ALTIBASE ADVANCE

ALTIBASE ADVANCE

CONTENTS

- DBMS TUNING
- ALTIBASE OPERATION
- MONITORING TOOL
- TECHNICAL SUPPORT

ALTIBASE ADVANCE

DBMS TUNING

CONTENTS

DBMS TUNING

DBMS TUNING

❖ DBMS 튜닝이란?

- DBMS의 성능 향상을 목적으로 진행하는 일련의 작업
- 어플리케이션 튜닝과 SERVER 튜닝으로 나뉨

❖ DBMS 튜닝 목표

- DBMS call을 최소화
- Prepare를 최소화
- 디스크 I/O를 최소화
- CPU 사용률을 최소화

❖ 관점에 따른 튜닝 종류

- Design 관점
- Application 관점
- DBMS SERVER 관점
- OS 관점

DBMS TUNING

❖ DBMS CALL 최소화

- ➢ 어플리케이션에서 DBMS로 호출하는 횟수가 많은가?
- ➢ DBMS call을 최소화하여 어플리케이션의 성능을 향상

❖ DBMS CALL 최소화 방법

- ➢ array processing으로 처리 : Array 단위 Fetch, Bulk Insert / Update / Delete
- ➢ Fetch call을 최소화 : 부분범위처리, ArraySize 조정

❖ DBMS CALL 최소화 예제

➢ APRE * C/C++

```
struct
    {
        char    gno[3][10+1];
        char    gname[3][20+1];
        char    goods_location[3][9+1];
        int     stock[3];
        double  price[3];
    } a_goods2;
…
EXEC SQL INSERT INTO GOODS VALUES (:a_goods2);
```

➢ JAVA

```
1. array processing
      for(…){
            …
            pstmt.addBatch();
      }
      pstmt.executeBatch();

2. Fetch call 최소화
      …
      stmt.setFetchSize(100);
       …
```

❖ PREPARE 최소화

- PREPARE 비용
  - 단순 SQL 처리시 약 70%정도가 Prepare 비용
- PREPARE 최소화 방법
  - C/C++/APRE* 프로그램 작성 시 bind 변수
  - Java 프로그램 작성 시 PreparedStatement를 이용
  - SQL Plan Cache에 적재되어있는 Execution Plan을 재 사용
- 실행 계획을 공유하지 못해 PREPARE를 다시 수행하는 경우
  - 공백이 다르거나 줄바꿈이 다른 경우
  - 주석이 다른 경우
  - 힌트가 다른 경우
  - 조건절 비교 값이 다른 경우
  - 대소문자가 다른 경우

❖ 디스크 I/O 최소화

➢ 디스크 DBMS I/O

- 디스크 DBMS는 한건의 레코드만 읽어도 page 단위로 I/O가 일어남

➢ 메모리 I/O vs 디스크 I/O

Memory I/O	Disk I/O
전기적인 신호에 의한 입출력	액세스 Arm이 움직이면서 헤드를 통해 입출력
속도가 빠르다	속도가 느리다

➢ 디스크 I/O 최소화 방법

- 자주 access되는 테이블은 메모리 테이블로 구성
- 필요한 최소 page만 읽도록 SQL 작성
- Buffer hit율 증가
- Random access 감소
- Sequential access 증가
- Single I/O page read와 multi I/O page read를 고려한 SQL이 index scan/ full table scan 중 유리한 쪽 선택

DBMS TUNING

❖ CPU 사용률 최소화

- 메모리 DBMS는 I/O 보다는 CPU사용률이 튜닝 factor
- SQL 1개 수행 동안 CPU 1EA(100%) 사용

❖ CPU 사용률 최소화 방법

- 메모리 테이블은 전체 건수를 SQL하는 경우라도 full table scan 보다는 index scan 이 빠름
- index scan을 하도록 SQL 작성

❖ 튜닝을 위한 점검 사항

➢ Application 관점

- TRANSACTION 증가
- 매번 connect - disconnect 반복
- SQL 실행 시 빈번한 prepare 수행
- full table scan 등의 오래 수행되는 SQL
- lock을 잡고 있는 SQL

➢ ALTIBASE 관점

- 체크포인트 I/O
- logfile writing
- service thread 병목
- 메모리 ager
- buffer
- SQL plan cache

➢ OS 관점

- OS 설정
- 디스크 I/O 성능

❖ ALTIBASE 튜닝 도구

- Explain Plan
- Profiling
- Meta 테이블 & 성능 뷰
- OS 유틸리티 or 명령어(ps, top, nmon, glance)

❖ ALTIBASE 튜닝 도구

- Explain Plan
  - SQL의 실행 계획을 확인하고자 할 때 설정
  - iSQL에서 EXPLAIN PLAN을 설정 후 확인 가능
- SQL 수행 후 출력하며, plan tree와 access 횟수 등을 출력

```
iSQL> ALTER SESSION SET EXPLAIN PLAN = ON;
```

- Plan tree 출력하지 않음

```
iSQL> ALTER SESSION SET EXPLAIN PLAN = OFF;
```

- SQL 수행하지 않고 Plan tree 만 출력

```
iSQL> ALTER SESSION SET EXPLAIN PLAN =ONLY;
```

❖ ALTIBASE 튜닝 도구

➢ Explain Plan 예제

- EXPLAIN PLAN = ON 설정

```
iSQL> ALTER SESSION SET EXPLAIN PLAN = ON;
iSQL> SELECT eno, ename, salary FROM employee WHERE eno=1;
ENO         ENAME                SALARY
------------------------------------------------
1           EJJUNG
1 row selected.
------------------------------------------------------------
PROJECT ( COLUMN_COUNT: 3, TUPLE_SIZE: 47 )
 SCAN ( TABLE: EMPLOYEE, INDEX: SYS_C0011065, ACCESS: 1, SELF_ID: 2 )
------------------------------------------------------------
```

- EXPLAIN PLAN = OFF 설정

```
iSQL> ALTER SESSION SET EXPLAIN PLAN  = OFF;
iSQL> SELECT eno, ename, salary FROM employee WHERE eno=1;
ENO         ENAME                SALARY
------------------------------------------------
1           EJJUNG
1 row selected.
```

❖ ALTIBASE 튜닝 도구

➢ Explain Plan 예제

- EXPLAIN PLAN = ONLY 설정

```
iSQL> ALTER SESSION SET EXPLAIN PLAN = ONLY;
iSQL> SELECT eno, ename, salary FROM employee WHERE eno=1;
ENO         ENAME                SALARY
------------------------------------------------
No rows selected.
------------------------------------------------------------
PROJECT ( COLUMN_COUNT: 3, TUPLE_SIZE: 47 )
 SCAN ( TABLE: EMPLOYEE, INDEX: SYS_C0011065, ACCESS: ??, SELF_ID: 2 )
------------------------------------------------------------
```

❖ ALTIBASE 튜닝 도구

- Profiling
  - 사용자가 수행한 SQL 실행 정보를 수집하는 기능
  - system level에서 property로 설정
  - Profiling 된 파일은 $ALTIBASE_HOME/trc 에 저장됨
- Profiling 으로 저장되는 정보
  - SQL를 수행한 CLIENT 정보
  - SQL 처리의 통계정보
    - 실행시각
    - SQL
    - 수행시간
    - 인덱스 정보
    - 버퍼 / 디스크 접근 비용
    - 실행 계획

❖ ALTIBASE 튜닝 도구

- Meta 테이블
  - 데이터 베이스 객체에 관한 모든 정보를 기록하기 위한 시스템 정의 테이블
  - 소유자는 SYSTEM_ (접속 불가)
  - DDL 수행 시 시스템에 의해 변경
- 성능 뷰
  - 인스턴스와 성능 관련 정보 저장
  - 시스템 메모리, 프로세스 상태, 세션, 버퍼 등의 최신 정보 제공
  - read only

❖ ALTIBASE 튜닝 도구

➢ OS 및 ALTIBASE 점검을 위한 OS 명령어 및 유틸리티

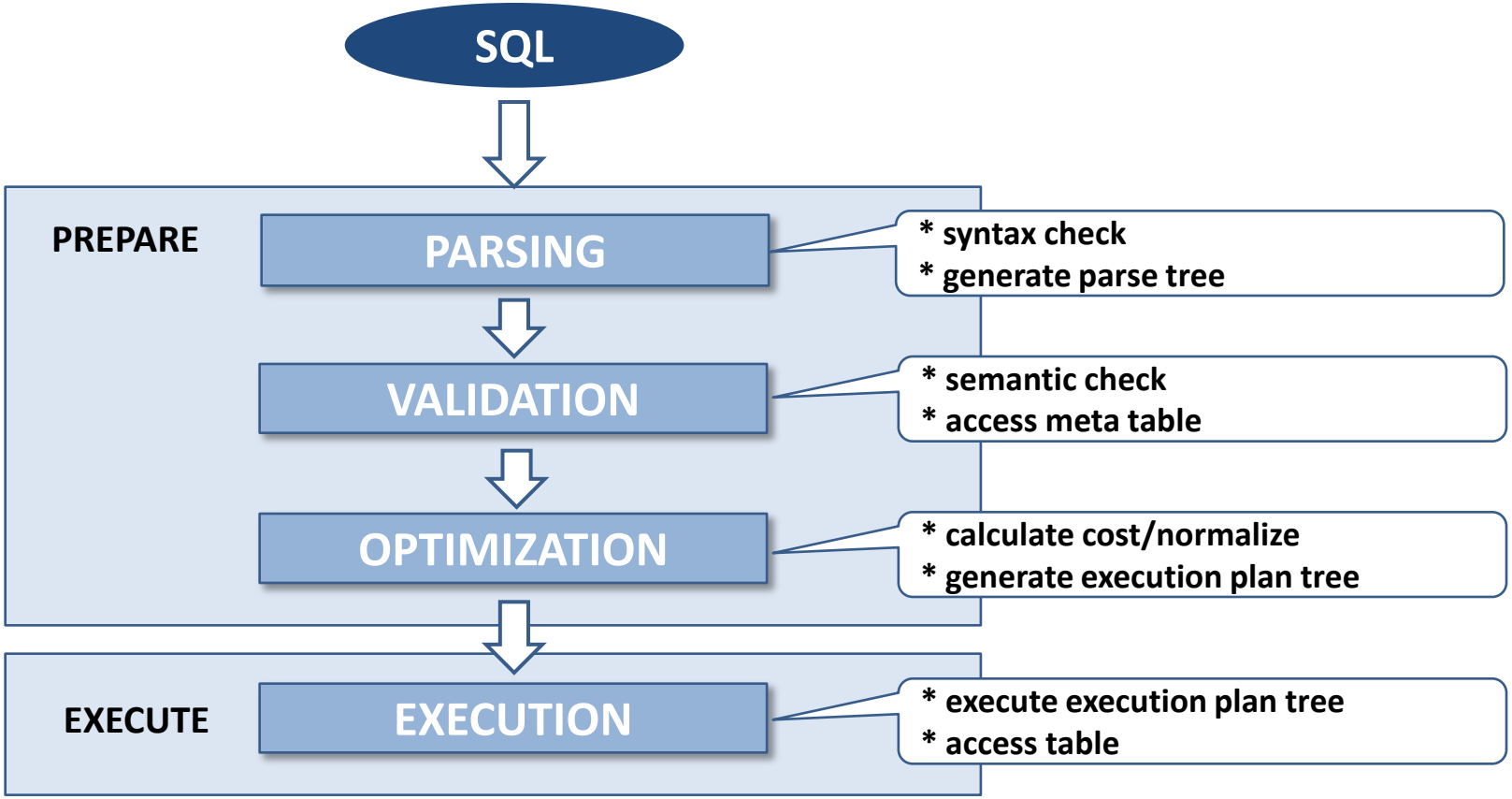
	AIX	HPUX	LINUX	Solaris
Performance monitor	top topas nmon	top glance	top	top
System activity reporter	sar	sar	sar	sar
Virtual Memory statistics	vmstat	vmstat	vmstat	vmstat
I/O statistics	iostat	iostat	iostat	iostat
Error log	errpt	dmesg	dmesg	emesg /var/adm/syslog

DBMS TUNING

SQL TUNING

SQL TUNING

❖SQL 처리 절차



❖ 디스크 테이블과 메모리 테이블의 SQL 처리 방법 차이

Item	Memory table	Disk table
Object identifier	Pointer	OID(RID)
Buffer management	N/A	Limited buffer
Join methods	One-pass algorithms	Multi-pass algorithms
Main cost	CPU	Disk
Index selection	Minimize record access	Minimize 디스크 I/O
Cost factor	T(R), V(R.a), etc	+ B(R), M

T(R) : 테이블의 레코드 수 **B(R) :** 테이블의 **page** 수
V(R.a) : 컬럼의 **Value Cardinality** **M : Memory Buffer** 수

- One- pass algorithms
  - 가용메모리 버퍼에 중간 결과를 모두 적재할 수 있을 때 사용
- Multi-pass algorithms
  - 중간 결과를 메모리상에 모두 적재할 수 없을 때 버퍼교체를 최소화 하기 위해 사용

❖ 인덱스

- 검색 시 성능 향상을 위해 테이블과는 별도로 저장되는 객체
- 인덱스 대상 컬럼 값을 sorting하여 저장

Memory Index	Disk Index
데이터베이스가 구동될 때마다 Memory에 생성	인덱스 생성 시점에 Disk에 생성
실제 테이블의 데이터에 대한 포인터만 저장 (16 bytes)	컬럼의 값과 테이블의 레코드 주소 값이 저장
DML 시 Logging을 하지 않음	DML 시 Logging

❖ ALTIBASE 인덱스 특징

- UNIQUE, PRIMARY KEY로 지정한 컬럼은 내부적으로 Unique 인덱스가 생성
- 테이블과 별도의 디스크에 분리 저장 권고
- Btree 인덱스와 Rtree 인덱스 만 지원
  - Rtree 인덱스 는 Geometry 를 위한 다차원 데이터 처리 시 사용
  - Reverse, Bitmap, Global partitioned 인덱스 등은 미 지원

※ 6.3.1 버전부터 제공

❖ FUNCTION BASED INDEX 개념

- 함수 또는 수식의 결과 값을 기반으로 생성하는 인덱스
- WHERE절 함수 또는 산술 표현을 자주 사용시 빠른 검색 속도 보장
- QUERY_REWRITE_ENABLE 1로 변경 후 인덱스 사용 가능

❖ FUNCTION BASED INDEX 생성 예제

```
iSQL> ALTER SESSION SET EXPLAIN PLAN = ON;
Alter success.
iSQL> CREATE INDEX  NVL_IDX  ON  employee (NVL(salary, 0));
Create success.
 iSQL> SELECT eno, ename, salary FROM employee
         2 WHERE (NVL(salary, 0)) < 1000000;
ENO         ENAME                SALARY
------------------------------------------------
1           EJJUNG
7           HJMIN                500000
8           JDLEE
13          KWKIM                980000
20          DIKIM
5 rows selected.
------------------------------------------------------------
PROJECT ( COLUMN_COUNT: 3, TUPLE_SIZE: 35, COST: 0.31 )
 SCAN ( TABLE: EMPLOYEE, FULL SCAN, ACCESS: 20, COST: 0.24 )
------------------------------------------------------------
```

❖ FUNCTION BASED INDEX 생성 예제

```
iSQL> ALTER SESSION SET QUERY_REWRITE_ENABLE = 1;
Alter success.
iSQL> SELECT eno, ename, salary FROM employee
         2 WHERE (NVL(salary, 0)) < 1000000;
ENO         ENAME                SALARY
-------------------------------------------------
1           EJJUNG
8           JDLEE
20          DIKIM
7           HJMIN                500000
13          KWKIM                980000
5 rows selected.
-------------------------------------------------------------
PROJECT ( COLUMN_COUNT: 3, TUPLE_SIZE: 35, COST: 0.07 )
 SCAN ( TABLE: EMPLOYEE, INDEX: NVL_IDX, RANGE SCAN, ACCESS: 5, COST: 0.00 )
-------------------------------------------------------------
```

SQL TUNING

❖ SEQUENTIAL ACCESS

- 레코드 간 물리적인 순서에 따라 차례대로 읽어 나가는 방식
- FULL table scan 시 레코드를 읽는 경우
- 인덱스 리프 노드에 위치한 데이터들을 읽는 경우

❖ RANDOM ACCESS

- 레코드 간 순서를 따르지 않고 한 건의 레코드를 읽기 위해 한 page 씩 접근하여 읽는 방식
- 인덱스에 저장되어 있는 물리적인 레코드의 주소를 참조하여 테이블의 레코드를 찾아가는 경우

❖ 튜닝 방법

- sequential access에 의한 선택 비중 증가
- random access 발생량 감소

❖ OPTIMIZER 최적화 과정

- 조건절 분류
- ACCESS 방법 결정
- JOIN 순서 결정
- JOIN 방법 결정
- GROUP / AGGREGATE 연산 수행 방법 결정
- DISTINCT 절 수행 방법 결정
- SET 절 수행 방법 결정
- ORDER BY 절 수행 방법 결정
- PROJECTION 수행 방법 결정

SQL TUNING

❖ SQL 튜닝 절차

- 오래 수행되는 SQL 확인
- explain plan 설정
- set timing on 설정
- 실행 계획 확인
- SQL 변경 or 인덱스 설정 or 힌트 등을 사용하여 SQL 튜닝

❖ SQL 튜닝 방법

- 가능한 한번만 PREPARE
- 효율적인 인덱스 사용 여부 확인
- Driving 테이블의 적합성 확인
- 인덱스가 필요하면 추가
- HINT 활용
- 부분 범위 처리 권고(limit 등 활용)

❖ SQL 튜닝 한계

- Modeling 단계에서 성능 및 튜닝에 대한 부분이 고려되지 않으면 SQL 튜닝만으로 성능 향상 기대 어려움
- 매우 큰 데이터 집합이 리턴 되거나, 거대한 볼륨 전체를 핸들링하는 업무 같은 경우 성능 향상 어려움
- 다음의 경우 아키텍쳐 기반 튜닝 필요
  - 디스크 I/O 분산
  - 잘못된 데이터베이스 스키마 변경
  - 데이터 베이스 CALL 횟수 최소화
  - 네트워크 부하 분산
  - SQL PLAN CACHE 활용

DBMS TUNING

SQL PLAN

SQL PLAN

❖ 실행계획이란?

- SQL이 실행될 때 필요한 처리 절차.
- Optimizer에 의해 생성

❖ 실행계획 확인

- EXPLAIN PLAN 설정

```
ALTER SESSION SET EXPLAIN PLAN = {ON|ONLY|OFF}
  - ON : 실행 계획 + 수행 결과
  - ONLY : 실행 계획 만
  - OFF : 수행 결과 만
```

- 예

```
iSQL> ALTER SESSION SET EXPLAIN PLAN = ON;
iSQL> SELECT eno, ename, salary FROM employee WHERE eno=1;
ENO         ENAME                SALARY
------------------------------------------------
1           EJJUNG
1 row selected.
------------------------------------------------------------
PROJECT ( COLUMN_COUNT: 3, TUPLE_SIZE: 35 )
 SCAN ( TABLE: EMPLOYEE, INDEX: __SYS_IDX_ID_163, ACCESS: 1, SELF_ID: 2 )
------------------------------------------------------------
```

SQL PLAN

❖ 실행계획 용어

- PLAN 을 읽는 방법은 가장 안쪽부터, 위에서 아래의 순서로 읽음
- PLAN 용어 설명

```
iSQL> SELECT /*+ USE_NL(T1, T2 ) */ T1.i1 FROM t1, t2 WHERE t1.i1=t2.i1;
I1
-------------
No rows selected.
-------------------------------------------------------------
PROJECT ( COLUMN_COUNT: 1, TUPLE_SIZE: 4 ) ------------------  (4)
 JOIN (REF_ID:2) ---------------------------------------------------------  (3)
  SCAN ( TABLE: T1, FULL SCAN, ACCESS: 2, SELF_ID: 1 )    -----  (1)
  SCAN ( TABLE: T2, INDEX: T2_I1, ACCESS: 0, SELF_ID: 2 ) ------ (2)
   [ VARIABLE KEY ]
   OR
    AND
     T1.I1 = T2.I1
```

- COLUMN_COUNT = PROJECTION(select) 되는 column 수
- TUPLE_SIZE = COLUMN size 의 합
- ACCESS : [n] = 테이블에서 SCAN한 RID (ROWID)의 건수를 의미
- PLAN NODE = SCAN, HASH, SORT, PROJ
- SELF_ID : plan tree 로 구성됐을 때의 노드 ID
- VARIABLE KEY : key range 하는 술어

SQL PLAN

❖ PREDICATE를 자세히 보고자 할 경우

- System level의 property 설정

```
ALTER SESSION SET TRCLOG_DETAIL_PREDICATE=1
```

- 예제

```
iSQL> ALTER SYSTEM SET TRCLOG_DETAIL_PREDICATE=1;
Alter success.
iSQL> ALTER SESSION SET EXPLAIN PLAN = ON;
Alter success.
iSQL> SELECT e.ename, d.dname
        2 FROM employee e, department d
        3 WHERE e.dno = d.dno
        4 AND e.ename='KSKIM';
ENAME                 DNAME
------------------------------------------------------------
KSKIM                 CUSTOMER SUPPORT DEPT
1 row selected.
------------------------------------------------------------
PROJECT ( COLUMN_COUNT: 2, TUPLE_SIZE: 54 )
 JOIN
  SCAN ( TABLE: EMPLOYEE E, FULL SCAN, ACCESS: 20, SELF_ID: 2 )
   [ FILTER ]                ← predicate 정보
   E.ENAME = 'KSKIM'
  SCAN ( TABLE: DEPARTMENT D, INDEX: __SYS_IDX_ID_322, ACCESS: 1, SELF_ID: 3 )
   [ VARIABLE KEY ]                ←  predicate 정보
   OR
    AND
     E.DNO = D.DNO
------------------------------------------------------------
```

SQL PLAN

❖WHERE절에서 조건절 처리 유형

```
Index : T1(i1, i2, i3)
SQL : SELECT *
            FROM T1
            WHERE ? = 1
                        AND i1 > 1
                        AND i2 = 2
                        AND i4 = 'abc'
                        AND EXISTS ( SELECT * FROM T2 WHERE T2.a1 = T1.i3 )
             ;
```

분류	처리순서	설명	example
Key Range	2	인덱스를 사용하여 범위 검색이 가능한 조건	T1.i1 > 1
Key Filter	3	범위 검색은 안되나 인덱스의 키 값을 비교하여 검사가 가능한 조건	T1.i2 = 1
Constant Filter	1	테이블의 데이터와 관계 없이 한 번의 검사만으로 판단이 가능한 조건	? = 1
Filter	4	데이터에 대한 직접적인 비교가 필요한 조건	T1.i4 = abc
Subquery Filter	5	Subquery를 포함하고 있는 조건	EXISTS ()

SQL PLAN

❖ 인덱스가 있더라도 사용할 수 없는 조건절

- 인덱스를 사용할 수 없는 연산자

```
WHERE enname LIKE '_K%';
WHERE enname NOT LIKE 'KIM%';
```

- 인덱스 칼럼에 포함되었더라도 변형을 한 경우

```
WHERE TO_CHAR(c1) = ‘a’;
WHERE SUBSTR(ename,1,4) = 'alti';
WHERE salary * 12 < 30000000;
```

- 데이터 타입이 다른 경우(일부의 데이터 타입)

```
WHERE start_flag = 1; (start_flag 의 타입이 CHAR/VARCHAR인 경우)
```

- 결합 인덱스일 경우 첫번째 컬럼의 인덱스를 탈수 있는 조건이 없는 경우

```
WHERE c2 = ‘a’; (c1, c2 순서로 결합인덱스를 생성한 경우)
```

- ALTIBASE Optimizer 가 Index SCAN COST 가 더 소요된다고 판단하였을 때

SQL PLAN

❖ 비교 연산자에 따른 인덱스 사용 여부

분류	비교 연산자	가능 여부	비고
단순 비교	=	O	
	!=	O	
	<	O	
	<=	O	
	>	O	
	>=	O	
범위 비교	BETWEEN	O	
	NOT BETWEEN	O	
멤버 비교	IN	O	
	NOT IN	O	
패턴 비교	LIKE	O	가능: T1.i1 LIKE 'abc%' 불가: T1.i1 LIKE '%abc'
	NOT LIKE	X	

SQL PLAN

분류	비교 연산자	가능 여부	비고
NULL 비교	IS NULL	O	
	IS NOT NULL	O	
존재 비교	EXISTS	X	
	NOT EXISTS	X	
Quantify ANY	=ANY	O	
	!=ANY	O	
	<ANY	O	
	<=ANY	O	
	>ANY	O	
	>=ANY	O	
Quantify ALL	=ALL	O	
	!= ALL	O	
	< ALL	O	
	<= ALL	O	
	> ALL	O	
	>= ALL	O	

SQL PLAN

❖ 인덱스 컬럼과 VALUE의 데이터 타입 간 호환 여부

- ➢ 인덱스가 있어도 데이터 타입이 호환되지 않으면 인덱스 사용 불가

KEY \ VALUE	CHAR	VARCHAR	SMALLINT	INTEGER	BIGINT	NUMERIC	FLOAT	REAL	DOUBLE	DATE	BLOB	NIBBLE	BYTE	GEOMETRY
CHAR	O	O	X	X	X	X	X	X	X	X	-	-	-	-
VARCHAR	O	O	X	X	X	X	X	X	X	X	-	-	-	-
SMALLINT	X	X	O	O	O	O	O	O	O	-	-	-	-	-
INTEGER	X	X	O	O	O	O	O	O	O	-	-	-	-	-
BIGINT	X	X	O	O	O	O	O	O	O	-	-	-	-	-
NUMERIC	O	O	O	O	O	O	O	O	O	-	-	-	-	-
FLOAT	O	O	O	O	O	O	O	O	O	-	-	-	-	-
REAL	X	X	O	O	O	O	O	O	O	-	-	-	-	-
DOUBLE	O	O	O	O	O	O	O	O	O	-	-	-	-	-
DATE	O	O	-	-	-	-	-	-	-	O	-	-	-	-
BLOB	-	-	-	-	-	-	-	-	-	-	O	-	-	-
NIBBLE	-	-	-	-	-	-	-	-	-	-	-	O	-	-
BYTE	-	-	-	-	-	-	-	-	-	-	-	-	O	-
GEOMETRY	-	-	-	-	-	-	-	-	-	-	-	-	-	O

SQL PLAN

ALTIBASE 에서 지원하는 조인 방법

- Nested loops 조인 계열
- Sort Merge 조인 계열
- Hash-based 조인 계열

SQL PLAN

NESTED LOOPS 조인 수행 방법

- 선행 테이블에서 조건에 만족하는 레코드 검색
- 선행 테이블의 조인 키 값으로 후행 테이블 조인 수행
- 선행 테이블 조건에 만족하는 모든 레코드에 대해 반복 수행

NESTED LOOPS 조인 용어

- Driving 테이블, Outer 테이블
  - 먼저 Scan 하는 선행 테이블
- Lookup 테이블, Inner 테이블
  - 선행테이블에서 조인조건에 만족하는 레코드와 연결하는 후행 테이블
  - Random access로 찾아감
  - 조인 컬럼에 인덱스가 있어야 효율적

NESTED LOOPS 조인 특성

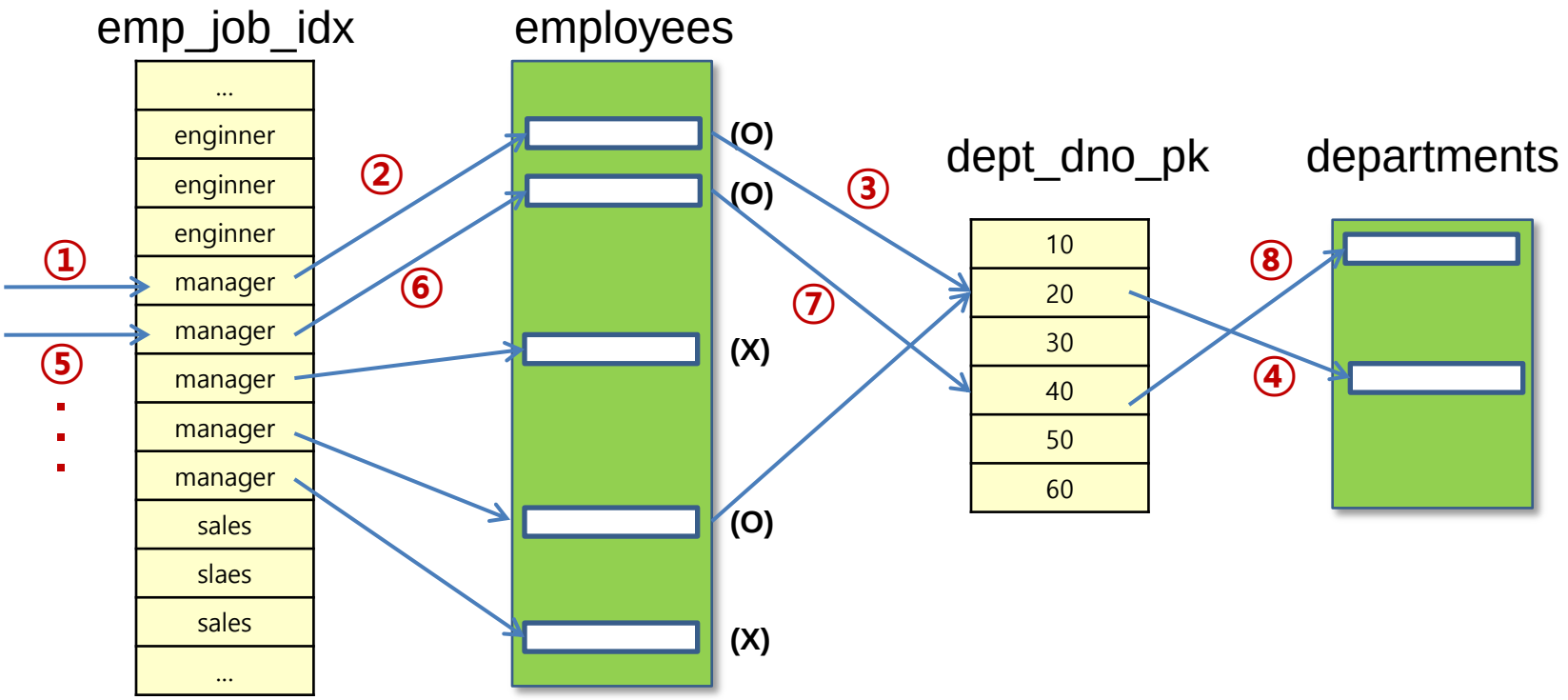
- Random access 위주 조인 방식
- 선행 테이블 조건에 만족하는 하나의 레코드 씩 순차적 진행
- OLTP 시스템에서 일차적으로 고려
- 부분 범위 처리시 유용

SQL PLAN

❖ NESTED LOOPS 조인

➢ 중첩 루프와 동일한 방식으로 동작

```
iSQL> SELECT * FROM employees e, departments d
        2 WHERE e.dno = d.dno
        3 AND e.emp_job = 'manager'
        4 AND e.salary > 2000000;
```



SQL PLAN

❖ SORT MERGE 조인 수행 방법

- outer 테이블 조건에 만족하는 레코드를 조인 컬럼 값으로 정렬
- inner 테이블 조건에 만족하는 레코드를 조인 컬럼 값으로 정렬
- outer 테이블 조인 컬럼 값과 같은 레코드를 inner 테이블에서 탐색하고, 다른 값이 나오는 순간 조인 중지
- outer 테이블 다음 레코드의 조인 컬럼 값과 같은 레코드를 inner 테이블에서 탐색하며 나오는 순간 조인 중지

❖ SORT MERGE 조인 특징

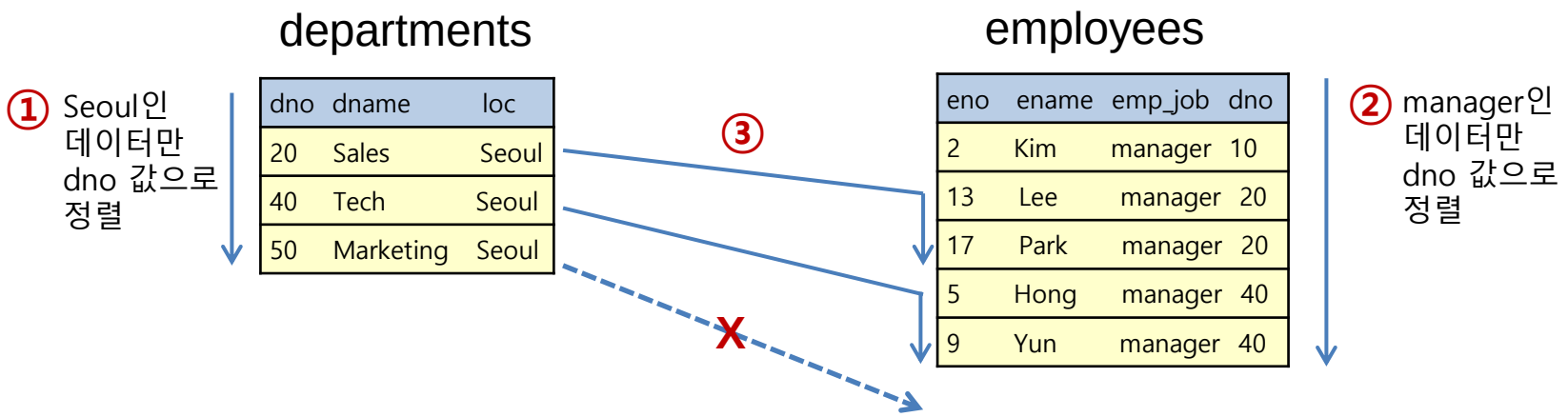
- 양쪽 테이블에 정렬이 수행된 후 조인을 수행
- 인덱스를 통해 미리 정렬이 되어있을 경우는 조인을 수행하지 않음
- Non equi 조인일 경우 Sort Merge 조인으로 수행

SQL PLAN

❖ SORT MERGE 조인

- Sort 단계 : 양쪽 집합을 조인 컬럼으로 정렬
- Merge 단계 : 정렬된 양쪽 집합을 병합

```
iSQL> SELECT * FROM employees e, departments d
        2 WHERE e.dno = d.dno
        3 AND e.emp_job = 'manager'
        4 AND d.loc='Seoul';
```



SQL PLAN

❖ HASH-BASED 조인 수행 방법

- 선행 테이블(Build) 조건에 맞는 레코드의 조인 컬럼에 Hash function 수행한 결과로 Hash map 생성
- 후행 테이블(Probe) 조건에 맞는 레코드의 조인 컬럼에 Hash function 수행한 후 Hash map을 탐색하며 조인 수행

❖ HASH-BASED 조인 특징

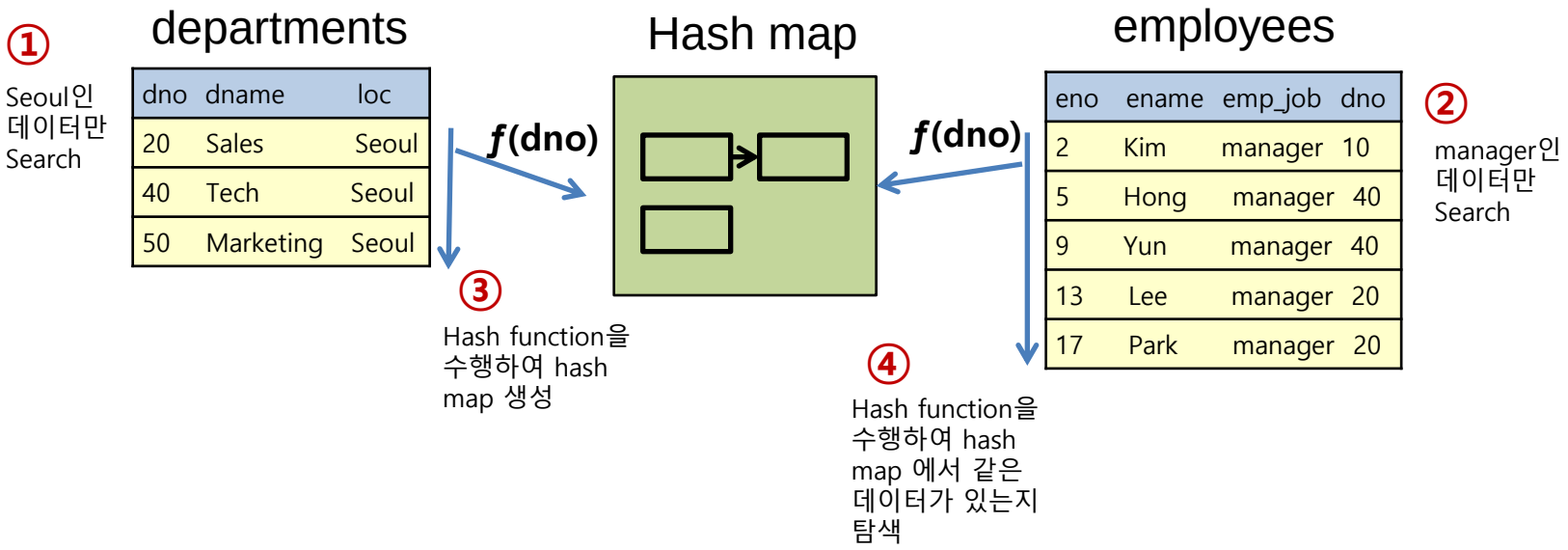
- NL 조인의 Random access와 SM 조인의 정렬 부담이 없음
- Hash function 수행 및 Hash map 생성 비용
- 조인 컬럼에 인덱스가 없을 경우 수행
- Equi join 에서만 사용
- 일반적으로 대용량의 테이블 조인 시 사용
- Build 테이블이 작아야 효과적(Hash map이 디스크에 생성될 경우 디스크 I/O 발생)

SQL PLAN

❖ HASH-BASED 조인

- 작은 테이블(Build)을 먼저 읽어 Hash map을 생성하고 큰 테이블(Probe)을 읽어 Hash map을 탐색하면서 조인

```
iSQL> SELECT * FROM employees e, departments d
        2 WHERE e.dno = d.dno
        3 AND e.emp_job = 'manager'
        4 AND d.loc='Seoul';
```



DBMS TUNING

SQL HINT

SQL HINT

❖ HINT 개념

- ➢ 옵티마이저가 올바른 실행계획을 생성할 수 있도록 유도

❖ HINT를 사용하는 SQL

- ➢ SELECT
- ➢ UPDATE
- ➢ DELTE
- ➢ MOVE
- ➢ INSERT(APPEND 힌트만 사용가능)

❖ HINT를 사용하는 방법

```
SELECT /*+ hint */ ~
UPDATE /*+ hint */ ~
DELETE /*+ hint */ ~
MOVE /*+ hint */ ~
INSERT /*+ hint */ ~
```

SQL HINT

❖ HINT 종류

힌트	설명
PUSH_PRED(view_name)	외부의 WHERE절의 조건 중 뷰와 관계된 조인 조건을 뷰 내부로 이동하여 처리
NO_MERGE(view)	뷰가 main query와 merge되는 것을 막을 때 사용
COST	비용을 고려한 실행 계획 생성
RULE	비용을 배제한 실행 계획 생성
ORDERED	FROM절에 나열된 순서대로 조인 순서를 결정
TEMP_TBS_MEMORY	SQL 처리 중에 생성되는 모든 중간 결과를 저장하기 위해 메모리 임시 테이블을 사용
TEMP_TBS_DISK	SQL 처리 중에 생성되는 모든 중간 결과를 저장하기 위해 디스크 임시 테이블을 사용
APPEND	Direct-Path INSERT가 수행
NO_PLAN_CACHE	생성된 플랜을 플랜 캐시에 저장하지 않음
KEEP_PLAN	한 번 생성된 플랜이 참조하는 테이블의 통계 정보가 변경되더라도 플랜이 재생성되는 것을 방지하고 그대로 사용하도록 함

SQL HINT

❖ HINT 종류

힌트	설명
FULL SCAN (table)	테이블에 이용 가능한 인덱스가 존재하더라도 인덱스를 사용하지 않고 테이블 전체를 스캔
INDEX ASC (table, index, index, ...), INDEX ASC (table index index ...)	명시된 인덱스를 사용하여 해당 테이블에 대해서 인덱스 스캔을 수행, 오름 차순으로 탐색
INDEX DESC (table, index, index, ...), INDEX DESC (table index index ...)	명시된 인덱스를 사용하여 해당 테이블에 대해서 인덱스 스캔을 수행, 내림 차순으로 탐색
INDEX (table, index, index, ...), INDEX (table index index ...)	명시된 인덱스를 사용하여 해당 테이블에 대해서 인덱스 스캔을 수행
NO INDEX (table, index, index, ...), NO INDEX (table index index ...)	명시된 인덱스를 사용해서 해당 테이블에 대한 인덱스 스캔을 수행하지 않도록
USE_NL (table, table)	Nested loops 조인 계열의 조인 방법을 사용
USE_HASH (table, table)	Hash-based 조인 계열의 조인 방법을 사용
USE_SORT (table, table)	Sort-based 조인 계열의 조인 방법을 사용
USE_MERGE (table, table)	Merge 조인 조인 계열의 조인 방법을 사용
NO_PUSH_SELECT_VIEW (table)	외부의 WHERE 절의 조건을 뷰 내부로 이동하여 처리하지 않음
PUSH_SELECT_VIEW (table)	외부의 WHERE절의 조건 중 가능한 것은 모두 뷰 내부로 이동하여 처리

SQL HINT

❖ FULL SCAN(TABLE)

➢ 이용 가능한 인덱스가 있더라도 테이블 데이터를 전체 스캔하여 검색

```
iSQL> SELECT /*+ FULL SCAN(tb02 ) */ DISTINCT col1 FROM tb02
      2 WHERE col3 ='testing'
      3 AND col1 NOT IN (1,2,3);
COL1
--------------
4
5
6
7
8
9
0
7 rows selected.
------------------------------------------------------------
PROJECT (COLUMN_COUNT: 1, TUPLE_SIZE: 4 )
 DISTINCT (ITEM_SIZE:16, ITEM_COUNT:7, DISK_PAGE_COUNT:3, ACCESS:7, SELF_ID:3, REF_ID: 2 )
  SCAN (TABLE: TB02, FULL SCAN, ACCESS: 1000000, DISK_PAGE_COUNT: 14272, SELF_ID: 2 )
   [ FILTER ]
   AND
    COL3 = 'testing'
    COL1 <>ALL (1, 2, 3)
------------------------------------------------------------
```

→ 디스크 테이블에 대해서는 거의 대부분의 row를 찾는 SQL에 대해서는 full table scan이 multi block IO를 실행하므로 성능이 빠를 수 있음

SQL HINT

❖ INDEX ASC

➢ 구문

```
INDEX ASC(TABLE, INDEX, INDEX, …)
INDEX ASC(TABLE INDEX INDEX …)
INDEX(TABLE, INDEX, INDEX, …)
INDEX(TABLE INDEX INDEX …)
```

➢ 지정된 인덱스를 생성된 순서대로 검색

```
iSQL> SELECT /*+ INDEX ASC(tb01 tb01_idx01) */ col2 FROM tb01
      2 WHERE col1=1
      3 LIMIT 1;
COL2
--------------
1
1 row selected.
------------------------------------------------------------
PROJECT ( COLUMN_COUNT: 1, TUPLE_SIZE: 4 )
 SCAN ( TABLE: TB01, INDEX: TB01_IDX01, ACCESS: 1, SELF_ID: 2 )
  [ FIXED KEY ]
  AND
   OR
    COL1 = 1
------------------------------------------------------------
```

→ 인덱스의 두 번째 칼럼의 min max 값을 인덱스를 이용해 찾을 때 첫 번째 조건에 해당하는 모든 값들에 대해 min max를 계산하므로, index asc 힌트와 limit 1 절을 이용하면 한 개의 레코드만 찾을 수 있어 성능 향상 기대

SQL HINT

❖ INDEX DESC

➢ 구문

```
INDEX DESC(TABLE, INDEX, INDEX, …)
INDEX DESC(TABLE INDEX INDEX …)
```

➢ 지정된 인덱스를 역순으로 검색

```
iSQL> SELECT /*+ INDEX DESC(tb01 tb01_idx01) */ col2 FROM tb01
      2 WHERE col1=1
      3 LIMIT 1;
COL2
-------------
999991
1 row selected.
------------------------------------------------------------
PROJECT ( COLUMN_COUNT: 1, TUPLE_SIZE: 4 )
 SCAN ( TABLE: TB01, INDEX: TB01_IDX01, ACCESS: 1, SELF_ID: 2 )
  [ FIXED KEY ]
  AND
   OR
    COL1 = 1
------------------------------------------------------------
```

→ 인덱스의 두 번째 칼럼의 min max 값을 인덱스를 이용해 찾을 때 첫 번째 조건에 해당하는 모든 값들에 대해 min max를 계산하므로, index desc 힌트와 limit 1 절을 이용하면 필요한 한 개의 레코드만 찾을 수 있으므로 성능 향상 기대

SQL HINT

❖ NO INDEX

➢ 구문

```
NO INDEX(TABLE, INDEX, INDEX, …)
NO INDEX(TABLE INDEX INDEX …)
```

➢ 지정된 인덱스를 사용하지 않음

```
iSQL> SELECT /*+ NO INDEX(tb01 tb01_idx02) */ DISTINCT col3 FROM tb01
      2 WHERE col1=1
      3 AND col3 = 'testing';
COL3
---------------------------------------------------
testing
1 row selected.
-------------------------------------------------------------
PROJECT ( COLUMN_COUNT: 1, TUPLE_SIZE: 52 )
 DISTINCT ( ITEM_SIZE:24, ITEM_COUNT:1, BUCKET_COUNT:1024, ACCESS:1, SELF_ID:3, REF_ID:2 )
  SCAN ( TABLE: TB01, INDEX: TB01_IDX01, ACCESS: 100000, SELF_ID: 2 )
   [ FIXED KEY ]
   AND
    OR
     COL1 = 1
   [ FILTER ]
   COL3 = 'testing'
-------------------------------------------------------------
```

SQL HINT

❖ USE_NL(TABLE, TABLE)

- Nested loops 조인 계열의 조인 방법을 사용
- 첫 번째 테이블을 먼저 스캔

```
iSQL> SELECT /*+ USE_NL(b, a) */ a.col1 FROM tb01 a , tb02 b
      2 WHERE a.col2 = b.col2
      3 AND a.col1 = b.col1
      4 AND a.col3 ='testing'
      5 AND b.col3 LIKE 'some%';
COL1
--------------
No rows selected.
------------------------------------------------------------
PROJECT ( COLUMN_COUNT: 1, TUPLE_SIZE: 4 )
 JOIN
 SCAN ( TABLE: TB02 B, INDEX:TB02_IDX02, ACCESS: 200, DISK_PAGE_COUNT: 14272, SELF_ID: 3 )
  [ FIXED KEY ]
  AND
   OR
    B.COL3 LIKE 'some%'
 SCAN ( TABLE: TB01 A, INDEX: TB01_IDX01, ACCESS: 200, SELF_ID: 2 )
  [ VARIABLE KEY ]
  OR
   AND
    A.COL1 = B.COL1
    A.COL2 = B.COL2
  [ FILTER ]
  A.COL3 = 'testing'
------------------------------------------------------------
```

SQL HINT

❖ USE_HASH(TABLE, TABLE)

- ➢ Hash-based 조인 방법을 사용
- ➢ HASH 노드의 테이블을 먼저 스캔

```
iSQL> SELECT /*+ USE_HASH(b, a) */ a.col1 FROM tb01 a, tb02 b
      2 WHERE a.col2 = b.col2
      3 AND a.col1 = b.col1
      4 AND a.col3 ='testing' AND b.col3 LIKE 'some%';
COL1
--------------
No rows selected.
------------------------------------------------------------
PROJECT ( COLUMN_COUNT: 1, TUPLE_SIZE: 4 )
 JOIN
  SCAN (TABLE: TB02 B, INDEX:TB02_IDX02, ACCESS:200, DISK_PAGE_COUNT:14272, SELF_ID:3 )
   [ FIXED KEY ]
   AND
    OR
     B.COL3 LIKE 'some%'
  HASH (ITEM_SIZE:24, ITEM_COUNT:999800, BUCKET_COUNT:1024, ACCESS:0, SELF_ID:4, REF_ID:2)
   [ FILTER ]
   AND
    A.COL1 = B.COL1
    A.COL2 = B.COL2
   SCAN ( TABLE: TB01 A, INDEX: TB01_IDX02, ACCESS: 999800, SELF_ID: 2 )
    [ FIXED KEY ]
    AND
     OR
      A.COL3 = 'testing'
------------------------------------------------------------
```

SQL HINT

❖ USE_MERGE(TABLE, TABLE)

- ➢ Sort Merge 조인 방법을 사용
- ➢ SORT 노드의 테이블을 먼저 스캔

```
iSQL> SELECT /*+ USE_MERGE(b, a) */ DISTINCT a.col1 FROM tb01 a , tb02 b
      2 WHERE a.col2 > b.col2 AND a.col3 LIKE 'some AND b.col3 LIKE 'some%'
      3 AND b.col1=a.col1 AND b.col1=1;
col1
--------------
1
1 row selected.
------------------------------------------------------------
PROJECT ( COLUMN_COUNT: 1, TUPLE_SIZE: 4 )
 DISTINCT(ITEM_SIZE:16, ITEM_COUNT:1, DISK_PAGE_COUNT 3, ACCESS: 1, SELF_ID:6, REF_ID:2)
  MERGE-JOIN
   [ VARIABLE KEY ]
   B.COL1 = A.COL1
   [ FILTER ]
   A.COL2 > B.COL2
   SCAN (TABLE:TB02 B, INDEX:TB02_IDX01, ACCESS:100000, DISK_PAGE_COUNT:14272, SELF_ID:3)
    [ FIXED KEY ]
    AND
     OR
      B.COL1 = 1
    [ FILTER ]
    B.COL3 LIKE 'some%'
   SORT ( ITEM_SIZE: 16, ITEM_COUNT: 200, ACCESS: 40000, SELF_ID: 4, REF_ID: 2 )
    SCAN ( TABLE: TB01 A, INDEX: TB01_IDX02, ACCESS: 200, SELF_ID: 2 )
     [ FIXED KEY ]
     AND
      OR
       A.COL3 LIKE 'some%'
------------------------------------------------------------
```

SQL HINT

❖ TEMP_TBS_MEMORY

- SQL 처리 중 생성되는 모든 중간 결과를 메모리 임시 테이블에 저장
- 사용 시 메모리 증가 가능

```
 iSQL> SELECT * FROM tb01 ORDER BY col2;
------------------------------------------------------------
PROJECT ( COLUMN_COUNT: 4, TUPLE_SIZE: 64 )
 SORT (ITEM_SIZE:72, ITEM_COUNT:10000, DISK_PAGE_COUNT:920, ACCESS:10000, SELF_ID:2, REF_ID:1)
  SCAN ( TABLE: TB01, FULL SCAN, ACCESS: 10000, DISK_PAGE_COUNT: 14272, SELF_ID: 1 )
------------------------------------------------------------

iSQL> SELECT /*+ TEMP_TBS_MEMORY */ * FROM tb01 ORDER BY col2;
------------------------------------------------------------
PROJECT ( COLUMN_COUNT: 4, TUPLE_SIZE: 64 )
 SORT ( ITEM_SIZE: 24, ITEM_COUNT: 10000, ACCESS: 10000, SELF_ID: 2, REF_ID: 1 )
  SCAN ( TABLE: TB01, FULL SCAN, ACCESS: 10000, DISK_PAGE_COUNT: 14272, SELF_ID: 1 )
------------------------------------------------------------
```

DBMS TUNING

ALTIBASE SERVER TUNING

❖ 로그파일 생성을 위한 TRANSACTION 대기

- prepare된 logfile 을 전부 사용 후 신규 logfile을 추가 생성할 때까지 TRANSACTION 대기로 인한 성능 저하 및 CPU 사용량 증가
- V$LFG의 lf_prepare_wait_count 값 확인

```
iSQL> SELECT lf_prepare_wait_count FROM V$LFG;

→ lf_prepare_wait_count 는 ALTIBASE 구동 이후 사용할 로그파일을 미리 만들어 두지 못해 로그 파일 생성되기를 기다린 횟수
```

- 디스크 속도 체크

```
Shell> time dd if=/dev/zero of=/altibase/altibase_home/logs/alti_test_file bs=8K count=1280
1280+0 records in
1280+0 records out
real    0m0.032s
user    0m0.000s
sys     0m0.033s
```

❖ 조치 사항

- PREPARE_LOG_FILE_COUNT 프로퍼티 값 조절
  - PREPARE_LOG_FILE_COUNT 은 logfile manager 쓰레드가 지정한 값 만큼 미리 logfile 생성
- PREPARE_LOG_FILE_COUNT 값이 클 수록 메모리 증가

❖ 체크포인트 발생 시 디스크 I/O에 대한 부하로 성능 저하

- ➢ altibase_sm.log 파일을 통해 체크포인트 진행 과정 확인
  - 체크포인트 시작

```
[CHECKPOINT-BEGIN]
```

  - 체크포인트 시작 LSN 기록

```
[CHECKPOINT-step2] Write BeginChkpt Log [0,89,6929497]
Active Tx Recovery LSN [0,3,3874568] :
Disk Buffer Oldest LSN [0,3,3874568] :
```

  - dirty page 대상 지정 및 TRANSACTION 로그의 sync, dirty page sync

```
[CHECKPOINT-step3] Flush Dirty Page(s)
[PRE-DirtyPageCount=0]
[NEW-DirtyPageCount=33036]
[DUP-DirtyPageCount=0]
Begin Sync For All-LFG - Request LSN [0,3,3875009]
Begin Bulk DB Sync (Prop.3200)
[FLU-DirtyPageCount=33036]
[REM-DirtyPageCount=0]
• PRE-DirtyPageCount: 이전에 썼던 Dirty Page 수
• NEW-DirtyPageCount: 새로 추가된 Dirty Page 수
• DUP-DirtyPageCount: New-DirtyPage중에서 PRE-DirtyPage와 중복되는 DirtyPage의 수
• Begin Sync For All-LFG - Request LSN: TRANSACTION로그를 Sync
• Begin Bulk DB Sync: dirty page sync
• FLU-DirtyPageCount: Flush 한 Dirty Page 수
• REM-DirtyPageCount: 2 벌의 데이터파일에 모두 기록되어 Dirty Page List에서 제거된 Page의 수
```

❖ 체크포인트 발생 시 디스크 I/O에 대한 부하로 성능 저하

- altibase_sm.log 파일을 통해 체크포인트 진행 과정 확인
  - datafile sync

```
[CHECKPOINT-step4] sync Database File
```

  - 체크포인트 종료 LSN 기록

```
[CHECKPOINT-step5] Write End_Chkpt Log [0,3,3974818]
```

  - 체크포인트 종료에 대한 TRANSACTION 로그 sync

```
[CHECKPOINT-step6] Sync Log File
Begin Sync For All-LFG - Request LSN [0,89,6885511]
```

  - 이중화로 보낼 TRANSACTION 로그의 SN 확인

```
[CHECKPOINT-step7] Check LogFiles That Is Not Needed
Replication MinSN368769
```

  - loganchor 에 체크포인트 수행 정보 기록

```
[CHECKPOINT-step8] Update and Flush Log Anchor
```

  - 불필요한 logfile 삭제

```
[CHECKPOINT-step8] Update and Flush Log Anchor
```

❖ 체크포인트 발생 시 디스크 I/O에 대한 부하로 성능 저하

- 체크포인트 로그 sample(altibase_sm.log)

```
[2011/10/05 13:51:01] [Thread-182894216896] [Level-9]
[CHECKPOINT BY SYSTEM]

[2011/10/05 13:51:01] [Thread-182894216896] [Level-9]
[CHECKPOINT-BEGIN]

... 중략

[2011/10/05 13:51:01] [Thread-182894216896] [Level-9]
[CHECKPOINT-summary] BeginChkptLSN=[0,153,3043510], EndChkptLSN=[0,153,3043951],
DiskRecLSN=[0,153,3043510]

[2011/10/05 13:51:01] [Thread-182894216896] [Level-9]
Minimum LSN = [0,153,3043510]

[2011/10/05 13:51:01] [Thread-182894216896] [Level-9]
[CHECKPOINT-END]
```

- 체크포인트가 수행 중에 [CHECKPOINT-step3] Flush Dirty Page(s) 혹은 [CHECKPOINT-step4] sync Database File가 오래 수행 중이라면 디스크 I/O를 모니터링

❖ 디스크 I/O

- ➢ sar, iostat을 통해 디스크 I/O 병목 확인

```
Shell> sar 1 3
02:32:26 PM       CPU     %user     %nice   %system   %iowait     %idle
02:32:30 PM       all     0.25      0.00     2.87      1.87        95.01
02:32:31 PM       all     0.12      0.00     6.24      6.99        86.64
02:32:32 PM       all     0.25      0.00     8.61      3.75        87.39

Shell> iostat 1
avg-cpu:  %user   %nice    %sys %iowait   %idle
           0.13    0.00    8.76   3.88     87.23
Device:            tps   Blk_read/s   Blk_wrtn/s   Blk_read   Blk_wrtn
sdb1           2821.78        63.37    608388.12         64     614472
```

❖ 조치 사항

- logfile과 datafile의 디스크 분리
- 체크포인트 관련 프로퍼티 조절

CHECKPOINT_BULK_WRITE_PAGE_COUNT	checkpoint 시 dirty page를 여러 번 나누어서 디스크에 저장할 수 있는데 이 때 한번에 디스크에 기록하는 page 수
CHECKPOINT_BULK_WRITE_SLEEP_SEC CHECKPOINT_BULK_WRITE_SLEEP_USEC	checkpoint 시에 한번 디스크에 기록한 후 sleep하는 시간 (두 값이 합산된 시간을 sleep)
CHECKPOINT_BULK_SYNC_PAGE_COUNT	checkpoint 시에 메모리와 디스크의 데이터를 일치시킬 page 단위

  - 기본 값보다 프로퍼티의 값을 줄여주면 디스크 I/O 분산 효과가 있지만 logfile이 증가할 수 있음
- 위의 사항을 적용했음에도 불구하고 성능향상이 없다면, 디스크 성능 한계치에 도달한 것으로 예측

❖ 디스크 테이블에 대한 BUFFER HIT 율 저하로 디스크 I/O증가

- 디스크 테이블 페이지를 적재할 때 한정된 크기의 buffer pool을 사용하므로 버퍼 교체가 이루어지고, 버퍼 교체가 일어나면 디스크 I/O로 인해 성능 저하
- 버퍼 관련 성능 뷰를 조회하여 버퍼 상황 확인

```
iSQL> SELECT hit_ratio 'HIT_RATIO(%)' , victim_search_warp FROM V$BUFFPOOL_STAT;
HIT_RATIO(%)           VICTIM_SEARCH_WARP
-------------------------------------------
99.8061076102763       0

→  VICTIM_SEARCH_WARP 값이 증가하고 있다면, flusher가 밀리고 있음
```

- 디스크 테이블 관련 SQL 중 대량의 페이지를 access하는 쿼리 튜닝
- BUFFER_AREA_SIZE 프로퍼티 변경

❖ 모든 SERVICE THREAD가 BUSY하면 새로운 SERVICE THREAD 생성으로 시스템 부하 증가

➢ V$SERVICE_THREAD를 통해 service thread 관련 부하 확인

```
iSQL> SELECT RPAD(type, 30), count(*) FROM V$SERVICE_THREAD GROUP BY type
        2 UNION ALL
        3 SELECT RPAD(name, 30), value1 FROM V$PROPERTY
        4 WHERE name LIKE 'MULTIPLEXING%_THREAD_COUNT';
RPAD(TYPE, 30)                 COUNT
-----------------------------------------------
SOCKET                         44
IPC                            10
MULTIPLEXING_THREAD_COUNT      8
MULTIPLEXING_MAX_THREAD_COUNT  1024
```

- SOCKET 항목의 수치가 MULTIPLEXING_THREAD_COUNT 항목의 수치보다 클 경우
  - 오래 수행되는 SQL 튜닝
  - MULTIPLEXING_THREAD_COUNT 프로퍼티 변경

❖ 메모리 AGER의 AGING으로 인한 성능 저하

- 메모리 테이블의 MVCC 기법으로 인해 생성되는 old versioning 정보를 메모리 ager가 삭제
- 메모리 ager 종류

MEM_LOGICAL_AGER	인덱스 페이지에 대한 versioning 정보 삭제
MEM_DELTHR	데이터 페이지에 대한 versioning 정보 삭제

- 메모리 ager가 밀린다면?
  - versioning 증가에 의한 메모리 증가
  - SQL 처리 시 versioning 정보들을 참조해야 하기 때문에 성능 저하
- V$MEMGC를 통해 메모리 ager 수행 여부 확인
- 메모리 ager 관련 프로퍼티
  - AGER_WAIT_MINIMUM
    - ager가 해야할 Job이 없을 경우 대기하는 최소시간
  - AGER_WAIT_MAXIMUN
    - ager가 해야할 Job이 없을 경우 대기하는 최대시간
  - ager는 작업이 없으면 MAX만큼 쉬고, 일이 있으면 MAX부터 1/2씩 쉬는 시간을 줄여 MIN까지 점차적으로 쉬는 시간을 줄여감.일이 없으면 다시 MAX까지 쉬는 시간을 늘려감

❖ 확인사항

- gc gap 조회

```
iSQL> SELECT gc_name, add_oid_cnt, gc_oid_cnt , add_oid_cnt - gc_oid_cnt gcgap FROM V$MEMGC;
ADD_OID_CNT          GC_OID_CNT           GCGAP
------------------------------------------------------------------
113                  113                  0
113                  113                  0
```

  - gc gap이 클수록 ager가 삭제해야 할 old version의 양이 많다는 의미

- Ager가 대기하는 TRANSACTION 정보

```
iSQL>  SELECT session_id, total_time, execute_time, tx_id, query
         2 FROM V$STATEMENT
         3 WHERE tx_id IN (SELECT id
         4                   FROM V$TRANSACTION
         5                  WHERE memory_view_scn = (SELECT minmemscnintxs FROM V$MEMGC LIMIT 1))
         6                                AND execute_flag = 1
         7 ORDER BY 2 DESC;
```

❖조치 사항

- GC 대상 TRANSACTION에서 수행하는 SQL 튜닝
- AGER_WAIT_MINIMUM, AGER_WAIT_MAXIMUM 값을 줄여 ager가 수행 빈도 증가

SUMMARY

- 성능 향상을 위해 ALTIBASE 관점에서 확인해야 하는 사항
  - logfile을 write하는 동안 대기하는가?
  - 체크포인트 시 디스크 I/O에 대한 병목이 큰가?
  - buffer의 hit율은 좋은가?
  - 메모리 ager가 잘 수행되는가?

DBMS TUNING

TRANSACTION TUNING

❖ 빈번한 PREPARE 수행

- Prepare-Validation-Optimization(PVO) 과정이 단순 DML일 경우 전제 SQL 처리과정의 60~70% 비중 차지
- V$SYSSTAT에서 execute 대비 prepare 수치 조회

```
iSQL> SELECT TO_CHAR(SYSDATE,'HH:MI:SS'), name, value FROM V$SYSSTAT
        2 WHERE name IN ('execute success count','prepare success count');
TO_CHAR(SYSDATE,'HH:MI:SS')  NAME                          VALUE
-------------------------------------------------------------------------
11:35:47                     execute success count         1225664
11:35:47                     prepare success count         234218
iSQL> /
TO_CHAR(SYSDATE,'HH:MI:SS')  NAME                          VALUE
-------------------------------------------------------------------------
11:36:47                     execute success count         1272912
11:36:47                     prepare success count         273319

→  47248(1272912-1225664) 문장 실행 시 39101(273319-234218) PREPARE를 실행(약 83%)
```

- prepare 비중이 많다면 바인딩 변수 사용 구조로 변경 권고

TRANSACTION TUNING

❖ 오래 수행되는 SQL(LONG TERM QUERY)

- SQL 처리하는 동안 해당 서비스 쓰레드가 CPU 1장(100%)을 사용
- CPU 부하의 가장 많은 원인
- V$STATEMENT를 통해 오래 수행되는 SQL 조회
- V$STATEMENT의 query는 16K까지만 보여주므로 전체 query를 조회하기 위해서는 V$SQLTEXT 로 확인

V$SQLTEXT

- 전체 SQL 확인할 때 조회하는 성능 뷰

칼럼	설명
SID	session ID
STMT_ID	statement ID
PIECE	SQL 조각 순서 (64byte단위로 나누어 저장되어 있음)
TEXT	실제 SQL 내용

- V$STATEMENT의 session_id, id 칼럼과 sid, stmt_id의 칼럼이 각각 같다는 조건을 이용하여 구문 작성

```
iSQL> SELECT text FROM V$SQLTEXT
        2 WHERE (sid,stmt_id) IN (SELECT session_id, id FROM V$STATEMENT WHERE id = 1123)
        3 ORDER BY piece;
```

TRANSACTION TUNING

❖ V$STATEMENT에서 QUERY와 TIME 정보 확인

```
iSQL> ALTER SYSTEM SET TIMED_STATISTICS=1;
iSQL> SET VERTICAL ON;
iSQL> SELECT query, execute_time, parse_time, total_time, optimize_time, validate_time,
        2              execute_success, mem_cursor_full_scan, mem_cursor_index_scan,
disk_cursor_full_scan,
        3              disk_cursor_index_scan
        4 FROM V$STATEMENT
        5 ORDER BY execute_time DESC
        6 LIMIT 10;

QUERY                  : select * from employee where ename= ' ALTIBASE'
EXECUTE_TIME           : 9
PARSE_TIME             : 151
TOTAL_TIME             : 423
OPTIMIZE_TIME          : 76
VALIDATE_TIME          : 97
EXECUTE_SUCCESS        : 1
MEM_CURSOR_FULL_SCAN   : 1
MEM_CURSOR_INDEX_SCAN  : 0
DISK_CURSOR_FULL_SCAN  : 0
DISK_CURSOR_INDEX_SCAN : 0
```

→ full table scan SQL로 대부분의 시간이 prepare 하는데 소요

TRANSACTION TUNING

❖ 오래 수행되는 SQL 튜닝

- 실행 계획을 확인하여 access 방식, join 방식, join 순서 등을 파악하여 튜닝
- 메모리 테이블은 table full scan 보다는 index scan이 더 유리

❖ 실행계획 확인 방법

- EXPLAIN PLAN 설정
  - ON : 실행결과 + 실행계획
  - ONLY : 실행계획만
  - OFF : 실행결과만

```
iSQL> ALTER SESSION SET EXPLAIN PLAN = ON;
iSQL> SELECT eno, ename, salary FROM employee WHERE eno=1;
ENO         ENAME                SALARY
------------------------------------------------
1           EJJUNG
1 row selected.
------------------------------------------------------------
PROJECT ( COLUMN_COUNT: 3, TUPLE_SIZE: 35 )
 SCAN ( TABLE: EMPLOYEE, INDEX: __SYS_IDX_ID_163, ACCESS: 1, SELF_ID: 2 )
------------------------------------------------------------
```

TRANSACTION TUNING

LOCK

- lock 부하로 인해 다른 TRANSACTION에 영향을 주어 성능 저하
- V$LOCK, V$LOCK_WAIT, V$TRANSACTION, V$STATEMENT 정보 확인
- TRANSACTION을 가급적 작은 단위로 나누어 lock 지속 시간 줄임
- dead lock이 발생할 수 있는 상황 방지

TRANSACTION TUNING

❖ LOCK 잡고 있는 SQL 확인

```
iSQL> SELECT tx.id tx_id, lw.wait_for_trans_id blocked_tx_id, l.lock_desc,
        2               DECODE(tx.first_update_time, 0, '0', to_char(to_date('1970010109','YYYYMMDDHH')
        3                     + tx.first_update_time / (60*60*24), 'MM/DD HH:MI:SS')) first_update_time,
        4               DECODE(tx.status, 0, 'BEGIN', 1, 'PRECOMMIT', 2, 'COMMIT_IN_MEMORY', 3, 'COMMIT',
        5                                4, 'ABORT', 5, 'BLOCKED', 6, 'END') status,
        6              st.query current_query
        7  FROM V$TRANSACTION tx, V$LOCK l
        8             LEFT OUTER JOIN V$LOCK_WAIT lw  ON l.trans_id = lw.trans_id
        9             LEFT OUTER JOIN (SELECT st.query,tx_id FROM V$STATEMENT st, V$SESSION ss
       11                                          WHERE ss.id = st.session_id ) st
       12                     ON l.trans_id = st.tx_id
       13 WHERE tx.id = l.trans_id;

TX_ID             : 103489
BLOCKED_TX_ID     :
LOCK_DESC         : IX_LOCK
FIRST_UPDATE_TIME : 09/02 14:42:35
STATUS            : BEGIN
CURRENT_QUERY     : update t1 set c1=1 where c1 between 1.11  and 1.112
TX_ID             : 4288
BLOCKED_TX_ID     : 103489
LOCK_DESC         : IX_LOCK
FIRST_UPDATE_TIME : 0
STATUS            : BLOCKED
CURRENT_QUERY     : update t1 set c1=1 where c1 =1.11
```

TRANSACTION TUNING

❖ LOCK 잡고 있는 세션 강제 종료

```
iSQL(sysdba)> ALTER DATABASE mydb SESSION CLOSE 10;

• mydb : 데이터베이스이름
• 10 : session_id
```

➢ 강제 세션 종료 시 세션 종료될 때까지 시간 소요 (rollback 처리 등)

TRANSACTION TUNING

❖ DURABILITY 설정방법

- altibase.properties 파일에서 관련 프로퍼티 변경 설정
  - COMMIT_WRITE_WAIT_MODE, LOG_BUFFER_TYPE 프로퍼티 설정
- 데이터베이스 구동 시 설정된 Durability 로 구동
- 데이터베이스 운영 중에 실시간으로 변경 불가능

TRANSACTION TUNING

❖ COMMIT_WRITE_WAIT_MODE =0

- OS Kernel 영역 로그버퍼를 사용하기 때문에 ALTIBASE 프로세스가 비정상 종료를 하더라도 TRANSACTION이 commit한 로그는 운영체제에 의해 로그파일에 반영
  - 성능지향 설정 방법
- OS의 crash 상황만 아니면 TRANSACTION durability 완벽 지원

❖ COMMIT_WRITE_WAIT_MODE = 1

- 로그를 프로세스 영역의 로그버퍼에 기록하고, 물리적인 로그파일에 기록하는 것을 보장하기 때문에 ALTIBASE 장애 시에도 durability 보장
- 모든 로그가 로그파일에 반영됨을 보장하기 때문에 어떠한 시스템 장애 상황에서도 완벽하게 TRANSACTION durability 보장
- durability level 중 성능이 가장 느림

TRANSACTION TUNING

❖ DURABILITY 설정 방법

목적	설명 및 설정 항목	
성능 위주 설정 (Durability = 3)	운영체제가 TRANSACTION 로그파일의 디스크 Sync를 보장하는 설정 운영체제의 Crash와 같은 장애를 제외한 모든 장애 경우에 대해 durability 보장	
	LOG_BUFFER_TYPE	0
	COMMIT_WRITE_WAIT_MODE	0
안정성 위주 설정 (Durability = 5)	ALTIBASE가 직접 디스크 Sync를 보장하는 설정 물리적인 디스크 장애 외에는 어떠한 장애라도 durability 보장	
	LOG_BUFFER_TYPE	1
	COMMIT_WRITE_WAIT_MODE	1

- COMMIT_WRITE_WAIT_MODE
  - ALTER SESSION 명령어로 변경 가능
- LOG_BUFFER_TYPE
  - 변경하려면 데이터베이스 재기동 필요

SUMMARY

- 성능 향상을 위해 application 관점에서 확인해야 하는 사항
  - TRANSACTION 양이 많은가?
  - connect - disconnect를 반복하는 구조인가?
  - prepare를 자주 하는가?
  - long term query가 수행되는가?
  - lock 경합이 빈번히 발생하는가?
  - update retry가 발생하는가?
  - 통신 방식은 무엇을 사용하는가?

DBMS TUNING

DBMS STATS

DBMS STATS

※ 6.1.1 버전부터 제공

❖ 개념

- DBMS의 통계 자료를 수집, 변경(설정), 삭제하는 기능
- 저장프로시저(Stored Procedure) 형태로 제공
- 통계 자료는 쿼리 옵티마이저가 최적화된 실행 계획을 만들기 위해 사용
- 개별 컬럼, 인덱스, 테이블 또는 시스템 별로 통계 자료 설정 및 삭제

❖ 유의 사항

- 통계 자료 수집 시, SERVER 부하 가중
- 통계 자료는 근사치를 수집
- 통계 자료를 수집한 대상 객체와 관련된 모든 쿼리의 실행 계획을 재구축하게 되어 성능 저하 발생 가능

DBMS STATS

❖ V$DBMS_STATS

- ➢ 데이터베이스 전체의 통계 정보를 보여주는 성능 뷰

Column name	Description
DATE	마지막으로 통계 정보를 수집한 일시
SAMPLE_SIZE	통계 정보 수집을 위해 선택된 샘플의 크기
NUM_ROW_CHANGE	마지막 통계 정보 수집 이후 행 개수의 변경된 양
TYPE	통계 수집 대상의 유형(S : 시스템, T : 테이블, I : 인덱스, C : 컬럼)
SREAD_TIME	하나의 페이지를 읽는 데 소요된 평균 시간
MREAD_TIME	여러 페이지를 한 번에 읽는 데 소요된 평균 시간
MREAD_PAGE_COUNT	여러 페이지를 한 번에 읽을 때 얽어 오도록 설정된 페이지의 수
HASH_TIME	해쉬 수행에 소요된 평균 시간
COMPARE_TIME	비교 수행에 소요된 평균 시간
STORE_TIME	메모리 임시 테이블에 저장하는데 소요된 평균 시간
TARGET_ID	통계 수집 대상이 된 테이블의 OID 또는 인덱스의 ID
COLUMN_ID	통계 수집 대상이 된 컬럼의 ID
NUM_ROW	통계 수집 대상(테이블 또는 인덱스)의 행의 개수
NUM_PAGE	통계 수집 대상(테이블 또는 인덱스)의 페이지 개수

DBMS STATS

❖ V$DBMS_STATS

Column name	Description
NUM_DIST	인덱스 또는 컬럼에서 중복되지 않은 유일한 값의 개수
NUM_NULL	컬럼에서 NULL의 개수
AVG_LEN	행 또는 컬럼의 평균 길이
ONE_ROW_READ_TIME	행 하나를 읽는 데 소요된 평균 시간
AVG_SLOT_COUNT	Leaf 노드 당 슬롯의 평균 개수
INDEX_HEIGHT	인덱스의 루트에서 leaf 노드까지의 깊이
CLUSTERING_FACTOR	인덱스에 부합하게 데이터가 정렬되어 있는 정도
MIN	인덱스 또는 컬럼의 최소 값
MAX	인덱스 또는 컬럼의 최대 값
META_SPACE	데이터 관리를 위해 사용된 공간의 크기
USED_SPACE	데이터 저장하기 위해 사용된 공간의 크기
AGEABLE_SPACE	나중에 aging 되어 재활용할 수 있는 공간의 크기
FREE_SPACE	테이블 또는 인덱스에 할당된 영역 중에서 사용 가능한 공간의 크기

※ V$DBMS_STATS 성능 View는 6.3.1 버전부터 제공

DBMS STATS

❖ DBMS STATS 프로시저

- ➢ 통계 자료 수집 및 실행 계획 재구축

Procedure Name	Description
GATHER_SYSTEM_STATS	데이터베이스 시스템에 대한 통계 자료 수집
GATHER_DATABASE_STATS	모든 테이블에 대한 통계 자료 수집
GATHER_TABLE_STATS	특정 테이블에 대한 통계 자료 수집
GATHER_INDEX_STATS	특정 인덱스에 대한 통계 자료 수집

※ 6.1.1 버전부터 제공

Procedure Name	Description
SET_SYSTEM_STATS	데이터베이스 시스템에 대한 통계 자료를 변경
SET_TABLE_STATS	특정 테이블에 대한 통계 자료를 변경
SET_INDEX_STATS	특정 인덱스에 대한 통계 자료를 변경
SET_COLUMN_STATS	특정 테이블의 컬럼에 대한 통계 자료를 변경

※ 6.3.1 버전부터 제공

DBMS STATS

❖ DBMS STATS 프로시저

- ➢ 개별적인 컬럼, 인덱스, 테이블 또는 시스템 관련 통계 자료를 조회

Procedure Name	Description
GET_SYSTEM_STATS	데이터베이스 시스템에 대한 통계 자료 조회
GET_DATABASE_STATS	모든 테이블에 대한 통계 자료 조회
GET_TABLE_STATS	특정 테이블에 대한 통계 자료 조회
GET_INDEX_STATS	특정 인덱스에 대한 통계 자료 조회

※ 6.5.1 버전부터 제공

Procedure Name	Description
DELETE_SYSTEM_STATS	데이터베이스 시스템에 대한 통계 자료를 삭제
DELETE_DATABASE_STATS	모든 테이블에 대한 통계 자료를 삭제
DELETE_TABLE_STATS	특정 테이블에 대한 통계 자료를 삭제
DELETE_INDEX_STATS	특정 인덱스에 대한 통계 자료를 삭제
DELETE_COLUMN_STATS	특정 테이블의 컬럼에 대한 통계 자료를 삭제

※ 6.5.1 버전부터 제공

ALTIBASE ADVANCE

ALTIBASE OPERATION

CONTENTS

- TRACE FILE
- PERFORMANCE VIEW
- OS CHECK LIST

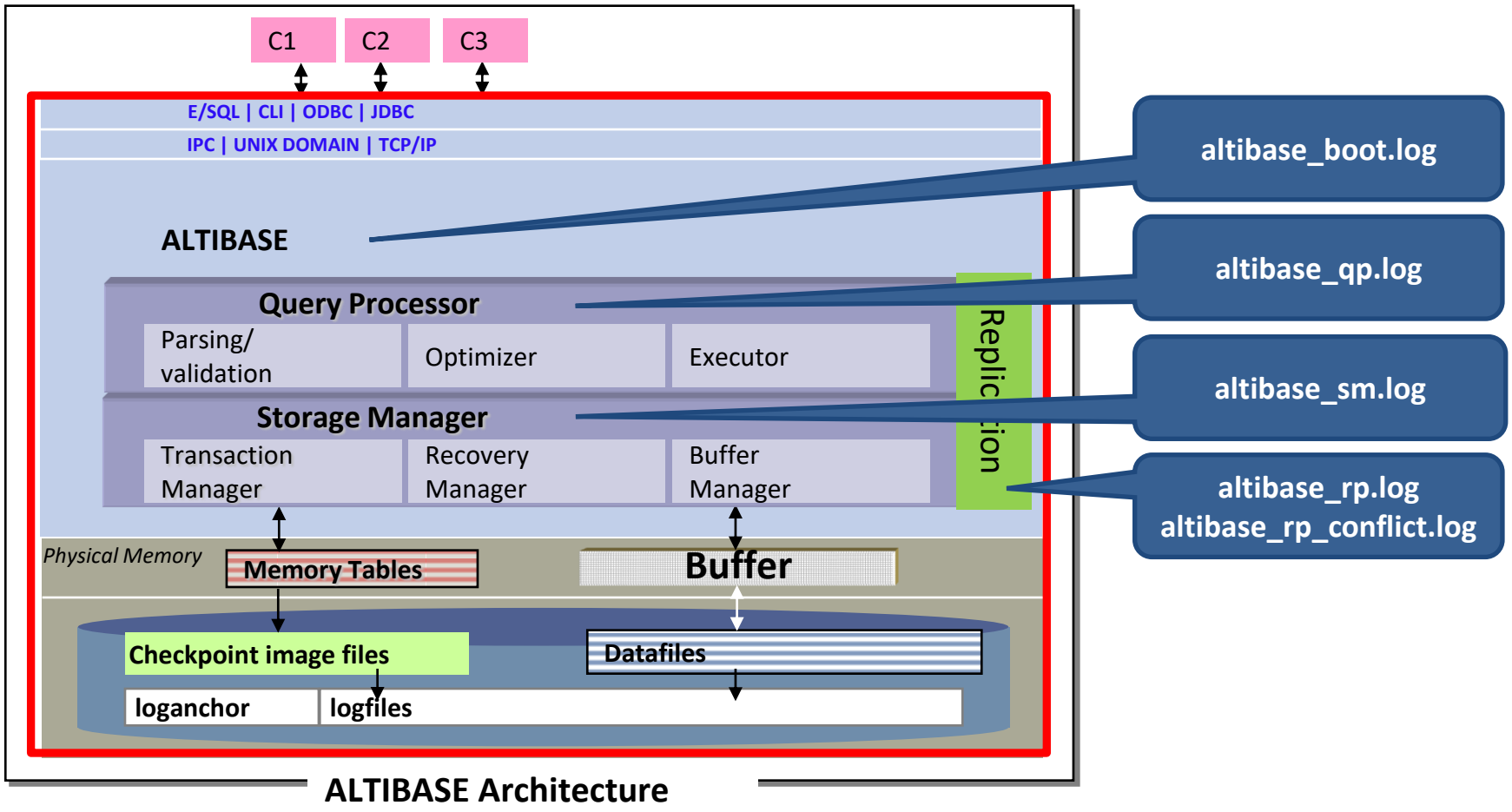
ALTIBASE OPERATION

TRACE FILE

TRACE FILE

ALTIBASE TRACE FILE

- $ALTIBASE_HOME/trc에 위치



ALTIBASE Architecture

TRACE FILE

❖ TRACE FILE 종류

파일 이름	주요 내용
altibase_boot.log	ALTIBASE가 동작된 상태 정보 구동 및 종료에 관련된 각종 step 정보 구동 중 디스크 부족으로 인한 에러 등 여러 가지 에러 상황에 대한 정보
altibase_error.log	비정상 종료가 발생할 경우 Call-Stack 정보
altibase_rp.log	Replication 모듈에서 발생하는 경고 및 에러 정보
altibase_rp_confilct.log	이중화 상태, 이중화 conflict 등의 정보
altibase_sm.log	Storage Manager 모듈에서 발생하는 경고 및 에러 정보 체크포인트, 백업 등의 정보
altibase_qp.log	Query Processor 모듈에서 발생하는 경고 및 에러 정보 DDL 문의 성공/실패 정보
altibase_id.log	시스템 레벨에서 발생하는 경고 및 에러 정보
altibase_mm.log	메인 모듈에서 발생하는 경고 및 에러 정보

TRACE FILE

❖ altibase_qp.log

- ➢ 사용자가 수행한 DDL문 정보
- ➢ ALTER SYSTEM 명령어로 property를 변경한 정보

❖ altibase_qp.log SAMPLE

- ➢ 테이블스페이스 DDL

```
[2012/10/27 13:32:48] [Thread-258] [Level-2]
[EXEC_DDL_BEGIN : CREATE TABLESPACE DISK_TBS DATAFILE 'disk_tbs001.dbf' SIZE 100M AUTOEXTEND OFF]
[2012/10/27 13:32:50] [Thread-258] [Level-2]
[EXEC_DDL_END : SUCCESS]
```

- ➢ 테이블 DDL

```
[2012/10/27 14:11:53] [Thread-515] [Level-2]
[EXEC_DDL_BEGIN : TRUNCATE TABLE TEST12]
[2012/10/27 14:11:53] [Thread-515] [Level-2]
[EXEC_DDL_END : SUCCESS]
```

- ➢ Property 변경

```
[2012/10/31 11:36:59] [Thread-515] [Level-0]
[SET-PROP] TIMED_STATISTICS=[1]
```

TRACE FILE

❖ altibase_sm.log

- 메모리 / 디스크 체크포인트 진행 및 수행 결과
- 인덱스 생성 진행 및 수행 결과
- 백업 진행 및 수행 결과
- ARCHIVE thread 오류 발생 정보
- 기타 디스크 관련 에러 정보

❖ altibase_sm.log SAMPLE(체크포인트)

```
-- CHECKPOINT가 발생한 이유에 대해 기록
   CHECKPOINT BY LOGFILE SWITCH COUNT, CHECKPOINT BY USER, CHECKPOINT BY TIME
[2018/02/06 17:23:05 55E][PID:16377][Thread-140449010857728][LWP-16414]
[CHECKPOINT BY TIME(6000 sec)]

-- CHECKPOINT 시작
[2018/02/06 17:23:05 55F][PID:16377][Thread-140449010857728][LWP-16414]
[CHECKPOINT-BEGIN]

-- BEGIN CHECKPOINT 기록
   메모리 데이터베이스의 recoveryLSN정보가 기록되고 차후 Recovery를 할 경우 이 정보 이용
[2018/02/06 17:23:05 560][PID:16377][Thread-140449010857728][LWP-16414]
[CHECKPOINT-step2] Write BeginChkpt Log [0,1413724,0]
```

TRACE FILE

```
-- CHECKPOINT 발생 시점에 가장 오래된 TRANSACTION이 기록한 첫 번째 BeginLog
Active Tx Recovery LSN [1413724,0]

-- CHECKPOINT 발생 시점에 버퍼에 접근한 가장 오래된 TRANSACTION이 기록한 첫 번째 BeginLog
Disk Buffer Oldest LSN [1413724,133152025]

-- 변경된 메모리데이터베이스의 DirtyPage Flush
   메모리에 대한 TRANSACTION 로그를 디스크로 쓰고, 변경된 데이터 페이지를 Checkpoint Image File에 저장
[2018/02/06 17:23:05 561][PID:16377][Thread-140449010857728][LWP-16414]
[CHECKPOINT-step3] Flush Dirty Page(s)

-- Ping-Pong CHECKPOINT를 수행하기 때문에 이전에 내렸던 Dirty Page 개수 기록
[2018/02/06 17:23:05 562][PID:16377][Thread-140449010857728][LWP-16414]
[PRE-DirtyPageCount=0]

-- 현재 CHECKPOINT 시점에 새롭게 추가된 DirtyPage 개수
[2018/02/06 17:23:05 563][PID:16377][Thread-140449010857728][LWP-16414]
[NEW-DirtyPageCount=11]

-- 이전 CHECKPOINT 와 현재 CHECKPOINT 시점에 중복되는 DirtyPage 개수
[2018/02/06 17:23:05 564][PID:16377][Thread-140449010857728][LWP-16414]
[DUP-DirtyPageCount=0]

-- DirtyPage 내리기 전, TRANSACTION 로그를 먼저 디스크로 sync
[2018/02/06 17:23:05 565][PID:16377][Thread-140449010857728][LWP-16414]
    + Begin Sync For All-LFG - Request LSN [0,1413781]

[2018/02/06 17:23:05 566][PID:16377][Thread-140449010857728][LWP-16414]
    + End Sync For All-LFG
```

TRACE FILE

```
-- flush한 DirtyPage 수
[2018/02/06 17:23:05 567][PID:16377][Thread-140449010857728][LWP-16414]
[FLU-DirtyPageCount=11]

-- 2벌의 데이터파일에 모두 기록되어 DirtyPageList에서 제거된 Page 수
[2018/02/06 17:23:05 568][PID:16377][Thread-140449010857728][LWP-16414]
[REM-DirtyPageCount=0]

-- CHECKPOINT Image 파일들을 모두 sync
[2018/02/06 17:23:05 569][PID:16377][Thread-140449010857728][LWP-16414]
[CHECKPOINT-step4] sync Database File

-- DISK I/O 통계 정보
=============================================================
SM IO STAT  - Checkpoint
DB SIZE      :       360448 Byte ( 11 Page)
LOG SIZE     :           57 Byte
TOTAL TIME   :      0 s 1735 us
LOG SYNC TIME:      0 s 290 us
DB FLUSH TIME:      0 s 1445us
      SYNC TIME :      0 s 938 us
      WAIT TIME :      0 s 0 us
      WRITE TIME:      0 s 507 us
LOG IO PERF  : 226.868890148546 MB/sec
DB IO PERF   : 237.889273356401 MB/sec
=============================================================
```

TRACE FILE

```
-- CHECKPOINT 완료를 메모리에 로깅
[2018/02/06 17:23:05 56B][PID:16377][Thread-140449010857728][LWP-16414]
[CHECKPOINT-step5] Write End_Chkpt Log [0,1413935,3579565754]

-- CHECKPOINT 과정에서 변경된 페이지에 접근한 TRANSACTION이 존재할 수 있고
 CHECKPOINT가 완료된 로그도 디스크로 Sync해야 하기 때문에 다시 한번 메모리상의 로그 Sync
[2018/02/06 17:23:05 56C][PID:16377][Thread-140449010857728][LWP-16414]
[CHECKPOINT-step6] Sync Log File

[2018/02/06 17:23:05 56D][PID:16377][Thread-140449010857728][LWP-16414]
    + Begin Sync For All-LFG - Request LSN [0,1413968]

[2018/02/06 17:23:05 56E][PID:16377][Thread-140449010857728][LWP-16414]
    + End Sync For All-LFG

-- 현재는 의미 없는 로그임으로 무시
[2018/02/06 17:23:05 56F][PID:16377][Thread-140449010857728][LWP-16414]
[CHECKPOINT-step7] Check LogFiles That Is Not Needed

-- sender가 어떤 오류가 발생했을 때 재전송을 시작해야 할 SN 정보 기록
[2018/02/06 17:23:05 570][PID:16377][Thread-140449010857728][LWP-16414]
Replication MinLSN [4294967295,4294967295]

-- RecoveryLSN에 대해 디스크 부분에 대한 RecoveryLSN정보를 loganchor 파일에 기록
[2018/02/06 17:23:05 571][PID:16377][Thread-140449010857728][LWP-16414]
[CHECKPOINT-step8] Update and Flush Log Anchor

-- 메모리상의 로그와 DirtyPage들을 디스크로 기록 했음으로 더 이상 복구에 필요하지 않은 로그 파일 삭제
[2018/02/06 17:23:05 572][PID:16377][Thread-140449010857728][LWP-16414]
[CHECKPOINT-step9] Remove Online Log File[None]
```

TRACE FILE

```
-- 모든 CHECKPOINT Image 파일 및 디스크 데이터파일에 RedoLSN 정보 기록
[2018/02/06 17:23:05 573][PID:16377][Thread-140449010857728][LWP-16414]
[CHECK DATABASE SID=0, PPID=0, FID=0]

[2018/02/06 17:23:05 574][PID:16377][Thread-140449010857728][LWP-16414]
LogAnchor SpaceID=0, SmVersion=100990977, DBFileHdr SpaceID=0, SmVersion=100990977,
[2018/02/06 17:23:05 575][PID:16377][Thread-140449010857728][LWP-16414]
RedoLSN=control[0,1413724], [0,1413724]

[2018/02/06 17:23:05 576][PID:16377][Thread-140449010857728][LWP-16414]
CreateLSN=control[0,540], [0,540]
…

-- CHECKPOINT의 Begin/End시점의 LSN 정보 기록, 디스크의 RedoLSN 정보 기록
[2018/02/06 17:23:05 583][PID:16377][Thread-140449010857728][LWP-16414]
[CHECKPOINT-summary] BeginChkptLSN=[0,1413724], EndChkptLSN=[0,1413935], DiskRecLSN=[0,1413724]

-- 현재 CHECKPOINT가 진행되는 시점에서 가장 오래된 TRANSACTION이 기록한 첫 번째 TRANSACTION 로그의 시작
위치 기록
[2018/02/06 17:23:05 584][PID:16377][Thread-140449010857728][LWP-16414]
Minimum LSN = [0,1413724]

-- CHECKPOINT 완료 기록
[2018/02/06 17:23:05 585][PID:16377][Thread-140449010857728][LWP-16414]
[CHECKPOINT-END]

-- 다음 CHECKPOINT 시간 기록
[2018/02/06 17:23:05 586][PID:16377][Thread-140449010857728][LWP-16414]
Sleep checkpoint thread ( next time : 2018-2-6 19:3:5 )
```

TRACE FILE

❖altibase_sm.log SAMPLE(ONLINE BACKUP)

```
[2019/03/07 16:08:26 633][PID:113130][Thread-140063711504128][LWP-113136]
DISK TABLESPACE SYS_TBS_DISK_DATA DATABASE /ssd/mbw/altibase_home/dbs/system001.dbf BACKUP TO
/ssd/mbw/work/imsi5/system001.dbf

[2019/03/07 16:08:27 634][PID:113130][Thread-140063711504128][LWP-113136]
DISK TABLESPACE SYS_TBS_DISK_UNDO DATABASE /ssd/mbw/altibase_home/dbs/undo001.dbf BACKUP TO
/ssd/mbw/work/imsi5/undo001.dbf

[2019/03/07 16:08:27 635][PID:113130][Thread-140063711504128][LWP-113136]
DISK TABLESPACE DISK_TBS DATABASE /ssd/mbw/altibase_home/dbs/disk_tbs001.dbf BACKUP TO
/ssd/mbw/work/imsi5/disk_tbs001.dbf

[2019/03/07 16:08:27 636][PID:113130][Thread-140063711504128][LWP-113136]
Waiting logfile0 to archive

[2019/03/07 16:08:32 637][PID:113130][Thread-140063711504128][LWP-113136]
Database-Level Backup Completed [SUCCESS]
```

TRACE FILE

❖ altibase_rp.log

- 이중화 시작/중지 및 장애 상황에 대한 정보
- REPLICATION_LOCK_TIMEOUT에 의한 오류 정보
- REPLICATION_RECEIVE_TIMEOUT에 의한 오류 정보

❖ altibase_rp.log SAMPLE

- 이중화 시작
  - Sender

```
[2019/03/18 10:59:02 617][PID:104294][Thread-140429585733376][LWP-104376]
[Sender] Replication REP1:0 Start... at [4295287685]
```

  - Receiver

```
[2019/03/18 10:58:53 613][PID:104294][Thread-140429440194304][LWP-104388]
[Receiver] Replication REP1 Started ...
```

TRACE FILE

➢ 이중화 시작

- Sender

```
[2019/03/18 11:00:43 61A][PID:104294][Thread-140429585733376][LWP-104376]
SEND Stop Message!

[2019/03/18 11:00:43 61B][PID:104294][Thread-140429585733376][LWP-104376]
SEND Stop Message SUCCESS!!!

[2019/03/18 11:00:43 61C][PID:104294][Thread-140429597861632][LWP-104375]
[SenderApply] Replication REP1 Normal End
[2019/03/18 11:00:43 61D][PID:104294][Thread-140429585733376][LWP-104376]
[Sender] Stop sender thread REP1:0 at [4295288674], Restart SN[4295287685]

→ Restart SN은 다음 이중화를 재시작할 때 해당 값이 기록된 로그부터 재전송을 하겠다는 의미
```

- Receiver

```
[2019/03/18 11:00:43 584][PID:7301][Thread-140021822617344][LWP-7407]
[Receiver] Replication Stop Message is arrived
[2019/03/18 11:00:43 585][PID:7301][Thread-140021822617344][LWP-7407]
[Receiver] RepName:REP1 is processed at [Last Processed SN:4295288674]
[2019/03/18 11:00:43 586][PID:7301][Thread-140021822617344][LWP-7407]
[Receiver] RepName:REP1 is received at [Last Received SN:4295288674]
[2019/03/18 11:00:43 587][PID:7301][Thread-140021822617344][LWP-7407]
[Receiver] Normal Stop
[2019/03/18 11:00:43 588][PID:7301][Thread-140021822617344][LWP-7407]
[Receiver] Replication REP1 Stopped ...
```

TRACE FILE

- 이중화 SYNC
  - Sender

```
[2019/03/18 11:04:30 637][PID:104294][Thread-140429597861632][LWP-104375]
[PJChild] [0] PJ_RUN
[2019/03/18 11:04:30 638][PID:104294][Thread-140429597861632][LWP-104375]
[PJChild] [0] TABLE TEST1 SYNC START
[2019/03/18 11:04:30 639][PID:104294][Thread-140429597861632][LWP-104375]
[PJChild] [0] TABLE: TEST1, PROCESSED: 5, TUPLE: 5              ← sync 할 대상 테이블 및 레코드건수
[2019/03/18 11:04:30 63A][PID:104294][Thread-140429597861632][LWP-104375]
[PJ Child] SEND Stop Message!

[2019/03/18 11:04:30 63B][PID:104294][Thread-140429597861632][LWP-104375]
[PJ Child] SEND Stop Message SUCCESS!!!                          ← sync 완료 후 sync 중지

[2019/03/18 11:04:30 63C][PID:104294][Thread-140429597861632][LWP-104375]
[PJChild] [0] PJ_EXIT
[2019/03/18 11:04:31 63D][PID:104294][Thread-140429440194304][LWP-104388]
Succeeded to insert data.

[2019/03/18 11:04:32 63E][PID:104294][Thread-140429440194304][LWP-104388]
Succeeded to rebuild indexes.

[2019/03/18 11:04:32 63F][PID:104294][Thread-140429440194304][LWP-104388]
[Sender] Replication REP1:0 Start... at [4295300681]            ← sender 자동 재 구동
```

TRACE FILE

- 이중화 SYNC
  - Receiver

```
[2019/03/18 11:04:30 58F][PID:7301][Thread-140021822617344][LWP-7407]
[Receiver] Replication REP1 Started ...                          ← receiver 자동 재 구동


[2019/03/18 11:04:30 590][PID:7301][Thread-140022755325696][LWP-7355]
[Receiver] Replication REP1 Started ...                          ← sync 용 receiver 구동


[2019/03/18 11:04:30 591][PID:7301][Thread-140022755325696][LWP-7355]
[Receiver] Replication Stop Message is arrived                   ← sync 용 receiver 종료
[2019/03/18 11:04:30 592][PID:7301][Thread-140022755325696][LWP-7355]
[Receiver] RepName:REP1 is processed at [Last Processed SN:18446744073709551615]
[2019/03/18 11:04:30 593][PID:7301][Thread-140022755325696][LWP-7355]
[Receiver] RepName:REP1 is received at [Last Received SN:0]
```

TRACE FILE

- Sender 비정상 종료
  - Sender 비정상 종료 되어 Sender 관련 로그 기록 불가
  - Receiver

```
[2019/03/21 13:43:19 680][PID:10233][Thread-139675223820032][LWP-10326]
ERR-7101a(errno=16) Connection closed                                    ← 단절 감지

[2019/03/21 13:43:19 680][PID:10233][Thread-139675223820032][LWP-10326]
ERR-7101a(errno=16) Connection closed

[2019/03/21 13:43:19 6DA][PID:2343][Thread-140022664775424][LWP-2456]
ERR-61048(errno=62) [Receiver] REP1 receiver has recvXLog error in run()

[2019/03/21 13:43:19 6DB][PID:2343][Thread-140022664775424][LWP-2456]
[Receiver] RepName:REP1 is processed at [Last Processed SN:313534027962]
[2019/03/21 13:43:19 6DC][PID:2343][Thread-140022664775424][LWP-2456]
[Receiver] RepName:REP1 is received at [Last Received SN:313534027962]
[2019/03/21 13:43:19 6DD][PID:2343][Thread-140022664775424][LWP-2456]
ERR-6104b(errno=62) [Receiver] REP1 receiver is ended (by thr_exit)

[2019/03/21 13:43:19 6DE][PID:2343][Thread-140022664775424][LWP-2456]
Error Stop!

[2019/03/21 13:43:19 6DF][PID:2343][Thread-140022664775424][LWP-2456]
[Receiver] Replication REP1 Stopped ...                                  ← Receiver 중단
```

TRACE FILE

Sender 복구 후

- Sender

```
[2019/03/21 13:46:12 667][PID:3713][Thread-140548569315072][LWP-3811]
[SenderApply] Replication REP1 Start
[2019/03/21 13:46:12 668][PID:3713][Thread-140548825384704][LWP-3784]
[Sender] Replication REP1:0 Start... at [322131036108]
```

- Receiver

```
[2019/03/21 13:46:12 675][PID:3713][Thread-140548544927488][LWP-3813]
[Receiver] Replication REP1 Started ...
```

TRACE FILE

- Receiver 비정상 종료
  - Receiver 비정상 종료 되어 Receiver 관련 로그 기록 불가
  - Sender

```
[2019/03/21 13:43:19 680][PID:10233][Thread-139675223820032][LWP-10326]
ERR-7101a(errno=16) Connection closed                                    ← 연결 단절 감지


[2019/03/21 13:43:19 681][PID:10233][Thread-139675599795968][LWP-10295]
ERR-620f0(errno=11) [Sender] Stop sender thread REP1:0 at [313534027962], Restart SN[313534025907]
                                                                         ← Sender 중단


[2019/03/21 13:43:19 682][PID:10233][Thread-139675599795968][LWP-10295]
[Sender] getNextLastUsedHostNo: from 192.168.1.62:40700 to 192.168.1.62:40700
[2019/03/21 13:43:19 683][PID:10233][Thread-139675599795968][LWP-10295]
ERR-71017(errno=111) Failed to invoke the connect() system function, errno=111
                                                                         ← 이중화 백업 라인 체크


[2019/03/21 13:43:19 684][PID:10233][Thread-139675599795968][LWP-10295]
[Sender] getNextLastUsedHostNo: from 192.168.1.62:40700 to 192.168.1.62:40700
[2019/03/21 13:43:19 685][PID:10233][Thread-139675599795968][LWP-10295]
ERR-61022(errno=111) [Sender] Sender Sleep : 60 seconds
                                                                         ← 다음 재 접속 시도까지
                                                    (REPLICATION_SENDER_SLEEP_TIMEOUT=60초) 만큼 Sleep
```

TRACE FILE

- 60초 후 Receiver 복구 안 되는 경우
  - 장애 시점 로그와 동일한 connection 오류 발생하고 이중화 백업 라인 시도 후 실패하게 되면 Sender 다시 sleep

```
[2019/03/21 13:44:19 686][PID:10233][Thread-139675599795968][LWP-10295]
ERR-71017(errno=111) Failed to invoke the connect() system function, errno=111

[2019/03/21 13:44:19 687][PID:10233][Thread-139675599795968][LWP-10295]
[Sender] getNextLastUsedHostNo: from 192.168.1.62:40700 to 192.168.1.62:40700
[2019/03/21 13:44:19 688][PID:10233][Thread-139675599795968][LWP-10295]
ERR-61022(errno=111) [Sender] Sender Sleep : 60 seconds

[2019/03/21 13:45:19 689][PID:10233][Thread-139675599795968][LWP-10295]
ERR-71017(errno=111) Failed to invoke the connect() system function, errno=111

[2019/03/21 13:45:19 68A][PID:10233][Thread-139675599795968][LWP-10295]
[Sender] getNextLastUsedHostNo: from 192.168.1.62:40700 to 192.168.1.62:40700
[2019/03/21 13:45:19 68B][PID:10233][Thread-139675599795968][LWP-10295]
ERR-61022(errno=111) [Sender] Sender Sleep : 60 seconds
```

- Receiver 복구 이후 로그는 Sender 복구 로그와 동일

TRACE FILE

- REPLICATION_RECEIVE_TIMEOUT 에러
  - Sender

```
-- Timeout이 발생한 기록
[2013/11/30 16:13:03] [Thread-6786] [Level-0]
ERR-61075(errno=16) Timeout exceed.

-- 반복적으로 재 접속이 발생
[2013/11/30 16:36:02] [Thread-6529] [Level-0]
ERR-620f0(errno=16) [Sender] Stop sender thread REP1:0 at [785419687], Restart SN[783217166]

[2013/11/30 16:36:07] [Thread-6529] [Level-0]
[Sender] getNextLastUsedHostNo: from 192.168.1.131:37585 to 192.168.1.131:37585
[2013/11/30 16:36:07] [Thread-6529] [Level-0]
ERR-7101a(errno=0) Connection closed

[2013/11/30 16:36:07] [Thread-6529] [Level-0]
ERR-61003(errno=0) Unable to read from a socket

[2013/11/30 16:36:07] [Thread-6529] [Level-0]
[Sender] getNextLastUsedHostNo: from 192.168.1.131:37585 to 192.168.1.131:37585
[2013/11/30 16:36:07] [Thread-6529] [Level-0]
ERR-61022(errno=0) [Sender] Sender Sleep : 60 seconds

[2013/11/30 16:37:07] [Thread-6529] [Level-0]
ERR-7101a(errno=0) Connection closed

[2013/11/30 16:37:07] [Thread-6529] [Level-0]
ERR-61003(errno=0) Unable to read from a socket
 …
```

TRACE FILE

- REPLICATION_RECEIVE_TIMEOUT 에러
  - Receiver
    - Receiver 데이터베이스의 lock 대기로 timeout 발생의 경우

```
[2013/11/30 16:38:07] [Thread-5956] [Level-0]
ERR-61030(errno=11) [Receiver] Failed to build meta information  …
```

    - Sender에 의해 재구동

```
[2013/11/30 16:45:18] [Thread-6486] [Level-0]
ERR-7101a(errno=32) Connection closed

[2013/11/30 16:45:18] [Thread-6486] [Level-0]
ERR-61048(errno=32) [Receiver] REP1 receiver has recvXLog error in run()

[2013/11/30 16:45:39] [Thread-6486] [Level-0]
ERR-71032(errno=76) Unable to shutdown the socket

[2013/11/30 16:45:39] [Thread-6486] [Level-0]
ERR-61087(errno=76) [Network] Shutdown link operation is failed

[2013/11/30 16:45:39] [Thread-6486] [Level-0]
ERR-6104b(errno=76) [Receiver] REP1 receiver is ended (by thr_exit)

[2013/11/30 16:45:39] [Thread-6486] [Level-0]
Error Stop!

[2013/11/30 16:45:39] [Thread-6486] [Level-0]
[Receiver] Replication REP1 Stopped ...
```

TRACE FILE

❖ altibase_rp_conflict.log

- conflict 발생 시 해당 SQL 및 conflict 유형 정보 기록
- Insert Duplicate 에러
  - Sender로부터 수신한 Insert 데이터와 동일한 PK를 가진 데이터가 Receiver 측 데이터베이스에 이미 존재할 경우 에러 기록

```
[2019/03/21 13:56:17 68A][PID:3713][Thread-140548544927488][LWP-3813]
ERR-11058(errno=62) The row already exists in a unique index.

[2019/03/21 13:56:17 68B][PID:3713][Thread-140548544927488][LWP-3813]
INSERT INTO SYS.TEST1 VALUES ( 100 ,  'aaaa' ); (TID : 20320)

[2019/03/21 13:56:17 68C][PID:3713][Thread-140548544927488][LWP-3813]
COMMIT (TID : 20320)
```

- Update Not Found 에러
  - Sender 로부터 수신한 Update에 해당하는 PK를 가진 데이터가 Receiver 측 데이터베이스에 존재하지 않을 경우 에러 기록

```
[2019/03/21 13:58:03 690][PID:3713][Thread-140548544927488][LWP-3813]
ERR-610f7(errno=62) [Receiver] Unable to find record in executeUpdate() function

[2019/03/21 13:58:03 691][PID:3713][Thread-140548544927488][LWP-3813]
UPDATE SYS.TEST1 SET C2 = 'bbbb' WHERE C1 = 101; (TID : 26464)

[2019/03/21 13:58:03 692][PID:3713][Thread-140548544927488][LWP-3813]
COMMIT (TID : 26464)
```

TRACE FILE

- Update Conflict 에러
  - Sender 에서 Update를 수행했던 시점의 변경 대상 컬럼이 가지는 변경 전 값과 Receiver 측 데이터베이스의 변경 대상 컬럼이 가지는 현재 값이 다를 경우 에러 기록
  - Sender 데이터베이스는 Update SQL에 대해 변경 대상 컬럼의 변경 전 / 후 값을 함께 전송하여 비교 후 Update 수행

```
[2019/03/21 13:57:12 68D][PID:3713][Thread-140548544927488][LWP-3813]
ERR-61035(errno=62) [Receiver] An update conflict occurred.

[2019/03/21 13:57:12 68E][PID:3713][Thread-140548544927488][LWP-3813]
UPDATE SYS.TEST1 SET C2 = 'bbbb' WHERE C1 = 100; (TID : 24416)

[2019/03/21 13:57:12 68F][PID:3713][Thread-140548544927488][LWP-3813]
COMMIT (TID : 24416)
```

- Delete Not Found 에러
  - Sender 로부터 수신한 Delete 에 해당하는 PK를 가진 데이터가 Receiver 측 데이터베이스에 존재하지 않을 경우 에러 기록

```
[2019/03/21 13:58:35 693][PID:3713][Thread-140548544927488][LWP-3813]
ERR-610f7(errno=62) [Receiver] Unable to find record in executeDelete() function

[2019/03/21 13:58:35 694][PID:3713][Thread-140548544927488][LWP-3813]
DELETE FROM SYS.TEST1 WHERE C1 = 101; (TID : 28512)

[2019/03/21 13:58:35 695][PID:3713][Thread-140548544927488][LWP-3813]
COMMIT (TID : 28512)
```

ALTIBASE OPERATION

PERFORMANCE VIEW

PERFORMANCE VIEW

ALTIBASE 성능 뷰

- 시스템 통계 정보
- session
- statement
- lock
- service thread
- 메모리 ager
- buffer
- logfile
- 테이블스페이스 usage
- 테이블 usage
- 메모리 usage
- 이중화

PERFORMANCE VIEW

V$SYSSTAT

- 시스템 상태 정보
- 3초마다 정보 갱신
- 적절한 주기를 두고 전, 후의 값을 비교하여 각 항목의 증가 추이 관찰
- 시스템의 부하 항목 확인
- 칼럼 정보

칼럼	내용
NAME	시스템 각 상태의 이름
VALUE	각 상태 값의 누적 치

- sample SQL

```
iSQL> SELECT TO_CHAR(SYSDATE,'HH:MI:SS'), name, value FROM V$SYSSTAT
       2  WHERE name IN ('logon cumulative', 'execute success count', 'prepare success count',
       3                 'memory table cursor full scan count', 'memory table cursor index scan count',
       4                 'disk table cursor full scan count', 'disk table cursor index scan count',
       5                 'update retry count', 'lock row retry count'
       6                 );
```

PERFORMANCE VIEW

❖ V$SESSTAT

- 접속된 모든 세션 통계치
- 접속이 끊겨진 세션의 정보는 삭제
- 칼럼 정보

칼럼	내용
SID	세션 ID
NAME	시스템 각 상태의 이름
VALUE	각 상태 값의 누적치

- sample SQL

```
iSQL> SELECT sid, TO_CHAR(SYSDATE,'HH:MI:SS'), name, value FROM V$SESSTAT
       2 WHERE name IN ('logon cumulative', 'execute success count', 'prepare success count',
       3                'memory table cursor full scan count', 'memory table cursor index scan count',
       4                'disk table cursor full scan count', 'disk table cursor index scan count',
       5                'update retry count', 'lock row retry count'
       6                );
```

PERFORMANCE VIEW

❖ V$SESSION

- ALTIBASE 내부에 생성된 CLIENT 세션 정보
- 칼럼 정보

칼럼	내용
ID	세션 ID
TRANS_ID	현재 사용하고 있는 TRANSACTION ID
TASK_STATE	WATING, READY, EXECUTING, QUEUE WAIT, QUEUE READY , UNKNOWN
COMM_NAME	클라이언트의 접속 정보
OPENED_STMT_COUNT	statement 개수
CLIENT_PID	CLIENT 프로세스 ID
AUTOCOMMIT_FLAG	autocommit 모드 확인(0/1)
SESSION_STATE	INIT, AUTH, SERVICE READY, SERVICE, END, ROLLBACK, UNKNOWN
CURRENT_STMT_ID	현재 수행중인 statement ID. recursive SQL은 제외
LOGIN_TIME	CLIENT 접속 시간 TO_CHAR(TO_DATE('1970010109','yyyymmddhh24') + login_time/60/60/24, 'yyyymmdd hh:mi:ss')

PERFORMANCE VIEW

❖ V$SESSIONMGR

- 세션 통계 정보
- 비정상 종료된 세션 누적 통계

칼럼	내용
IDLE_TIMEOUT_COUNT	발생된 idle timeout의 횟수
QUERY_TIMEOUT_COUNT	발생된 query timeout의 횟수
DDL_TIMEOUT_COUNT	발생된 DDL timeout의 횟수
FETCH_TIMEOUT_COUNT	발생된 fetch timeout의 횟수
UTRANS_TIMEOUT_COUNT	발생된 utrans timeout의 횟수
SESSION_TERMINATE_COUNT	sysdba에 의해 강제로 연결이 끊긴 횟수

PERFORMANCE VIEW

- sample SQL
  - 접속 유형 별 세션 수

```
iSQL> SELECT  SUBSTR(comm_name, 1,4) con_type, COUNT(*) cnt
        2 FROM   V$SESSION
        3 GROUP BY SUBSTR(comm_name, 1,4);
CON_TYPE                                  CNT
------------------------------------------------------------------
TCP                                       105
UNIX                                      1
IPC                                       100
```

  - 비정상 종료된 세션 누적치

```
iSQL> SELECT  login_timeout_count,  idle_timeout_count, query_timeout_count,
        2               fetch_timeout_count, utrans_timeout_count, session_terminate_count
        3 FROM  V$SESSIONMGR;

LOGIN_TIMEOUT_COUNT     : 0
IDLE_TIMEOUT_COUNT      : 0
QUERY_TIMEOUT_COUNT     : 10
FETCH_TIMEOUT_COUNT     : 0
UTRANS_TIMEOUT_COUNT    : 0
SESSION_TERMINATE_COUNT : 0
```

PERFORMANCE VIEW

❖ V$STATEMENT

- 현재 연결된 세션 별로 가장 최근에 실행된 구문(statement) 정보
- LONG RUN QUERY 확인
- time 정보를 수집하기 위해서는 TIMED_STATISTICS=1로 설정
- time 정보는 microsec 단위
- 칼럼 정보

칼럼	설명
ID	statement ID
SESSION_ID	세션 ID
TX_ID	해당 SQL이 속해있는 TRANSACTION ID
QUERY	실제 SQL 문장 (16Kbyte까지만 표현)
TOTAL_TIME	SQL이 최종 수행된 시점의 전체 소요시간
EXECUTE_TIME	SQL이 DBMS 내부에서 처리된 순수 소요시간 (SELECT의 경우 첫번째 FETCH가 일어나지 전까지 수행된 시간)
FETCH_TIME	최종 SQL이 데이터베이스와 프로그램간 발생한 통신의 누적 소요 시간
PARSE_TIME	parse 하는데 소요된 시간
VALIDATE_TIME	validation 체크하는데 소요된 시간

PERFORMANCE VIEW

➢ 칼럼 정보

칼럼	설명
EXECUTE_SUCCESS	동일한 SQL이 성공적으로 수행된 누적횟수(동일 세션에서 누적치)
PROCESS_ROW	INSERT/UPDATE/DELETE/MOVE 문으로 처리된 레코드 수
EVENT	대기 이벤트 명
WAIT_TIME	대기로 인해 소모된 시간을 의미하며 millisecond 단위
SECOND_IN_TIME	대기로 인해 소모된 시간을 의미하며 second단위

➢ sample SQL

- 전체 statement 개수

```
iSQL> SELECT COUNT(*) stmt_cnt FROM V$STATEMENT;
```

- 쿼리 정보

```
iSQL> ALTER SYSTEM SET TIMED_STATISTICS=1;
iSQL> SELECT session_id, id stmt_id, tx_id,
        2               (parse_time+validate_time+optimize_time) prepare_time, fetch_time, execute_time,
        3               total_time, execute_flag,
        4               DECODE(last_query_start_time, 0, '-', TO_CHAR(TO_DATE('1970010109', 'yyyymmddhh')
        5             + last_query_start_time / (24*60*60), 'mm/dd hh:mi:ss'))
        6               last_start_time, NVL(LTRIM(query), 'NONE') query
        7 FROM V$STATEMENT
        8 ORDER BY execute_time DESC ;
```

PERFORMANCE VIEW

V$LOCK

- 모든 테이블에 대한 lock 노드 정보
- 칼럼 정보

칼럼	설명
LOCK_ITEM_TYPE	LOCK이 획득 된 대상의 종류 (TBL, TBS, DBF)
TBS_ID	LOCK이 획득 된 대상이 속한 테이블스페이스 ID
TABLE_OID	LOCK이 획득 된 테이블의 OID
TRANS_ID	TRANSACTION ID
LOCK_DESC	LOCK 유형 (X_LOCK, IX_LOCK, IS_LOCK)
IS_GRANT	테이블 level의 LOCK을 획득했는지 여부 (1: 획득 0: 대기)

PERFORMANCE VIEW

V$LOCK_WAIT

- TRANSACTION 간의 lock 대기 정보
- 칼럼 정보

칼럼	설명
TRANS_ID	TRANSACTION의 고유번호
WAIT_FOR_TRANS_ID	TRANS_ID가 대기하고 하는 TRANSACTION의 고유번호

PERFORMANCE VIEW

❖ V$TRANCSACTION

- ➢ 현재 수행중인 모든 TRANSACTION 정보
- ➢ 주로 V$LOCK, V$MEMGC등 다른 뷰와 조인해서 사용
- ➢ 칼럼 정보

칼럼	설명
ID	TRANSACTION ID
SESSION_ID	세션 ID
STATUS	0: BEGIN, 1: PRECOMMIT, 2: COMMIT_IN_MEMORY, 3: COMMIT, 4: ABORT, 5: BLOCKED, 6: END
LOG_TYPE	0: 일반, 1: 이중화 관련
FIRST_UPDATE_TIME	최초 변경이 일어난 시각TO_CHAR(TO_DATE('1970010109','YYYYMMDDHH') + first_update_time / (60*60*24), 'MM/DD HH:MI:SS'))
DDL_FLAG	0: non-DDL, 1: DDL
FIRST_UNDO_NEXT_LSN_FILENO	TRANSACTION이 처음 기록한 로그의 파일 번호
UPDATE_SIZE	UPDATE 시 작성된 로그의 크기 LOCK_ESCALATION_MEMORY_SIZE 설정 값보다 크면 in-place MVCC로 동작
MEMORY_VIEW_SCN	메모리 테이블에 대해 열려있는 커서의 view SCN 중 가장 작은 값

PERFORMANCE VIEW

- sample SQL
  - lock 잡고 있는 SQL 확인

```
iSQL> SELECT tx.id TX_ID, lw.wait_for_trans_id BLOCKED_TX_ID, l.lock_desc,
        2              DECODE(tx.log_type, 0, st.db_username, 'REPLICATION') USER_NAME,
        3              DECODE(tx.first_update_time, 0, '0', to_char(to_date('1970010109','YYYYMMDDHH')
        4              + tx.first_update_time / (60*60*24), 'MM/DD HH:MI:SS')) FIRST_UPDATE_TIME,
        5              DECODE(tx.status, 0, 'BEGIN', 1, 'PRECOMMIT', 2, 'COMMIT_IN_MEMORY',3, 'COMMIT',
        6               4, 'ABORT', 5, 'BLOCKED', 6, 'END') STATUS, st.query current_query
        7  FROM V$TRANSACTION tx, V$LOCK l  LEFT OUTER JOIN V$LOCK_WAIT lw ON l.trans_id = lw.trans_id
        8              LEFT OUTER JOIN (SELECT st.query,tx_id, ss.db_username
        9                                 FROM V$STATEMENT st, V$SESSION ss
       10                                WHERE ss.id = st.session_id ) st
       11             ON l.trans_id = st.tx_id
       12 WHERE tx.id = l.trans_id;

TX_ID             : 103489
BLOCKED_TX_ID     :
LOCK_DESC         : IX_LOCK
USER_NAME         : SYS
FIRST_UPDATE_TIME : 09/02 14:42:35
STATUS            : BEGIN
CURRENT_QUERY     : update t1 set c1=1 where c1 between 1.11  and 1.112
TX_ID             : 4288
BLOCKED_TX_ID     : 103489
LOCK_DESC         : IX_LOCK
FIRST_UPDATE_TIME : 0
STATUS            : BLOCKED
CURRENT_QUERY     : update t1 set c1=1 where c1 =1.11
```

PERFORMANCE VIEW

V$SERVICE_THREAD

- service thread 정보
- 칼럼 정보

칼럼	설명
TYPE	service thread 접속 방법 SOCKET: TCP 혹은 Unix Domain 방식 IPC: IPC 방식
STATE	service thread 현재 상태 NONE: service thread 초기화 상태 POLL: service thread가 이벤트를 대기하는 상태 QUEUE-WAIT: service thread가 queue를 대기하는 상태 EXECUTE: service thread가 statement를 수행중인 상태
RUN_MODE	service thread 운영 모드 SHARED: TCP로 연결된 작업을 처리 DEDICATED: IPC로 연결된 작업을 처리
SESSION_ID	service thread가 수행중인 session ID
STATEMEMT_ID	service thread가 수행중인 statement ID
TASK_COUNT	service thread에 할당된 session 개수
REDAY_TASK_COUNT	Service thread가 요청을 처리해 주기를 대기하고 있는 session 수

PERFORMANCE VIEW

- sample SQL
  - service thread의 상태 확인

```
iSQL> SELECT RPAD(type, 30), state, count(*)
        2 FROM V$SERVICE_THREAD
        3 GROUP BY type, state;
RPAD(TYPE, 30)                 STATE       COUNT
------------------------------------------------------------------------
SOCKET                         EXECUTE     1
SOCKET                         POLL        7
IPC                            EXECUTE     1
IPC                            POLL        9
```

PERFORMANCE VIEW

❖ V$MEMGC

- 메모리 garbage collection 정보 확인
- aging 할 대상 증가 여부 확인
- GC가 대기하는 TRANSACTION 조회
  - V$TRANSACTION 과 V$STATEMENT와 조인
- 칼럼 정보

칼럼	설명
GC_NAME	GC 이름 MEM_LOGICAL_AGER는 인덱스의 old version을 삭제하는 GC MEM_DELTHR는 테이블 레코드의 old version을 삭제하는 GC
MINMEMSCNINITX	메모리 관련 TRANSACTION 중 가장 작은 view SCN ALTIBASE 가장 오래된 old version의 번호
ADD_OID_CNT	aging을 위해 추가된 OID list 개수
GC_OID_CNT	aging 된 OID list 개수

PERFORMANCE VIEW

- sample SQL
  - 메모리 ager의 gap 증가 확인

```
iSQL> SELECT gc_name, add_oid_cnt, gc_oid_cnt , add_oid_cnt - gc_oid_cnt gcgap
        2 FROM V$MEMGC;
GC_NAME                ADD_OID_CNT          GC_OID_CNT           GCGAP
------------------------------------------------------------------------
MEM_LOGICAL_AGER       1275                 1275                 0
MEM_DELTHR             1275                 1275                 0
```

  - **GC GAP(ADD_OID_CNT - GC_OID_CNT) 값이 증가되고 있다면, aging 할 대상이 증가**

PERFORMANCE VIEW

❖ V$LFG

- 로그파일이 미처 생성되지 못해 TRANSACTION이 로그파일이 생성될 때까지 대기한 수 확인
- 칼럼 정보

칼럼	설명
CUR_WRITE_LF_NO	기록 로그 파일번호
LF_PREPARE_COUNT	미리 생성한 로그파일의 수
LF_PREPARE_WAIT_COUNT	로그 스위치 시 대기한 횟수
END_LSN_FILE_NO	restart redo가 시작될 LSN의 파일번호
END_LSN_OFFSET	restart redo가 시작될 LSN의 오프셋

- sample SQL
  - 로그 파일이 생성되기를 기다린 수

```
iSQL> SELECT lf_prepare_wait_count FROM V$LFG;
LF_PREPARE_WAIT_COUNT
-----------------------
0
```

PERFORMANCE VIEW

V$MEM_TABLESPACES

- 메모리 테이블스페이스 정보
- 메모리 테이블스페이스의 사용량 확인
- 칼럼 정보

칼럼	설명
SPACE_ID	테이블스페이스 ID
SPACE_NAME	테이블스페이스 이름
MAXSIZE	테이블스페이스 최대 크기 DECODE(MAXSIZE, 0, ALLOC_PAGE_COUNT*PAGE_SIZE, MAXSIZE)
ALLOC_PAGE_COUNT	테이블스페이스의 전체 페이지 개수
FREE_PAGE_COUNT	테이블스페이스의 free 페이지 개수

PERFORMANCE VIEW

- sample SQL
  - 메모리 테이블스페이스 사용량 조회

```
iSQL> SELECT space_id, space_name, autoextend_mode,
        2                DECODE(maxsize, 140737488322560, 'UNDEFINED',
        3                                  0, alloc_page_count*32/1024,
        4                                  maxsize/1024/1024)  'MAX(M)',
        5               alloc_page_count * 32/1024  'TOTAL(M)',
        6               (alloc_page_count-free_page_count)*32/1024 'ALLOC(M)',
        7               ((alloc_page_count-free_page_count)*32/1024)
        8              /DECODE(maxsize, 0, alloc_page_count*32/1024, maxsize/1024/1024)  'USAGE(%)'
        9  FROM V$MEM_TABLESPACES;
```

- MAX(M) - 메모리 테이블스페이스 maxsize
  테이블스페이스 생성 시 지정
- TOTAL(M) - 메모리 테이블스페이스가 현재까지 할당 받은 크기
- ALLOC(M) - 메모리 테이블스페이스가 현재까지 할당 받은 페이지 중 빈 페이지를 제외한 사용 공간
- USAGE(%) - 메모리 테이블스페이스가 최대로 할당할 수 있는 크기 대비 사용중인 공간에 대한 백분율

PERFORMANCE VIEW

❖ V$TABLESPACES

- 전체 테이블스페이스(디스크 / 메모리) 정보
- 칼럼 정보

칼럼	설명
ID	테이블스페이스 ID
NAME	테이블스페이스 이름
TYPE	테이블스페이스 타입
STATE	테이블스페이스 상태
SEGMENT_MANAGEMENT	테이블스페이스에서 세그먼트를 생성할 때 어떤 타입으로 생성할 것인지 나타냄 (AUTO, BITMAP, CIRCULAR)
TOTAL_PAGE_COUNT	테이블스페이스의 전체 페이지 개수
ALLOCATE_PAGE_COUNT	테이블스페이스에 할당된 페이지 개수
PAGE_SIZE	테이블스페이스의 페이지 크기
EXTENT_PAGE_COUNT	extent의 페이지 개수

PERFORMANCE VIEW

❖ V$DATAFILES

- 디스크 테이블스페이스 데이터파일 정보
- 데이터파일의 사용량 확인
- 칼럼 정보

칼럼	설명
ID	데이터파일 ID
NAME	데이터파일 경로와 이름
SPACEID	데이터파일이 속한 테이블스페이스 ID
MAXSIZE	데이터파일 생성 시 지정한 MAXSIZE AUTOEXTEND OFF 이면 0
INITSIZE	데이터파일 생성 시 지정한 SIZE
NEXTSIZE	데이터파일 생성 시 지정한 NEXT AUTOEXTEND OFF 이면 0
CURRSIZE	데이터파일의 현재 크기 AUTOEXTEND OFF이면 INITSIZE와 같음
AUTOEXTEND	0: OFF, 1: ON
STATE	1: 오프라인, 2: 온라인, 4: 백업시작, 8: 백업종료, 128: 삭제(dropped)

PERFORMANCE VIEW

- sample SQL
  - 디스크 테이블스페이스의 데이터파일 별 사용량 확인

```
iSQL> SELECT b.name tbs_name, a.id 'FILE#', a.name datafile_name,
        2               currsize*8/1024 'ALLOC(M)',
        3               ROUND(CASE2(a.maxsize=0, currsize, a.maxsize)*8/1024) 'MAX(M)',
        4               DECODE(autoextend, 0, 'OFF', 'ON') 'AUTOEXTEND'
        5 FROM V$DATAFILES a,
        6             V$TABLESPACES b
        7 WHERE b.id = a.spaceid
        8 ORDER BY b.name, a.id;
```

- ALLOC(M) - 데이터파일의 현재 크기
- MAX(M) - 데이터파일이 AUTOEXTEND ON으로 생성되었을 경우에 확장될 수 있는 최대 크기
  AUTOEXTEND OFF 로 생성되었다면 생성 시 크기

PERFORMANCE VIEW

❖ V$MEMTBL_INFO

- 메모리 테이블 정보
- 메모리 테이블 사용량 확인
- 칼럼 정보

칼럼	설명
TABLESPACE_ID	테이블스페이스 ID
TABLE_OID	테이블 식별자 system_.sys_tables_와 조인하여 table_name 확인
FIXED_ALLOC_MEM	테이블에서 할당한 고정 영역의 메모리 크기
FIXED_USED_MEM	테이블에서 실제 사용하고 있는 고정 영역의 메모리 크기
VAR_ALLOC_MEM	테이블에서 할당한 가변 영역의 메모리 크기
VAR_USED_MEM	테이블에서 실제 사용하고 있는 가변 영역 메모리 크기
UNIQUE_VIOLATION_COUNT	UNIQUE 제약조건이 위반된 횟수
UPDATE_RETRY_COUNT	UPDATE 시 재시도한 횟수
DELETE_RETRY_COUNT	DELETE 시 재시도한 횟수

PERFORMANCE VIEW

- sample SQL
  - 휘발성 테이블스페이스와 메모리 테이블스페이스에 속한 테이블 사용량 확인

```
iSQL> SELECT a.user_name, b.table_name, d.name tablespace_name,
        2               (c.fixed_alloc_mem + c.var_alloc_mem)/(1024*1024) 'ALLOC(M)',
        3               (c.fixed_used_mem + c.var_used_mem)/(1024*1024) 'USED(M)',
        4               (c.fixed_used_mem + c.var_used_mem)/(c.fixed_alloc_mem + c.var_alloc_mem)*100
        5                       'EFFICIENCY(%)'
        6 FROM SYSTEM_.SYS_USERS_ a,  SYSTEM_.SYS_TABLES_ b, V$MEMTBL_INFO c, V$TABLESPACES d
        7 WHERE a.user_name <> 'SYSTEM_'
        8 AND b.table_type = 'T'
        9 AND a.user_id = b.user_id
       10 AND b.table_oid = c.table_oid
       11 AND b.tbs_id = d.id
       12 ORDER BY 4 DESC ;
```

- ALLOC(M) – 테이블이 할당 받은 메모리 합계
  (FIXED_ALLOC_MEM+VAR_ALLOC_MEM)
- USED(M) – 테이블이 할당 받은 페이지중에서 “ 실제로 데이터가 적재된 페이지 ” 의 메모리 합계
  (FIXED_USED_MEM+VAR_USED_MEM)
  예를 들어, ALLOC이 100M인 테이블에 전체 DELELE 수행하면 ALLOC은 변함없으나 USED는 0에 가까움
- EFFICIENCY(%) – 테이블이 소유한 페이지 중 “ 실제로 데이터가 적재된 페이지 ” 에 대한 백분율로 공간 효율성을 나타냄

PERFORMANCE VIEW

❖ V$DISKTBL_INFO

- 디스크 테이블 정보
- 디스크 테이블 사용량 확인
- 칼럼 정보

칼럼	설명
TABLESPACE_ID	테이블스페이스 ID
TABLE_OID	테이블 식별자 system_.sys_tables_와 조인하여 table_name 확인
DISK_PAGE_CNT	테이블에서 데이터를 갖고 있는 페이지 개수

- sample SQL

```
iSQL> SELECT user_name, a.table_name, d.name tbs_name,
        2               ROUND((b.disk_page_cnt*8)/1024) 'ALLOC(M)'
        3 FROM SYSTEM_.SYS_TABLES_ a, V$DISKTBL_INFO b, SYSTEM_.SYS_USERS_ c, V$TABLESPACES d
        4 WHERE a.table_oid = b.table_oid
        5 AND a.user_id = c.user_id
        6 AND a.tbs_id=d.id
        7 AND c.user_name <> 'SYSTEM_'
```

PERFORMANCE VIEW

V$REPSENDER

- 이중화 sender 정보
- 이중화 sender가 동작 중일 때만 확인 가능
- 칼럼 정보

칼럼	설명
REP_NAME	이중화 이름
STATUS	sender의 현재 상태 0: STOP, 1: RUN, 2: RETRY
NET_ERROR_FLAG	네트워크 오류 여부 0: OK 1: ERROR
REPL_MODE	이중화가 설정된 모드 EAGER, LAZY
XSN	remote로 전송한 로그의 일련번호
START_FLAG	이중화 start 시 지정한 옵션 NORMAL: 0, QUICK: 1, SYNC: 2, SYNC_ONLY: 3, SYNC RUN : 4, SYNC END: 5, RECOVERY from Replication : 6, OFFLINE: 7

PERFORMANCE VIEW

V$REPRECEIVER

- 이중화 receiver 정보
- remote SERVER의 sender가 start되면 확인 가능
- 칼럼 정보

칼럼	설명
REP_NAME	이중화 이름
APPLY_XSN	Receiver가 적용중인 로그의 SN
INSERT_SUCCESS_COUNT	Receiver가 성공적으로 적용한 INSERT 로그레코드의 수
INSERT_FAILURE_COUNT	Receiver가 적용에 실패한 INSERT 로그레코드의 수
UPDATE_SUCCESS_COUNT	Receiver가 성공적으로 적용한 UPDATE 로그레코드의 수
UPDATE_FAILURE_COUNT	Receiver가 적용에 실패한 UPDATE 로그레코드의 수
DELETE_SUCCESS_COUNT	Receiver가 성공적으로 적용한 DELETE 로그레코드의 수
DELETE_FAILURE_COUNT	Receiver가 적용에 실패한 DELETE 로그레코드의 수

- SUCCESS_COUNT, FAILURE_COUNT
  - TIMED_STATISTICS=1 로 설정되어있는 경우만 확인 가능

PERFORMANCE VIEW

❖ V$REPGAP

- 이중화 sender의 작업 로그와 local SERVER에 생성된 최근 로그 파일간의 차이
- 이중화가 정상적으로 수행되고 있는지, 밀리지는 않는지 확인할 때 사용
- 이중화 sender가 동작 중일 때만 확인 가능
- 칼럼 정보

칼럼	설명
REP_NAME	이중화 이름
REP_SN	현재 전송 중인 로그 레코드 식별 번호
REP_LAST_SN	마지막 로그 레코드 식별 번호
REP_GAP	REP_LAST_SN 과 REP_SN 의 차이
READ_FILE_NO	현재 읽고 있는 로그 파일 번호
START_FLAG	이중화 start 시 지정한 옵션 NORMAL : 0, QUICK : 1, SYNC: 2, SYNC_ONLY: 3, SYNC RUN : 4, SYNC END: 5, RECOVERY from Replication : 6, OFFLINE : 7, PARALLEL : 8

PERFORMANCE VIEW

- sample SQL
  - Sender 확인

```
iSQL> SELECT rep_name, status FROM V$REPSENDER;
REP_NAME                                  STATUS
------------------------------------------------------------
REP1                                      1

→ status가 1이 아닐 경우 이중화는 정상적이 않음
```

  - Receiver 확인

```
iSQL> SELECT rep_name FROM V$REPRECEIVER;
REP_NAME
---------------------------------
REP1

→ 데이터가 조회되지 않을 경우 receiver가 정상적이지 않음
```

  - 이중화 갭 확인

```
iSQL> SELECT rep_name, rep_last_sn, rep_sn,  rep_last_sn-rep_sn FROM V$REPGAP;
REP_NAME     REP_LAST_SN        REP_SN             REP_LAST_SN-REP_SN
------------------------------------------------------------------------
REP          12638584           -1                 12638585

→ rep_sn 값이 -1 이면 원격 SERVER에 이중화를 생성하지 않음
    rep_last_sn-rep_sn 값이 증가하는데 rep_sn 값이 변하지 않으면 이중화 장애
```

ALTIBASE OPERATION

OS CHECK LIST

OS CHECK LIST

ALTIBASE 프로세스 확인

```
Shell> ps -ef | grep "altibase -p" | grep -v grep
altibase   11300     1  0 Sep22 ?      00:00:10 /altibase/altibase_home/bin/altibase -p boot from admin
```

CPU 사용률

```
Shell> ps -o pcpu -p 11300
%CPU
 10.1
```

메모리 사용률

```
Shell> ps -o vsz -p 11300
 VSZ
8447864
```

디스크 사용률(ex. linux)

```
Shell> df -k
 Filesystem           1K-blocks      Used    Available Use% Mounted on
/dev/sda1             20641788     1625008   17968140   9% /
none                   8211648     3924128    4287520  48% /dev/shm
/dev/sdb1           1663084532  1395923292  182681432  89% /home
/dev/sdb2           1701178400   462146088 1152617476  29% /alti_data
/dev/sda3            386418928   101322240  265467668  28% /alti_logs
/dev/sda5             20641788      669688   18923460   4% /opt
…
```

OS CHECK LIST

❖ OS 시스템 로그

➢ 시스템 로그를 통해 ALTIBASE가 OS에 의해 종료되었는지 확인

운영제제	확인할 시스템 로그
SUN	/var/adm/message 파일
HP	/var/adm/syslog/syslog.log 파일
AIX	errpt -a
LINUX	/var/log/message 파일

ALTIBASE ADVANCE

MONITORING TOOL

MONITORING TOOL

❖ CONTENTS

- AMS(ALTIBASE MONITORING SYSTEM)
- SQUIRREL SQL CLIENT

MONITORING TOOL

AMS(ALTIBASE MONITORING SYSTEM)

AMS(ALTIBASE MONITORING SYSTEM)

❖ 지원 OS 및 ALTIBASE 버전

구분	지 원	지원가능
OS 종류	Linux 계열 (Ubuntu , CentOS, Redhat)	System Resource 모니터링 가능 ALTIBASE 모니터링 가능
	Linux 계열 외 (HP , AIX , SUN)	System Resource 모니터링 불가 ALTIBASE 모니터링 가능
ALTIBASE 버전	A5 (5.3.5 버전 이상)	
	A6	
	A7	Shard Mornitoring 요청시 추가 가능

❖ 특징

- Open Source 활용으로 상용 S/W 에 대한 추가 비용 없음
- 50여개의 모니터링 메트릭스 제공으로 용도별 선별 사용
- 데이터 저장소로 빠른 시계열(TIME SERIES) 데이터베이스 활용으로 모니터링 대상 데이터베이스 부하 경감
- 사용자가 직접 다양한 chart 및 dashboard 구성 가능

※ 다운로드 URL : https://github.com/bsshin71/ams

AMS(ALTIBASE MONITORING SYSTEM)

❖ AMS HOME DASHBOARD 예시 화면

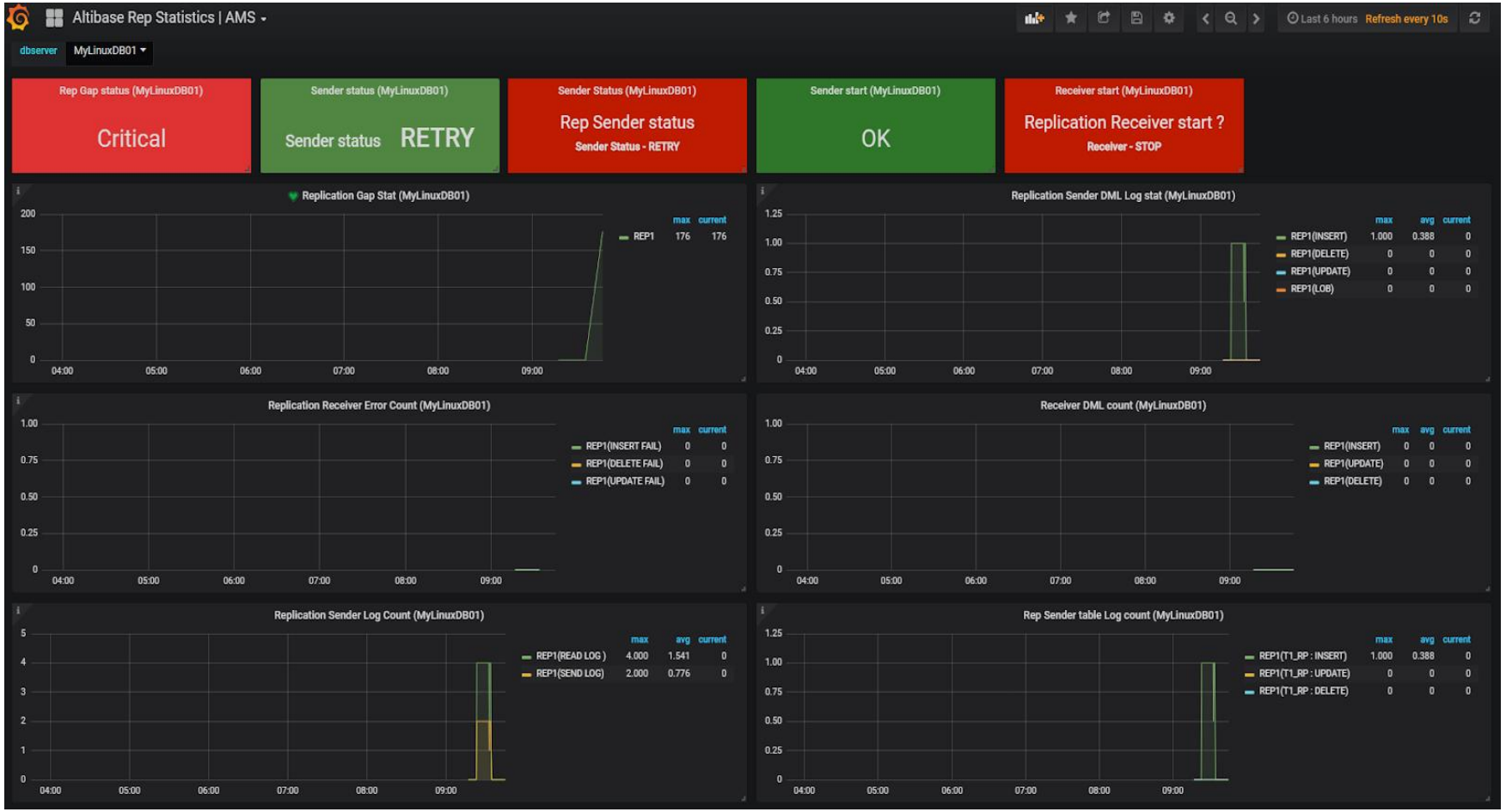
- ➢ 관심 chart 추가 및 변경 가능
- ➢ Layout 일부 변경 가능 (Html 코드 삽입)
- ➢ 여러 대의 데이터베이스 SERVER 추가 가능

AMS(ALTIBASE MONITORING SYSTEM)

❖ AMS SYSTEM OVERVIEW DASHBOARD 전체 예시 화면

AMS(ALTIBASE MONITORING SYSTEM)

❖ AMS 이중화 전체 예시 화면



MONITORING TOOL

SQUIRREL SQL CLIENT

SQUIRREL SQL CLIENT

❖ 시스템 요구 사항

- ALTIBASE : ALTIBASE 6 이상
- SQUIRREL : 3.7.1 Standard

❖ 특징

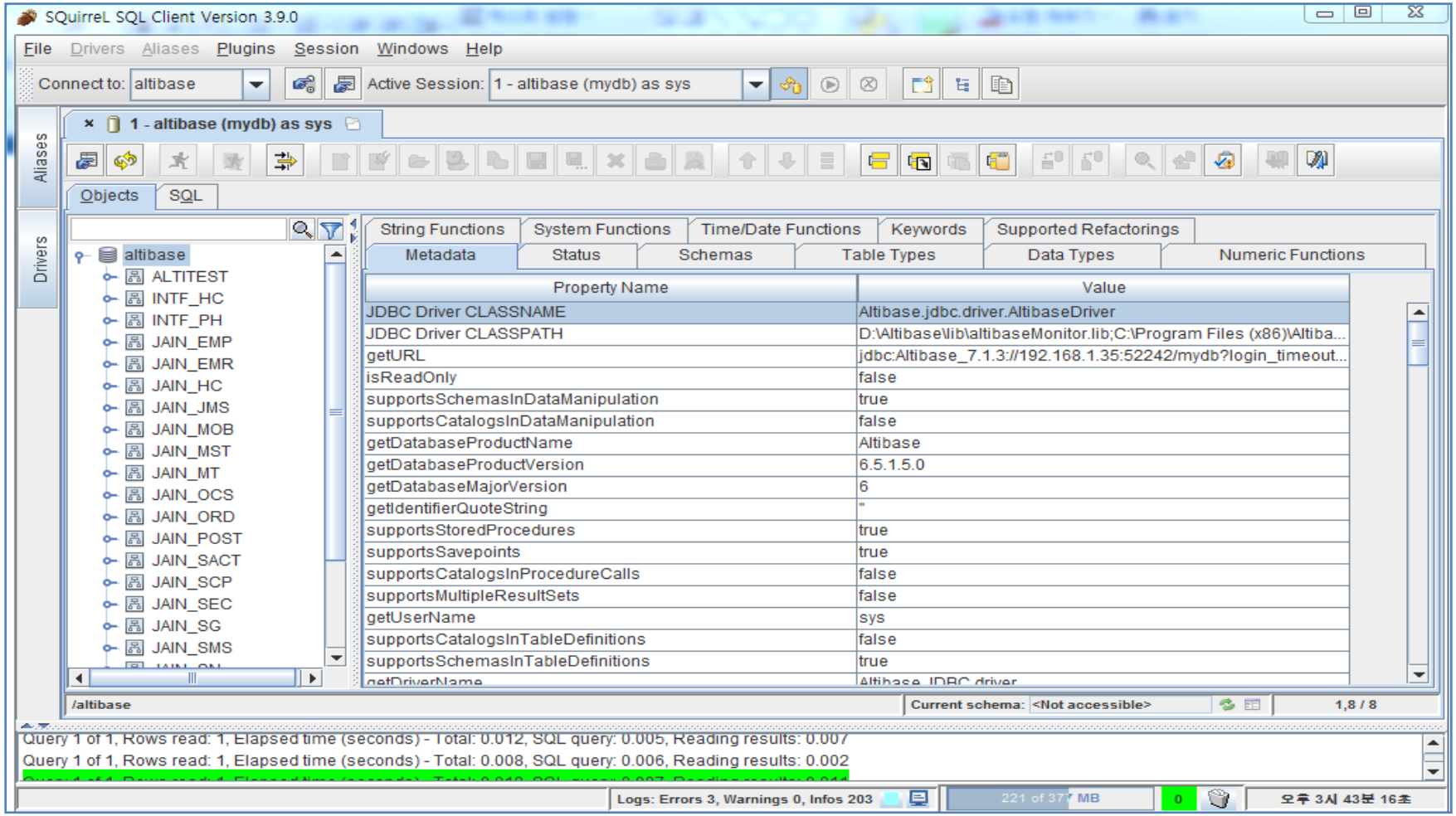
- ALTIBASE 에 접속하여 SQUIRREL SQL CLIENT 를 사용하기 위한 plug-in
- 공식 plug-in 이 아니기 때문에 별도로 사용자가 직접 다운로드

❖ 설치 방법

- https://jaist.dl.sourceforge.net/project/squirrel-sql/1-stable/3.7.1/squirrel-sql-3.7.1-standard.jar 에서 CLIENT 파일 다운로드
- Driver 등록 후 (Altibase.jar) ALTIBASE 연동 가능

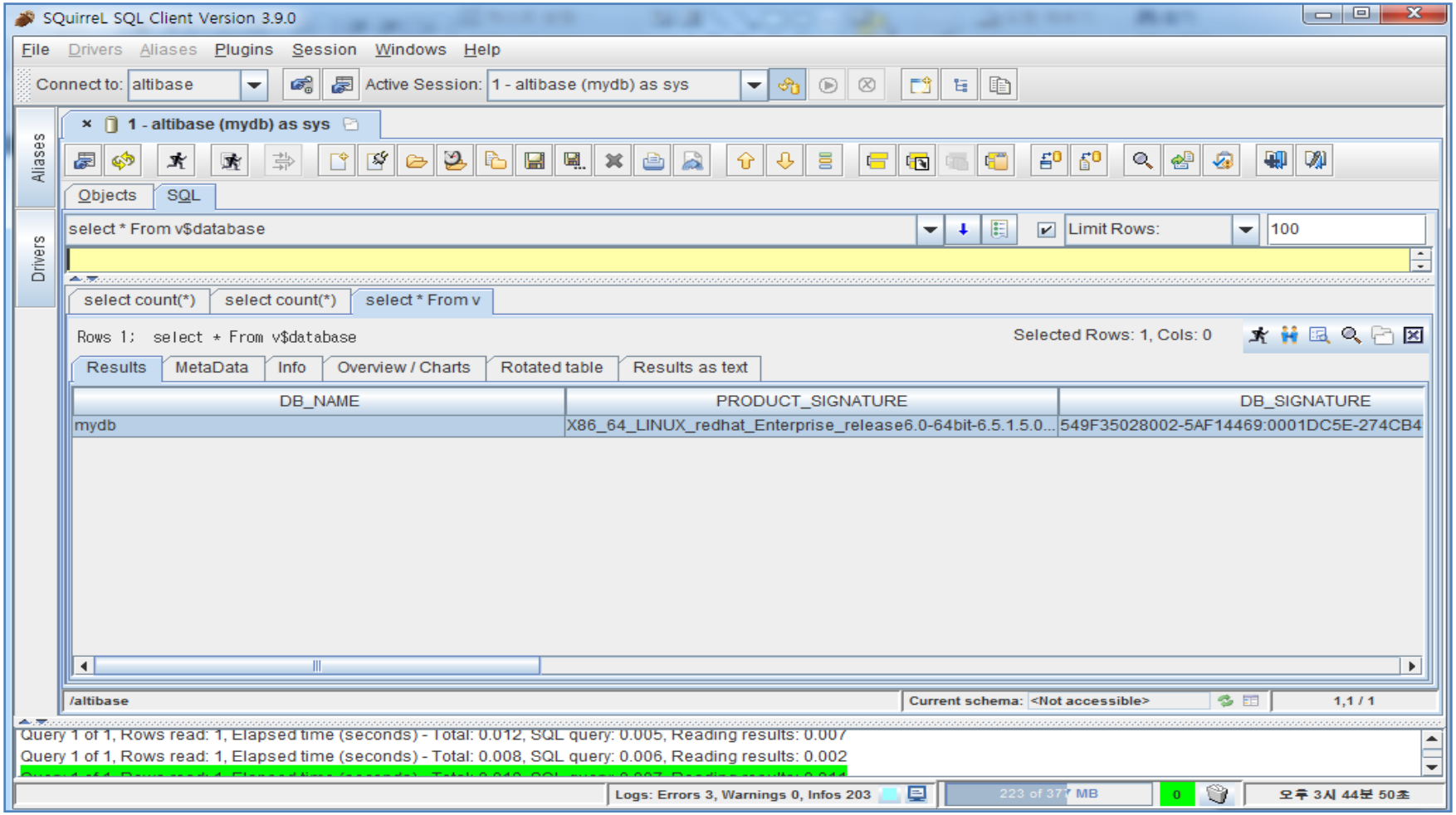
SQUIRREL SQL CLIENT

❖ SQUIRREL OBJECT 화면



SQUIRREL SQL CLIENT

❖ SQUIRREL SQL 화면



ALTIBASE ADVANCE

TECHNICAL SUPPORT

MONITORING TOOL

CONTENTS

- ALTIBASE WIKIPEDIA
- ALTIBASE CUSTOMER SERVICE

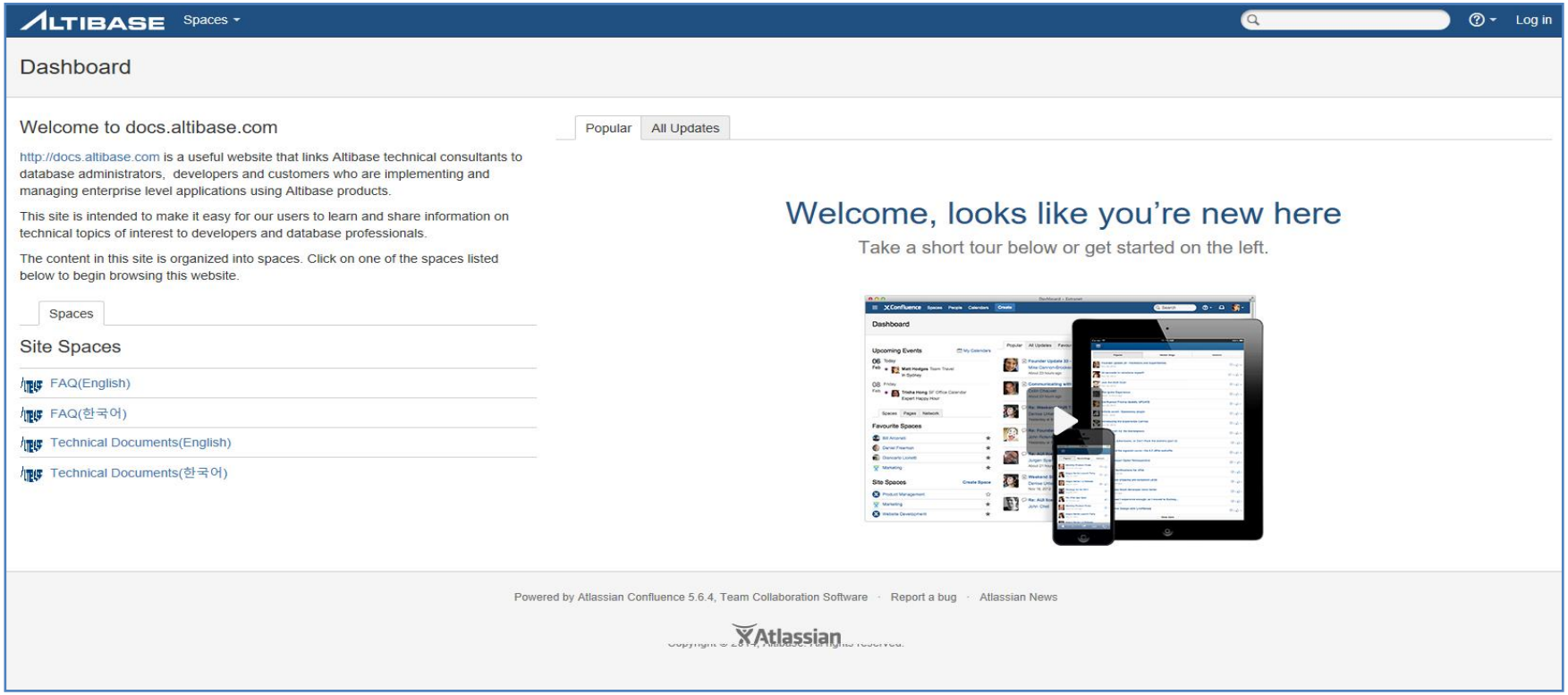
TECHNICAL SUPPORT

ALTIBASE WIKIPEDIA

ALTIBASE WIKIPEDIA

❖ AID SITE

- ALTIBASE 운영 및 개발 관련 정보 제공
  - FAQ
  - TECHNICAL DOCUMENTS
- URL : aid.altibase.com

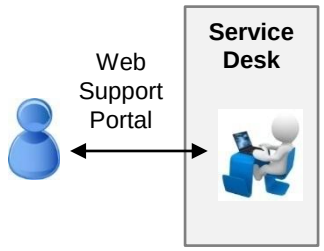
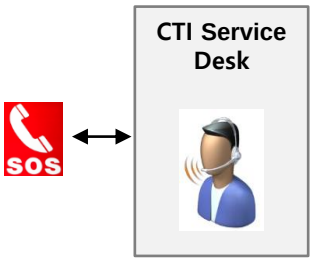
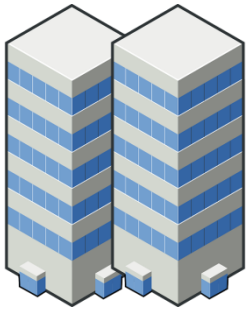


TECHNICAL SUPPORT

ALTIBASE CUSTOMER SERVICE

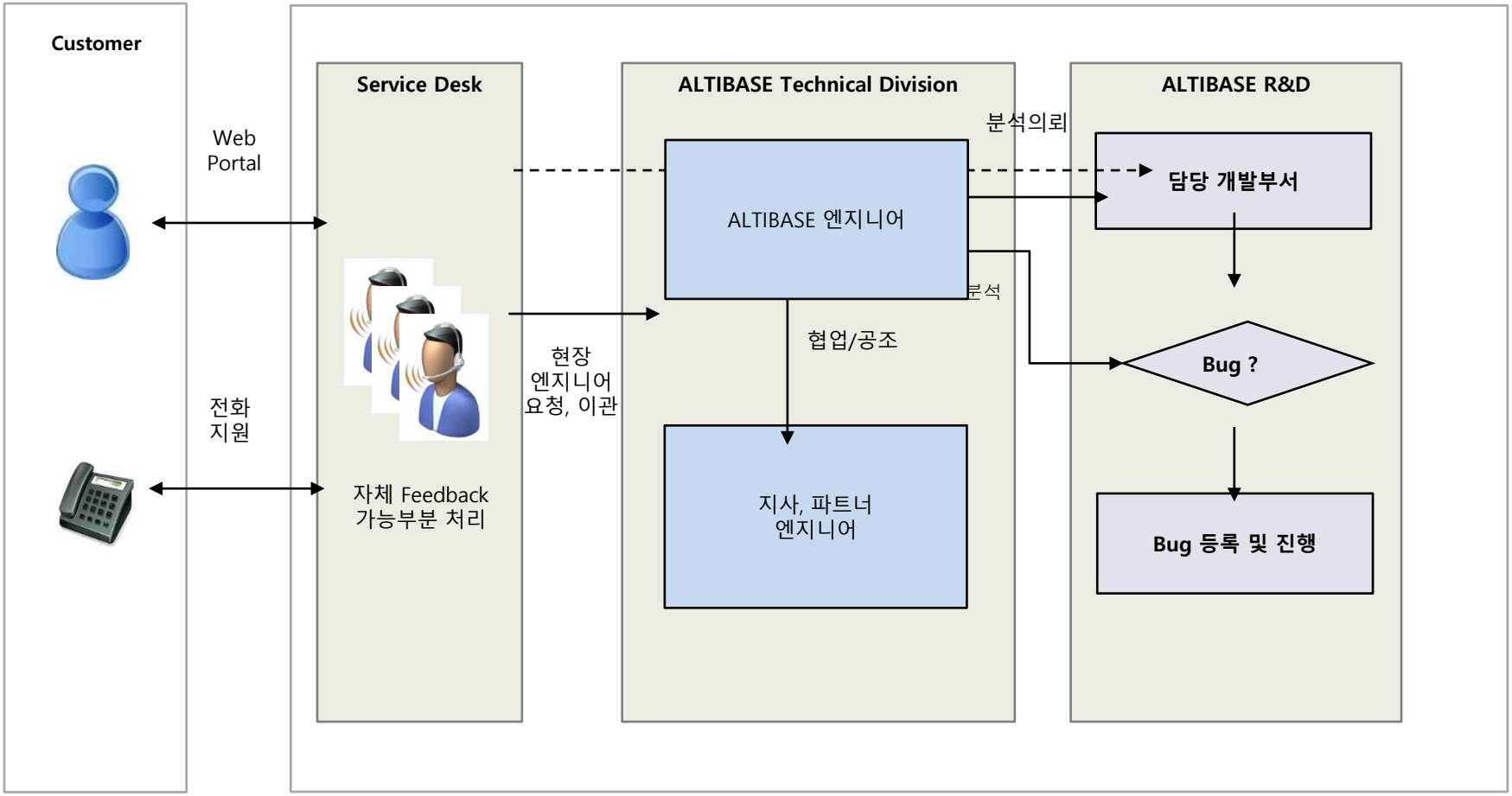
ALTIBASE CUSTOMER SERVICE

❖ ALTIBASE는 일관 되고 신속한 기술 지원 제공

On-Line	On-Call	On-Remote	On-Site
			
1. 기술지원 포털 http://support.altibase.com 2. 등록된 요청에 대한 엔지니어 지정 3. 기술지원 및 이력 관리	1. 기술지원 콜센터 (02) 2082-1114 2. 요청에 대한 전화응대 3. 기술지원 및 지원이력 관리	1. On-Line/On-Call 을 통한 요청 접수 2. 원격지원을 위한 엔지니어 지정 3. 원격지원을 위한 URL 접속 안내 http://rsup.net/altibase 4. 기술지원 및 지원이력 관리	1. On-Line/On-Call 을 통한 요청 접수 2. 엔지니어 지정 및 현장 방문 3. 기술지원 및 지원이력 관리
❖ 다운로드 서비스 • 제품 • Release Notes • Patch Notes • Manual • White Paper ❖ FAQ 제공	❖ 긴급상황을 대비하여 7 X 24 유지 ❖ SID 를 획득한 고객만 이용가능	❖ 긴급상황을 대비하여 7 X 24 유지 ❖ 웹브라우저를 통하여 엔지니어가 고객의 화면을 함께 보면서 지원	❖ SID 를 획득한 고객만 이용가능

ALTIBASE CUSTOMER SERVICE

❖ ALTIBASE는 일관 되고 신속한 기술 지원 제공



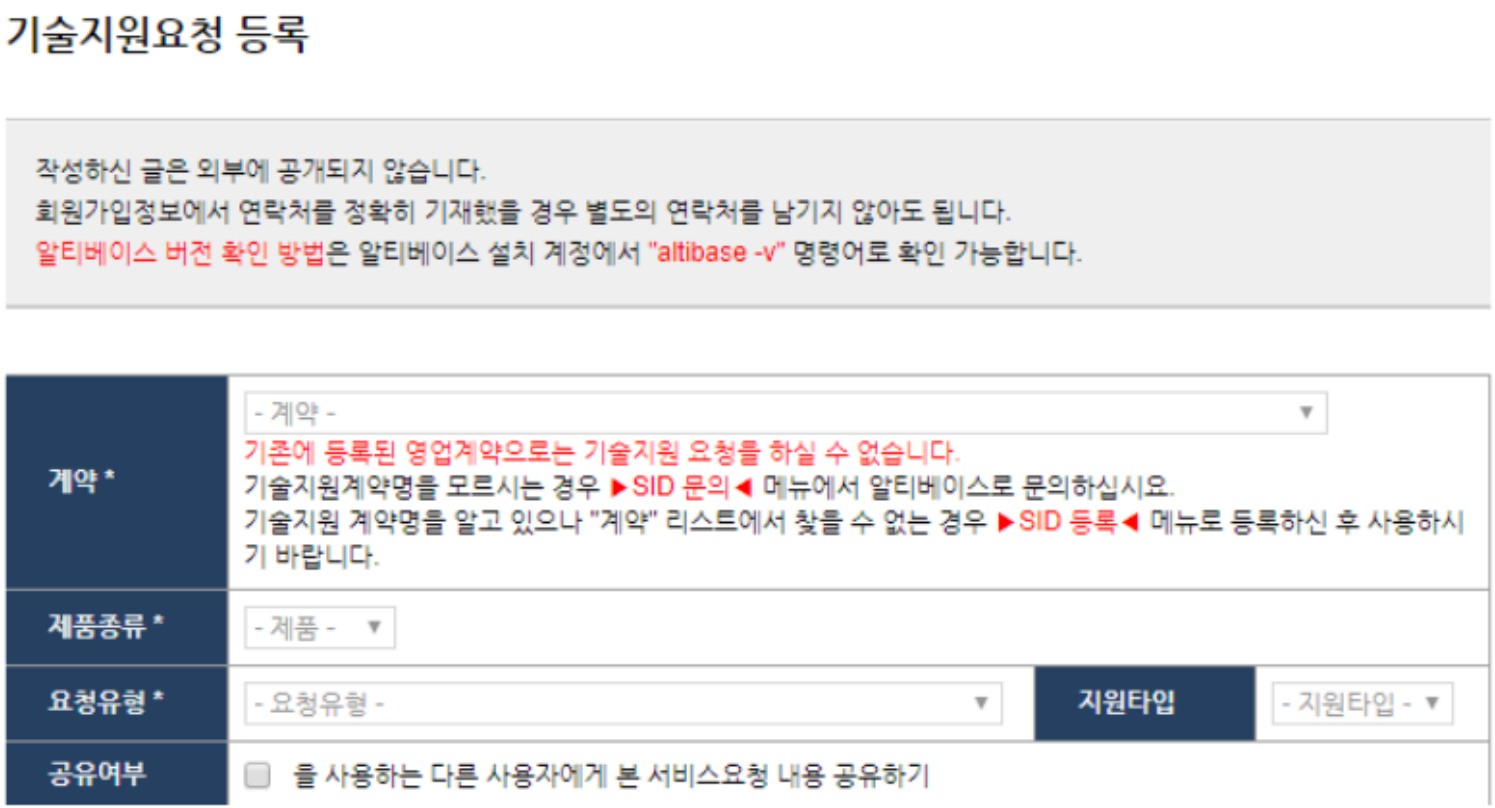
❖ ALTIBASE 기술서비스 요청 방법

- support.altibase.com 에서 간단한 회원 가입
- [고객서비스] -> [SID 등록] 에서 회사명을 검색후 해당 SID 추가

ALTIBASE CUSTOMER SERVICE

❖ ALTIBASE는 일관 되고 신속한 기술 지원 제공

➢ [고객서비스] -> [기술지원요청] -> [기술지원 요청 등록] 을 통하여 기술 지원 요청



Thank you!

ALTIBASE