INFORMIX 운영자 manual
(Informix Dynamic Server)

2003-06-05

Table of Contents

Informix Dynamic Server 의 특징

Informix Database Server 은 multi-threaded architecture 로 구성

- DBMS 를 수행하는데 fewer process 가 필요로 함.
- 사용자가 증가할수록, 서버는 사용자에 맞는 최소한의 resource 를 필요로 한다. ,
- 서버 측면에서는 resource 를 적게 사용
- 클라이언트 측면에서는 여러 process 를 사용할 수 있다는 장점을 지님.

Informix Database System Architecture

- Shared memory, process, and disk 를 한 개의 instance 라고 한다. 그러므로 한 개의 machine 에 여러 개의 instance 가 존재 할 수 있다.

1. **Shared Memory**:
   1) Resident Portion: DB 의 속도를 향상 시키기 위하여 디스크로부터의 caching 작업과 buffer pool 과 system information 을 포함한다.
   2) Virtual Portion: Process 에 필요한 maintaining and controlling the resources 하며, thread 와 session 그리고 thread 와 session 에 의하여 사용되는 data 를 포함.
   3) Message Portion: C/S 간에 communication 을 구성, 일종의 mailbox 역할을 한다.
2. **Process**: Database server 를 구성하고 있는 요소. VP 로 구성되어있으며, 각각의 VP 의 단위는 특정한 임무를 책임지는 VP class 에 속한다. Thread 를 schedule 한다.
3. **Disk**: Server 에 할당된 한 개 이상의 disk space 단위의 집합. 모든 data 와 system information 이 저장되어있다.

Disk 의 구성

1) **Physical spaces**:
   ① Chunks: 서버에 할당된 연속된 공간, DISK (Raw disk, Cooked file; mounted disk)
   ② Page: Chunk 가 IDS 에 할당되면, logically 게 할당되는 I/O 의 기본단위, 시스템에 따라 다르지만 보통 2k, 4k byte 로 구성
   ③ Extent: Table 에 할당되는 disk space 단위, 최소한의 단위는 4 page 로 구성되어 있으며, default 8 page
2) **Logical spaces**: (반드시 physical space 가 있어야 함)
   ① Dbspace: Logical collection of chunks, 각각의 dbspace 는 chunk 에 할당 되어있어야 한다. 즉 physical 한 disk 에 back-up 과 restore 의 단위, mirroring 의 단위, database, table 의 생성 장소, 한 개의 server 에 2047 개의 dbspace 까지 가능함.
   ② Tblspaces: Table 에 관련된 page 의 집합으로 single table, index 혹은 fragment 에 사용된다.
   ③ Blobspace : Simple Large Object(text, byte)를 저장하는데 사용되는 공간
   ④ Sbspace: Smart Large Object 를 저장하는 공간으로 logging 이나 caching 이 가능하다.

1. 기본적인 용어

1) Page

- All data & system information이 dbspace내의 chunk에 page단위로 저장
- On-Line system에서 I/O 와 data 저장의 기본 단위
- Size는 OS에 의존한다.(주로 2k, 혹 4k)

2) Extent

- table에 할당되는 disk space 단위
- page1(Bitmap page), page2(index page), ...
- EXTENT SIZE : When the table is created.(default 8 page)
- NEXT SIZE : added to the table. (default 8 page)
- size는 짝수개의 page로

3) Chunk

- On-Line system에서 이용 가능한 physical disk space 단위

※ Cooked / Raw Devices

- buffering interface : A Cooked file is a UNIX file.
- none buffering interface = *character-special* interface : raw device.

4) Blobpage

- page의 수.
- blobpage size는 blobspace 생성시 명시.

5) Dbspace

- 하나 이상의 chunk로 구성.
- back-up과 restore의 단위.
- mirroring의 단위.
- Databases, Table의 생성 장소.

6) Tblspace

- table에 관련된 page의 집합.

7) Blobspace

- BLOBs data를 저장하는 chunk로 구성

8) Database Logging

종류	flag
• No Logging	N
• Buffered Logging	B
• Unbuffered Logging	U
• MODE ANSI Unbuffered Logging	A

) No Logging

- transaction이 지원되지않지만 performance측면에서는 좋음.
- 단위 작업에 대한 복구는 불가능함.

(Buffered Logging

- logical log buffer가 full 됐을 때 logical log에 기록한다.
- Disk에 logical log를 기록하기 전에 fail이 일어날 경우 복구가 불가능하다.

) Unbuffered Logging

- transaction 단위로 logical log buffer의 내용을 logical log에 기록
- 즉 한 개의 transaction이 끝나면 곧바로 disk에 write를 한다.
- Performance는 떨어짐.

(ANSI 표준 MODE

- Unbuffered mode와 동일,
- 다른 db와 연동 시 사용

```
$ ontape -s -B stores7       ➔ DB의 Logging mode 변경하고자 할 때
              -B : Buffered      -U : Unbuffered
              -A : ANSI           -N : No Logging
              stores7 : DataBase Name

$ onmonitor -> Status -> DataBase
Press ESC to return to the Status Menu.
               Use arrow keys to move the cursor.

                              DATABASES
                                                              When        Log
         Database Name        Owner       In Dbspace          Created     Status

         sysmaster            informix    rootdbs             05/27/97    U
         stores7              informix    datadbs             03/05/98    B
```

9) Informix Data Type

CHAR(n), CHARACTER(n)

문자나 문자열, 숫자, 기호 등을 저장할 수 있는 Data type.

실제 사용되는 크기와 상관 없이 항상 n byte 만큼 disk space 를 차지한다.

Character type 으로 저장된 숫자는 연산이 불가능하다.

최대 길이 32,767 byte 까지 허용

INTEGER, INT8, SMALLINT

정수값을 저장할 수 있는 Data type

INTEGER : 4 byte (±2,147,483,647, 2^31±-1)

INT8 : (8:64bit, 10:32bit) (±9,223,372,036,854,775,807, 2^63±-1)

SMALLINT : 2byte(±32,767)

Decimal(p,s)

실수값을 저장할 수 있는 고정 소수점 Data type

전체 자리수와 소수점 이하 자리수를 사용자가 정의

precision : 전체 자리수

scale : 소수점 이하 자리수

유효 자리수 - 32 자리, 자동으로 반올림된다.

scale 홀수 (p + 4) / 2, scale 짝수 (p + 3) / 2

MONEY(p,s)

화폐단위를 포함하는 자료값을 저장할 수 있는 Data type

내부적으로는 Decimal 로 정의됨 (고정 소수점 32 자리)

Default value : MONEY(16,2)

DBMONEY 환경변수를 지정함으로써 보여지는 화폐단위를 바꿀 수 있다.

SERIAL(s), **SERIAL8**(s)

system 에 의해 자동적으로 sequential number 가 할당되는 Data type

내부적으로는 Integer 가 정의되고, Starting number 는 사용자가 정의.

하나의 테이블에는 두개이상의 serial column 이 존재할 수 없다.

DATE

날짜를 저장할 수 있는 Data type

내부적으로는 integer 로 정의됨 (1900 년 1 월 1 일 : 1)

DBDATE 환경 변수를 지정함으로써 보여지는 format 을 바꿀 수 있다.

Default : mdy4/

입력이나 연산 수행 시에는 인용부호를 사용해야 한다.

DATETIME (Largest qualifier) **TO** (smallest qualifier)

Date type 보다 더 정확한 시, 분, 초등의 시간까지 표현 가능한 Data type.

INTERVAL Largest qualifier(n) **TO** smallest qualifier(n)

기간을 저장하는 데 사용하는 Data type

VARCHAR(Max,Min)

실제로 사용되는 만큼 disk space 를 차지하는 Character Data type

Max : Maximum length (최대 255 byte)

Min : minimum length

TEXT

ASCII file 을 저장할 수 있는 Data type.

2^{31} byte 까지 저장 가능

BYTE

Binary type Data 를 저장할 수 있는 Data type

Smart Large Object

BLOB

Binary Data 저장. 4 tera byte 까지 가능

CLOB

text type data 저장 가능.

Informix 기동 과 종료

1) Informix 기동 방법

Informix Server 를 기동하는 방법은 oninit 을 이용하는 방법과 onmonitor 를 이용하는 두 가지 방법이 있다. Informix server 를 기동하기 위해서는 Informix 가 요구하는 환경(예: INFORMIXDIR 등..) 이 설정되어 있는 상태에서 informix user 나 root user 로 가능하다.

가) oninit 을 이용한 기동방법

① Informix 로 login
② env 명령어를 통해 환경을 확인
(INFORMIXDIR.INFORMIXSERVER,ONCONFIG 는 반드시 확인)
③ oninit 을 실행
④ onstat -m 명령을 통해 Server 가 정상 기동 되었음을 확인

나) onmonitor 을 이용한 기동방법

① Informix 로 login
② env 명령어를 통해 환경을 확인
(INFORMIXDIR.INFORMIXSERVER,ONCONFIG 는 반드시 확인)
③ onmonitor 를 실행
④ 메뉴 중 Mode 메뉴 선택
⑤ 메뉴 중 Start-up 메뉴 실행(3 번째 라인에 quiescent 표시)
⑥ 메뉴 중 On-Line 메뉴 실행(3 번째 라인에 On-Line 표시)
⑦ onmonitor 종료 후 onstat -m 명령을 통해 메시지 확인

2) Informix 종료 방법

Informix Server 를 종료하는 방법은 onmode 를 이용하는 방법과 onmonitor 를 이용하는 두 가지 방법이 있다. Informix server 를 종료하기 위해서는 Informix 가 요구하는 환경(예: INFORMIXDIR 등..) 이 설정되어 있는 상태에서 informix user 나 root user 로 가능하다.

가) onmode 를 이용한 종료방법

① Informix 로 login
② env 명령어를 통해 환경을 확인
③ (INFORMIXDIR.INFORMIXSERVER,ONCONFIG 는 반드시 확인)
④ onmode -ky 을 실행
⑤ onstat - 명령을 통해 Server 가 정상 종료 되었음을 확인
⑥ Shared memory not initialized 라는 메시지 확인

나) onmonitor 을 이용한 종료 방법

① Informix 로 login
② env 명령어를 통해 환경을 확인

③ (INFORMIXDIR.INFORMIXSERVER,ONCONFIG 는 반드시 확인)
④ onmonitor 를 실행
⑤ 메뉴 중 Mode 메뉴 선택
⑥ 메뉴 중 Take-Offline 메뉴 실행(3 번째 라인에 off-line 표시)
⑦ onmonitor 종료 후 onstat - 명령을 통해 메시지 확인
⑧ Shared memory not initialized 라는 메시지 확인

3. 운영 상태 보기

Informix Server 의 운영 상태를 확인하는 방법은 onstat 명령어를 통한 방법과 onmonitor 를 이용한 방법 그리고 online log message 를 통해 확인 할 수 있다. DBA 는 수시로 이러한 명령어를 통해 Server 의 정상 동작 유무를 확인하여 조치 하여야 한다.

1) onstat 명령의 사용 방법

onstat 은 Informix Server 의 운영 상태를 monitor 할 수 있는 utility 로 자세한 사용방법은 매뉴얼을 참조 바라며, 이 문서상에서는 운영자에게 반드시 필요한 몇 가지 명령어에 대해서만 기술합니다.

① Informix Server 운영 상태 확인 방법 : onstat -

```
Informix Dynamic Server Version 7.30.FC4      -- On-Line -- Up 15:07:43 -- 20304 Kbytes
```

설명 : 7.30FC4 -> Version 정보
On-Line -> 운영 모드
Up 15:07:43 -> Server 기동 후 경과 시간
20304Kbytes -> Server 가 사용하는 Memory 량

② Informix Server 운영 log 보기 : onstat -m

```
Web Oct   20 19:01:06 1998
19:01:06  Event alarms enabled.    ALARMPROG = '/home/informix/etc/no_log.sh'
19:01:12  DR: DRAUTO is 0 (Off)
19:01:12  Informix Dynamic Server Version 7.30.FC4      Software Serial Number DTI#J000000
19:01:13  Informix Dynamic Server Initialized -- Shared Memory Initialized.
19:01:13  Physical Recovery Started.
19:01:13  Physical Recovery Complete: 0 Pages Restored.
19:01:13  Logical Recovery Started.
19:01:16  Logical Recovery Complete. 0 Committed, 0 Rolled Back, 0 Open, 0 Bad Locks
19:01:16  Dataskip is now OFF for all dbspaces
19:01:16  On-Line Mode
19:01:16  Checkpoint Completed:    duration was 0 seconds.
```

설명 : onstat -m option 은 server message log 에 있는 마지막 내용을 보여주며, 즉 server 상의 운영 정보를 보여주며, 이 내용은 informix 가 설치되어 있는 home directory 밑의 online.log 파일에 저장되어 있다. DBA 는 위의 내용을 수시로 점거하여 ERROR 나 Assert Fail 의 메시지가 있는 경우 해당 내용을 조치하여야 한다.

참고) tail - f $INFORMIXDIR/online.log

③ 사용자 보기 : onstat -u

Informix Dynamic Server Version 7.30.FC4 -- On-Line -- Up 16:36:40 -- 12112 Kbytes
Userthreads

address	flags	sessid	user	tty	wait	tout	locks	nreads	nwrites
20044e028	---P--D	1	informix	-	0	0	0	15	5
20044e6c0	---P--D	1	informix	-	0	0	0	0	2
20044e6c0	---P--D	1	informix	-	0	0	0	2	
20044e6c0	---P--D	1	informix	-	0	0	0	0	2
20044e6c0	---P--D	1	informix	-	0	0	0	0	2
20044e6c0	---P--D	1	informix	-	0	0	0	0	2

6 active, 128 total, 19 maximum concurrent.

Position 1:	B	Waiting on a buffer
	C	Waiting on a checkpoint
	G	Waiting on write of the Logical Log buffer
	L	Waiting on a lock
	S	Waiting on a mutex
	T	Waiting on a transaction
	X	Waiting on a transaction cleanup
	Y	Waiting on a condition
Position 3:	A	DBSpace backup thread
	B	Begin work
	P	Preparing/prepared
	X	XA prepare
	C	Commiting/committed
	R	Aborting/aborted
	H	Heuristic aborting/aborted
Position 4:	P	Primary thread for a session
Position 5:	R	Reading (RSAM Call)
	X	Critical Write
Position 7:	B	BTree clean thread
	C	Terminated user thread waiting for cleanup
	D	Daemon thread
	F	Page cleaner thread (flusher)
	M	On-Monitor thread

설명 : Server 상에 접속하여 사용하고 있는 사용자 정보를 보여준다.
6 active : 현재 사용자 수
19 maximum concurrent: Sever 기동 후 최대 19user 까지 사용함.

Nice to know:

Onstat -g ses [session #]: Print session information

Onstat -g sql [session #]: Print SQL information

④ Dbspace 정보 보기 : onstat -d

```
Informix Dynamic Server Version 7.30.FC4      -- On-Line -- Up 15:28:34 -- 20304 Kbytes

Dbspaces
address          number     flags      fchunk     nchunks    flags      owner      name
20044c1c0        1          1          1          1          N          informix   rootdbs
20044c7d8        2          2001       2          1          N T        informix   tempdbs
20044c8c0        3          1          3          1          N          informix   cdrqueue
20044c9a8        4          1          4          1          N          informix   datadbs
 4 active, 2047 maximum

Chunks
address          chk/dbs    offset     size       free       bpages     flags      pathname
20044c2a8        1  1       0          150000     43481                 PO-        /DBSPACE/rootdbs
20044c508        2  2       0          50000      49947                 PO-        /DBSPACE/tempdbs
20044c5f8        3  3       0          100000     99883                 PO-        /DBSPACE/cdrqueue
20044c6e8        4  4       0          425000     421407                PO-        /DBSPACE/datadbs1
 4 active, 2047 maximum
```

0x01	No mirror
0x02	Mirror
0x04	Down/disable mirror chunks
0x08	Newly mirrored
0x10	BLOBSpace
0x20	BLOBSpace on removable media
0x40	BLOBSpace on optical media
0x80	BLOBSpace has been dropped
0x100	BLOBSpace is optical stage blob
0x200	Space is being physically recovered
0x400	Space has been physically recovered
0x800	Space is being logically recovered
0x1000	A table in the DBSpace has been dropped
0x2000	Temp DBSpace
0x4000	BLOBSpace is being archived.
0x8000	Smart BLOBSpace
0x10000	Either physical or logical log changed

Position 1: M	Mirrored	
	N	Not mirrored
Position 2: X	Newly mirrored	
	D	Down
	P	Physically recovered, waiting for logical recovery
	L	Being logically recovered
	R	Being recovered
Position 3: B	BLOBSpace	
	T	Temporary

설명 : Server 상에 할당된 dbspace 정보 및 chunk 정보를 보여준다. 유의하여 볼 내용은 CHUNK 정보의 flag 정보가 PD 로 된 chunk 는 down 되었음을 나타내므로 원인 규명 후 조치한다. chunk 의 free space 가 적은 경우 해당 dbspace 에 chunk 를 add 해 주거나 space 조정 작업을 해주어야 한다. Chunk 추가 방법은 다음 장을 참조 바람.

Nice to know:

Onstat -g iof: Print disk I/O statistics by chunk/file

```
Informix Dynamic Server 2000 Version 9.20.UC2    -- On-Line -- Up 16:35:32 --
11264 Kbytes

AIO global files:
gfd pathname              totalops  dskread dskwrite  io/s
  3 rootdbs_ids920_ksroh      32        23        9     0.0
  4 front01-rear01-vol01      2         2         0     0.0
```

Oncheck -pe: Print extents report (-ce)

⑤ 로그 정보 보기 : onstat -l

```
Physical Logging
Buffer bufused    bufsize    numpages numwrits pages/io
   P-2  0            32          3009       97          31.02
         phybegin physize    phypos     phyused    %used
         10003f     25000      4628       0          0.00

Logical Logging
Buffer bufused    bufsize    numrecs    numpages numwrits recs/pages pages/io
   L-3  0            32          345992     16135      1174       21.4          13.7
            Subsystem       numrecs    Log Space used
            OLDRSAM         345992     31252264

address             number     flags      uniqid     begin          size
sed      %used
200543d88           1            U-B----    1       1061e7         10000      10000
100.00
200543da4           2            U-B----    2       1088f7         10000      10000     100.00
200543dc0           3            U---C-L    3       10b007         10000      1109      11.09
200543ddc           4            F------    0       10d717         10000      0         0.00
200543df8           5            F------    0       10fe27         10000      0         0.00
200543e14           6            F------    0       112537         10000      0         0.00
200543e30           7            F------    0       114c47         10000      0         0.00
200543e4c           8            F------    0       117357         10000      0         0.0

Position 1: A       Newly added
            F       Free log
            U       Used log
Position 3: B       Backed up
```

설명 : 로그는 FULL 이 발생하지 않도록 주기적으로 monitor 후 백업하여 주어야 한다. 위의 내용은 20MB 로그 8 개가 잡혀있으며 로그번호 1, 2 은 다 쓴 후 백업(U-B) 이 된 상태고 현재 3 번을 사용 중(U - C-L) 이며 나머지는 사용하지 않은(F ----) 임을 나타낸다.

Nice to know:logical log 를 추가하거나, drop, physical log 의 위치를 변경할 때

```
Onparams: onparams   { -a -d DBspace [-s size]            |
                       -d -l logid [-y]                   |
                       -p -s size [-d DBspace] [-y] }
        -a   - Add a logical log
        -d   - Drop a logical log
        -p   - Change physical log size and location
        -y   - Automatically responds "yes" to all prompts
```

```
Onmode -l (Force to next logical log)
Onmode -c (Do Checkpoint)
```

2) onmonitor 를 이용한 상태 감시

onmonitor 상의 status 메뉴상에서도 다양한 운영상의 정보를 monitor 할 수 있다. 자세한 onmonitor 사용방법은 매뉴얼을 참조 바람.

```
STATUS:    Profile   Userthreads   Spaces   Databases   Logs   Archive   ...
Display dbspace and chunk information.
----------------------------On-Line------- Press CTRL-W for Help. --------
```

4. 디스크 추가 삭제

DBA 는 수시로 dbspace 사용량을 check 하여 필요시 새로운 디스크를 dbspace 에 할당해 주어야 한다. 디스크 할당 방법에는 새로운 dbspace 를 생성하는 방법과 기존의 dbspace 에 추가하는 방법이 있다.

1) 새로운 dbspace 생성 방법

① 할당하고자 하는 raw disk 또는 파일 시스템을 준비
(예: /dev/rdsk/disk1 또는 /home/data/datadbs1)

② 해당 디스크의 owner:group 을 informix 로 바꾼다.

chown informix /dev/rdsk/disk1

chgrp informix /dev/rdsk/disk1

③ 해당 디스크의 mod 을 660 로 바꾼다.

chmod 660 /dev/rdsk/disk1

④ onmonitor 를 실행 후 Dbspaces -> Create 선택.

```
CREATE DBSPACE
Dbspace Name [newdbspace                    ]     Mirror [N ]     Temp [N ]

     PRIMARY CHUNK INFORMATION:
     Full Pathname [/dev/rdsk/disk1                                      ]
           Offset [          0] Kbytes              Size [    2000000] Kbytes

  MIRROR CHUNK INFORMATION:
     Full Pathname [                                                ]
           Offset [          0] Kbytes
```

Dbspace Name : 생성하고자 하는 dbspace 의 이름

TEMP : dbspace 를 temp dbspace 로 생성할 것인지 설정[Y/N]

MIRROR : dbspace 의 복사본 dbspace 를 지정할 것인지 설정[Y/N]

Full Path : 실제 사용할 disk path 정보 입력(반드시 full path 사용)

Offset : disk 상의 사용할 시작 위치 지정(처음부터 사용하는 경우 0)

Size : 실제 할당할 disk 량

MIRROR 를 [Y]를 설정한 경우 복사본 disk 정보를 입력해야 함.

2) 기존의 dbspace 에 추가 방법

① 할당하고자 하는 raw disk 또는 파일 시스템을 준비

② (예: /dev/rdsk/disk1 또는 /home/data/datadbs1)

③ 해당 디스크의 owner:group 을 informix 로 바꾼다.

④ chown informix /dev/rdsk/disk1

⑤ chgrp informix /dev/rdsk/disk1

⑥ 해당 디스크의 mod 을 660 로 바꾼다.

⑦ chmod 660 /dev/rdsk/disk1

⑧ onmonitor 를 실행 후 Dbspaces -> Add_chunk -> 해당 dbspace 선택

⑨ dbspace 생성과 동일한 정보 입력.

- Onspaces 를 이용

Onspaces: create, drop, mirror dbspaces

Onspaces -c -d dbspace -p /dev/rdsk/disk1 -o 20000 -s 100000

```
Usage:
 onspaces   { -a spacename -p pathname -o offset -s size [-m path offset]
                        { { [-Mo mdoffset] [-Ms mdsize] } | -U }
                            [-Df default-list] } |
                -c { -d DBspace [-t]
                        -p pathname -o offset -s size [-m path offset] } |
                    { -b BLOBspace -g pagesize
                        -p pathname -o offset -s size [-m path offset] } |
                    { -S SBLOBspace
                        -p pathname -o offset -s size [-m path offset]
                        [-Mo mdoffset] [-Ms mdsize] [-Df default-list] } |
                    { -x Extspace -l Location } |
                -d spacename [-p pathname -o offset] [-f] [-y]
                -f[y] off [DBspace-list] | on [DBspace-list] |
                -m spacename {-p pathname -o offset -m path offset [-y] |
                            -f filename} |
                -r spacename [-y] |
                -s spacename -p pathname -o offset {-O | -D} [-y] }
                -ch SBLOBspace {-Df | -ch} flags
                -cl SBLOBspace

    -a - Add a chunk to a DBspace, BLOBspace or SBLOBspace
    -c - Create a DBspace, BLOBspace, SBLOBspace or Extspace
    -d - Drop a DBspace, BLOBspace, SBLOBspace, Extspace, or chunk
    -f - Change dataskip default for specified DBspaces
    -m - Add mirroring to an existing DBspace, BLOBspace or SBLOBspace
    -r - Turn mirroring off for a DBspace, BLOBspace or SBLOBspace
    -s - Change the status of a chunk
    -ch - Change SBLOBspace characteristics
    -cl - Clean SBLOBspace (delete stray LOs)
```

5. 백업 과 복구

- Incremental Backup
  - Level-0 : backup 이 시작된 시점의 server system 의 모든 data 를 backup.
  - Level-1 : Level-0 backup 이후에 변경된 data 만 backup.
  - Level-2 : 마지막 Level-0 나 Level-1 로 backup 된 data 중 변경된 data 만 backup.

- backup 의 step
  ① logical log 의 free space 가 적어도 반 이상 남아있는지 check.
  ② Server 가 checkpoint 를 시작.
  ③ Chunk 별로 write 된 page 단위로 backup 받을 list 를 구성.
  ④ Physical log page 를 temporary table 에 기록.
  ⑤ Internal backup thread 가 backup data 생성 시작.

1) DATA 백업 방법
  ① onconfig 파일의 TAPEDEV 값을 확인한다.
  ② 해당 Device 에 테이프를 넣는다.
  ③ ontape -s -L 0 (또는 1,2) 를 실행
  ④ 종료 후 테이프에 Label 작성 후 보관

2) LOG 백업 방법
  ① onconfig 파일의 LTAPEDEV 값을 확인한다.
  ② 해당 Device 에 테이프를 넣는다.
  ③ ontape -a (또는 -c) 를 실행
  ④ 종료 후 테이프에 Label 작성 후 보관

3) DATA 복구 방법
  ① onconfig 파일의 TAPEDEV 값을 확인한다.
  ② 해당 Device 에 테이프를 넣는다.
  ③ ontape -r 을 실행
  ④ Salvage Log 백업 (Y/N) : N
  ⑤ Continue Restore ?(Y/N) : Y
  ⑥ Log Restore ?(Y/N) : Y
  ⑦ LOG 백업 받은 테이프 넣는다.
  ⑧ Enter
  ⑨ 종료
  ⑩ onstat -m 으로 메시지 확인

6. 장애 확인 방법과 조치

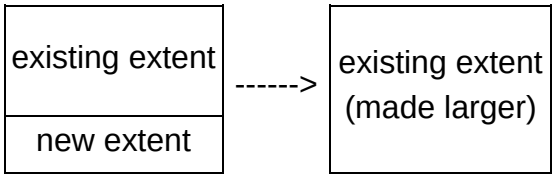
1) Server 비정상 종료 시 조치
   ① onstat - 명령을 통해 On-Line 여부 확인
   ② Off-Line 인 경우 Restart (oninit)
   ③ online.log 파일상의 에러 내용 확인
   ④ 로그상의 af.xxxxx 파일 확인
   ⑤ online.log 상의 장애 내용 과 af.xxxx 파일을 보관
   ⑥ 원인 진단 및 Informix Tech Support Center 연락
2) Server 정상 이나 Connection 이 안되는 경우
   ① onstat - 명령을 통해 On-Line 여부 확인
   ② onstat -a > onstat.a 실행
   ③ onmode -ky 로 종료후 재기동(oninit)
   ④ online.log 파일 확인
   ⑤ online.log 상의 장애 내용(있는 경우) 과 onstat -a 파일을 보관
   ⑥ 원인 진단 및 Informix Tech Support Center 연락

3) DISK 장애로 DATA ACCESS 불가
   ① DISK 장애 여부 확인 및 교체
   ② 테이프에서 복구 여부 결정(Informix Tech Support 연락 후 결정)
   ③ 장애 DISK 가 rootdbs 인 경우는 전체 복구
   ④ ontape -r
   ⑤ 장애 DISK 가 rootdbs 가 아닌 다른 Dbspace 인 경우
   ⑥ ontape -r dbspace name
   ⑦ online.log 및 onstat -d 명령 통해 복구 여부 확인

7. SQL

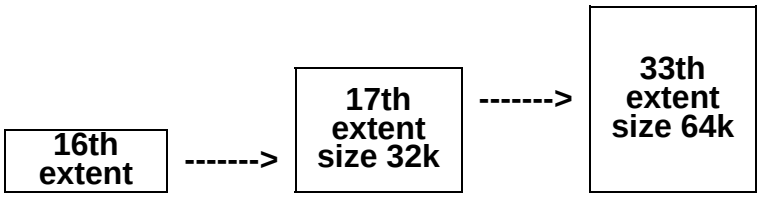
- DB 생성전 결정할 사항
  . The logging mode
  . ANSI standard compliance
  . database가 들어갈 dbsapce를 지정(default root dbsapce)
- DataBase Create
  1. CREATE DATABASE *dbname* [IN *dbspace*] [WITH [LOG, BUFFERED LOG, LOG MODE ANSI]]
  2. 예
     . CREATE DATABASE zzz
       root dbspace, no logging, DB명은 zzz
     . CREATE DATABASE zzz IN dbsmaster WITH LOG
       dbsmaster dbspace, unbuffered logging, DB명은 zzz
- Tables and Dbspace
  1. database가 있는 dbspace와는 다른 dbsapce에 table을 생성할수 있다.
     이 경우 system catalog는 database에 data는 지정한 dbspace에 저장.

- Extent Growth
  1. Concatenation



new extent is physically adjacent to existing extent.

  2. Doubling



extent size automatically double every 16 extents

  3. Manual Modification
     ALTER TABLE 문장을 사용 변경한다.

  4. 만약 dbspace의 여유분이 없을 경우는 minimum available space만 사용된다.

- Table Lock Modes

. Lock mode는 table 생성시 또는 ALTER 구문을 사용 변경할 수 있다.

1. Page locking

. concurrency, availability가 감소.

2. Row locking

. locking에 대한 resource 할당이 증가.

- Creating a Table

```
CREATE TABLE tabname(
    col1 SERIAL NOT NULL,    Columns and their data types
    col2 TEXT IN TABLE,
    col3 BYTE IN blobdbs,
    col4 ...)
    IN dbsname               Location of the table
    EXTENT SIZE 64
    NEXT SIZE 32             Size of the initial and next extent in kilobyte
    LOCK MODE row            Lock level
```

1. CREATE TABLE 문장은 다음과 같은 일을 한다.

. **systables**, **syscolumns** table에 table과 column의 정보를 insert

. dbspace에 table을 위한 extent size 만큼의 space를 할당.

. table lock level을 set

- Create INDEXS

1. Index 생성시에는 별도의 dbspace에 생성 한다.

```
create table chur   (
                    id integer,
                    name char(20),
                    phone char(20),
                    address char(100)
   )   in datadbs1 extent size 32 next size 32 lock mode row;

   create unique index x_chur_0 on chur(id) in ixdbs1;
   alter table chur add constraint primary key (id) constraint c_chur_0;

   create unique index x_chur_1 on chur(name) in ixdbs1;
   alter table chur add constraint unique (name) constraint c_chur_1;
```

- Storing BLOB data

1. BLOB data는 tablespace 또는 blobspace 저장할 수 있다.

2. tablespace에 저장할 경우 BLOB은 다른 data가 들어있는 row에 함께 저장할 수

없다. BLOBs will use their own pages.

3. BLOB data가 2 page보다 작은 경우는 tablespace에 저장하여도 performance엔 지장이 없다.

- Creating a Temporary Table

1. TEMP TABLE
   . session이 살아있는 동안 가능
   . DataBase가 close되면 delete된다(또는 DROP TABLE 문장)
   . systables, syscolumns에 insert 않됨.
   . ALTER TABLE 문장 사용 불가.
   . INDEX create.
   . 반든시 No Logging mode
   CREATE TEMP TABLE *temptable*(col1 CHAR(3), ...) **WITH NO LOG**;
2. DBSPACETEMP 환경변수

- Altering a Table

1. Add a column
   . 기존 data들의 column값은 NULL.
   . BEFORE를 사용 column의 위치를 지정할 수 있다.
   . ALTER TABLE *tablename* ADD *col5* INTEGER BEFORE *col3*
2. Modify a column
   . data type, the length or allow/disallow null
   . ALTER TABLE *tablename* MODIFY *col2* INTEGER NOT NULL
3. Drop a column
   . ALTER TABLE *tablename* DROP *col2*

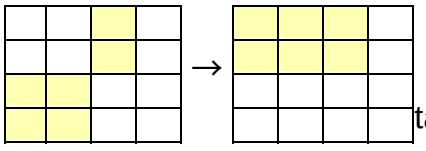
- In-Place ALTER TABLE:ADD column

. $oncheck -pT *database:table*

- Altering the Next Extent Size and Lock Mode

. ALTER TABLE *tablename* MODIFY NEXT SIZE *300*
. ALTER TABLE *tableanme* LOCK MODE *(ROW)*

- Data Space Reclamation



the table to be re-written, freeing unused space for other tables

. ALTER INDEX *indexname* TO CLUSTER

. 이 방법 보다는 data unload 후 Table 재생성 data load 하는게 좋다.

- Renaming Columns, Tables, and Database
  1. Rename a column
     . RENAME COLUMN *tablename.colname* TO *colrename*
     . sysviews, syschecks table의 viewtext 자동 변경
  2. Rename a table
     . RENAME TABLE *tablename* TO *tablerename*
     . references to the table within any views are changed.
     . CREATE TRIGGER *triggername*
       INSERT ON *tablename1* REFERENCING NEW AS new ← **event**
       FOR EACH ROW (INSERT INTO *tablename2*); ← **action**
       event내의 tablename1은 자동 변경되지만,
       action내의 tablename2는 변경않됨
  3. Rename a database
     . RENAME DATABASE *dbname* TO *dbrename*

- Dropping Tables and Databases
  1. Drop Table
     . All references to the table in system catalog tables are deleted
     . table에 제공된 space is freed
     . DROP TABLE *tablename*
  2. Drop Database
     . The system catalog tables are dropped
     . 모든 table에 제공된 space is freed
     . The data is no longer accessible
     . DROP DATABASE *dbname*

- DBSCHEMA Utility

. DBSCHEMA는 database나 table에 대한 CREATE TABLE, CREATE INDEX, GRANT,

CREATE SYNONYM, CREATE VIEW 문장을 포함한 SQL command file을 만드는데

사용한다.

. USAGE:

dbschema [-t *tabname*] [-s *user*] [-p *user*] [-r *rolename*] [-f *procname*]
[-hd *tabname*] **-d** *dbname* [-ss] [*filename*]

-t	table or view, table 전체는 all
-s	synonyms created by user name, "all" for all users
-ss	generate server specific syntax (lock mode, extent size, dbspace name)
-p	permissions granted to user name, "all" for all users
-r	create and grant of the role,"all" for all roles (Not a valid option for SE)
-f	stored procedure name, "all" for all stored procedures
-d	database name
-hd	Histograms of the distribution for columns of of a specified table, a specific table column, or \"all\" for all tables.

Fragmentation

- Fragmentation
  1. The distribution of data from one table across separate dbspaces.

- Advantages of Fragmentation
  1. Parallel scans
  2. Balanced I/O
  3. Finer granularity of archives and restores
  4. Higher availability
     . to skip unavailable fragments in a table.

- Types of Distribution Schemes
  . FRAGMENT 구문은 EXTENT 또는 LOCK MODE 구분 앞에 위치해야한다.
  1. Round Robin
     . CREATE TABLE *tablename*(
     col1 INTEGER, ...)
     FRAGMENT BY ROUND ROBIN IN *dbspace1*, *dbspace2*
     EXTENT SIZE ...;
     . dbspace는 적어도 2개 이상 기술해야한다.
     . You should calculate EXTENT SIZE and NEXT SIZE for an average size fragment.
     . 장점 : column값에 의존하지 않기 때문에 update시 row의 이동이 없다.
     단점 : Query optimizer is not able to eliminate fragments

  2. Expression
     . CREATE TABLE *tablename*(
     col1 INTEGER, ...)
     FRAGMENT BY EXPRESSION
     col1 > 1000 IN *dbspace1*,
     col1 > 500 IN *dbspace2,*
     REMAINDER IN *dbspace3*
     EXTENT SIZE ...;
     . row의 column을 평가하여 조건에 맞는 dbspace에 저장한다.
     . REMAINDER는 ELSE 구문과 같다.
     . 조건에 맞는 구문이 여러개인 경우는 조건에 맞는 첫 번째 dbspace에 저장한다.
     . >, <, >=, <=, IN, BETWEEN, AND, OR
     . 장점 : Fragments may be eliminated from query scans.
     Data can be segregated to support a archiving strategy.
     fragment level의 grant 권한을 부여.

Unequal data distribution can be created to offset an unequal frequency of access.

. 단점 : CPU resource를 요구

- Fragmenting by Expression :Guidelines
  1. 항상 scan하기 때문에 REMAINDER절은 피해라
  2. Attempt to balance I/O across disks.
  3. expression은 간단하게
  4. 조건문의 조건검색은 가급적 적게 하도록
  5. data type conversion 조건문은 피해라.
  6. Optimize data loads by placing the most frequently accessed fragment first in your fragmentation statement.
  7. 중요한 이익이 기대되지 않으면 fragment하지 마라.

- Fragmented Indexes
  . index를 round robin 방식 처럼 fragment할 수는 없다.
  . index and data pages are separate.
  . dbspace를 명시하지 않은 경우 the index will default to the same fragmentation strategy as the table.

- CREATE INDEX Statement

```
CREATE INDEX indexname ON tablname (col1)
    FRAGMENT BY EXPRESSION
    col1   > 1000 IN dbspace1,
    col2 <= 1000 IN dbspace2;
```

- ROWIDS
  . fragment된 table의 rowid는 unique하지 않다.
  . rowid를 unique하게 하려면 WITH ROWIDS를 사용한다.
  . CREATE TABLE tablename(...)
    WITH ROWIDS
    FRAGMENT ...
  . informix는 unique row access에서 rowid보다는 primary key 사용을 추천한다.
  . 각각의 row에 rowid column 4 byte가 add.

- Initializing a New Fragmentation
  1. non-fragmentation --> fragmentation
     . ALTER FRAGMENT ON TABLE *tablename*
       INIT FRAGMENT BY ROUND ROBIN IN *dbspace1, dbspace2*
  2. round robin --> expression
     . ALTER FRAGMENT ON TABLE *tablename*

INIT FRAGMENT BY EXPRESSION
MOD(col1, 2) = 0 *dbspace1,*
MOD(col1, 2) = 1 *dbspace2*

3. fragmentation --> non-fragmentation
. ALTER FRAGMENT ON TABLE *tablename*
INIT IN *dbspace1*

- Adding an Additional Fragment
. ADD 명령이 실행되는 동안 해당 row는 새로운 dbspace로 이동
1. Expression
. ALTER FRAGMENT ON TABLE *tablename*
ADD cpl1 > 3000 IN *dbspace4*
. ALTER FRAGMENT ON TABLE *tablename*
ADD cpl1 <= 3000 OR col1 = 3500 IN *dbspace3* **BEFORE** *dbspace4*
. BEFORE, AFTER구분을 사용 조건문의 위치를 정할 수 있다.
. BEFORE, AFTER구분이 없는 경우는 조건문 끝에 위치한다.
단 REMAINDER절 앞에 위치한다.
. 위 예제에서 col1의 값이 3200인 경우 row는 dbspace3로
만약, dbspace3을 추가할 때 **BEFORE** 구문을 사용하지 않았을 경우
row는 dbspace4로 저장된다.
2. Round robin
. ALTER FRAGMENT ON TABLE *tablename*
ADD *dbspace4*

- Dropping a Fragment
. drop 하려는 fragment에 있는 모든 row 또는 index를 다른 fragment로
옮기고 fragment는 drop 된다.
. expression fragment에서는 remainder fragment로
drop 하려는 fragment에 있는 row가 이동한다.
. ALTER FRAGMENT ON TABLE *tablename* DROP *dbspace1*
. ALTER FRAGMENT ON INDEX *indexname* DROP *dbspace1*
. Dropping the number of fragment below two is not allowed
- Modifying an Existing Fragment
. 조건문(expression) 또는 dbspace는 MODIFY 구문으로 수정 가능.
. 조건문(expression) 변경
ALTER FRAGMENT ON TABLE *tablename*
MODIFY ***dbspace1*** TO col1 > 3000 IN ***dbspace1***
. dbspace 변경
ALTER FRAGMENT ON TABLE *tablename*
MODIFY ***dbspace3*** TO REMAINDER IN ***dbspace5***

. MODIFY시 row의 이동이 있는데, 이 row에 대해 조건문(expression)에 맞는 fragment가 없는 경우는 error return and ALTER FRAGMENT will fail

- Attaching and Detaching Tables
  1. Attach
     . ALTER FRAGMENT ON TABLE *table1*
       ATTACH *table1, table2*
     . table1과 table2는 identical schema, different dbspace
     . No referential, primary key, unique, or NOT NULL constraint
     . table2는 serial column이 table1은 check constraint가 없어야 한다.
  2. Detach
     . ALTER FRAGMENT ON TABLE *table1*
       DETACH *dbspace2 table2*
     . DETACH command will not work on tables with rowids
- Skipping Inaccessible Fragment

Concurrency Control

- Types of Concurrency
  . Read Concurrency
  . Update Concurrency
  . Lock의 종류
    1. Exclusive lock
    2. Shared lock
    3. Update lock

- Read Concurrency : Four isolation Levels
- Dirty Read
  . No lock check (∴ dirty data일수있다)
  . No lock (∴ data 가 변할수있다)
- Committed Read
  . Lock check (∴ 적어도 commit data)
  . No lock
- Cursor Stability
  . Lock check
  . Row lock (cusor가 읽고있는 data)
  . cursor를 사용하지 않으면 Committed read와 같다.
- Repeatable Read
  . 질의에 만족하는 모든 row를 lock
- Setting the Level of isolation
  . SET ISOLATION TO [*dirty read, committed read, cursor stability, repeatable read*]
  . logging mode가 아닌 경우는 Dirty Read고 바꿀수없다
  . session이 살아있는 동안 유지된다.

- Update Concurrency : Locking Granularity
  1. Database level : DATABASE stores7 EXCLUSIVE;
  2. Table level : LOCK TABLE customer IN [SHARE | EXCLUSIVE] MODE;
     UNLOCK TABLE customer;
  3. Page level
  4. Row level
  5. Key level
- SET LOCK MODE TO [WAIT | NOT WAIT | WAIT 20];

Sample Query

- Using Additional Functions

1. SELECT customer_num, call_dtime FROM cust_calls WHERE call_dtime >= “1994-07-01 00:00”;

   SELECT customer_num, call_dtime FROM cust_callss WHERE call_dtime >= **DATE**(‘7/1/94’);

2. SELECT customer_num, **MONTH**(call_dtime) FROM cust_calls;

3. SELECT customer_num, MONTH(call_dtime) FROM cust_calls;

   WHERE MONTH(call_dtime) = MONTH(**CURRENT**)

   AND YEAR(call_dtime) < YEAE(**CURRENT**);

4. SELECT customer_num, MONTH(call_dtime) FROM cust_calls;

   WHERE MONTH(call_dtime) = MONTH(**TODAY**)

   AND YEAR(call_dtime) < YEAE(**TODAY**);

5. SELECT customer_num, **EXTEND**(call_dtime, month to minute) FROM cust_calls;

6. SELECT TO_CHAR(call_dtime, ‘%A %B %d %Y %R’) FROM cust_calls;

   Result : (expression) Sunday June 12 1994 08:20

%A : Full weekday name	%B : Full month name
%d : Day of month as a decimal	%m : Month as a decimal
%Y : Year as a 4-digit decimal	%R : Time of day in 24-hour format

7. SELECT * FROM cust_calls

   WHERE call_dtime >= **TO_DATE**(‘1994-06-12 08:20’, ‘%Y-%m-%d %H:%M’);

8. SELECT company FROM customer WHERE **UPPER**(city) = “PALO ALTO”;

9. SELECT company FROM customer WHERE **LOWER**(city) = “palo alto”;

10. SELECT company FROM customer WHERE **INITCAP**(city) = “Palo Alto”;

11. SELECT **phone[1,3]**, customer_num FROM customer WHERE **phone[1,3]** = ‘415’;

12. SELECT **“customer name is “ || lname** FROM customer;

13. SELECT **‘(‘ || phone[1,3] || ‘)’ || phone[5,12]**, fname, lname FROM customer;

14. SELECT **TRIM(fname)** || lname FROM customer;

15. SELECT **HEX(256)** FROM dummy_table;

16. SELECT **TAN(1)** FROM dummy_table;

17. SELECT **ABS(-256)** FROM dummy_table;

18. SELECT **MOD(257,8)** FROM dummy_table;

19. SELECT company, **NVL(address2,”No Second line”)** FROM customer;

20. SELECT company, state, **CASE WHEN state="CA" THEN "Region1"**

WHEN state="AZ" THEN "Region2"

ELSE"Region3" END region FROM customer;

21. SELECT company, state **DECODE(state,"CA:,"Region1","AZ","Region2","Region3")**

FROM customer;

22. SELECT **LENGTH(fname)**, fname FROM customer;

23. SELECT **USER, SITENAME, DBSERVERNAME**, * FROM dummy_table;

24. SELECT c.customer_num, lname, order_num FROM customer c, **OUTER** orders o

WHERE c.customer_num = o.customer_num;

25. SELECT * FROM manufact WHERE manu_code **IN**

(SELECT manu_code FROM manufact WHERE lead_time > "1");

26. SELECT * FROM manufact WHERE manu_code **NOT IN**

(SELECT manu_code FROM manufact WHERE lead_time > "1");

27. **INSERT INTO** closed_orders **SELECT** * FROM orders WHERE paid_date IS NOT NULL;

28. **UPDATE** orders SET ship_instruct = 'ASAP' WHERE order_date > '6/1/94'

AND customer_num **IN (SELECT** customer_num FROM customer WHERE city='Redwood City');

29. **LOAD FROM** 'new_state.unl' INSERT INTO state;

30. **LOAD FROM** 'new_state.unl' **DELIMITER '|**' INSERT INTO state;

31. **LOAD FROM** 'new_state.unl' INSERT INTO customer**(customer_num,fname,lname)**;

32. **UNLOAD TO** 'new_state.unl' **DELIMITER '|'** SELECT * FROM customer;

33. **UNLOAD TO** 'new_state.unl' SELECT * FROM customer WHERE customer_num > 120;

34. INSERT INTO customer(fname,lname) VALUES ('Donald','Duck');

SELECT UNIQUE **DBINFO('sqlca.sqlerrd1')** FROM sysdefaults;

35. INSERT INTO customer(fname,lname) VALUES ('Donald','Duck');

SELECT UNIQUE **DBINFO('sqlca.sqlerrd2')** FROM sysdefaults;

- Remote Connection

1. CONNECT TO **'db@server'** USER 'username' USING passwd;
2. DATABASE **"//server/db"**
3. SELECT a.*, b.* FROM customer a, **testdb:customer b**

WHERE a.customer_num = b.num;

4. SELECT * FROM orders, **testdb@server:tabname**

   WHERE orders.key = **testdb@server:tabname.key**;

- Query EXPLAIN

1. Query Optimization
   A. Read fewer rows
   B. Avoid sorts
   C. Sort fewer rows
   D. Sort on simple keys
2. SET EXPLAIN ON; SET EXPLAIN OFF;

```
SQL:  New  Run  Modify  Use-editor  Output  Choose  Save  Info  Drop  Exit
Run the current SQL statements.
---------------------- stores7@dbserver ------ Press CTRL-W for Help --------

SET EXPLAIN ON;
SELECT * FROM customer WHERE zipcode matches '94*';
SELECT * FROM customer WHERE zipcode matches '*40';
SET EXPLAIN OFF;
```

$ vi sqexplain.out

```
QUERY:
------
SELECT * FROM customer WHERE zipcode matches '94*'
Estimated Cost: 1
Estimated # of Rows Returned: 6

   1) informix.customer: INDEX PATH
      (1) Index Keys: zipcode     (Serial, fragments: ALL)
            Lower Index Filter: informix.customer.zipcode MATCHES '94*'
QUERY:
------
SELECT * FROM customer WHERE zipcode matches '*40'
Estimated Cost: 2
Estimated # of Rows Returned: 6
   1) informix.customer: SEQUENTIAL SCAN
            Filters: informix.customer.zipcode MATCHES '*40'
```

3. (o)SELECT * FROM customer WHERE zipcode MATCHES ‘94*’;
   ≠ (x)SELECT * FROM customer WHERE zipcode MATCHES ‘*94*’;
4. (o)SELECT * FROM customer WHERE zipcode[1,2] = ‘94’;
   ≠ (x)SELECT * FROM customer WHERE zipcode[2,3] = ‘40’;
5. (o)SELECT * FROM customer, order WHERE customer.customer_num = orders.customer_num
   ≠ (x)SELECT * FROM customer, order WHERE customer.text = orders.text
6. SELECT * FROM customer WHERE EXISTS
   (SELECT customer_num FROM orders
   WHERE orders.customer_num = customer.customer_num
   AND order_date = date('1998/6/18'));
   ≠ SELECT c.* FROM customer c, orders o
   WHERE c.customer_num = o.customer_num AND o.order_date = date('1998/6/18');

- Optimizer Directives

1. --+ Directives
2. {+ Directives }
3. /*+ Directives */

4. Directives
   A. Join-Order Directive
   - ORDERED : join-order

```
SELECT --+ ORDERED
name, title, salary, dname
FROM dept, job, emp
Where title = 'clerk'
AND loc = 'Palo Alto'
AND emp.dno = dept.dno
AND emp.job= job.job
```

   B. Access-Method Drective :
   - INDEX : Perform a full-table scan.
   - AVOID_INDEX : Do not use any of the indexes listed.
   - FULL : Use the index specified to access the table.
   - AVOID_FULL : Do not perform a full-table scan on the listed table.

```
SELECT {+INDEX(EMP dept_no)}

SELECT {+AVOID_INDEX(EMP loc_no, job_no), AVOID_FULL(EMP)}

SELECT {+FULL(e)} name, salary FROM emp e;

SELECT {+AVOID_FULL(e), INDEX(e salary_idx)} name, salary
    FROM   emp e   WHERE   e.dno = 1   AND e.salary > 5000;

SELECT {+EXPLAIN AVOID_FULL(customer)} *
    FROM   customer;
```

   C. Join-Method Drective :
   - USE_NL : Nested Loop Join
   - USE_HASE : Hash Join
   - AVOID_NL :
   - AVOID_HASE :

```
SELECT /*+ USE_HASH (dept /BUILD)
Force the optimizer to use the dept table to
construct a hash table */
name, title, salary, dname
FROM emp, dept, job
WHERE loc = 'Phoenix'
AND emp.dno = dept.dno
AND emp.job = job.job
```

```
SELECT --+USE_NL(dept)
          Name, title, salary, dname
    FROM emp, dept, job
 WHERE loc='Palo Alto'
     AND emp.dno = dept.dno
     AND emp.job = job.job
```

D. Optimization-Goal Drective

- ALL_ROWS:
- FIRST_ROWS :

```
SELECT --+ FIRST_ROWS
lname, fname FROM customer
ORDER BY sname
```

```
SELECT {+FIRST_ROWS
Return the first screenful of rows as fast as possible}
FIRST 50 lname, fname, bonus
FROM emp
ORDER BY bonus DESC
```

```
SELECT --+FIRST_ROWS INDEX(test ix_mul)
FIRST 10 a,b,fix_test(a,b)
FROM test
ORDER BY 4;
```