3장. DBMS 관리

3.1 백업 및 복구 (Backup & Recovery)

3.1.1 Logging

3.1.2 User Exit

3.1.3 Backup

3.1.4 Recovery

3.2 데이터 이동 (Data Moving)

3.2.1 Export

3.2.2 Import

3.2.3 Load

3.3 REORG & RUNSTATS

3.3.1 REORGCHK

3.3.2 REORG

3.3.3 RUNSTATS

3.1 Backup & Recovery

3.1.1 Logging

(1) 기본용어

가. Transaction

여러개의 SQL문으로 구성되는 기본적인 작업단위로 Unit of Work(UOW)이라고도 한다. 한 UOW내의 SQL문들은 UOW종료시까지 그 실행이 보류되었다가 종료시에 일괄적으로 Database에 반영되거나 혹은 취소된다. 즉, 한 UOW내의 SQL문들은 전부 성공적으로 수행되거나, 혹은 모두 그 실행이 취소되어야 한다. UOW의 종료를 표시하는 시점을 Consistency Point라고 한다.

나. Commit

UOW내의 모든 SQL문장이 처리했던 내용을 Database에 반영한 후 UOW를 종료한다.

다. Rollback

UOW내의 모든 SQL문장의 처리 내용을 취소한 후 UOW를 종료한다.

라. Log

UOW내의 모든 SQL문장이 처리했던 내용을 Database에 실제로 반영하기 전에 그 변경 전후의 Image를 임시로 기록해 두는 File

마. Recoverable

Database Recovery에 Log file이 적용될 수 있는 경우를 Recoverable Database라고하고 그렇지 못한 경우를 Non-Recoverable Database라고 한다.

(2) Log의 종류

가. Active Log : Log file내에 아직 database에 반영되지 않은 정보가 있는 log
나. Inactive Log : Log file내의 모든 정보가 Database에 반영된 log
다. On-line Archived Log : Inactiva시 Database log Directory에 저장되는 Log
라. Off-line Archived Log : Inactive시 Database log Directory 이외의 Directory에 저장되는 log

(3) Logging 방식

비교항목	Circular Logging	Archival Logging
정의	Inactive log를 재사용하는 Logging 방법	Inactive log를 재사용 하지 않고 보관해 둠으로써 Recovery에 log를 적용시키는 logging 방법
설정방법	logretain off 이고 userexit off 상태임 (Database Configuration)	logretain on이고 userexit on or off (Database Configuration)
Log file 할당	preallocate	dynamic allocate
Inactive log 처	Reuse	Archive
Log full 조건	activr log = logprimary + logsecond	active log = logprimary
연관 parameter	logprimary / logsecond	logprimary / loghead / lognext
Backup 방법	Off Line Full Database	Off Line / On Line Full Database / Tablespace
Recovery 방법	Crash / Restore	Crash / Restore / Rollforward

(4) Log File의 관리

가. Log file은 S0000000.LOG에서 S9999999.LOG까지 순차적으로 사용된다.

나. Logging 방법이 Archival에서 Circular로, 또는 Circular에서 Archival로 변경되면 Log file은 S0000000.LOG에서부터 다시 사용된다.

다. 10,000,000개의 log file을 다 사용한 경우에는 S0000000.LOG에서부터 다시 사용된다.

라. Recovery에 log file이 적용되면 적용된 마지막 log file은 Truncate되고, 다음 번호의 log file부터 다시 사용된다.

마. Archive Log를 delete하면 Error

(5) Logging방식 변경

가. Circular Logging으로 설정

```
$ db2 update db cfg db명 using logretain off
$ db2 update db cfg db명 using userexit off
```

나. Archival Logging으로 설정

```
$ db2 update db cfg db명 using logretain on
$ db2 update db cfg db명 using userexit on
  or db2 update db cfg db명 using userexit off
```

다. DataBase Directory 확인

```
$ db2 get db directory on $HOME
 Local Database Directory on /home/db2inst1
 Number of entries in the directory = 1
 Database 1 entry :
 Database alias          = SAMPLE
 Database name           = SAMPLE
 Database directory      = SQL00001
 Database relese level= 6.00
 Commit          =
 Directory entry type  = Home
```

라. DataBase Log file 확인

```
$ Is - lia $HOME/db2inst1/SQL00001/SQLLOGDIR
```

3.1.2 User Exit

가. Archival logging 방식을 취하려면 Logretain변수를 이용하는데, Userexit변수를 함께 쓰면 Archival file을 다른 device로 이동시킬 수 있다.

나. Database Configuration Parameter 조정

```
$ db2 update db cfg db명 using logprimary 10
$ db2 update db cfg db명 using logretain on
$ db2 update db cfg db명 using userexit on
```

다. db2uexit Program 실행 module 생성

- $HOME/sqllib/samples/c/db2uexit.cdisk를 작업 Directory에 db2uexit.c로 복사한다.
- db2uexit.c file을 적당하게 수정한다.
- xlc Compiler를 이용하여 db2uexit 실행 module을 생성한다.

```
$ vi db2uexit.c
        #define ARCHIVE_PATH          "/archive/"
        #define RETRIEVE_PATH                 "/retrieve/"
        #define AUDIT_ACTIVE          1
        #define ERROR_ACTIVE          1
        #define AUDIT_ERROR_PATH    "/logback/"
        #define AUDIT_ERROR_ATTR    "a"
$ xlc -o db2uexit db2uexit.c
$ mv db2uexit $HOME/sqllib/bin
```

3.1.3 Backup

(1) Backup 개요

가. 기능 : Local 혹은 Remote node에 있는 Database의 Image를 Copy해주는 utility

나. 권한 : SYSADM, SYSCTRL, SYSMAINT Authority가 필요

다. Backup Level

(가) Full DB backup : 특정 Database 전체의 Image를 backup

(나) Tablespace backup : 특정 Database내의 1개 이상의 Tablespace Image를 backup

라. Backup Mode

(가) On-Line backup : Share Mode 상태에서 backup 실행

(나) Off-Line backup : Exclusive Mode에서 backup 실행

마. Configuration Parameter

(가) backbufsz : backup을 실행하는 동안 사용되는 buffer의 크기로서 기본값은 1024 page이다.

(나) num_ioservers : backup을 실행하는 동안 Data prefetch 나 Asynchronous I/O를 담당하는 I/O Server의 개수로서 기본값은 3이다.

바. Specific Processor

(가) Buffer Manipulator : database로부터 backup buffer로 backup image를 I/O하는 processor로서 database당 1개씩 존재한다.

(나) Media I/O Controller : backup buffer로부터 Tape등의 특정 device로 backup image를 I/O하는 processor로서 database다 1개씩 존재한다.

사. Backup Image

(가) Database alias : backup의 대상이 되는 database alias명을 나타낸다

(나) Type : backup의 type으로 `0'은 full database, '3'은 tablespace, '4'는 load copy를 나타낸다.

(다) Instance Name : database가 속한 Instance명을 나타낸다.

(라) Reserved : Reserved field로서 값은 `0'이다.

(마) Timestamp : backup이 실행된 timestamp를 나타낸다.

(바) Sequential Number : File의 extension을 나타내는 일련번호

<< 참고사항 >>

1. Temporary Container를 포함한 Tablespace는 Backup할 수 없다.
2. Tablespace Backup, On-Line, Backup은 Archival Logging상태에서만 가능하다.

(2) Backup 방법

가. Full DB Off-line Backup

▷ Circular logging이거나 Archival logging이거나 무관

```
$ db2 backup database SAMPLE
 Backup successful.  The timestamp for the backup
 image is : 1996090988811
$ Is -lia SAMPLE*
 6212 -rw ------- | inst1 4242424 9월 01
              SAMPLE.0.inst1. 1996090988811
```

나. Full DB On-line Backup

▷ Archival logging이어야 함

```
$ db2 backup database SAMPLE online
 Backup successful. The timestamp for the backup
 image is : 1996090988811
$ Is -lia SAMPLE *
 6212 -rw ------- | inst1 4242424 9월 01
              SAMPLE. 0. inst1. 1996090988811.001
```

다. Tablespace Off-line Backup

▷ Archival logging이어야 함

```
$ db2 backup database SAMPLE tablespace xx01dept
 Backup successful. The timestamp for the backup
 image is : 1996090988811
$ Is -lia SAMPLE *
 6212 -rw ------- | inst1 4242424 9월 01
              SAMPLE. 0. inst1. 1996090988811.001
```

라. Tablespace On-line Backup

▷ Archival logging이어야 함

```
$ db2 backup database SAMPLE tablespace
                        xx01det online
 Backup successful. The timestamp for the backup
 image is : 1996090988811
$ ls -lia SAMPLE*
 6212 -rw ------ | inst1 4242424 9월 01
              SAMPLE. 0. inst1. 1996090988811.001
```

3.1.4 Recovery

(1) Recovery 종류

- Crash Recovery
- Restore Recovery
- Roll Forward Recovery

가. Crash Recovery

(가) 기능 : System Power failure나 Application error 발생 등으로 인해 Database가 Inconsistent한 상태가 된 경우 Active Log를 다시 적용함으로써 Database 상태를 Consistent하게 유지시키는 Utility

(나) 권한 : 특별한 Authority는 필요없음

(다) Configuration Parameter : autorestart 기본값은 `on'으로 Crash recovery가 필요한 경우 자동적으로 recovery작업을 실행한다.

나. Restore Recovery

(가) 기능 : 특정 시간에 생성된 backup image를 이용하여 database를 backup 당시의 image와 동일하게 복구 혹은 새로 생성해주는 Utility

(나) 권한 :

ㄱ. 기존복구 : SYSADM, SYSCTRL, SYSMAINT 권한필요

ㄴ. 신규생성 : SYSADM, SYSCTRL 권한필요

(다) Restore Level

ㄱ Full DB backup : 특정 database 전체의 image를 restore

ㄴ Tablespace backup : 특정 database 내의 1개 이상의 Tablespace image를 restore

ㄷ Recovery History File : 특정 database의 recovery History file만 restore

(라) Restore Mode

ㄱ. On-Line restore : Share Mode 상태에서 restore

ㄴ. Off-Line restore : Exclusive Mode 상태에서 restore

(마) Configuration Parameter

ㄱ. restbufsz : restore를 실행하는 동안 사용되는 buffer의 크기로서 기본값은 1024페이지이다.

ㄴ. num_ioservers : restore를 실행하는 동안 data prefetch나 asyncronous I/O를 담당하는 I/O server의 개수로서 기본값은 3이다.

(바) Specific Processor

ㄱ. Buffer Mainpulator : restore buffer로부터 database로 backup image를 I/O하는 processor로서 database당 1개씩 존재한다.

ㄴ. Media I/O Controller : Tape등의 특정 device로부터 restore buffer로 backup image를 I/O하는 processor로서 database당 1개씩 존재한다.

(사) Redirected Restore Recovery

ㄱ. Container 변경 : tablespace에 할당된 containers를 재배열할 때

다. Rollforward Recovery

(가) 기능 : 특정 시간에 생성된 backup image와 Active Log를 이용하여 database를 backup 생성 이후의 특정 기간이나 가장 최근의 image로 복구 혹은 새로 생성해주는 recovery 방법

(나) 권한

ㄱ. 기존복구 : SYSADM, SYSCTRL, SYSMAINT권한 필요

ㄴ. 신규생성 : SYSADM, SYSCTRL 권한 필요

(다) Rollforward Level

ㄱ Log 적용 안함 : Log를 한 개도 적용하지 않는 방식으로 Off-line Full database backup image에만 적용가능

ㄴ 특정시간까지 적용 : Log 내용중 특정시간까지의 내용만 적용하는 방법

ㄷ Log 끝까지 적용 : 가장 최근까지의 Log내용을 적용하는 방법

(2) Recovery 방법

가. CRASH Recovery

▷ Autorestart option 확인

```
$ db2 get db cfg for DB 명 | grep AUTO
Auto restart enabled                    (AUTORESTART) = on
```

▷ 실행명령어

```
$ db2 restart db DB명
```

나. RESTORE Recovery

▷ 현재의 Logging상태가 Circular Logging인지 확인

```
$ db2 get db cfg for DB명
Log retain for recovery enabled      (LOGRETAIN)   = OFF
User exit for logging enabled        (USEREXIT)    = OFF
```

▷ Full DB off-line backup image인 경우

```
$ db2 restore database SAMPLE
$ db2 restore database from / dbbackup
$ db2 restore database taken at 1996090988811
$ db2 restore database to / xx
$ db2 restore database into xxdb
```

다. ROLLFORWARD Recovery

▷ 현재의 Logging상태가 Archival Logging인지 확인

```
$ db2 get db cfg for DB명
Log retain for recovery enabled      (LOGRETAIN)   = ON
User exit for logging enabled        (USEREXIT)    = OFF
```

▷ Full DB off-line backup image인 경우

```
<< log를 전혀 적용시키지 않는 경우 >>
$ db2 restore database sample without rolling forward
<< 특정 CPU 시간까지의 log만 적용하는 경우 >>
$ db2 restore database SAMPLE
$ db2 rollforward database sample to 1996-09-09-09:30:00
                                        and stop
<< 현재까지의 log를 모두 적용시키는 경우 >>
$ db2 restore database sample
$ db2 restore database sample to end of logs and stop
```

▷ Full DB on-line backup image인 경우 반드시 log 적용

```
<< 특정 CPU 시간까지의 log만 적용하는 경우 >>
$ db2 restore database SAMPLE
$ db2 rollforward database sample to 1996-09-09-09:30:00
                                        and stop
<< 현재까지의 log를 모두 적용시키는 경우 >>
$ db2 restore database sample
$ db2 restore database sample to end of logs and stop
```

▷ Tablespace backup image인 경우에는 현재까지의 log를 모두 적용시켜야 한다.

```
$ db2 restore database SAMPLE
$ db2 restore database sample to end of logs and stop
```

<< 참고사항 >>

▷ CPU Time : 현재시간에서 9시간을 빼야함.
(현재시간 - 9 = CPU Time)

(3) Recovery History File

가. 기능 : 특정 database에 행한 Backup / Restore / Load
작업에 대한 정보를 기록하는 file

나. Recovery History File의 구조

Column명	Type	설 명
OPERATION	Char(1)	작업형태 : B = Backup R = Restore U = Unload L = Load
OBJECT	Char(1)	작업범위 : D = Full Database P = Tablespace T = Table
OBJECT_PART	Char(17)	작업범위 : timestamp(14char) + sequence(3char)
OPTYPE	Char(1)	F=off-lienbackup N=on-line backup R=load replace A=load append C=load copy blank = 기타
DEVICE_TYPE	Char(1)	D=disk, K=diskette, T=tape A=ADSM, U=userexit, O=other
FIRST_LOG	Char(12)	rollforward recovery시 사용될 최초의 log 번호
LAST_LOG	Char(12)	back당시 마지막 log 번호
BACKUP_ID	Char(14)	backup id
SCHEMA	Char(8)	unload/load 작업시 사용되는 table name Qualifies
TABLE_NAME	Char(18)	" table name
NUM_ TABLESPACE	Char(3)	backup/restore시 포함될 Tablespace 갯수
LOCATION	Char(255)	backup/load copy/unload의 결과 file
COMMENT	Char(30)	주석

다. 권한 : SYSADM, SYSCTRL, SYSMAINT나 DBADM 권한이 필요하다.

라. History file의 내용 확인

ㄱ. List History command를 이용하여 Recovery History file의 내용을 확인한다.

ㄴ. 특정시간까지 적용 : Log 내용중 특정시간까지의 내용만 적용하는 방법

ㄷ. Log 끝까지 적용 : 가장 최근까지의 Log내용을 적용하는 방법

마. History file의 관리

ㄱ. Prune History command : History file중 특정 Timestamp 이전의 정보를 삭제한다.

ㄴ. Update History command : history file의 내용을 수정한다.

ㄷ. REC_HIS_RETENTN parameter : history file의 size를 지정한다.

바. History file의 복구

ㄱ. Database Recovery 작업시 Recovery History file만을 복구하는 Option을 이용한다.

사. Recovery History file의 내용 확인

▷ list backup command

```
$ db2 list backup all for DB명
$ db2 list backup since 199609 for DB명
$ db2 list backup containing for DB명, Table명
```

▷ list history command

```
$ db2 list history all for DB명
$ db2 list history since 199609 for DB명
$ db2 list history containing for DB명. Table명
```

아. Recovery History file의 내용 관리

▷ prune history comand

```
$ db2 prune history timestamp
$ db2 prune history 199609
$ db2 prune history 199609 with force option
```

▷ update history command

```
$ db2 update history for object_part with
    new_location device type new_device_type
    cmment new_coment
$ db2 update history for 1996090988881101 with
    location / dbbackup device type D comment testing
```

▷ update REC_HIS_RETENTN command

```
$ db2 update db cfg db명 using
      REC_HIS_RETENTN 100
```

3.2 Data Moving

3.2.1 Export

▷ table의 데이터 내용을 DEL, IXF, WSF 형태로 반출할 수 있다. 반출된 내용은 import 나 load 유틸리티를 사용하여 테이블로 다시 반입할 수 있다.

<예 : org 테이블의 데이터를 edu.exp 라고 하는 ixf 형태의 파일로 export>

```
$ db2 export to educ.exp of ixf messages msg
  select * from org
```

3.2.2 Import

▷ 일단 export 명령으로 반출된 파일이나 이미 DEL,ASC,IXF,WSF 형태로 존재하는 파일은 LOAD 명령(WSF유형만 지원 안됨)이나 IMPORT 명령으로DB의 TABLE로 데이타를 이동시킬 수 있다.

<예: educ.exp 파일로부터 데이터를 import하여 usreid.org라는 새로운 테이블 작성 >

```
$ db2 import from educ.exp of ixf messages
```

▷ method n (column 명) 옵션을 부여할 수도 있다. 그 옵션은 export된 table과 다른 column을 주는 경우에 사용할 수 있는 option 임

3.2.3 Load

▷ LOAD는 대량의 데이터를 빠른 속도로 테이블로 옮기는데 사용되는 utility 이다.

<예 : educ.exp 파일로부터 orgg테이블로 데이터를 LOAD>

```
$ db2 load from educ.exp of ixf messages

   msg.load remote file educ
```

▷ LOAD는 IMPORT 보다 속도가 빠르다. 왜냐하면, 우선 Load utility는 log를 기록하지 않고 물리적인 페이지 단위로 데이터를 올리는 반면 Import는 log를 기록하며 내부적으로는 SQL Insert 문장이 반복적으로 수행하여 데이터를 올린다. INDEX가 생성되는 방법에 있어서도 IMPORT는 IMPORT 되는 레코드 단위로 한번에 하나의 INDEX를 만들지만, LOAD는 데이타의 LOAD 단계가 모두 끝난 이후에 BUILD 단계에서 한꺼번에 INDEX를 만드는 특징을 갖고 있기 때문이다.

▷ 특기할 만한 옵션은 다음과 같은 것들이 있다.

```
method ( L : 컬럼의 위치 (start,end)
         N : 컬럼의 이름
         P : 컬럼의 순서      )

restart ( B : index creation
          D : error 인 레코드 제거
          N : N 번째 레코드 부터 시작 )
```

- v7에서는 restart 명령을 발행할 때 option을 주지 않아도 자동적으로 error가 발생한 시점으로부터 시작을 한다.

3.3 REORG & RUNSTATS

3.3.1 REORGCHK

▷ REORG utility는 레코드의 insert/delete등으로 인하여 데이터 및 index 페이지의 물리적인 정열 순서가 엉키는 것을 새롭게 정열시켜주는 utility이다. REORGCHK unility는 데이터베이스를 검사하여 REORG를 할 필요가 있는지를 체크하는 utility이다. 데이터 페이지와 index 페이지의 순서가 서로 맞지 않으면 성능의 저하를 유발하게 되므로 REORGCHK를 정기적으로 수행하여 데이터베이스의 물리적인 상태를 확인하는 것이 필요하다.

▷ Syntax

```
db2 -r out_file reorgchk current statistics on table tbl_name 또는
db2  reorgchk  update  statistics  on  table  system
```

▷ Sample Result

```
Table statistics:
 F1: 100*OVERFLOW/CARD < 5
 F2: 100*TSIZE / ((FPAGES-1) * 4020) > 70
 F3: 100*NPAGES/FPAGES > 80
CREATOR NAME   CARD OV NP FP TSIZE F1 F2 F3 REORG
 ------------------------------------------------------------
-------------------
SYSIBM SYSCHECKS        -   -  -  -    -    -   -   -   ---
SYSIBM SYSDATATYPES   13   0   1   1   1027   0    -   100   ---
SYSIBM SYSFUNCTIONS 104   0   8   8    728   0   2    100   -*-
SYSIBM SYSINDEXES     57   17   3   5   9063   29   56   60 ***

Index statistics:
 F4: CLUSTERRATIO or normalized CLUSTERFACTOR > 80
 F5: 100*(KEYS*(ISIZE+ 10)+ (CARD-KEYS)*4) / (NLEAF*4096) > 50
 F6: 90*(4000/(ISIZE+ 10)**(NLEVELS-2))*4096/ (KEYS*(ISIZE+ 10)+ (CARD-
     KEYS)*4)<100
CREATOR   NAME CARD   LEAF   LVLS ISIZE   KEYS   F4   F5   F6 REORG
------------------------------------------------------------
-------------------
Table: SYSIBM.SYSCHECKS
SYSIBM   IBM37              -      -      -      -       -    -    -     - ---
```

▷ 위의 결과에서 REORG컬럼의 내용이 별표(*) 이면 그 테이블은 reorg가 필요하고 하이폰(-)이면 정상이므로 reorg를 할 필요가 없다.

3.3.2 REORG

▷ REORG명령은 해당 테이블의 데이터를 Physical 하게 재정렬하여 조회의 Performance 를 향상시킨다

▷ Syntax

```
db2 reorg table 테이블명 [INDEX 색인명] [USE 테이블공간명]

db2 reorg table tbl_name index idx_name use tempspace1
```

3.3.3 RUNSTATS

▷ RUNSTATS 는 데이터베이스의 통계정보를 최신 상태로 갱신하는데 사용된다.
DB2 optimizer는 access path를 결정할 때 바로 이 통계정보를 참조하므로 대량의 데이터가 삽입되거나 또는 지워졌을 경우 항상 RUNSTATS를 실행시켜서 통계정보를 갱신하는 것이 중요하다. 또한 REORGCHK utility도 이 통계정보를 참조하여 REORG 필요 유무를 판단하므로 REORGCHK를 수행하기전에 항상 RUNSTATS를 먼저 수행하는 것이 좋다.

▷ 즉 RUNSTATS → REORGCHK → REORG → RUNSTATS 순서로 항상 함께 사용하는 것이 바람직하다.

```
db2  runstats on table 테이블명 [WITH DISTRIBUTION

        [AND [DETAILED] {INDEXES ALL | INDEX 색인명}] |

        {AND | FOR} [DETAILED] {INDEXES ALL | INDEX 색인명}]
```