les registres spéciaux
- 2 registres
  - CURRENT TEMPORAL SYSTEM_TIME ==> TIMESTAMP(12), nullable
    - DB2 ajoute implicitement "FOR SYSTEM_TIME AS OF CURRENT TEMPORAL SYSTEM_TIME"
  - CURRENT TEMPORAL BUSINESS_TIME ==> TIMESTAMP(12), nullable
    - DB2 ajoute implicitement "FOR BUSINESS_TIME AS OF CURRENT TEMPORAL BUSINESS_TIME"
  - Défaut = NULL: DB2 ne modifie pas les ordres SQL
- Paramètres du BIND pour déterminer si les registres spéciaux sont pris en compte
  - SYSTIMESENSITIVE (YES / NO)
  - BUSTIMESENSITIVE (YES / NO)

DB2 V11: Tables temporelles: support Audit

- Avec DB2 10, les tables temporelles système ont permis de connaitre:
  - **QUAND** les données étaient modifiées

- Avec DB2 V11 (PM99683), des informations utiles en cas d'Audit peuvent être obtenues
  - **QUI** a modifié les données
  - **QUELLE** action (type d'ordre SQL) a été effectuée

- Nouvelle syntaxe pour:
  - définir des colonnes "non-deterministic generated expression"
  - Ajouter une clause ON DELETE ADD EXTRA ROW qui est utilisée quand la table avec une période temporelle système contient des colonnes "non-deterministic generated expression"

DB2 V11: Tables temporelles et Audit: exemple

```
CREATE bank_account_stt
(account_no INT NOT NULL,
balance INT,
user_id VARCHAR(128) GENERATED ALWAYS AS (SESSION_USER),
op_code VARCHAR(1) GENERATED ALWAYS AS (DATA CHANGE OPERATION),
sys_start TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW BEGIN,
sys_end TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW END,
trans_id TIMESTAMP(12) GENERATED ALWAYS AS TRANSACTION START ID,
...
PERIOD SYSTEM_TIME(sys_start, sys_end));

CREATE bank_account_hist
(account_no INT NOT NULL,
balance INT,
user_id VARCHAR(128),
op_code VARCHAR(1),
sys_start TIMESTAMP(12) NOT NULL,
sys_end TIMESTAMP(12) NOT NULL,
trans_id TIMESTAMP(12),      ...);

ALTER TABLE bank_account_stt   ADD VERSIONING   USE HISTORY TABLE bank_account_hist
        ON DELETE ADD EXTRA ROW;
```

DB2 V11: Tables temporelles et Audit: exemple

- On 15 June 2010, user KWAN INSERT INTO BANK_ACCOUNT_STT (account_no, balance) VALUES (100000,1)
- After the insert:

account_no	balance	user_id	op_code	sys_start	sys_end
100000	1	KWAN	I	2010-06-15	9999-12-30

  – only the date portion of the TIMESTAMP(12) columns is shown.

  BANK_ACCOUNT_HIST Empty

- Later, on 1 December 2011, user HAAS uses the following statement to update the row:

  UPDATE BANK_ACCOUNT_STT SET balance = balance + 9 where account_no=100000;

account_no	balance	user_id	op_code	sys_start	sys_end
100000	10	HAAS	U	2011-12-01	9999-12-30

  BANK_ACCOUNT_HIST

account_no	balance	user_id	op_code	sys_start	sys_end
100000	1	KWAN	I	2010-06-15	2011-12-01

DB2 V11: Tables temporelles et Audit: exemple

- On 20 December 2013, user THOMPSON uses the following statement to delete the row:
  - DELETE FROM BANK_ACCOUNT_STT;
- This deletion results in the following data in the tables:
- BANK_ACCOUNT_STT Empty

account_no	balance	user_id	op_code	sys_start	sys_end

- BANK_ACCOUNT_HIST

account_no	balance	user_id	op_code	sys_start	sys_end
100000	1	KWAN	I	2010-06-15	2011-12-01
100000	10	HAAS	U	2011-12-01	9999-12-30
100000	10	THOMPSON	D	2013-12-20	2013-12-20

DB2 V11: Variables Globales

- Pour partager des informations entre ordres SQL

- Similaires aux registres spéciaux
- Maintenues par DB2 au delà des COMMIT/ROLLBACK
- Peut être créée et référencée dans la même unité de travail
  - Ne sera pas disponible globalement avant la fin de l'unité de travail
  - CREATE VARIABLE **Charge_Rate** DECIMAL(4,2) DEFAULT 0.00;
- Sa valeur peut être modifiée par:
  - SET, SELECT INTO, ou VALUES INTO
  - Un argument d'un parameter OUT ou INOUT dans un ordre CALL

- Exemple: Une procédure stockée peut utiliser une variable globale pour la renseigner et cette variable pourra ensuite être utilisée par une autre application dans un ordre SQL.

- Remarque: Le niveau de compatibilité doit être V11R1

DB2 V11: Variables Globales

- Exemple:

```
-- create a global variable
CREATE VARIABLE Charge_Rate DECIMAL(4,2) DEFAULT 0.00;
-- create a procedure that determines charge rate
CREATE PROCEDURE Loan_Charge_Rate (IN ID CHAR(5))
BEGIN
   ..
  SELECT SCORE INTO Cust_Score FROM CUSTOMER WHERE ACCOUNT = ID;
  IF Cust_Score = 'Good' THEN SET Charge_Rate = 1.0;
  ELSE SET Charge_Rate = 3.0;
END;
-- calling application
myApp: PROCEDURE(Buyer, Amount);
    ...
    CALL Loan_Charge_Rate(Buyer);
    UPDATE CUSTOMER SET BALANCE = BALANCE + (Amount * Charge_Rate);
END myAPP;
```

DB2 V11: Archivage transparent

- Archivage automatique des lignes supprimées via le positionnement d'une variable globale
- Les applications peuvent récupérer les lignes de la table de base uniquement ou bien également celles de la table Archive via le positionnement d'une variable globale
  - DB2 convertit automatiquement le SQL avec UNION ALL
- Activation via CREATE ou ALTER TABLE…ENABLE ARCHIVE
- Les Tables Temporelles et à Archivage transparent sont mutuellement exclusives
- Remarque: envisager le REORG DISCARD pour le premier archivage

DB2 V11: Archivage transparent

Table de Base

```
CREATE TABLE POLICY_BASE
(EMPL VARCHAR(4) NOT NULL,
 TYPE VARCHAR(4),
 PLCY VARCHAR(4) NOT NULL,
 COPAY VARCHAR(4),
 START_DATE   DATE NOT NULL,
 TIMESTAMP1   TIMESTAMP NOT NULL GENERATED ALWAYS
   FOR EACH ROW ON UPDATE AS ROW CHANGE TIMESTAMP,
 PRIMARY KEY (EMPL,PLCY));
```

Table Archive

```
CREATE TABLE POLICY_ARCHIVE
(EMPL VARCHAR(4) NOT NULL,
 TYPE VARCHAR(4),
 PLCY VARCHAR(4) NOT NULL,
 COPAY VARCHAR(4),
 START_DATE   DATE NOT NULL,
 TIMESTAMP1 TIMESTAMP NOT NULL GENERATED ALWAYS
   FOR EACH ROW ON UPDATE AS ROW CHANGE TIMESTAMP,
 PRIMARY KEY (EMPL,PLCY));
```

OU

```
CREATE TABLE POLICY_ARCHIVE
  LIKE POLICY_BASE
  INCLUDING ROW CHANGE TIMESTAMP;
```

Activer l'archivage

```
ALTER TABLE POLICY_BASE ENABLE ARCHIVE USE POLICY_ARCHIVE;
```

* Créer la table de base

  avec colonne TIMESTAMP NOT NULL GENERATED ALWAYS FOR EACH ROW ON UPDATE AS ROW CHANGE TIMESTAMP

* Créer la table archive (TYPE=R)

  avec colonne TIMESTAMP NOT NULL GENERATED ALWAYS FOR EACH ROW ON UPDATE AS ROW CHANGE TIMESTAMP

* Indiquer l'association entre les 2 tables

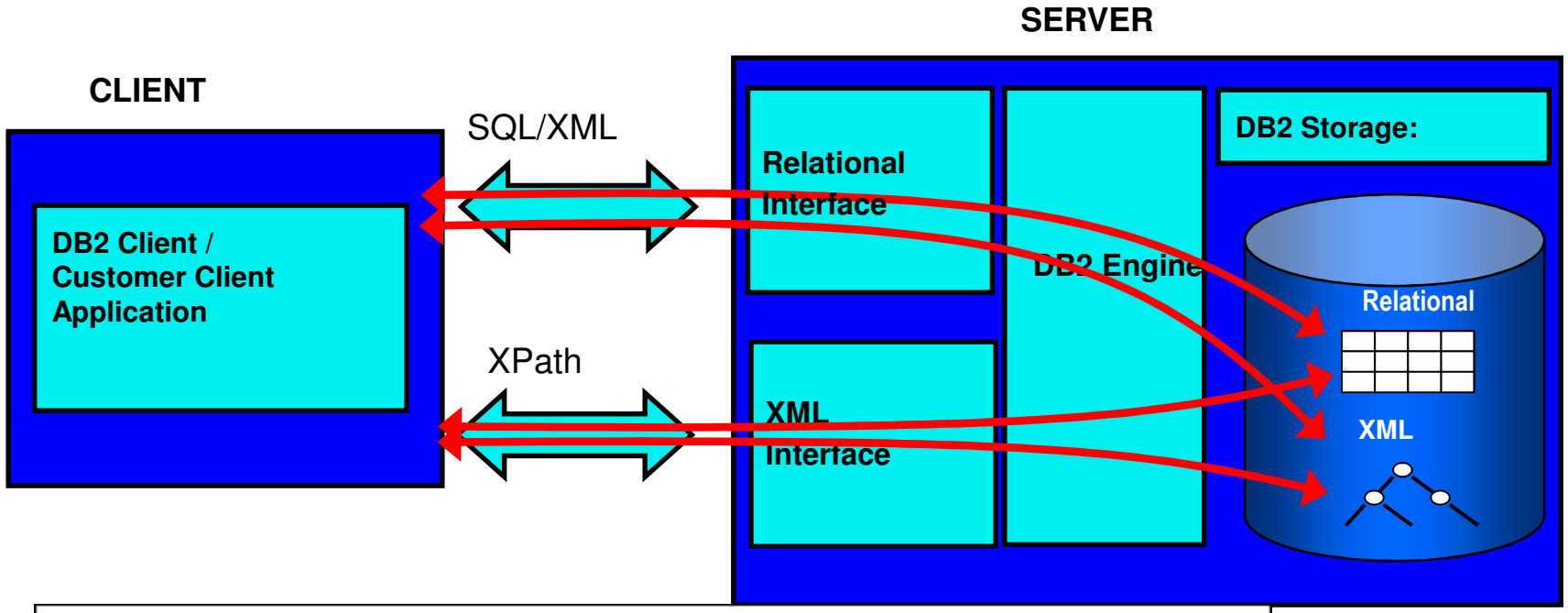
DB2 V11: Archivage transparent

- Variables globales
  - SYSIBMADM.GET_ARCHIVE
    - Détermine si les SELECTs sur la table de base accèdent automatiquement (via UNION ALL) à la table archive associée
    - 'Y' ➔ UNION ALL
    - Les Packages doivent être bindés avec ARCHIVESENSITVE(YES)
  - SYSIBMADM.MOVE_TO_ARCHIVE
    - Détermine si les lignes supprimées (DELETE) de la table de base sont insérées dans la table archive associée
    - 'Y': INSERT et UPDATE pas autorisés sur la table de base
    - 'E': INSERT et UPDATE autorisés sur la table de base

- Restrictions
  - LOAD REPLACE pas autorisé
  - On ne peut pas faire de ALTER, RENAME, DROP sur les colonnes
  - Pas de table CLONE

DB2 V9: XML natif dans DB2

SQL et XML sur un pied d'égalité




```
CREATE TABLE PURCHASEORDER (
    POID       BIGINT NOT NULL,
    STATUS     VARCHAR(10) NOT NULL WITH DEFAULT 'UNSHIPPED',
    CUSTID     BIGINT,
    ORDERDATE DATE,
    PORDER     XML )
PARTITION BY RANGE
(POID ASC)
(PARTITION 1 ENDING ( 00223344 ) ,
 PARTITION 2 ENDING ( 11223344 ) ,
 PARTITION 3 ENDING ( 22334455) )
IN XMLDB.PORDER ;
```

The

DB2 V9: XMLQUERY & XMLEXISTS

POID	...	CUSTID	PORDER
11223344	...	1002	<PurchaseOrder PoNum="11223344" OrderDate="2007-07-18" Status="Unshipped"> ... <items> <item> <partid>100-100-01</partid> <name>Snow Shovel, Basic 22 inch</name> ... </item> <item> <partid>100-103-01</partid> <name>Snow Shovel, Super Deluxe 26 inch</name> ... </item> ... </PurchaseOrder>
11223345	...	1002	<PurchaseOrder PoNum="11223345" OrderDate="2007-07-18" Status="Unshipped"> ... <items> <item> <partid>100-100-01</partid> <name>Snow Shovel, Basic 22 inch</name> <quantity>4</quantity> <price>9.99</price> </item>

1. Evaluate all rows according to XMLEXISTS condition

2. select matching rows

3. Select columns to return and evaluate XMLQUERY function

Resultset:

POID	ITEMS
11223344	<name>Snow Shovel, Basic 22 inch</name><name>Snow Shovel, Super Deluxe 26 inch</name>

```
SELECT POID, XMLQUERY('/PurchaseOrder/items/item/name' PASSING PORDER) AS ITEMS
from purchaseorder where XMLEXISTS('/PurchaseOrder[@PoNum=11223344]' PASSING PORDER)
```

DB2 V9: autres fonctions XML

Table PURCHASEORDER

OrderNr	Item	Quantity	Price
11223344	Snow Shovel, Basic 22 inch	4	9.99
11223344	Snow Shovel, Super Deluxe 26 inch	2	49.99

```
SELECT (XMLELEMENT ( name "PurchaseOrder",
    XMLATTRIBUTES ( ORDERNR AS "PoNum" ),
    XMLAGG (
        XMLELEMENT ( NAME "item", ITEM)
        ORDER BY ITEM) ))
FROM PURCHASEORDER GROUP BY ORDERNR
```

Pour créer un document XML

```
<PurchaseOrder PoNum="11223344">
    <item>Snow Shovel, Basic 22 inch<item>
    <item>Snow Shovel, Super Deluxe 26 inch<item>
</PurchaseOrder>
```

DB2 V9: Quelques fonctions SQL/XML

- XMLCAST: transforme (cast) une valeur en type XML en valeur type VARCHAR pour pouvoir faire des comparaisons
- XMLTABLE: insère des valeurs d'un document XML dans une table DB2
- XMLELEMENT: crée un élément XML
- XMLATTRIBUTES: utilisé à l'intérieur de XMLELEMENT pour créer des attributs
- XMLCONCAT: concatène une liste de valeurs XML
- XMLAGG: groupe un ensemble de lignes XML

DB2 V10: XMLMODIFY

Pour mettre à jour uniquement les données nécessaires dans le XML

```
UPDATE customer
SET info = XMLMODIFY( '
replace value of node /customerinfo/addr/zipcode with “90111” ' )
WHERE cid = 1000;
```

```
<customerinfo>
    <name>John Smith</name>
    <addr country=“Canada">
        <street>Fourth</street>
        <city>Calgary</city>
        <state>Alberta</state>
        <zipcode>M1T 2A9</zipcode>
    </addr>
    <phone type="work">963-289-
4136</phone>
</customerinfo>
```

```
<customerinfo>
    <name>John Smith</name>
    <addr country=“Canada">
        <street>Fourth</street>
        <city>Calgary</city>
        <state>Alberta</state>
        <zipcode>90111</zipcode>
    </addr>
    <phone type="work">963-289-
4136</phone>
</customerinfo>
```

DB2 V11: XQUERY

- FLOWR expressions
  - Rétrofitté en V10
  - Donne aux développeurs plus de facilité à écrire des applications XML dans un langage familier

1) Create table

```
CREATE TABLE ORDERS (ORDERXML XML);
```

2) Insert XML document into table

```
INSERT INTO ORDERS (ORDERXML) VALUES
 ('<order orderDate="2011-05-18">
     <shipTo country="US"> </shipTo>
     <billTo country="US"> </billTo>
     <items>
       <item partno="872-AA">
         <quantity>2</quantity>
         <price>39.99</price> </item>
       <item partno="926-AA">
         <quantity>2</quantity>
         <price>74.99</price></item>
       <item partno="945-ZG">
         <quantity>1</quantity>
         <price>178.99</price></item>
     </items>
   </order>');
```

3) Select items, part numbers and amount for orders whose items are ordered in an amount greater than $100

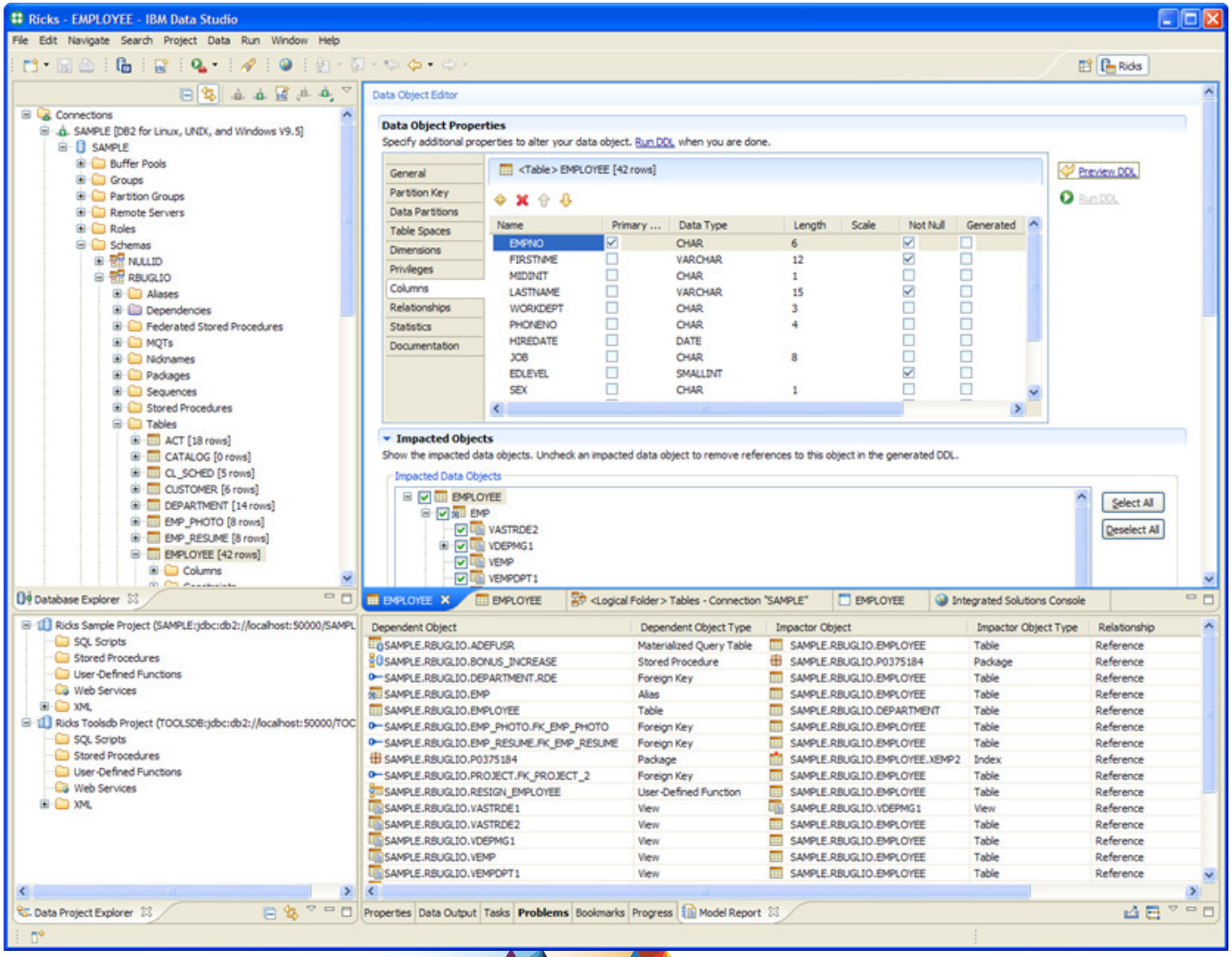
```
SELECT XMLQUERY(
  'for $i in $O/order/items/item
   let $p:=$i/price, $q:=$i/quantity,
       $amount:=$p * $q
   where $amount >100
   order by $amount
   return <item partno="{$i/@partno}">
            {$amount}
          </item>'
PASSING ORDERS.ORDERXML as "O")
FROM ORDERS;
```

4) Output from FLOWR expression

```
<item partno="926-AA">149.98</item>
<item partno="945-ZG">178.99</item>
```

IBM Data Studio

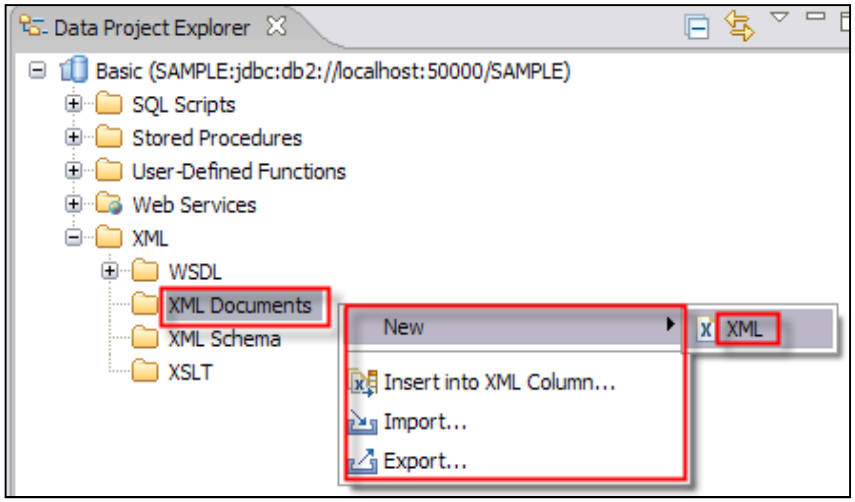
A Consistent and Productive work environment

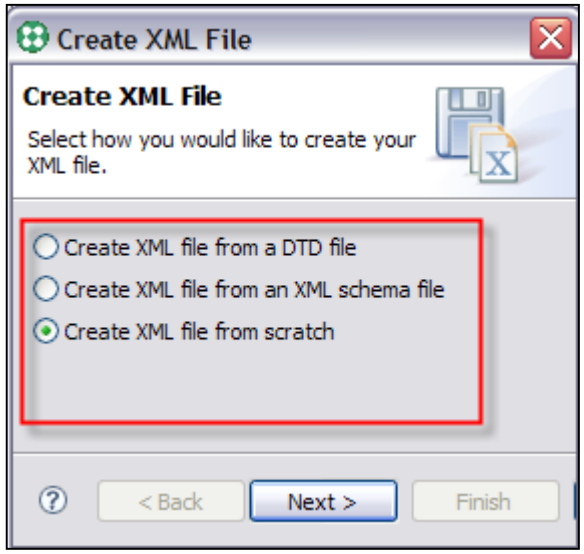


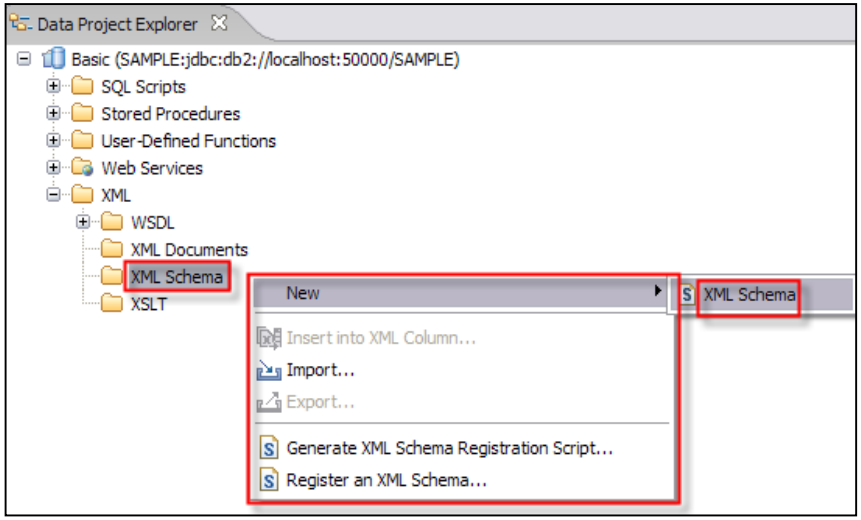
The next step in big data starts with IBM.

Data Studio: *XML Management*

XML Editors







- Easily perform all your XML needs
  - Create XML documents and schemas
  - Import and export XML documents and schemas
  - Insert XML documents into XML columns
  - Generate XML Schema registration scripts
  - Register XML Schema

Extensive XML support and capabilities

DB2 11 Resources

- DB2 11 Announcement
  - http://www-01.ibm.com/common/ssi/cgi-bin/ssialias?infotype=AN&subtype=CA&htmlfid=897/ENUS213-376&appname=USN
- DB2 11 Technical Overview Redbook (SG24-8180)
  - http://www.redbooks.ibm.com/abstracts/sg248180.html?Open
- DB2 11 Performance Topics Redbook (SG24-8222)
  - http://www.redbooks.ibm.com/abstracts/sg248222.html?Open.
- DB2 11 links: https://www.ibm.com/software/data/db2/zos/family/db211/
  - Case study: BMW Group
  - Case study: JN Data
  - Whitepaper: "DB2 11 for z/OS: Unmatched Efficiency for Big Data and Analytics"
  - Whitepaper: "How DB2 11 for z/OS Can Help Reduce Total Cost of Ownership"
  - Free eBook available for download
    - http://ibm.co/160vQgM

Merci !

The next step in big data starts with IBM.

IBM