Q & A (vol.10)

USER ID와 PASSWORD를 지정하는 CONNECT문 ; ESQL/C

Q1 **데이터베이스에 CONNECT 할 때 사용자와 패스워드를 미리 지정하여 해당 데이터베이스에 접속하는 사용자를 제한하고 싶습니다. CONNECT 문에 USER와 PASSWORD를 줄 수 있는 걸로 알고 있는데 어떻게 사용해야 합니까?**

A CONNECT 문에서 USING 절을 사용하여 PASSWORD를 지정할 수 있습니다. 아래의 sample program을 참고하십시오.

이 sample program은 $InformixDIR/demo/esqlc/demo1.ec 파일을 수정하여 작성한 것입니다.

```
#include



EXEC SQL define FNAME_LEN         15;

EXEC SQL define LNAME_LEN         15;



main()

{
```

```
EXEC SQL BEGIN DECLARE SECTION;

    char fname[ FNAME_LEN + 1 ];

    char lname[ LNAME_LEN + 1 ];

    char username[15];

    char passwd[15];

EXEC SQL END DECLARE SECTION;



printf( "DEMO1 Sample ESQL Program running.₩n₩n");

EXEC SQL WHENEVER ERROR STOP;

printf("User Name : ");

scanf( "%s",&username);

printf("Password  : ");

scanf( "%s",&passwd);
```

```
EXEC SQL connect to 'stores7' user :username

using :passwd;



EXEC SQL declare democursor cursor for

    select fname, lname

        into :fname, :lname

        from customer

        where lname < "C";



EXEC SQL open democursor;

for (;;)

    {

      EXEC SQL fetch democursor;

      if (strncmp(SQLSTATE, "00", 2) != 0)
```

```
            break;



        printf("%s %s₩n",fname, lname);

    }

  if (strncmp(SQLSTATE, "02", 2) != 0)

      printf("SQLSTATE after fetch is %s₩n", SQLSTATE);



  EXEC SQL close democursor;

  EXEC SQL free democursor;

  EXEC SQL disconnect current;

  printf("₩nDEMO1 Sample Program over.₩n₩n");

}
```

9.2x에서 데이터베이스명, 테이블명의 최대 길이

Q2 IDS 2000을 사용하고 있는데, 18 바이트까지만 사용 가능했던 이전 버전과는 달리 허용 가능한 데이터베이스명, 테이블명이 꽤 길어졌다고 들었습니다. 몇 글자까지 사용할 수 있습니까?

A IDS 2000은 Informix Dynamic Server 2000 Version 9.2x 로 Informix의 최신 버전이라 할 수 있습니다.

질문의 내용처럼 이전 버전(IDS 7.x, IDS 9.1x)에서는 데이터베이스명, 테이블명을 18 바이트까지만 사용 가능했었습니다. 이는 데이터베이스와 테이블에 관한 정보를 저장하는 system catalog인 sysdatabases와 systables의 schema가 18바이트로 지정되어 있기 때문입니다. 그러나 IDS 2000에서는 이에 관련된 내용이 확장되어서 128바이트까지 사용 가능해졌습니다.

다음은 7.24.UC7에서 sysdatabases와 systables의 각 column의 정보와, IDS 2000(9.20.UC1)에서의 sysdatabases와 systables의 각 column의 정보를 질의한 내용입니다.

먼저 7.24.UC7의 내용입니다.

```
kor-k400:/bigfs> onstat -

Informix OnLine Version 7.24.UC7 -- On-
Line -- Up 20:02:26 -- 17808

Kbyteskor-k400:/bigfs> dbaccess - -
> database sysmaster;
Database selected.

> info columns for sysdatabases;
```

Column	Type	Nulls

Column name	Type	Nulls
name	**char(18)**	**yes**
partnum	integer	yes
owner	char(8)	yes
created	date	yes
is_logging	integer	yes
is_buff_log	integer	yes
is_ansi	integer	yes
is_nls	integer	yes
flags	smallint	yes

> info columns for systables;

Column name	Type	Nulls
tabname	**char(18)**	**yes**
owner	char(8)	yes
partnum	integer	yes
tabid	serial	yes
rowsize	smallint	yes
ncols	smallint	yes
nindexes	smallint	yes
nrows	integer	yes
created	date	yes
version	integer	yes
tabtype	char(1)	yes
locklevel	char(1)	yes

npused	integer	yes
fextsize	integer	yes
nextsize	integer	yes
flags	smallint	yes
site	char(18)	yes
dbname	char(18)	yes

다음은 9.20.UC1의 내용입니다.

```
/CS1/byrhee> onstat -

Informix Dynamic Server 2000 Version 9.20.UC1 -- On-
Line -- Up 00:29:01 -- 9216 Kbytes

/CS1/byrhee> dbaccess - -
> database sysmaster;
Database selected.

> info columns for sysdatabases;
```


is_buff_log	integer	yes
is_ansi	integer	yes
is_nls	integer	yes
flags	smallint	yes
> info columns for systables;		
Column name	Type	Nulls
tabname	**varchar(128,0)**	**yes**
owner	char(32)	yes
partnum	integer	yes
tabid	serial	yes
rowsize	smallint	yes
ncols	smallint	yes
nindexes	smallint	yes
nrows	integer	yes
created	date	yes
version	integer	yes
tabtype	char(1)	yes
locklevel	char(1)	yes
npused	integer	yes
fextsize	integer	yes
nextsize	integer	yes
flags	smallint	yes
site	varchar(128,0)	yes
dbname	varchar(128,0)	yes
type_xid	integer	yes

am_id	integer	yes

위에서 각각 강조 처리된 부분의 내용을 참고하십시오.

데이터베이스 이름을 저장하게 되는 sysdatabases의 name 칼럼이 CHAR(18)에서 CHAR(128)로 바뀌었고, 테이블 이름을 저장하게 되는 systables의 tabname 칼럼은 CHAR(18)에서 VARCHAR(128,0)으로 변경되었습니다. 따라서 데이터베이스명과 테이블명이 128 바이트까지 가능해졌습니다.

실제로 128 바이트 길이로 테이블을 만든 후 dbaccess에서 테이블 정보를 확인해 보면 다음과 같이 표시되는 것을 볼 수 있을 것입니다.

```
/CS1/byrhee> dbaccess



DBACCESS:  Query-language  Connection  Database  Table  Session  Exit

Use SQL query language.

---------------------------------- Press CTRL-W for Help --------



SQL:  New  Run  Modify  Use-editor  Output  Choose  Save  Info  Drop  Exit

Gives information on tables in the database.
```

-------- stores_demo@byrhee_920t -------- Press CTRL-W for Help ------

INFO FOR TABLE >>

Choose a table with the Arrow Keys, or enter a name, then press Return.

-------- stores_demo@byrhee_920t -------- Press CTRL-W for Help ------

abc name

abcdefghijklmnopq+ orders

addr sdjflaksjdlfjalsk+

call_type state

catalog stock

cust_calls

```
customer

items

manufact
```

dbaccess를 실행하여 테이블 정보를 확인할 수 있는 "Info" 메뉴를 선택하면 위와 같이 테이블명이 긴 경우에는 마지막 부분에 "+" 기호가 생김을 확인할 수 있을 것입니다.

따라서 긴 테이블명을 사용하고자 할 경우, 물론 127 바이트가 같고 마지막 1 바이트만 달라도 서로 다른 테이블로 생성이 되지만, 위와 같이 화면에 표시되는 것은 18 바이트이므로 앞에서 18 바이트 중 어느 하나를 다르게 하는 것이 구분하기에는 좋을 것입니다.

Stored Procedure에서 트랜잭션 처리 방법

Q3 Stored procedure에서 트랜잭션 처리를 하고 싶습니다. 즉, 특정 작업을 트랜잭션 처리하여 만일 오류가 발생하면 rollback할 수 있도록 하고자 합니다. stored procedure를 작성하는 방법을 알고 싶습니다.

```
                    mem_count smallint,

                    primary key(code) constraint pk code);
```

code	department	mem_count
001	Technical	0
002	Sales	0
003	Finance	0
004	Manufacturing	0
005	Admin	0

```
create table emp(num serial,

                    name char(10),

                    depart_code char(3),

                    primary key(num) constraint pk_num,
```

```
                    foreign key(depart_code)

                        references depart constraint fk_code);



create procedure trans_test(v_name char(10),v_code char(3))

  begin

      on exception

        rollback work;

      end exception;



  begin work;

      insert into emp

            values(0,v_name,v_code);

      update depart
```

```
                    set mem_count=mem_count+1

                    where code=v_code;

    commit work;

    end

end procedure;
```

depart 테이블에 위와 같이 데이터가 들어가 있다고 가정할 때, emp 테이블에는 depart_code가 foreign key로 지정되어 있으므로 depart 테이블에 들어있는 값 외에는 입력되지 않습니다.

```
execute procedure trans_test("byrhee","001");
execute procedure trans_test("wghwang","003");
execute procedure trans_test("shlee","006");
```

위와 같이 sotred procedure를 실행한 후 각 테이블의 내용을 질의해 보십시오. 마지막 execute문은 depart 테이블에 depart_code가 "006"인 내용은 없으므로 rollback하여 실제로 insert되지 않았음을 확인할 수 있을 것입니다.

1	byrhee	001
2	wghwang	003

select * from depart;

code	department	mem_count
001	Technical	1
002	Sales	0
003	Finance	1
004	Manufacturing	0
005	Admin	0

CGI 프로그램의 CONNECT 오류

Q4 **Linux에서 CGI(in C)를 웹에서 불러 데이터베이스에 접속하여 해당 데이터를 보여주려고 합니다. 그런데 CONNECT가 잘 되질 않습니다.**

실제로 어플리케이션 안에서 데이터베이스에 CONNENCT하는 문장은 다음과 같이 구현하였습니다.

```
EXEC SQL BEGIN declare section;

    char ccc[256];

EXEC SQL END declare section;
```

```
EXEC SQL connect to test_db@linux;



EXEC SQL declare cursor_list cursor for

select distinct aa into :ccc from test;



printf(" <select size=1 name=₩'AA₩' onchange=₩'selecting()₩'> ₩n");



EXEC SQL open cursor_list;

for (;;)

{

    EXEC SQL fetch cursor_list;

            if (strncmp(SQLSTATE, "00", 2) != 0)
```

```
                    break;

            printf(" <option value=₩'%s₩'>%s</option>₩n",ccc,ccc);

    }



            printf(" </select>₩n");

    EXEC SQL close cursor_list;

    EXEC SQL free curosr_list;



    EXEC SQL disconnect current;
```

웹 브라우저가 아니라 UNIX command line에서 실행하면 잘 됩니다. 그러나 웹 상에서 실행하여 보면 전혀 데이터를 가져오지 못합니다. 왜 이런 현상이 일어나는지 궁금합니다.

다른 환경설정 같은 것이 필요한 것인지, 그렇다면 어떤 것을 주어야 하는지 알고 싶습니다.

A 이유는 간단합니다. 환경이 제대로 설정이 되지 않은 것이지요. UNIX command line에서는 실행이 된다고 하셨는데, 이는 login 하면서 InformixDIR, InformixSERVER, PATH 등 Informix 관련 환경변수들이 설정이 되었기 때문에 가능한 것입니다. 그러나 웹 상에서 실행할 때에는 일반적으로 사용자 계정이 "NOBODY"이기 때문에 Informix 관련 환경변수가 해당 사용자에게는 설정되지 않습니다.

따라서 CGI 프로그램 내에 putenv() 함수를 사용하여 Informix 관련 환경변수들과 기타 필요한 환경변수들을 설정해야만 합니다. InformixDIR, InformixSERVER는 반드시 필요한 환경변수이며, LOCALE 관련 변수와 ONCONFIG가 설정되어 있다면 이 변수들도 모두 설정하여 주면 됩니다.

다음의 예를 참고하십시오.

```
putenv("InformixSERVER=maserv_tcp");
putenv("InformixDIR=/bigfs/IDS724UC7");
putenv("DB_LOCALE=ko_kr.ksc");
putenv("SERVER_LOCALE=ko_kr.ksc");
putenv("CLIENT_LOCALE=ko_kr.ksc");
putenv("ONCONFIG=onconfig.csdb");
```

임시 테이블과 temp dbspace

Q5 Temp dbspace를 만든 후 $ONCONFIG 파일에 등록하고 DB server를 재가동했습니다. 그런 다음 임시 테이블을 만들었는데, 이 temp dbspace를 사용하지 않는 것 같습니다. Temp dbspace를 사용하지 못하는 이유가 무엇인지요?

- 또는 hash 조인이 사용되면서 모든 데이터를 공유 메모리에 올려 놓고 작업할 수 없는 환경에서 hash 테이블을 생성하는 경우,
- 색인 만들 때 생성되며,
- warm restore시에 논리 로그 파일 역시 임시 파일로 생성됩니다.

임시 테이블은

- SELECT문에 INTO TEMP 절을 사용할 때나,
- CREATE TEMP TABLE문으로 임시 테이블을 생성할 때,
- 그리고 BLOB 데이터가 애플리케이션으로부터 DB Server에 있는 내장 프로시져로 전달될 때와 광역 BLOB 데이터 변수가 선언되고 값이 할당될 때 임시테이블이 생성됩니다.

이러한 임시 파일과 임시 테이블은 사용되는 영역에도 차이가 있습니다.

우선 임시 파일은

- PSORT_DBTEMP 환경변수 (한 개 이상의 디렉토리)
- DBSPACETEMP 환경변수 (한 개 이상의 dbspace)
- DBSPACETEMP 구성 매개변수 ($ONCONFIG 파일 ; 한 개 이상의 dbspace)
- /tmp
-

의 순으로 사용됩니다.

또한 임시 테이블을 위한 공간으로는

- DBSPACETEMP 환경변수 (한 개 이상의 dbspace)
- DBSPACETEMP 구성 매개변수 ($ONCONFIG 파일 ; 한 개 이상의 dbspace)
- ROOT dbspace 또는 해당 데이터베이스가 생성된 dbspace 의 순으로 사용됩니다.

그러나 임시 테이블의 경우, 해당 데이터베이스가 로깅되어 있는지 아닌지에 따라 임시 테이블이 생성되는 위치에 차이가 있습니다. 즉, 해당 데이터베이스가 로깅되지 않고 DBSPACETEMP가 지정되어 있다면, 임시 테이블은 DBSPACETEMP에 지정된 영역에 생성됩니다. 반면 해당 데이터베이스가 로깅되어 있고 DBSPACETEMP가 지정되어 있다면, 임시테이블은 ROOT dbspace에 생성됩니다. 데이터베이스가 로깅된다면 임시 테이블 역시 로깅되기 때문입니다. 이 경우 ROOT dbspace에 생성되는 것을 방지하기 위해서 WITH NO LOG을 사용하여 임시 테이블은 로깅하지 않도록 할 수도 있습니다.

예를 들어, DBSPACETEMP 환경변수가 tempdbs로 설정되어 있고 로깅한 데이터베이스에 임시 테이블을 만든다고 가정한다면 아래의 문장에 따라 저장되는 영역이 달라집니다.

$)env

InformixSERVER=byrhee_731t

DBSPACETEMP=tempdbs

ONCONFIG=onconfig.byrhee

InformixDIR=/CS1/IDS731UC6

PATH=/CS1/IDS731UC6/bin:.:/usr/ccs/bin ... ...

... ...

/CS1/byrhee> onstat -d i 데이터베이스를 생성하지 않은 초기의 디스크 정보

Informix Dynamic Server Version 7.31.UC6-- On-Line -- Up 00:15:40 -- 8896 Kbytes

Dbspaces

address	number	flags	fchunk	nchunks	flags	owner	name

a12a5e8	3	2001	3	1	N T	Informix	tempdbs

3 active, 2047 maximum

Chunks

address	chk/dbs	offset	size	free	bpages	flags	pathname
a12a210	1	1	0	10000	6705	PO-	/CS1/byrhee/rootdbs731
a12a368	2	2	0	5000	4947	PO-	/CS1/byrhee/dbsp1
a12a448	3	3	0	1500	1447	PO-	/CS1/byrhee/tempdbs

3 active, 2047 maximum

/CS1/byrhee> dbaccessdemo7 -dbspace dbsp1 -log

DBACCESS Demonstration Database Installation Script

Dropping existing stores7 database ....

Creating stores7 database ....

Database created.

... ...

-> stores7이라는 데모 데이터베이스를 dbsp1에 unbeffered 로깅 모드로 만들어 주는

명령으로 이 명령을 실행하면 stores7에 관련된 테이블, 색인이 dbsp1에 생성됩니다.

/CS1/byrhee> onstat -d	<-	데모 데이터베이스를 dbsp1에 생성한 후의 디스크 정보, dbsp1의 크기가 감소

Informix Dynamic Server Version 7.31.UC6-- On-Line -- Up 00:15:40 -- 8896 Kbytes

Informix Dynamic Server Version 7.31.UC6-- On-Line -- Up 00:15:40 -- 8896 Kb

address	number	flags	fchunk	nchunks	flags	owner	name
a12a150	1	1	1	1	N	Informix	rootdbs
a12a528	2	1001	2	1	N	Informix	dbsp1
a12a5e8	3	2001	3	1	N T	Informix	tempdbs

3 active, 2047 maximum

Chunks

address	chk/dbs	offset	size	free	bpages	flags	pathname
a12a210	1	1	0	10000	6697	PO-	/CS1/byrhee/rootdbs731
a12a368	2	2	0	5000	4603	PO-	/CS1/byrhee/dbsp1
a12a448	3	3	0	1500	1447	PO-	/CS1/byrhee/tempdbs

3 active, 2047 maximum

> create temp table tp2 (col1 smallint, col2 char(10));Table created.

/CS1/byrhee> onstat -d	<-	위 문장을 사용하여 임시 테이블 tp2를 생성한 후의 디스크 정보, dbsp1의 크기가 감소

Informix Dynamic Server Version 7.31.UC6-- On-Line -- Up 00:15:40 -- 8896 Kbytes

Dbspaces

address	number	flags	fchunk	nchunks	flags	owner	name
a12a150	1	1	1	1	N	Informix	rootdbs
a12a528	2	1001	2	1	N	Informix	dbsp1
a12a5e8	3	2001	3	1	N T	Informix	tempdbs

3 active, 2047 maximum

Chunks

address	chk/dbs	offset	size	free	bpages	flags	pathname

a12a210	1	1	0	10000	6697	PO-	/CS1/byrhee/rootdbs731
a12a368	2	2	0	5000	4595	PO-	/CS1/byrhee/dbsp1
a12a448	3	3	0	1500	1439	PO-	/CS1/byrhee/tempdbs

3 active, 2047 maximum

> create temp table tp4 (col1 smallint, col2 char(10)) with no log;Temporary table created.

/CS1/byrhee> onstat -d	<-	위 문장을 사용하여 임시 테이블 tp4를 생성한 후의 디스크 정보, tempdbs의 크기가 감소

Informix Dynamic Server Version 7.31.UC6-- On-Line -- Up 00:15:40 -- 8896 Kbytes

Dbspaces

address	number	flags	fchunk	nchunks	flags	owner	name
a12a150	1	1	1	1	N	Informix	rootdbs
a12a528	2	1001	2	1	N	Informix	dbsp1
a12a5e8	3	2001	3	1	N T	Informix	tempdbs

3 active, 2047 maximum

Chunks

address	chk/dbs	offset	size	free	bpages	flags	pathname
a12a210	1	1	0	10000	6697	PO-	/CS1/byrhee/rootdbs731
a12a368	2	2	0	5000	4595	PO-	/CS1/byrhee/dbsp1
a12a448	3	3	0	1500	1431	PO-	/CS1/byrhee/tempdbs

3 active, 2047 maximum

위의 예에서 생성되는 임시 테이블에 따라 사용되는 dbspace가 다르고, 또한 테이블의 기본 익스텐트 크기가 8 페이지이므로 각각의 dbspace의 크기가 8페이지씩 사용되어 "free" 사이즈가 8페이지씩 감소되어지는 것을 볼 수 있을 것입니다.

따라서 질문의 내용처럼, DBSPACETEMP가 지정되어 있는데도 그 지정된 영역을 사용하지 않는다면 로깅된 데이터베이스이기 때문일 것입니다.

위에서 살펴본 것처럼 임시 파일이나 임시 테이블 모두를 위해서는 DBSPACETEMP와 같이 임시 저장 공간이 있어야 합니다. 이러한 임시 자장 공간을 특별히 지정해 주지 않으면 /tmp를 사용하거나 ROOT dbspace를 사용하여 성능에 악영향을 미칠 수도 있습니다. 따라서 I/O의 효율을 높이기 위해서는 일반적으로 ROOT dbspace와 사용자 DB 영역인 일반 dbspace, 그리고 temp dbspace로 나누어 사용하도록 하는 것이 좋습니다. 또한 로깅된 데이터베이스를 사용하는 경우에 임시 테이블을 생성한다면 WITH NO LOG를 주어 임시 테이블을 로깅하지 않도록 하는 것도 중요하다고 할 수 있을 것입니다.