Altibase FAQ | SQL & Schema Objects

Useful SQL

Miscellaneous queries

State of a service thread

This query shows the state of a service thread.

Column Name	Description
RUN_MODE	the type of service thread. (dedicated or shared)
STATE	currently working state (executing / pool)
CNT	the number of threads in the run_mode and state

Altibase V4

```
SELECT  TYPE RUN_MODE,
        STATE,
        COUNT(*) CNT
FROM    V$SERVICE_THREAD
GROUP BY RUN_MODE, STATE
```

Altibase V5

```
SELECT  RUN_MODE,
        STATE,
        COUNT(*) CNT
FROM    V$SERVICE_THREAD
GROUP BY RUN_MODE, STATE
```

Lock & Transaction Information

This query shows the lock & transaction information including replication transactions.

Column Name	Description
TX_ID	current transaction id
BLOCKED_TX_ID	the transaction id which current transaction is waiting for
STATUS	current transaction's status
USER_NAME	the user name who has issued the current transaction
SESSION_ID	the session id which has issued the current transaction
CLIENT_IP	client IP address
AUTOCOMMIT	autocommit

© Altibase Corp.

Column Name	Description
LOCK_DESC	Lock description
FIRST_UPDATE_TIME	the first update time of current transaction
TABLE_NAME	table_name
CURRENT_QUERY	current query in the current transaction
DDL	DDL or not
LOGFILE#	logfile number

Altibase V4

Icon

Altibase V4 does not support this features.

Altibase V5

```
SELECT  TX.ID TX_ID,
        WAIT_FOR_TRANS_ID BLOCKED_TX_ID,
        DECODE(TX.STATUS, 0, 'BEGIN', 1, 'PRECOMMIT', 2, 'COMMIT_IN_MEMORY', 3, 'COMMIT', 4, 'ABORT', 5,
'BLOCKED', 6, 'END') STATUS,
        DECODE(TX.LOG_TYPE, 0, U1.USER_NAME, 'REPLICATION') USER_NAME,
        DECODE(TX.LOG_TYPE, 0, TX.SESSION_ID, RT.REP_NAME) SESSION_ID,
        DECODE(TX.LOG_TYPE, 0, ST.COMM_NAME, RR.PEER_IP) CLIENT_IP,
        DECODE(ST.AUTOCOMMIT_FLAG, 1, 'ON', 'OFF') AUTOCOMMIT,
        L.LOCK_DESC,
        DECODE(TX.FIRST_UPDATE_TIME, 0, '0', TO_CHAR(TO_DATE('1970010109','YYYYMMDDHH') +
TX.FIRST_UPDATE_TIME / (606024), 'MM/DD HH:MI:SS')) FIRST_UPDATE_TIME,
        U2.USER_NAME||'.'||T.TABLE_NAME TABLE_NAME,
        DECODE(TX.LOG_TYPE, 0, SUBSTR(ST.QUERY, 1, 10), 'REMOTE TX_ID '||REMOTE_TID) CURRENT_QUERY,
        DECODE(TX.DDL_FLAG,0, 'NON-DDL', 'DDL') DDL,
        DECODE(TX.FIRST_UNDO_NEXT_LSN_FILENO, -1, '-', TX.FIRST_UNDO_NEXT_LSN_FILENO) 'LOGFILE#'
FROM      V$TRANSACTION TX,
        V$LOCK L LEFT OUTER JOIN
        (
          SELECT ST.*,
                 SS.AUTOCOMMIT_FLAG,
                 SS.DB_USERID,
                 SS.COMM_NAME
          FROM   V$STATEMENT ST,
                 V$SESSION SS
          WHERE  SS.ID = ST.SESSION_ID
                 AND SS.CURRENT_STMT_ID = ST.ID
         ) ST ON L.TRANS_ID = ST.TX_ID LEFT OUTER JOIN
         V$REPRECEIVER_TRANSTBL RT ON L.TRANS_ID = RT.LOCAL_TID LEFT OUTER JOIN
         V$REPRECEIVER RR ON RT.REP_NAME = RR.REP_NAME LEFT OUTER JOIN
         V$LOCK_WAIT LW ON L.TRANS_ID = LW.TRANS_ID LEFT OUTER JOIN
         SYSTEM_.SYS_USERS_ U1 ON ST.DB_USERID = U1.USER_ID,
         SYSTEM_.SYS_TABLES_ T LEFT OUTER JOIN
         SYSTEM_.SYS_USERS_ U2 ON T.USER_ID = U2.USER_ID
WHERE    TX.ID = L.TRANS_ID AND T.TABLE_OID = L.TABLE_OID AND TX.STATUS != 6 -- 6:END
ORDER BY TX.ID, ST.ID, TX.FIRST_UPDATE_TIME DESC ;
```

Redo log files

This query shows the information of transaction log files

© Altibase Corp.

Column Name	Description
OLDEST_LOG_FILE	the oldest log file in LOG_DIR
CURRENT_LOG_FILE	the current log file number which service thread is writing to.
LOG_FILE_GAP	the interval between CURRENT_LOG_FILE and OLDEST_LOG_FILE

```
SELECT  OLDEST_ACTIVE_LOGFILE OLDEST_LOGFILE,
        CURRENT_LOGFILE CURRENT_LOGFILE,
        CURRENT_LOGFILE-OLDEST_ACTIVE_LOGFILE LOGFILE_GAP
FROM V$ARCHIVE ;
```

Cumulative count of total transaction waits due to logging

This query shows the number of total waits due to lack of the log space exhaustion for logging. If the value returns non-zero value, you'd better increase PREPARE_LOG_FILE_COUNT in altibase.properties.

```
SELECT  LF_PREPARE_WAIT_COUNT
FROM V$LFG;
```

Memory Ager GAP

This query shows the memory ager thread's information.

Column Name	Description
GC_NAME	the Ager name. (LOGICAL_AGER for index aging, DELTHRD for data aging)
SCNOFTAIL	The commit SCN of the tail in garbage collection OID list
MINMEMSCNINTXS	The lowest of the view SCNs for memory-related transactions
GC_GAP	the OID count which ager has to proceed

```
SELECT  GC_NAME,
        SCNOFTAIL,
        MINMEMSCNINTXS,
        ADD_OID_CNT-GC_OID_CNT GC_GAP
FROM V$MEMGC ;
```

Finding a query which blocks ager from aging.

Aging process will be blocked if there's a transaction which refers the aging target. This query shows the query which blocks from aging.

Column Name	Description
SESSION_ID	the session identifier
TOTAL_TIME	total elapsed time
EXECUTE_TIME	the elapsed time for query execution

© Altibase Corp.

Column Name	Description
TX_ID	transaction identifier
QUERY	query string

```
SELECT  SESSION_ID,
        TOTAL_TIME,
        EXECUTE_TIME,
        TX_ID,
        QUERY
FROM        V$STATEMENT
WHERE TX_ID IN  (
                                    SELECT ID
                                    FROM V$TRANSACTION
                                    WHERE MEMORY_VIEW_SCN = (
                                           SELECT MINMEMSCNINTXS
                                           FROM V$MEMGC
                                           LIMIT 1
                                            )
                                ) AND EXECUTE_FLAG = 1
ORDER BY 2 DESC ;
```

Memory status

This query shows the memory state arranged by ALTIBASE module.

Column Name	Description
NAME	the name of ALTIBASE module
ALLOC_MAX(M)	max alloc size of memory
ALLOC(M)	current size of memory

```
SELECT  NAME,
          ROUND(MAX_TOTAL_SIZE/1024/1024) 'ALLOC_MAX(M)',
          ROUND(ALLOC_SIZE/1024/1024) 'ALLOC(M)'
FROM      V$MEMSTAT
ORDER BY 3 DESC ;
```

Total memory usage of ALTIBASE

This query shows the total memory state.

Column Name	Description
ALLOC_MAX(M)	max alloc size of memory
ALLOC(M)	current size of memory

```
SELECT  ROUND(SUM(MAX_TOTAL_SIZE)/1024/1024) 'ALLOC_MAX(M)',
        ROUND(SUM(ALLOC_SIZE)/1024/1024) 'ALLOC(M)'
FROM V$MEMSTAT ;
```

© Altibase Corp.

SQL about Objects

Memory Table

This query returns the object information about a memory table.

Column Name	Description
USER_NAME	the object owner
TABLE_NAME	the object name
TABLESPACE_NAME	the tablespace name which this object belongs to
TABLE_OID	the table object identifier
ALLOC(M)	allocated size of the memory table (Megabyte)
USED(M)	actually used size of the memory table (Megabyte)
EFFICIENCY(%)	store efficiency of the object

Altibase V4

```
SELECT  A.USER_NAME,
        B.TABLE_NAME,
        'SYS_TBS_MEMORY' TABLESPACE_NAME,
        C.TABLE_OID,
        ROUND((C.FIXED_ALLOC_MEM + C.VAR_ALLOC_MEM)/(1024*1024),2) 'ALLOC(M)',
        ROUND((C.FIXED_USED_MEM + C.VAR_USED_MEM)/(1024*1024),2) 'USED(M)',
        ROUND((C.FIXED_USED_MEM + C.VAR_USED_MEM)/(C.FIXED_ALLOC_MEM + C.VAR_ALLOC_MEM)*100,2) 'EFFICIENCY(%)'
FROM    SYSTEM_.SYS_USERS_ A,
        SYSTEM_.SYS_TABLES_ B,
        V$MEMTBL_INFO C
WHERE   A.USER_NAME <> 'SYSTEM_'
        AND B.TABLE_TYPE = 'T'
        AND A.USER_ID = B.USER_ID
        AND B.TABLE_OID = C.TABLE_OID;
```

Altibase V5

```
SELECT  A.USER_NAME,
        B.TABLE_NAME,
        D.NAME TABLESPACE_NAME,
        C.TABLE_OID,
        ROUND((C.FIXED_ALLOC_MEM + C.VAR_ALLOC_MEM)/(1024*1024),2) 'ALLOC(M)',
        ROUND((C.FIXED_USED_MEM + C.VAR_USED_MEM)/(1024*1024),2) 'USED(M)',
        ROUND((C.FIXED_USED_MEM + C.VAR_USED_MEM)/(C.FIXED_ALLOC_MEM + C.VAR_ALLOC_MEM)*100,2) 'EFFICIENCY(%)'
FROM    SYSTEM_.SYS_USERS_ A,
        SYSTEM_.SYS_TABLES_ B,
        V$MEMTBL_INFO C,
        V$TABLESPACES D
WHERE   A.USER_NAME <> 'SYSTEM_'
        AND B.TABLE_TYPE = 'T'
        AND A.USER_ID = B.USER_ID
        AND B.TABLE_OID = C.TABLE_OID
        AND B.TBS_ID = D.ID ;
```

© Altibase Corp.

Queue Table

This query returns the object information about a queue table.

Column Name	Description
USER_NAME	the object owner
TABLE_NAME	the object name
TABLESPACE_NAME	the tablespace name which this object belongs to
TABLE_OID	the table object identifier
ALLOC(M)	allocated size of the memory table (Megabyte)
USED(M)	actually used size of the memory table (Megabyte)
EFFICIENCY(%)	store efficiency of the object

Icon

Queue table structure is the same as a memory table except its functionality.

Altibase V4

```
SELECT  A.USER_NAME,
        B.TABLE_NAME,
        'SYS_TBS_MEMORY' TABLESPACE_NAME,
        C.TABLE_OID,
        ROUND((C.FIXED_ALLOC_MEM + C.VAR_ALLOC_MEM)/(1024*1024),2) 'ALLOC(M)',
        ROUND((C.FIXED_USED_MEM + C.VAR_USED_MEM)/(1024*1024),2) 'USED(M)',
        ROUND((C.FIXED_USED_MEM + C.VAR_USED_MEM)/(C.FIXED_ALLOC_MEM + C.VAR_ALLOC_MEM)*100,2) 'EFFICIENCY(%)'
FROM    SYSTEM_.SYS_USERS_ A,
        SYSTEM_.SYS_TABLES_ B,
        V$MEMTBL_INFO C
WHERE   A.USER_NAME <> 'SYSTEM_'
        AND B.TABLE_TYPE = 'Q'
        AND A.USER_ID = B.USER_ID
        AND B.TABLE_OID = C.TABLE_OID;
```

Altibase V5

```
SELECT  A.USER_NAME,
        B.TABLE_NAME,
        D.NAME TABLESPACE_NAME,
        C.TABLE_OID,
        ROUND((C.FIXED_ALLOC_MEM + C.VAR_ALLOC_MEM)/(1024*1024),2) 'ALLOC(M)',
        ROUND((C.FIXED_USED_MEM + C.VAR_USED_MEM)/(1024*1024),2) 'USED(M)',
        ROUND((C.FIXED_USED_MEM + C.VAR_USED_MEM)/(C.FIXED_ALLOC_MEM + C.VAR_ALLOC_MEM)*100,2) 'EFFICIENCY(%)'
FROM    SYSTEM_.SYS_USERS_ A,
        SYSTEM_.SYS_TABLES_ B,
        V$MEMTBL_INFO C,
        V$TABLESPACES D
WHERE   A.USER_NAME <> 'SYSTEM_'
        AND B.TABLE_TYPE = 'Q'
        AND A.USER_ID = B.USER_ID
        AND B.TABLE_OID = C.TABLE_OID
        AND B.TBS_ID = D.ID ;
```

© Altibase Corp.

Memory table / Queue table Index information

This query returns the index information of a memory table and a queue object.

Column Name	Description
USER_NAME	the object owner
OBJECT_TYPE	the type of object
OBJECT_NAME	the object name
TABLESPACE_NAME	the tablespace name which this object belongs to
INDEX_NAME	the table object identifier
INDEX_TYPE	the table object identifier

Icon

Estimating of memory index is very simple in most cases. Altibase index just consists of pointers, thus you just multiply 16 bytes by the number of table rows to get the size of each index.

Altibase V4

```
SELECT  C.USER_NAME,
        DECODE(F.TABLE_TYPE, 'Q', 'QUEUE', 'T', 'TABLE') OBJECT_TYPE,
        TABLE_NAME OBJECT_NAME,
        'SYS_TBS_MEMORY' TABLESPACE_NAME,
        E.INDEX_NAME,
        RPAD(CASE2(E.INDEX_TYPE=1, 'B-TREE', 'R-TREE'),10,' ') INDEX_TYPE
FROM    V$INDEX B,
        SYSTEM_.SYS_USERS_ C,
        SYSTEM_.SYS_INDICES_ E,
        SYSTEM_.SYS_TABLES_ F
WHERE   B.INDEX_ID = E.INDEX_ID
        AND E.USER_ID = C.USER_ID
        AND F.USER_ID = E.USER_ID
        AND F.TABLE_OID = B.TABLE_OID
        AND C.USER_NAME <> 'SYSTEM_';
```

Altibase V5

```
SELECT  C.USER_NAME,
        DECODE(F.TABLE_TYPE, 'Q', 'QUEUE', 'T', 'TABLE') OBJECT_TYPE,
        TABLE_NAME OBJECT_NAME,
        D.SPACE_NAME TABLESPACE_NAME,
        E.INDEX_NAME,
        RPAD(CASE2(E.INDEX_TYPE=1, 'B-TREE', 'R-TREE'),10,' ') INDEX_TYPE
FROM    V$INDEX B,
        SYSTEM_.SYS_USERS_ C,
        V$MEM_TABLESPACES D,
        SYSTEM_.SYS_INDICES_ E,
        SYSTEM_.SYS_TABLES_ F
WHERE   B.INDEX_ID = E.INDEX_ID
        AND E.USER_ID = C.USER_ID
        AND F.USER_ID = E.USER_ID
        AND F.TBS_ID = D.SPACE_ID
        AND F.TABLE_OID = B.TABLE_OID
        AND C.USER_NAME <> 'SYSTEM_';
```

© Altibase Corp.

Disk Table

This query returns the object information about a disk table.

Column Name	Description
USER_NAME	the object owner
TABLE_NAME	the object name
TABLESPACE_NAME	the tablespace name which this object belongs to
TABLE_OID	the table object identifier
ALLOC(M)	allocated size of the memory table (Megabyte)

```
SELECT  USER_NAME,
        A.TABLE_NAME,
        D.NAME TBS_NAME,
        A.TABLE_OID,
        ROUND((B.DISK_PAGE_CNT*8)/1024) 'ALLOC(M)'
FROM    SYSTEM_.SYS_TABLES_ A,
        V$DISKTBL_INFO B,
        SYSTEM_.SYS_USERS_ C,
        V$TABLESPACES D
WHERE   A.TABLE_OID = B.TABLE_OID
        AND A.USER_ID = C.USER_ID
        AND A.TBS_ID=D.ID
        AND C.USER_NAME <> 'SYSTEM_';
```

Disk Index

This query returns the object information about a disk index.

Column Name	Description
USER_NAME	the object owner
TABLE_NAME	the object name
TBS_NAME	the tablespace name which this object belongs to
INDEX_NAME	the name of index
INDEX_TYPE	the type of index
ALLOC(M)	allocated size of the memory table (Megabyte)

Altibase V4

```
SELECT  USER_NAME,
        TABLE_NAME,
        D.NAME TBS_NAME,
        E.INDEX_NAME,
        RPAD(CASE2(E.INDEX_TYPE=1,'B-TREE', 'R-TREE'),10,' ') INDEX_TYPE,
        ROUND(D.A_EXTENT_PAGE_COUNTD.PAGE_SIZEA.EXTENT_TOTAL_COUNT/1024/1024) 'ALLOC(M)'
FROM    V$SEGMENT A,
        V$INDEX B,
        V$TABLESPACES D,
        SYSTEM_.SYS_USERS_ C,
```

© Altibase Corp.

```
        SYSTEM_.SYS_INDICES_ E,
        SYSTEM_.SYS_TABLES_ F
WHERE   A.SEGMENT_DESC = B.INDEX_SEG_DESC
        AND B.INDEX_ID = E.INDEX_ID
        AND E.USER_ID = C.USER_ID
        AND A.SPACE_ID = D.ID
        AND F.TBS_ID = D.ID
        AND A.SEGMENT_TYPE='INDEX'
        AND F.USER_ID = E.USER_ID
        AND F.TABLE_OID = B.TABLE_OID ;
```

Altibase V5

```
SELECT  USER_NAME,
        TABLE_NAME,
        D.NAME TBS_NAME,
        E.INDEX_NAME,
        RPAD(CASE2(E.INDEX_TYPE=1,'B-TREE', 'R-TREE'),10,' ') INDEX_TYPE,
        ROUND(D.EXTENT_PAGE_COUNTD.PAGE_SIZEA.EXTENT_TOTAL_COUNT/1024/1024) 'ALLOC(M)'
FROM    V$SEGMENT A,
        V$INDEX B,
        V$TABLESPACES D,
        SYSTEM_.SYS_USERS_ C,
        SYSTEM_.SYS_INDICES_ E,
        SYSTEM_.SYS_TABLES_ F
WHERE   A.SEGMENT_PID = B.INDEX_SEG_PID
        AND B.INDEX_ID = E.INDEX_ID
        AND E.USER_ID = C.USER_ID
        AND A.SPACE_ID = D.ID
        AND F.TBS_ID = D.ID
        AND A.SEGMENT_TYPE='INDEX'
        AND F.USER_ID = E.USER_ID
        AND F.TABLE_OID = B.TABLE_OID ;
```

Sequence

This query returns the object information about a sequence.

Column Name	Description
USER_NAME	the object owner
SEQ_NAME	the object name
MIN	the minimum value of the sequence
CURRENT_SEQ	the current value of the sequence
MAX	the maximum value of the sequence
INCREMENT	increment property of the sequence
IS_CYCLE	can be cycled
CACHE	the cache size of the sequence

```
SELECT  USER_NAME,
        TABLE_NAME SEQ_NAME,
        MIN_SEQ MIN,
        CURRENT_SEQ,
        MAX_SEQ MAX,
```

© Altibase Corp.

```
        INCREMENT_SEQ INCREMENT,
        IS_CYCLE,
        CACHE_SIZE CACHE
FROM    V$SEQ A,
        SYSTEM_.SYS_USERS_ B,
        SYSTEM_.SYS_TABLES_ C
WHERE   A.SEQ_OID = C.TABLE_OID
        AND B.USER_ID = C.USER_ID
        AND B.USER_NAME <> 'SYSTEM_' ;
```

Synonym

This query returns the object information about a synonym.

Column Name	Description
SYNONYM_OWNER	the object owner
SYNONYM_NAME	the object name
OBJECT_OWNER	the minimum value of the sequence
OBJECT_NAME	the current value of the sequence
LAST_DDL_TIME	the last DDL occured time

Altibase V4

```
SELECT  NVL(USER_NAME, 'PUBLIC') SYNONYM_OWNER,
        SYNONYM_NAME,
        SCHEMA_NAME OBJECT_OWNER,
        OBJECT_NAME
FROM    SYSTEM_.SYS_SYNONYMS_ A LEFT OUTER JOIN
        SYSTEM_.SYS_USERS_ B ON A.USER_ID = B.USER_ID
WHERE
                A.SCHEMA_NAME != 'SYSTEM_' ;
```

Altibase V5

```
SELECT  NVL(USER_NAME, 'PUBLIC') SYNONYM_OWNER,
        SYNONYM_NAME,
        OBJECT_OWNER_NAME OBJECT_OWNER,
        OBJECT_NAME,
        TO_CHAR(A.LAST_DDL_TIME, 'YYYY-MM-DD HH:MI:SS') LAST_DDL_TIME
FROM    SYSTEM_.SYS_SYNONYMS_ A LEFT OUTER JOIN
        SYSTEM_.SYS_USERS_ B ON A.SYNONYM_OWNER_ID = B.USER_ID
WHERE   OBJECT_OWNER_NAME <> 'SYSTEM_' ;
```

PSM (Stored Procedures / Function / Typeset)

This query returns the object information about a PSM.

Column Name	Description
USER_NAME	the object owner
PSM_TYPE	the type of PSM

© Altibase Corp.

Column Name	Description
PSM_NAME	the name of PSM
STATUS	the status of PSM
LAST_DDL_TIME	the last DDL occured time

Altibase V4

```
SELECT  A.USER_NAME,
        DECODE(OBJECT_TYPE, 0, 'PROCEDURE', 1, 'FUNCTION', 3, 'TYPESET') PSM_TYPE,
        PROC_NAME PSM_NAME,
        DECODE(STATUS, 0, 'VALID', 'INVALID') STATUS
FROM    SYSTEM_.SYS_USERS_ A,
        SYSTEM_.SYS_PROCEDURES_ B
WHERE A.USER_ID = B.USER_ID AND A.USER_NAME <> 'SYSTEM_' ;
```

Altibase V5

```
SELECT  A.USER_NAME,
        DECODE(OBJECT_TYPE, 0, 'PROCEDURE', 1, 'FUNCTION', 3, 'TYPESET') PSM_TYPE,
        PROC_NAME PSM_NAME,
        DECODE(STATUS, 0, 'VALID', 'INVALID') STATUS,
        TO_CHAR(B.LAST_DDL_TIME, 'YYYY-MM-DD HH:MI:SS') LAST_DDL_TIME
FROM    SYSTEM_.SYS_USERS_ A,
        SYSTEM_.SYS_PROCEDURES_ B
WHERE A.USER_ID = B.USER_ID AND A.USER_NAME <> 'SYSTEM_' ;
```

View

This query returns the object information about a view.

Column Name	Description
USER_NAME	the object owner
VIEW_NAME	the type of PSM
STATUS	the status of PSM
LAST_DDL_TIME	the last DDL occured time

Altibase V4

```
SELECT  A.USER_NAME,
        B.TABLE_NAME VIEW_NAME,
        DECODE(STATUS, 0, 'VALID', 'INVALID') STATUS
FROM    SYSTEM_.SYS_USERS_ A,
        SYSTEM_.SYS_TABLES_ B,
        SYSTEM_.SYS_VIEWS_ C
WHERE   A.USER_ID = B.USER_ID
        AND B.TABLE_ID = C.VIEW_ID
        AND B.TABLE_TYPE = 'V' ;
```

© Altibase Corp.

Altibase V5

```
SELECT  A.USER_NAME,
        B.TABLE_NAME VIEW_NAME,
        DECODE(STATUS, 0, 'VALID', 'INVALID') STATUS,
        TO_CHAR(B.LAST_DDL_TIME, 'YYYY-MM-DD HH:MI:SS') LAST_DDL_TIME
FROM    SYSTEM_.SYS_USERS_ A,
        SYSTEM_.SYS_TABLES_ B,
        SYSTEM_.SYS_VIEWS_ C
WHERE   A.USER_ID = B.USER_ID
        AND B.TABLE_ID = C.VIEW_ID
        AND B.TABLE_TYPE = 'V' ;
```

System Privileges

This query returns the object information about system privileges.

Column Name	Description
GRANTEE	the grantee
GRANTOR	the granter
PRIV_NAME	the name of privilege

```
SELECT  A.USER_NAME GRANTEE,
        C.USER_NAME GRANTOR,
        REPLACE(D.PRIV_NAME, '_', ' ') PRIV_NAME
FROM    SYSTEM_.SYS_USERS_ A,
        SYSTEM_.SYS_GRANT_SYSTEM_ B,
        SYSTEM_.SYS_USERS_ C,
        SYSTEM_.SYS_PRIVILEGES_ D
WHERE   C.USER_NAME <> 'SYSTEM_'
        AND B.GRANTEE_ID = A.USER_ID
        AND B.GRANTOR_ID = C.USER_ID
        AND B.PRIV_ID = D.PRIV_ID ;
```

Object Privileges

This query returns the information about object privileges.

Column Name	Description
GRANTEE	the grantee
GRANTOR	the granter
OBJECT_OWNER	the object owner
OBJECT_NAME	the object name
OBJECT_TYPE	the object type
PRIV_NAME	the name of privilege
GRANTABLE	is grant-able

© Altibase Corp.

```
SELECT  A.USER_NAME GRANTEE,
        C.USER_NAME GRANTOR,
        F.USER_NAME OBJECT_OWNER,
        E.TABLE_NAME OBJECT_NAME,
        E.TABLE_TYPE OBJECT_TYPE,
        REPLACE(D.PRIV_NAME, '_', ' ') PRIV_NAME,
        DECODE(B.WITH_GRANT_OPTION, 0, 'NO', 'YES') GRANTABLE
FROM    SYSTEM_.SYS_USERS_ A,
        SYSTEM_.SYS_GRANT_OBJECT_ B,
        SYSTEM_.SYS_USERS_ C,
        SYSTEM_.SYS_PRIVILEGES_ D,
        SYSTEM_.SYS_TABLES_ E,
        SYSTEM_.SYS_USERS_ F
WHERE   C.USER_NAME <> 'SYSTEM_'
        AND B.GRANTEE_ID = A.USER_ID
        AND B.GRANTOR_ID = C.USER_ID
        AND B.PRIV_ID = D.PRIV_ID
        AND B.OBJ_ID = E.TABLE_ID
        AND E.USER_ID = F.USER_ID
ORDER BY GRANTEE, GRANTOR, OBJECT_OWNER, OBJECT_TYPE, OBJECT_NAME, PRIV_NAME ;
```

Constraint list

This query returns the information about object constraints.

Column Name	Description
USER_NAME	the object owner
OBJECT_NAME	the object name
OBJECT_TYPE	the object type
CONST_NAME	the constraint name
CONST_type	the constraint type
COLUMN_NAME	the column name

```
SELECT  USER_NAME,
        TABLE_NAME OBJECT_NAME,
        DECODE(B.TABLE_TYPE, 'T', 'TABLE', 'Q', 'QUEUE', 'V', 'VIEW', 'SEQUENCE') OBJECT_TYPE,
        C.CONSTRAINT_NAME CONST_NAME,
        DECODE(C.CONSTRAINT_TYPE,0, 'FK', 1, 'NOT NULL', 2, 'UNIQUE', 3, 'PK', 4, 'NULL', 5, 'TIMESTAMP', 6,
'LOCAL UNIQUE') CONST_TYPE,
        COLUMN_NAME
FROM    SYSTEM_.SYS_USERS_ A,
        SYSTEM_.SYS_TABLES_ B,
        SYSTEM_.SYS_CONSTRAINTS_ C,
        SYSTEM_.SYS_COLUMNS_ D,
        SYSTEM_.SYS_CONSTRAINT_COLUMNS_ E
WHERE   A.USER_ID=C.USER_ID
        AND B.TABLE_ID = C.TABLE_ID
        AND A.USER_ID = D.USER_ID
        AND A.USER_ID = E.USER_ID
        AND B.TABLE_ID = D.TABLE_ID
        AND B.TABLE_ID = E.TABLE_ID
        AND C.CONSTRAINT_ID = E.CONSTRAINT_ID
        AND D.COLUMN_ID = E.COLUMN_ID
        AND A.USER_NAME <> 'SYSTEM_' ;
```

© Altibase Corp.

Primary Key, Foreign Key, Unique Constraint

This query returns the information about object constraints.

Column Name	Description
USER_NAME	the object owner
TABLE_NAME	the table name
CONST_TYPE	the constraint type
CONST_NAME	the constraint name
INDEX_NAME	the index name
R_TABLE	the referenced table
R_INDEX	the referenced index

```
SELECT  A.USER_NAME,
        B.TABLE_NAME,
        DECODE(C.CONSTRAINT_TYPE,0, 'FK', 2, 'UNIQUE', 3, 'PK', 4, 'NULL') CONST_TYPE,
        C.CONSTRAINT_NAME CONST_NAME,
        DECODE(D.INDEX_NAME,C.CONSTRAINT_NAME,NULL,INDEX_NAME) INDEX_NAME,
        (SELECT TABLE_NAME FROM SYSTEM_.SYS_TABLES_ WHERE TABLE_ID = C.REFERENCED_TABLE_ID) R_TABLE,
        (SELECT INDEX_NAME FROM SYSTEM_.SYS_INDICES_ WHERE INDEX_ID = C.REFERENCED_INDEX_ID) R_INDEX
FROM     SYSTEM_.SYS_USERS_ A,
         SYSTEM_.SYS_TABLES_ B,
         SYSTEM_.SYS_CONSTRAINTS_ C LEFT OUTER JOIN
         SYSTEM_.SYS_INDICES_ D ON C.INDEX_ID = D.INDEX_ID
WHERE    C.TABLE_ID = B.TABLE_ID
         AND A.USER_NAME <> 'SYSTEM_'
         AND C.USER_ID = A.USER_ID
         AND C.CONSTRAINT_TYPE IN (3, 0, 2, 6) --PK, FK, UNIQUE, LOCAL UNIQUE
ORDER BY TABLE_NAME, CONST_TYPE ;
```

Index column list

This query returns the information about index columns.

Column Name	Description
USER_NAME	the object owner
TABLE_NAME	the table name
INDEX_NAME	the index name
COLUMN_NAME	the column name
COL_ORDER	column order (when composite index)

```
SELECT  D.USER_NAME,
        C.TABLE_NAME,
        B.INDEX_NAME,
        E.COLUMN_NAME,
        A.INDEX_COL_ORDER COL_ORDER,
        DECODE(A.SORT_ORDER, 'A', 'ASC', 'D', 'DESC') SORT
FROM    SYSTEM_.SYS_INDEX_COLUMNS_ A,
```

© Altibase Corp.

```
        SYSTEM_.SYS_INDICES_ B,
        SYSTEM_.SYS_TABLES_ C,
        SYSTEM_.SYS_USERS_ D,
        SYSTEM_.SYS_COLUMNS_ E
WHERE   D.USER_NAME <> 'SYSTEM_'
        AND C.TABLE_TYPE = 'T'
        AND A.INDEX_ID = B.INDEX_ID
        AND A.TABLE_ID = C.TABLE_ID
        AND A.USER_ID = D.USER_ID
        AND A.COLUMN_ID = E.COLUMN_ID
ORDER BY USER_NAME, TABLE_NAME, INDEX_NAME, COL_ORDER ;
```

Index information

This query returns the information about an index.

Column Name	Description
USER_NAME	the object owner
INDEX_NAME	the index name
INDEX_ID	the index name
TABLE_NAME	the table name
TBS_NAME	the tablespace which this object belongs to
IS_UNIQUE	unique index or not
COLUMN_CNT	the number of columns

```
SELECT  A.USER_NAME,
        C.INDEX_NAME,
        C.INDEX_ID,
        B.TABLE_NAME,
        NVL(D.NAME, 'SYS_TBS_MEMORY') TBS_NAME,
        C.IS_UNIQUE,
        C.COLUMN_CNT
FROM    SYSTEM_.SYS_USERS_ A,
        SYSTEM_.SYS_TABLES_ B,
        SYSTEM_.SYS_INDICES_ C LEFT OUTER JOIN
        V$TABLESPACES D ON C.TBS_ID = D.ID
WHERE   A.USER_NAME <> 'SYSTEM_'
        AND B.TABLE_TYPE = 'T'
        AND C.TABLE_ID = B.TABLE_ID
        AND C.USER_ID = A.USER_ID
ORDER BY B.TABLE_NAME, C.INDEX_NAME ;
```

SQL about Replication

Replication Sender

This query returns the replication sender status.

© Altibase Corp.

Column Name	Description
REP_NAME	the name of replication object
REMOTE_IP	the IP address of remote replicated server
REMOTE_REP_PORT	the port number of remote replicated server
STATUS	the status of replication sender
NETWORK	check whether network error or not
XSN	XLOG Sequence Number: the final position in a log file from which logs were transmitted by the replication sender

```
SELECT  REP_NAME,
        PEER_IP REMOTE_IP,
        PEER_PORT REMOTE_REP_PORT,
        DECODE(STATUS, 0, 'STOP', 1, 'RUN', 2, 'RETRY') AS STATUS,
        DECODE(NET_ERROR_FLAG, 0, 'OK', 'ERROR') AS NETWORK,
        XSN
FROM V$REPSENDER ;
```

Replication Receiver

Column Name	Description
REP_NAME	the name of replication object
REMOTE_IP	the IP address of remote replicated server
REMOTE_REP_PORT	the port number of remote replicated server
APPLY_XSN	applied XLOG Sequence Number: the final position in a log file from which logs were applied by the replication receiver

```
SELECT  REP_NAME,
          PEER_IP REMOTE_IP,
          PEER_PORT REMOTE_REP_PORT,
          APPLY_XSN
FROM V$REPRECEIVER ;
```

Replication gap

Column Name	Description
REP_NAME	the name of replication object
REMOTE_IP	the IP address of remote replicated server
REMOTE_REP_PORT	the port number of remote replicated server
APPLY_XSN	applied XLOG Sequence Number: the final position in a log file from which logs were applied by the replication receiver

```
SELECT  REP_NAME,
            REP_SN,
```

© Altibase Corp.

```
            REP_LAST_SN,
            REP_GAP,
            READ_FILE_NO,
            START_FLAG
FROM    V$REPGAP ;
```

Replication status

Column Name	Description
REP_NAME	the name of replication object
REMOTE_IP	the IP address of remote replicated server
REP_GAP	the interval between the log record that was most recently written due to a transaction on the local server and the log record that is currently being sent by the replication Sender thread.
RESTART_XSN	the XSN from which the Sender thread must begin sending logs when replication is started.
SENDER	the sender status
RECEIVER	the receiver status

```
SELECT  A.REPLICATION_NAME REP_NAME,
        D.HOST_IP REMOTE_IP,
        NVL(TO_CHAR(E.REP_GAP), '-') AS REP_GAP,
        A.XSN RESTART_XSN,
        DECODE(B.PEER_PORT, NULL, 'OFF', 'ON') AS SENDER,
        DECODE(C.PEER_PORT, NULL, 'OFF', 'ON') AS RECEIVER
FROM    SYSTEM_.SYS_REPL_HOSTS_ D ,
        SYSTEM_.SYS_REPLICATIONS_ A LEFT OUTER JOIN
        V$REPSENDER B ON A.REPLICATION_NAME = B.REP_NAME
        LEFT OUTER JOIN V$REPRECEIVER C ON A.REPLICATION_NAME = C.REP_NAME LEFT OUTER JOIN
        (
            SELECT REP_NAME,
                   MAX(REP_GAP) REP_GAP
            FROM   V$REPGAP
            GROUP BY REP_NAME
         ) E ON A.REPLICATION_NAME = E.REP_NAME
WHERE   A.REPLICATION_NAME = D.REPLICATION_NAME
ORDER BY REP_NAME ;
```

Log buffer or file status occupied by an unsent XLOG

This query shows that the logbuffer/logfile status which is occupied by an unsent XLOG.

Altibase V4

Icon

ALTIBASE V4 does not support this feature.

Altibase V5

```
SELECT  CASE2((BUFFER_MIN_SN < READ_SN), 'REP BUFFER '¦¦ROUND((BUFFER_MAX_SN-READ_SN)/(BUFFER_MAX_SN-
BUFFER_MIN_SN)*100,2)¦¦' % LEFT ', (SELECT TO_CHAR(CUR_WRITE_LF_NO - READ_FILE_NO) FROM    V$LFG, V$REPGAP))
```

© Altibase Corp.

```
LOGFILE_FOR_REP
FROM      V$REPLOGBUFFER ;
```

Replication table list

The list of tables that participate in a replication.

Column Name	Description
REP_NAME	the name of replication object
LOCAL_TABLE	the local table name
REMOTE_TABLE	the remote table name

```
SELECT  REPLICATION_NAME REP_NAME,
        LOCAL_USER_NAME||'.'||LOCAL_TABLE_NAME LOCAL_TABLE,
        REMOTE_USER_NAME||'.'||REMOTE_TABLE_NAME REMOTE_TABLE
FROM    SYSTEM_.SYS_REPL_ITEMS_
ORDER BY 1, 2 ;
```

SQL about Sessions

Total number of sessions

This query returns the number of sessions which are currently established.

```
SELECT  COUNT(*) TOTAL_SESSION_CNT
FROM    V$SESSION ;
```

Session Information

This query returns the information about a session.

Column Name	Description	
SESSSION_ID	the unique identifier of the session	
USER_NAME	the database user name	
CLIENT_IP	the client IP address	
CLIENT_APP_INFO	the type of client application	not supported in Altibase 4
CLIENT_PID	the client process identifier	
SESSION_STATE	the status of the session	
AUTOCOMMIT	the autocommit mode	
LOGIN_TIME	the time when the session is established	
IDLE_TIME	the time when the session has been idle	

© Altibase Corp.

Column Name	Description	
CURRENT_QUERY	the query that the session has last executed	not supported in Altibase 4

Altibase 4

```
SELECT  A.ID SESSION_ID,
        A.DB_USERNAME USER_NAME,
        REPLACE2(REPLACE2(A.COMM_NAME, 'SOCKET-', NULL), '-SERVER', NULL) CLIENT_IP,
        A.CLIENT_PID,
        A.SESSION_STATE,
        DECODE(A.AUTOCOMMIT_FLAG, 1, 'ON', 'OFF') AUTOCOMMIT,
        DECODE(A.LOGIN_TIME, 0, '-', TO_CHAR(TO_DATE('1970010109', 'YYYYMMDDHH') + A.LOGIN_TIME / (246060),
'MM/DD HH:MI:SS')) LOGIN_TIME,
        DECODE(A.IDLE_START_TIME, 0, '-', TO_CHAR(TO_DATE('1970010109', 'YYYYMMDDHH') + A.IDLE_START_TIME /
(246060), 'MM/DD HH:MI:SS')) IDLE_TIME
FROM    V$SESSION A ;
```

Altibase 5

```
SELECT  A.ID SESSION_ID,
        A.DB_USERNAME USER_NAME,
        REPLACE2(REPLACE2(A.COMM_NAME, 'SOCKET-', NULL), '-SERVER', NULL) CLIENT_IP,
        A.CLIENT_APP_INFO,
        A.CLIENT_PID,
        A.SESSION_STATE,
        DECODE(A.AUTOCOMMIT_FLAG, 1, 'ON', 'OFF') AUTOCOMMIT,
        DECODE(A.LOGIN_TIME, 0, '-', TO_CHAR(TO_DATE('1970010109', 'YYYYMMDDHH') + A.LOGIN_TIME / (246060),
'MM/DD HH:MI:SS')) LOGIN_TIME,
        DECODE(A.IDLE_START_TIME, 0, '-', TO_CHAR(TO_DATE('1970010109', 'YYYYMMDDHH') + A.IDLE_START_TIME /
(246060), 'MM/DD HH:MI:SS')) IDLE_TIME,
        NVL(LTRIM(B.QUERY), 'NONE') CURRENT_QUERY
FROM    V$SESSION A LEFT OUTER JOIN
        V$STATEMENT B ON A.CURRENT_STMT_ID = B.ID ;
```

Session information connected as SYSDBA

This query returns the information about a session which connects as SYSDBA.

Column Name	Description	
SESSSION_ID	the unique identifier of the session	
USER_NAME	the database user name	
CLIENT_IP	the client IP address	
CLIENT_APP_INFO	the type of client application	not supported in Altibase 4
CLIENT_PID	the client process identifier	
SESSION_STATE	the status of the session	
AUTOCOMMIT	the autocommit mode	
LOGIN_TIME	the time when the session is established	
IDLE_TIME	the time when the session has been idle	

© Altibase Corp.

Column Name	Description	
CURRENT_QUERY	the query that the session last executed	not supported in Altibase 4

Altibase 4

```
SELECT  A.ID SESSION_ID,
        A.DB_USERNAME USER_NAME,
        REPLACE2(REPLACE2(A.COMM_NAME, 'SOCKET-', NULL), '-SERVER', NULL) CLIENT_IP,
        A.CLIENT_PID,
        A.SESSION_STATE,
        DECODE(A.AUTOCOMMIT_FLAG, 1, 'ON', 'OFF') AUTOCOMMIT,
        DECODE(A.LOGIN_TIME, 0, '-', TO_CHAR(TO_DATE('1970010109', 'YYYYMMDDHH') + A.LOGIN_TIME / (246060),
'MM/DD HH:MI:SS')) LOGIN_TIME,
        DECODE(A.IDLE_START_TIME, 0, '-', TO_CHAR(TO_DATE('1970010109', 'YYYYMMDDHH') + A.IDLE_START_TIME /
(246060), 'MM/DD HH:MI:SS')) IDLE_TIME
FROM    V$SESSION A
WHERE   A.SYSDBA_FLAG = 1;
```

Altibase 5

```
SELECT  A.ID SESSION_ID,
        A.DB_USERNAME USER_NAME,
        REPLACE2(REPLACE2(A.COMM_NAME, 'SOCKET-', NULL), '-SERVER', NULL) CLIENT_IP,
        A.CLIENT_APP_INFO,
        A.CLIENT_PID,
        A.SESSION_STATE,
        DECODE(A.AUTOCOMMIT_FLAG, 1, 'ON', 'OFF') AUTOCOMMIT,
        DECODE(A.LOGIN_TIME, 0, '-', TO_CHAR(TO_DATE('1970010109', 'YYYYMMDDHH') + A.LOGIN_TIME / (246060),
'MM/DD HH:MI:SS')) LOGIN_TIME,
        DECODE(A.IDLE_START_TIME, 0, '-', TO_CHAR(TO_DATE('1970010109', 'YYYYMMDDHH') + A.IDLE_START_TIME /
(246060), 'MM/DD HH:MI:SS')) IDLE_TIME,
        NVL(LTRIM(B.QUERY), 'NONE') CURRENT_QUERY
FROM    V$SESSION A LEFT OUTER JOIN
        V$STATEMENT B ON A.CURRENT_STMT_ID = B.ID
WHERE   A.SYSDBA_FLAG = 1;
```

SQL about Statements

Total number of statements

This query returns the number of statements that currently are established.

```
SELECT  COUNT(*) STATEMENT_CNT
FROM    V$STATEMENT;
```

Statement information

This query returns the information about statements, sorted in descending order by the execution time.

Column Name	Description
SESSSION_ID	the unique identifier of the session

© Altibase Corp.

Column Name	Description
STMT_ID	the unique identifier of the statement in the session
TX_ID	the transaction identifier if the statement participates in a transaction
PREPARE_TIME	the time taken to perform preparing the statement. (microsecond)
FETCH_TIME	the time taken to perform a fetch operation (microsecond)
EXECUTE_TIME	the time taken to execute the statement (microsecond)
TOTAL_TIME	the total elapsed time
EXECUTE_FLAG	set 1 if the statement is currently executed
LAST_START_TIME	the time when the statement last started
QUERY	the query string

```
SELECT  SESSION_ID,
        ID STMT_ID,
        TX_ID,
        (PARSE_TIME+VALIDATE_TIME+OPTIMIZE_TIME) PREPARE_TIME,
        FETCH_TIME,
        EXECUTE_TIME,
        TOTAL_TIME,
        EXECUTE_FLAG,
        DECODE(LAST_QUERY_START_TIME, 0, '-', TO_CHAR(TO_DATE('1970010109', 'YYYYMMDDHH') +
LAST_QUERY_START_TIME / (246060), 'MM/DD HH:MI:SS')) LAST_START_TIME,
        NVL(LTRIM(QUERY), 'NONE') QUERY
FROM    V$STATEMENT
ORDER BY EXECUTE_TIME DESC ;
```

The number of currently running statements

This query returns the number of statements that are currently executing.

```
SELECT  COUNT(*) AS ACTIVE_STATEMENT_CNT
FROM    V$STATEMENT
WHERE   EXECUTE_FLAG = 1 ;
```

Information about currently running statements

This query returns the statement information about those that are currently executing.

Column Name	Description
SESSSION_ID	the unique identifier of the session
STMT_ID	the unique identifier of statement in the session
TX_ID	the transaction identifier if the statement participate in a transaction
PREPARE_TIME	the time taken to perform preparing the statement. (microsecond)
FETCH_TIME	the time taken to perform a fetch operation (microsecond)
EXECUTE_TIME	the time taken to execute the statement (microsecond)

© Altibase Corp.

Column Name	Description
TOTAL_TIME	the total elapsed time
EXECUTE_FLAG	set 1 if the statement is currently executed
LAST_START_TIME	the time when the statement last started
QUERY	the query string

```
SELECT  SESSION_ID,
        ID STMT_ID,
        TX_ID,
        (PARSE_TIME+VALIDATE_TIME+OPTIMIZE_TIME) PREPARE_TIME,
        FETCH_TIME,
        EXECUTE_TIME,
        TOTAL_TIME,
        DECODE(LAST_QUERY_START_TIME, 0, '-', TO_CHAR(TO_DATE('1970010109', 'YYYYMMDDHH') +
LAST_QUERY_START_TIME / (246060), 'MM/DD HH:MI:SS')) LAST_START_TIME,
        NVL(LTRIM(QUERY), 'NONE') QUERY
FROM    V$STATEMENT
WHERE   EXECUTE_FLAG = 1
ORDER BY EXECUTE_TIME DESC ;
```

Long running query (over 10 seconds)

This query returns the statement information about queries that are considered long running. (the queries with elapsed time over 10 seconds)

Column Name	Description
SESSSION_ID	the unique identifier of the session
STMT_ID	the unique identifier of statement in the session
TX_ID	the transaction identifier if the statement participate in a transaction
PREPARE_TIME	the time taken to perform preparing statement. (microsecond)
FETCH_TIME	the time taken to perform a fetch operation (microsecond)
EXECUTE_TIME	the time taken to execute statement (microsecond)
TOTAL_TIME	the total elapsed time
EXECUTE_FLAG	set 1 if the statement is currently executed
LAST_START_TIME	the time when the statement last started
QUERY	the query string

```
SELECT  SESSION_ID,
        ID STMT_ID,
        TX_ID,
        (PARSE_TIME+VALIDATE_TIME+OPTIMIZE_TIME) PREPARE_TIME,
        FETCH_TIME,
        EXECUTE_TIME,
        TOTAL_TIME,
        DECODE(LAST_QUERY_START_TIME, 0, '-', TO_CHAR(TO_DATE('1970010109', 'YYYYMMDDHH') +
LAST_QUERY_START_TIME / (246060), 'MM/DD HH:MI:SS')) LAST_START_TIME,
```

© Altibase Corp.

```
        NVL(LTRIM(QUERY), 'NONE') QUERY
FROM    V$STATEMENT
WHERE   EXECUTE_FLAG = 1 AND EXECUTE_TIME/1000000 > 10
ORDER BY EXECUTE_TIME DESC ;
```

Long running transaction's last statement information (over 1 minute)

In the case of long running transactions in Altibase, this query shows the information for the last query of a transaction with an elapsed time over 1 minute.

Column Name	Description
SESSSION_ID	the unique identifier of the session
CLIENT_IP	the client IP address
CLIENT_PID	the client process identifier
UTRANS_TIME	the time taken during the transaction. (microsecond)
EXECUTE_TIME	the time taken to execute statement (microsecond)
TOTAL_TIME	the total elapsed time
LAST_START_TIME	the time when the statement last started
QUERY	the query string

```
SELECT  ST.SESSION_ID,
        SS.COMM_NAME CLIENT_IP,
        SS.CLIENT_PID,
        (BASE_TIME - TR.FIRST_UPDATE_TIME) AS UTRANS_TIME,
        EXECUTE_TIME,
        TOTAL_TIME,
        DECODE(LAST_QUERY_START_TIME, 0, '-', TO_CHAR(TO_DATE('1970010109', 'YYYYMMDDHH') +
LAST_QUERY_START_TIME / (246060), 'MM/DD HH:MI:SS')) LAST_START_TIME,
        NVL(LTRIM(ST.QUERY), 'NONE') QUERY
FROM    V$TRANSACTION TR,
        V$STATEMENT ST,
        V$SESSIONMGR,
        V$SESSION SS
WHERE   TR.ID = ST.TX_ID
        AND ST.SESSION_ID = SS.ID
        AND TR.FIRST_UPDATE_TIME != 0 -- 0:READ ONLY TRANSACTION
        AND (BASE_TIME - TR.FIRST_UPDATE_TIME) > 60
ORDER BY UTRANS_TIME DESC ;
```

Information about a query running a FULL SCAN

This query shows the information about a query that is running a FULL SCAN. You need to be careful and check the SQL if there is a FULL SCAN plan node in the statement.

Column Name	Description
SESSSION_ID	the unique identifier of the session
CLIENT_IP	the client IP address

© Altibase Corp.

Column Name	Description
CLIENT_PID	the client process identifier
LAST_START_TIME	the time when the statement last started
PREPARE_TIME	the time taken to perform preparing statement. (microsecond)
FETCH_TIME	the time taken to perform a fetch operation (microsecond)
EXECUTE_TIME	the time taken to execute statement (microsecond)
TOTAL_TIME	the total elapsed time
QUERY	the query string

```
SELECT  SESSION_ID,
        S.COMM_NAME CLIENT_IP,
        S.CLIENT_PID,
        DECODE(LAST_QUERY_START_TIME, 0, '-', TO_CHAR(TO_DATE('1970010109', 'YYYYMMDDHH') +
LAST_QUERY_START_TIME / (246060), 'MM/DD HH:MI:SS')) LAST_START_TIME,
        (PARSE_TIME+VALIDATE_TIME+OPTIMIZE_TIME) PREPARE_TIME,
        FETCH_TIME,
        EXECUTE_TIME,
        TOTAL_TIME,
        NVL(LTRIM(QUERY), 'NONE') QUERY
FROM    V$STATEMENT T,
        V$SESSION S
WHERE   S.ID = T.SESSION_ID
        AND (MEM_CURSOR_FULL_SCAN > 0 OR DISK_CURSOR_FULL_SCAN > 0)
        AND UPPER(QUERY) NOT LIKE '%INSERT%'
ORDER BY EXECUTE_TIME DESC ;
```

SQL about Tablespaces

The usage of a Memory tablespace

This query returns the information about a memory tablespace.

Columns	Description
TBS_ID	the unique identifier of the tablespace
TBS_TYPE	the type of the tablespace
TBS_NAME	the name of the tablespace
MAX(M)	the max size of the tablespace (Megabyte)
TOTAL(M)	the total size of the tablespace (Megabyte)
ALLOC(M)	the allocated size of the tablespace (Megabyte)
USED(M)	actually used size of the tablespace (Megabyte)
USAGE(%)	the tablespace usages. (USED(M) / MAX(M)) * 100. if MAX(M) is unlimited, it assumes that MAX(M) is MEM_MAX_DB_SIZE in altibase.properties.

© Altibase Corp.

Columns	Description
STATE	the state of the tablespace
AUTOEXTEND	whether the tablespace can be extended automatically

Altibase V4

Icon

Altibase V4 does not support User Memory Tablespaces. There's only one memory tablespace including system dictionary tablespace.

Altibase V5

```
SELECT  ID TBS_ID,
        DECODE(TYPE, 0, 'MEMORY_DICTIONARY', 1, 'MEMORY_SYS_DATA', 2, 'MEMORY_USER_DATA', 8,
'VOLATILE_USER_DATA') TBS_TYPE,
        NAME TBS_NAME,
        DECODE(MAXSIZE, 140737488322560, 'UNDEFINED', MAXSIZE) 'MAX(M)',
        ROUND(ALLOCATED_PAGE_COUNT * PAGE_SIZE / 1024 / 1024, 2) 'TOTAL(M)',
        ROUND(NVL(M.ALLOC_PAGE_COUNT-M.FREE_PAGE_COUNT,TOTAL_PAGE_COUNT)*PAGE_SIZE/1024/1024,2) 'ALLOC(M)',
        MT.USED 'USED(M)',
        DECODE(MAXSIZE, 140737488322560, ROUND((M.ALLOC_PAGE_COUNT-M.FREE_PAGE_COUNT)PAGE_SIZE/
MEM_MAX_DB_SIZE100,2),
        ROUND((M.ALLOC_PAGE_COUNT-M.FREE_PAGE_COUNT)PAGE_SIZE/MAXSIZE100,2)) 'USAGE(%)',
        DECODE(STATE,1,'OFFLINE',2,'ONLINE',5,'OFFLINE BACKUP',6,'ONLINE BACKUP',128,'DROPPED', 'DISCARDED')
STATE,
        DECODE(AUTOEXTEND_MODE,1,'ON','OFF') 'AUTOEXTEND'
FROM    V$DATABASE D,
        V$TABLESPACES T,
        V$MEM_TABLESPACES M,
        (
          SELECT TABLESPACE_ID,
                 ROUND(SUM((FIXED_USED_MEM + VAR_USED_MEM))/(1024*1024),3) USED
          FROM   V$MEMTBL_INFO
          GROUP BY TABLESPACE_ID
        ) MT
WHERE T.ID = M.SPACE_ID AND ID = MT.TABLESPACE_ID ;
```

Total memory tablespace usage

This query returns the total memory tablespace usage.

Columns	Description
MAX(M)	the max size of the tablespace (Megabyte)
TOTAL(M)	the total size of the tablespace (Megabyte)
ALLOC(M)	allocated size of the tablespace (Megabyte)
USED(M)	actually used size of the tablespace (Megabyte)
USAGE(%)	the tablespace usages. (USED(M) / MAX(M)) * 100. if MAX(M) is unlimited, it assumes that MAX(M) is MEM_MAX_DB_SIZE in altibase.properties.

```
SELECT  MEM_MAX_DB_SIZE/1024/1024 'MAX(M)',
        ROUND(MEM_ALLOC_PAGE_COUNT*32/1024, 2) 'TOTAL(M)',
```

© Altibase Corp.

```
        TRUNC((MEM_ALLOC_PAGE_COUNT-MEM_FREE_PAGE_COUNT)*32/1024, 2) 'ALLOC(M)',
        (SELECT ROUND(SUM((FIXED_USED_MEM + VAR_USED_MEM))/(1024*1024),3) FROM V$MEMTBL_INFO) 'USED(M)',
        TRUNC(((MEM_ALLOC_PAGE_COUNT-MEM_FREE_PAGE_COUNT)321024)/MEM_MAX_DB_SIZE, 4)*100 'USAGE(%)'
FROM V$DATABASE ;
```

Disk Tablespace Usage

This query returns the disk tablespace usage information.

Columns	Description
TBS_ID	the unique identifier of the tablespace
TBS_TYPE	the type of the tablespace
TBS_NAME	the name of the tablespace
MAX(M)	the max size of the tablespace (Megabyte)
TOTAL(M)	the total size of the tablespace (Megabyte)
ALLOC(M)	allocated size of the tablespace (Megabyte)
AUTOEXTEND	whether the tablespace can be extended automatically

Altibase V4

```
SELECT  ID TBS_ID,
        DECODE(TYPE, 1, 'DISK_SYS_DATA', 2, 'DISK_USER_DATA', 3, 'DISK_SYS_TEMP', 4, 'DISK_USER_TEMP', 5,
'DISK_UNDO') TBS_TYPE,
        NAME TBS_NAME,
        ROUND(D.MAX * PAGE_SIZE / 1024 /1024, 2) 'MAX(M)',
        ROUND(TOTAL_PAGE_COUNT * PAGE_SIZE / 1024 / 1024, 2) 'TOTAL(M)',
        ROUND(ALLOCATED_PAGE_COUNT * PAGE_SIZE / 1024 / 1024,2) 'ALLOC(M)',
        ROUND(ALLOCATED_PAGE_COUNT / D.MAX * 100, 2) 'USAGE(%)',
        D.AUTOEXTEND
FROM    V$TABLESPACES T,
        (
         SELECT     SPACEID,
                 SUM(DECODE(MAXSIZE, 0, CURRSIZE, MAXSIZE)) AS MAX,
                 DECODE(MAX(AUTOEXTEND),1,'ON','OFF') 'AUTOEXTEND'
         FROM V$DATAFILES
         GROUP BY SPACEID
         ) D
WHERE T.ID = D.SPACEID ;
```

Altibase V5

```
SELECT  ID TBS_ID,
        DECODE(TYPE, 3, 'DISK_SYS_DATA', 4, 'DISK_USER_DATA', 5, 'DISK_SYS_TEMP', 6, 'DISK_USER_TEMP', 7,
'DISK_UNDO') TBS_TYPE,
        NAME TBS_NAME,
        ROUND(D.MAX * PAGE_SIZE / 1024 /1024, 2) 'MAX(M)',
        ROUND(TOTAL_PAGE_COUNT * PAGE_SIZE / 1024 / 1024, 2) 'TOTAL(M)',
        DECODE(TYPE, 7, ROUND((SELECT (SUM(TOTAL_EXTENT_COUNTPAGE_COUNT_IN_EXTENT)  PAGE_SIZE)/1024/1024 FROM
V$UDSEGS)+(SELECT (SUM(TOTAL_EXTENT_COUNTPAGE_COUNT_IN_EXTENT)  PAGE_SIZE)/1024/1024 FROM V$TSSEGS),2),
ROUND(ALLOCATED_PAGE_COUNT * PAGE_SIZE / 1024 / 1024,2)) 'ALLOC(M)',
        DECODE(TYPE, 7, ROUND(((SELECT SUM(TOTAL_EXTENT_COUNTPAGE_COUNT_IN_EXTENT) FROM V$UDSEGS)+(SELECT
SUM(TOTAL_EXTENT_COUNTPAGE_COUNT_IN_EXTENT) FROM V$TSSEGS)) / D.MAX 100, 2), ROUND(ALLOCATED_PAGE_COUNT /
D.MAX  100, 2)) 'USAGE(%)', DECODE(STATE,1,'OFFLINE',2,'ONLINE',5,'OFFLINE BACKUP',6,'ONLINE BACKUP',
128,'DROPPED', 'DISCARDED') STATE,
```

© Altibase Corp.

```
        D.AUTOEXTEND
FROM    V$TABLESPACES T,
        (
         SELECT     SPACEID,
                 SUM(DECODE(MAXSIZE, 0, CURRSIZE, MAXSIZE)) AS MAX,
                 DECODE(MAX(AUTOEXTEND),1,'ON','OFF') 'AUTOEXTEND'
         FROM V$DATAFILES
         GROUP BY SPACEID
         ) D
WHERE T.ID = D.SPACEID ;
```

The datafile information

This query returns the information about data files containing tablespaces.

Columns	Description
TBS_NAME	the name of the tablespace
FILE#	the file number of datafiles containing the tablespace. It starts with 0.
DATAFILE_NAME	the name of the datafile
ALLOC(M)	allocated size of the datafile (Megabyte)
MAX(M)	the max size of the datafile (Megabyte)
AUTOEXTEND	can be extended automatically

```
SELECT  B.NAME TBS_NAME,
        A.ID 'FILE#',
        A.NAME DATAFILE_NAME,
        CURRSIZE*8/1024 'ALLOC(M)',
        ROUND(CASE2(A.MAXSIZE=0, CURRSIZE, A.MAXSIZE)*8/1024) 'MAX(M)',
        DECODE(AUTOEXTEND, 0, 'OFF', 'ON') 'AUTOEXTEND'
FROM    V$DATAFILES A,
        V$TABLESPACES B
WHERE   B.ID = A.SPACEID
ORDER BY B.NAME, A.ID ;
```

I/O statistics of datafiles

This query returns I/O statistics for datafiles.

Altibase V4

Icon

Altibase V4 does not support V$FILESTAT performance view to retrieve the I/O statistics of datafiles.

Altibase V5

```
SELECT  NAME TBS_NAME,
        A.PHYRDS PHY_READ,
        A.PHYWRTS PHY_WRITE,
        A.PHYRDS+A.PHYWRTS PHY_TOTAL,
        TRUNC(A.PHYRDS/READ_SUM*100,2) 'READ(%)',
        TRUNC(A.PHYWRTS/WRITE_SUM*100,2) 'WRITE(%)',
        TRUNC( (A.PHYRDS+A.PHYWRTS) / (READ_SUM+WRITE_SUM) * 100 , 2) 'TOTAL(%)',
```

© Altibase Corp.

```
        A.AVGIOTIM AVG_IO_TIME
FROM      V$FILESTAT A,
          V$DATAFILES B,
          (
                SELECT  SUM(PHYRDS) READ_SUM,
                                SUM(PHYWRTS) WRITE_SUM
                FROM V$FILESTAT
          ) C
WHERE   A.SPACEID = B.SPACEID
          AND A.FILEID = B.ID
          AND READ_SUM > 0
          AND WRITE_SUM > 0
ORDER BY A.PHYRDS+A.PHYWRTS DESC, ROWNUM DESC ;
```

© Altibase Corp.