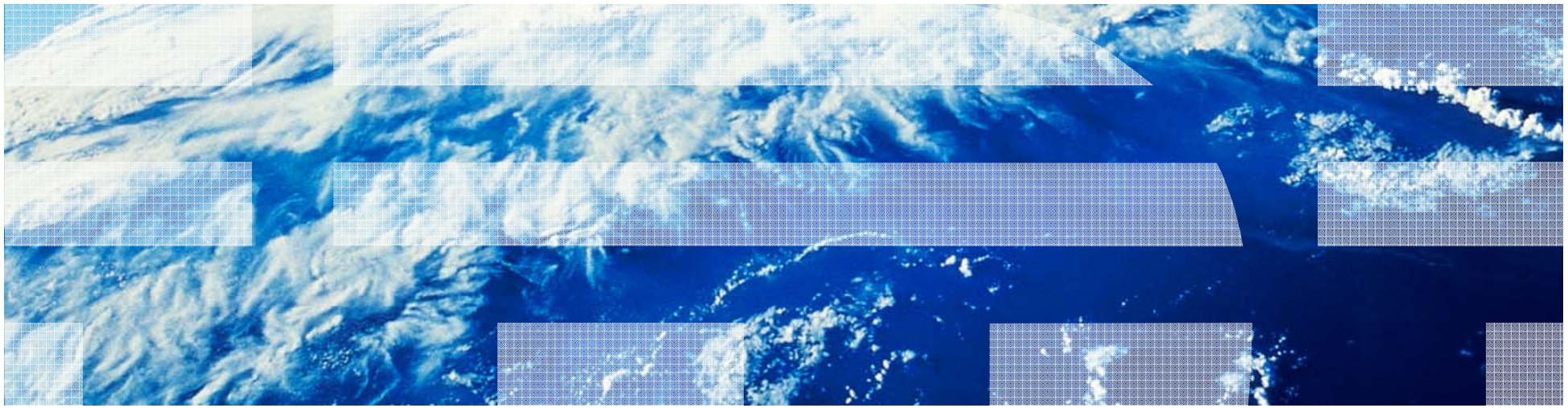
IBM

Information Management

Moving data



Unit objectives

After completing this unit, you should be able to:

- Discuss using the INSERT SQL statement to populate tables
- Explain the differences between IMPORT and LOAD processing
- Explain the EXPORT, IMPORT, and LOAD command options
- Create and use Exception Tables and Dump-Files
- Check table status using LOAD QUERY
- Describe Load Pending and Set Integrity Pending status for a table
- Use the SET INTEGRITY command
- Discuss the db2move and db2look commands
- Use the ADMIN_MOVE_TABLE procedure to move a table to different table spaces
- List some of the features of the Ingest utility for continuous data ingest

Review INSERT statement

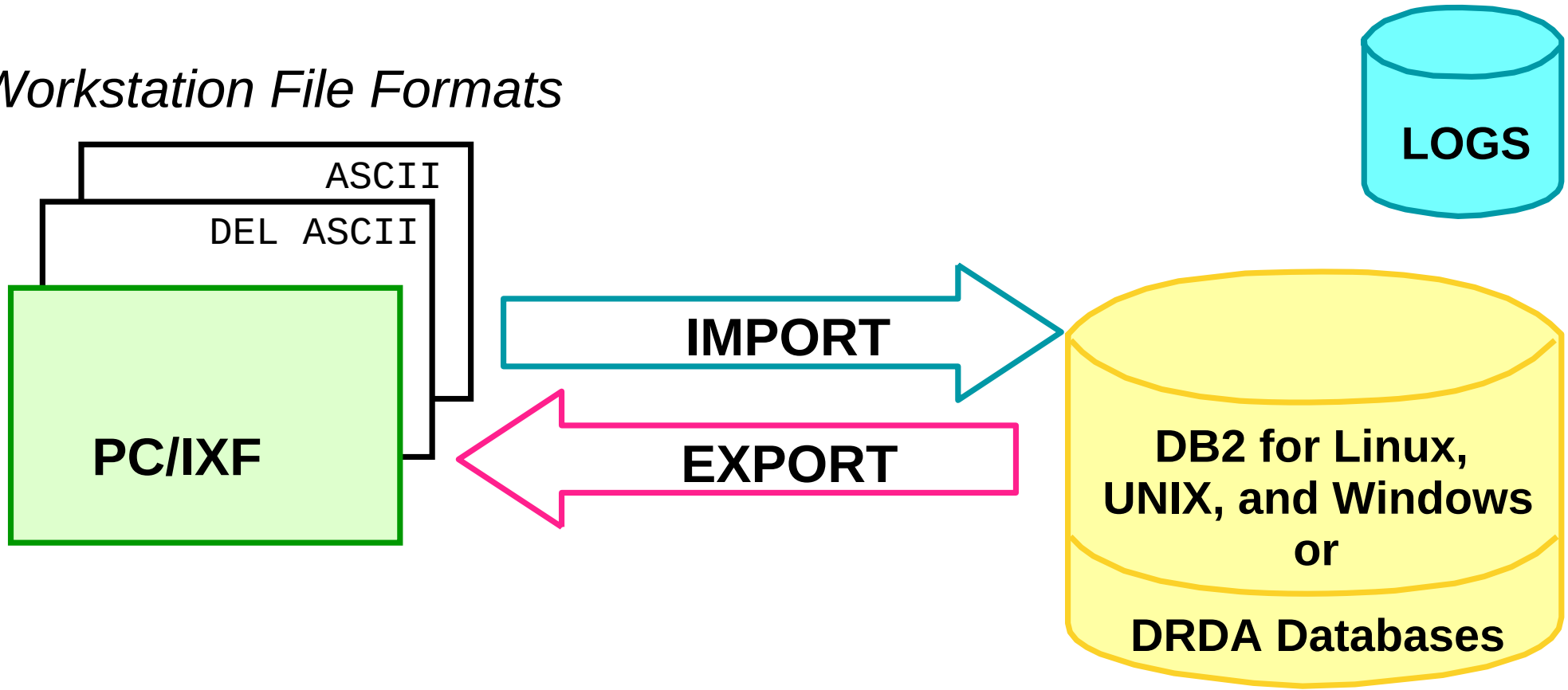
- The SQL INSERT statement can insert one or more rows of data into tables, nicknames, or views:
  - SQL overhead is imposed, such as obeying RI, or Check Constraints, or Uniqueness, or executing triggers.
  - As INSERTs occur, the activity is also stored in logs
- The SQL INSERT might not be the best or fastest method to load massive amounts of data into a database

Example inserts:

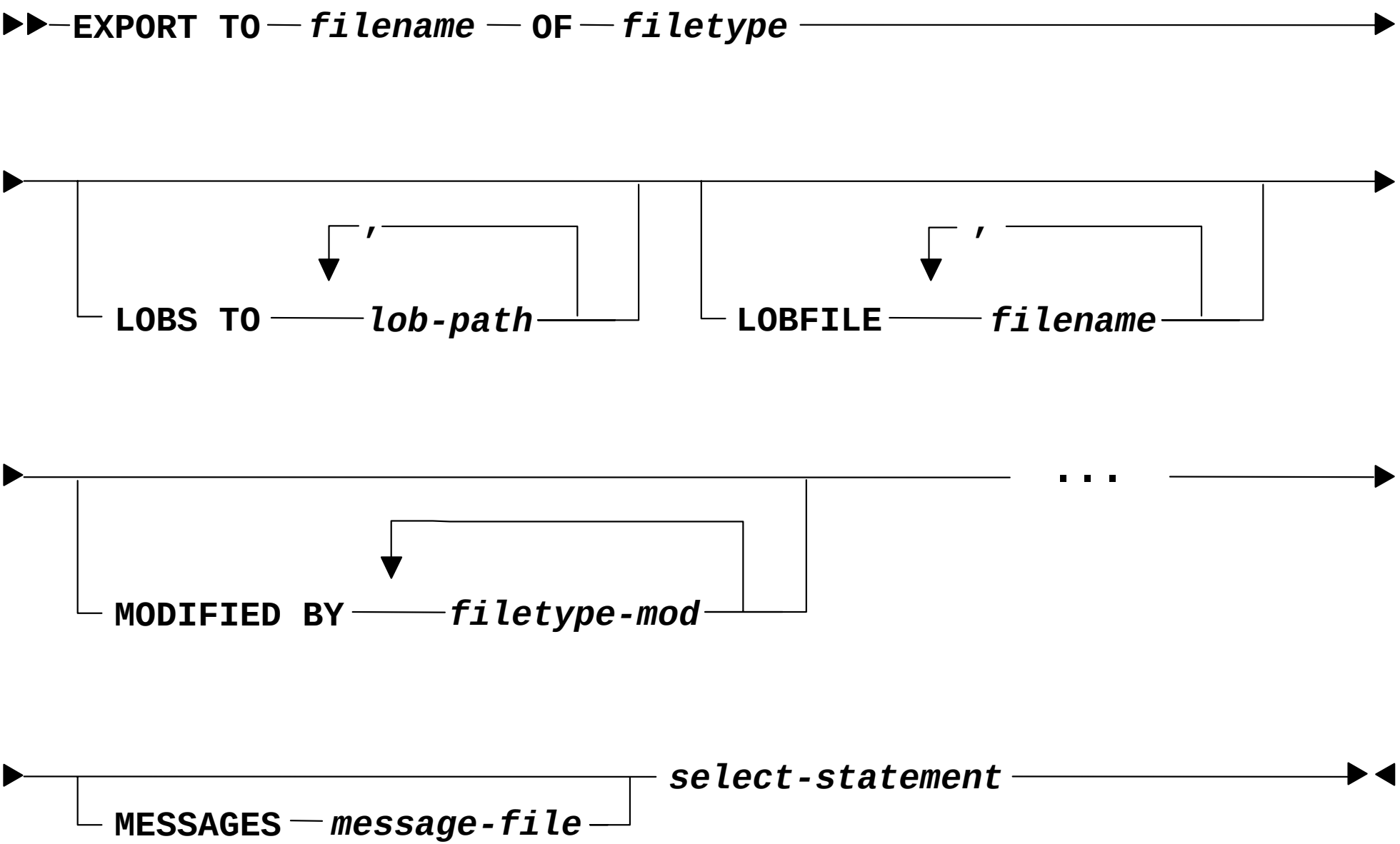
```
insert into artists (artno, name,classification)
values (100, 'Patti & Cart Wheels', 'S') ;


Insert into emptemp select * from employee ;
```

EXPORT/IMPORT overview

- Must be connected prior to call

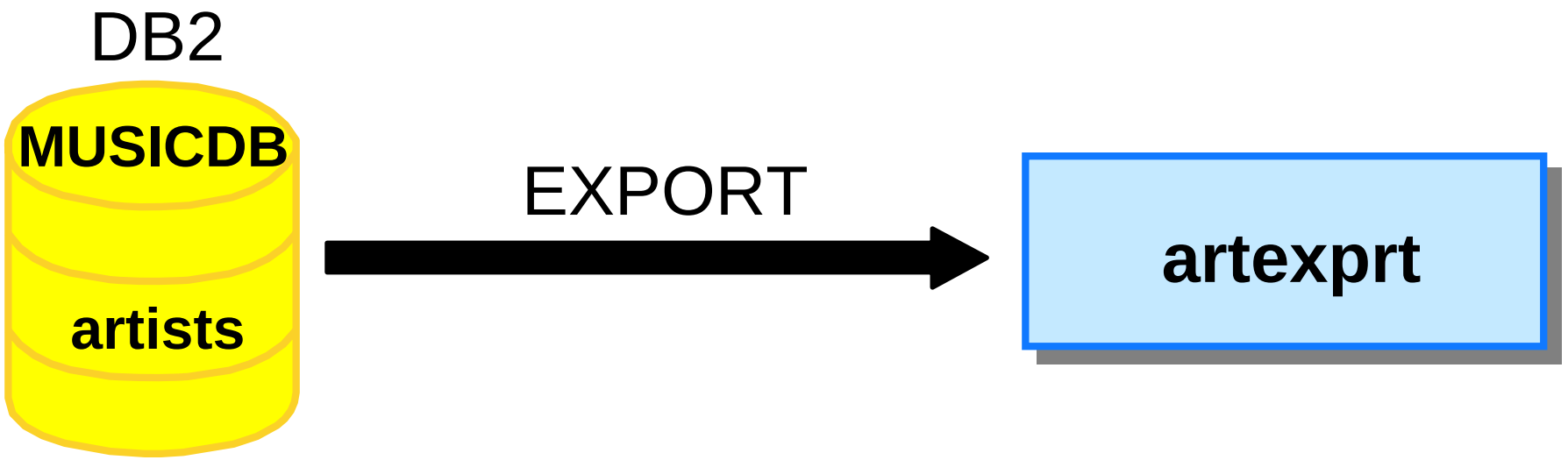


EXPORT command syntax (Basic)



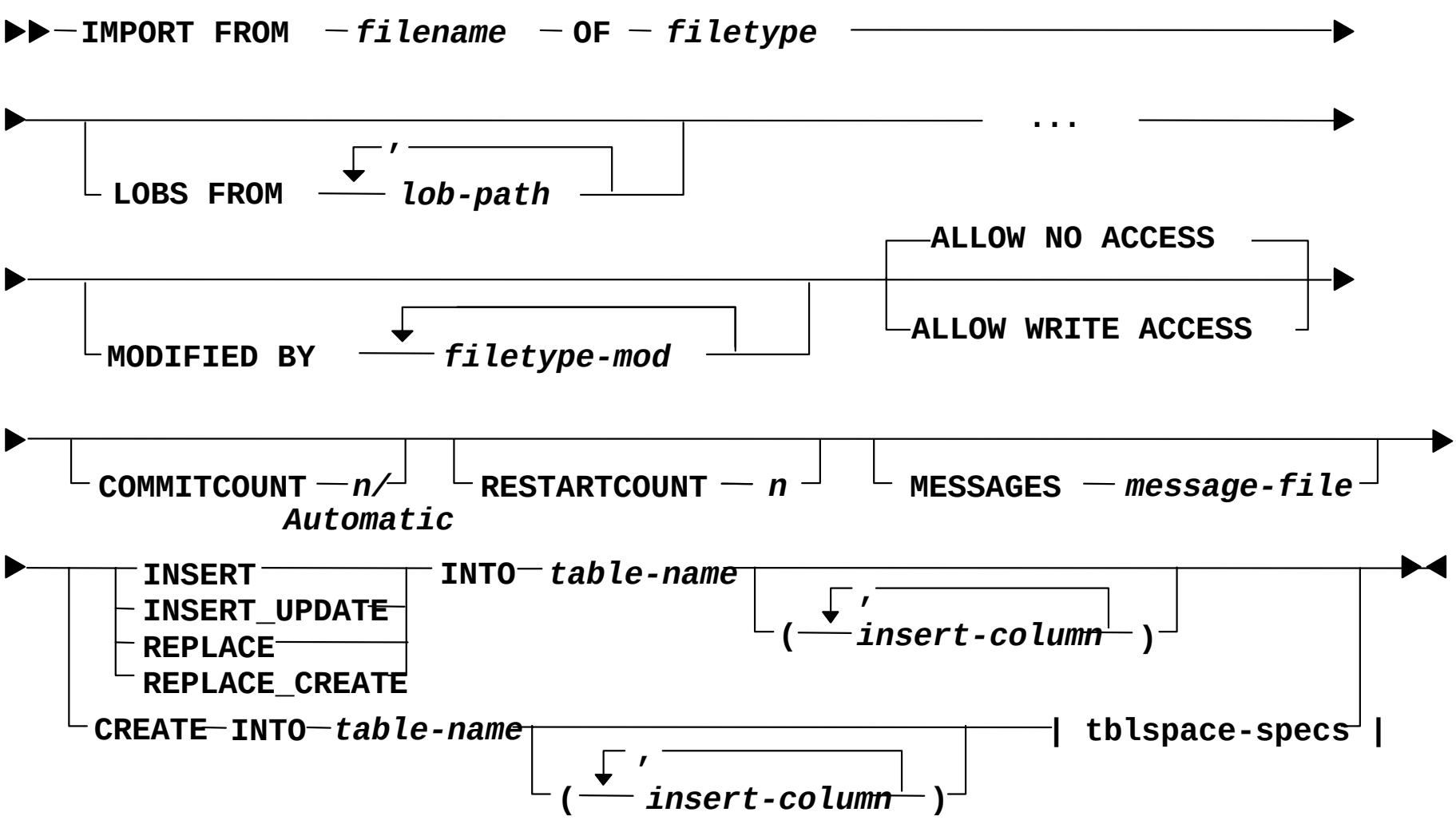
EXPORT command example

- Exports data from database tables to file
- Check message for error or warning messages




```
db2  connect  to  musicdb
db2  export  to artexprt of ixf messages artmsg
  select artno,  name  from artists
```

IMPORT command syntax (Basic)



Import Utility Example

db2 import from myfile.ixf of ixf messages msg.txt
insert into staff

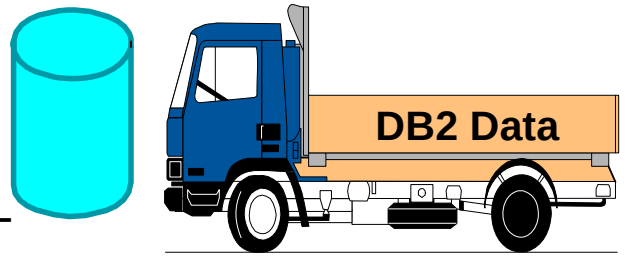
```
SQL3150N  The H record in the PC/IXF file has product "DB2    01.00", date
"19970220", and time "140848".

SQL3153N  The T record in the PC/IXF file has name "myfile",
qualifier "        ", and source "            ".

SQL3109N  The utility is beginning to load data from file "myfile".

SQL3110N  The utility has completed processing.  "58" rows were read from
  the
input file.

SQL3221W  ...Begin COMMIT WORK. Input Record Count = "58".

SQL3222W  ...COMMIT of any database changes was successful.

SQL3149N  "58" rows were processed from the input file.  "58" rows were
successfully inserted into the table.  "0" rows were rejected.
```

Differences between IMPORT and LOAD

IMPORT	LOAD
• Slow when moving large amounts of data • Creation of table/index supported with IXF format • Import into tables, views, aliases • Option to ALLOW WRITE ACCESS • All rows logged • Triggers fired, constraints enforced • Inserts can use space freed by deleted rows	• Faster for large amounts of data • Tables and indexes must exist • Load into tables only • Existing data can still be seen by read applications • Minimal logging; can make copy • Triggers not fired; unique constraints enforced, RI and check constraints via SET INTEGRITY • LOAD builds new extents

Four phases of Load

1. *LOAD*



Load data into tables
Collect index keys / sort
Consistency points at SAVECOUNT
Invalid data rows in dump file; messages in message file

2. *BUILD*

Indexes created or updated

3. *DELETE*

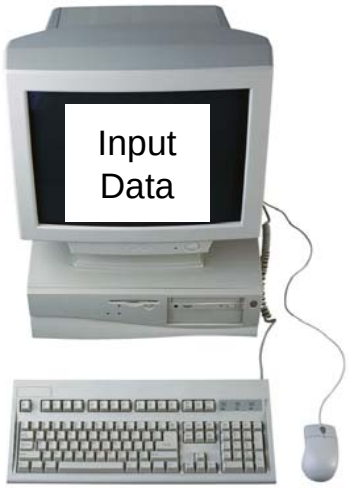
Unique Key Violations placed in Exception Table
Messages generated for unique key violations
Deletes Unique Key Violation Rows

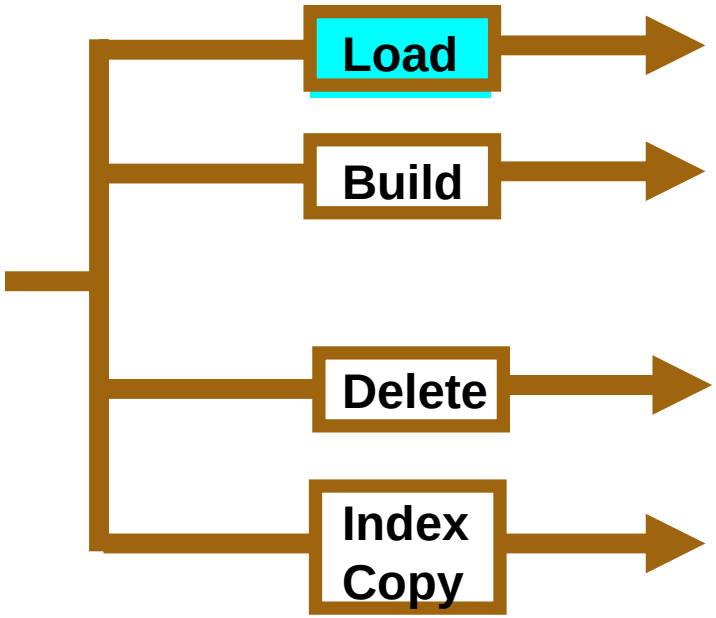
4. *INDEX COPY*

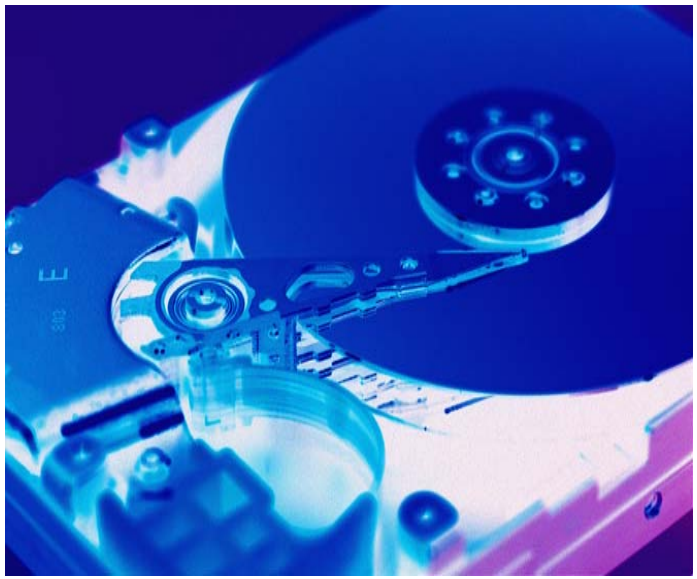
Copy indexes from temp table space to index table space

LOAD: Load phase

- During the LOAD phase, data is stored in a table and index keys are collected
- Save points are established at intervals
- Messages indicate the number of input rows successfully loaded
- If a failure occurs in this phase, use the RESTART option for LOAD to restart from the last successful consistency point

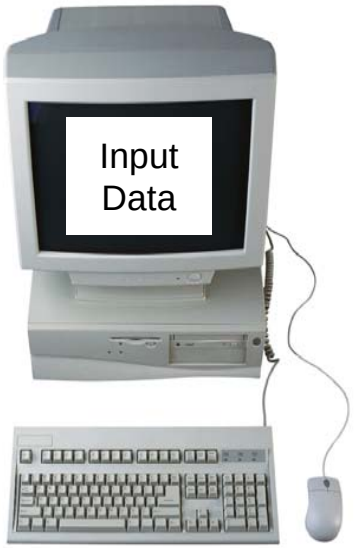


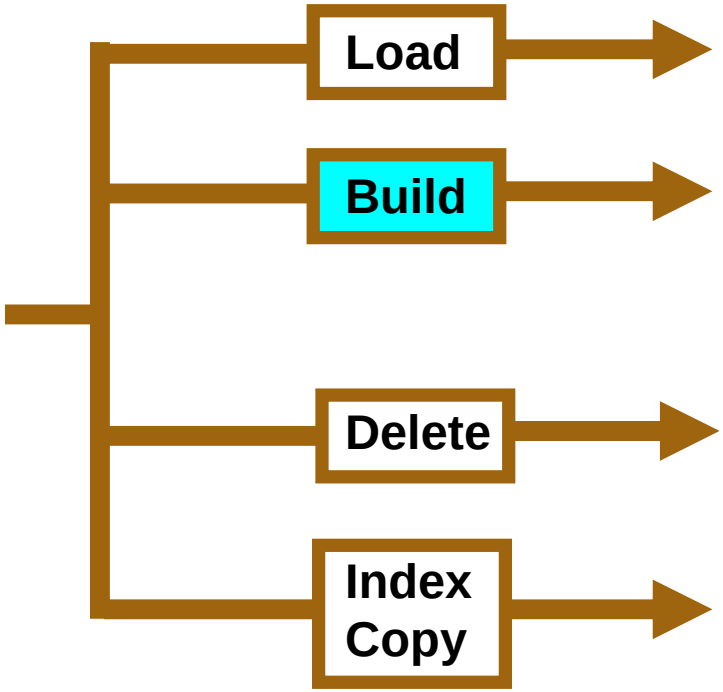




LOAD: Build phase

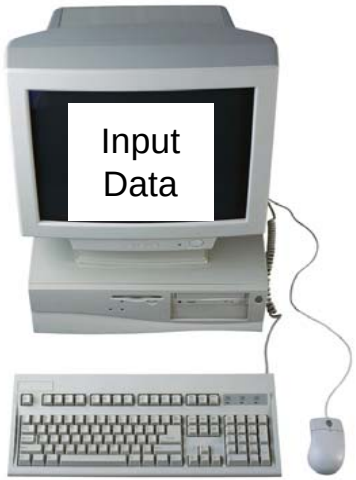
- During the BUILD phase, indexes are created based on the index keys collected in the Load phase
- The index keys are sorted during the Load phase
- If a failure occurs during this phase, LOAD restarts from the BUILD phase

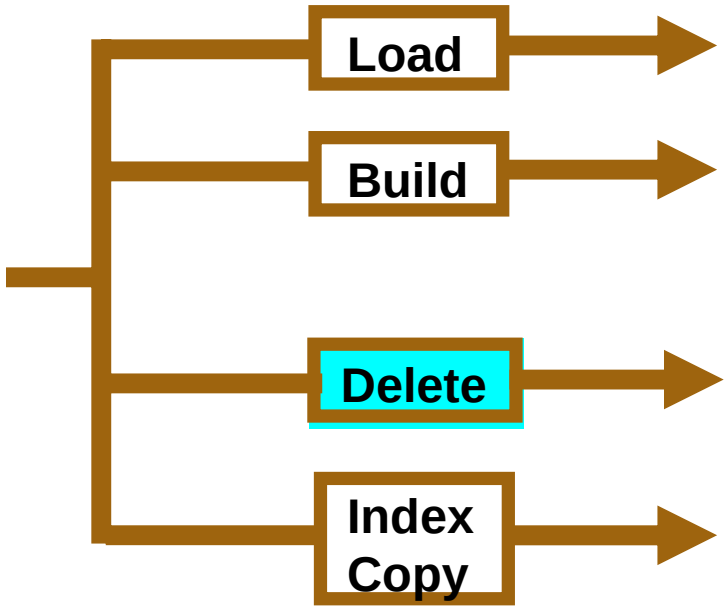




LOAD: Delete phase

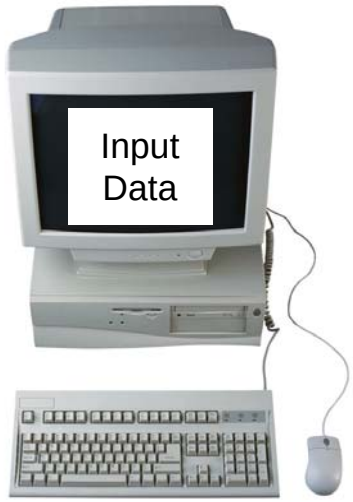
- During the DELETE phase, all rows that have violated a unique constraint are deleted
- If a failure occurs, LOAD restarts from the DELETE phase
- Once the database indexes are rebuilt, information about the rows containing the invalid keys is contained in an exception table, **WHICH SHOULD BE CREATED BEFORE LOAD**
- Finally, any duplicate keys found are deleted

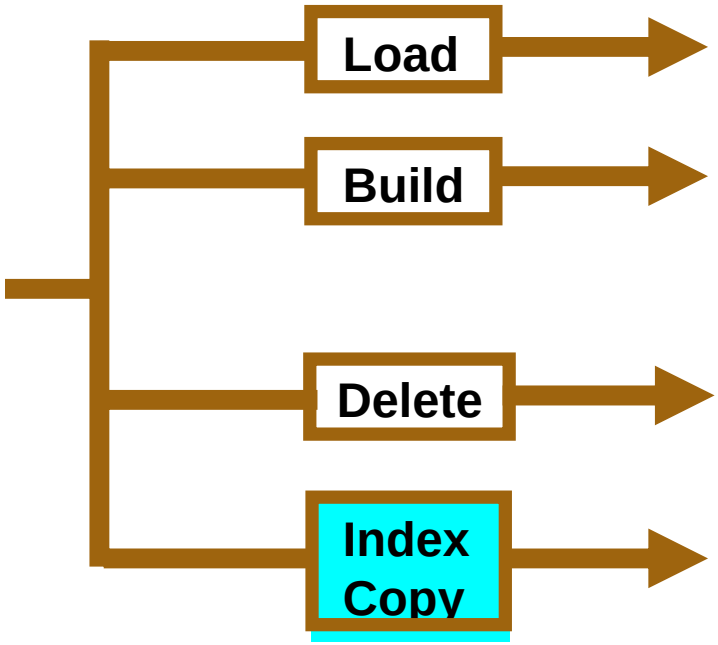


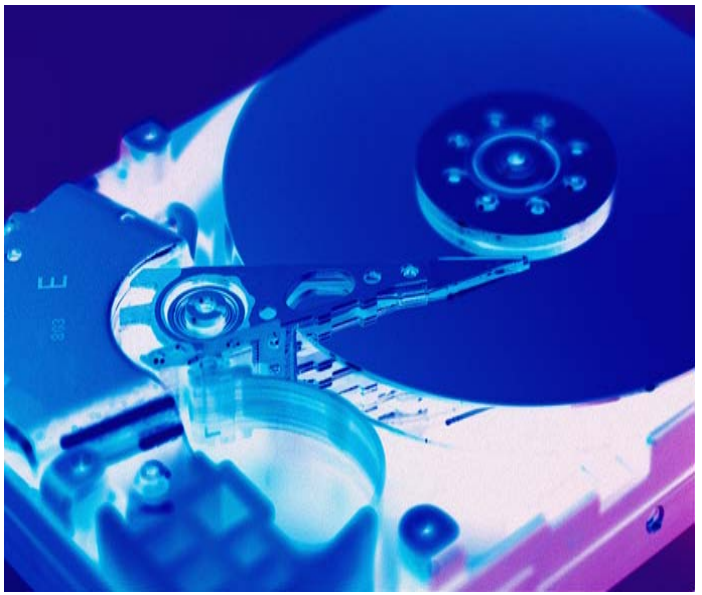


LOAD: Index Copy phase

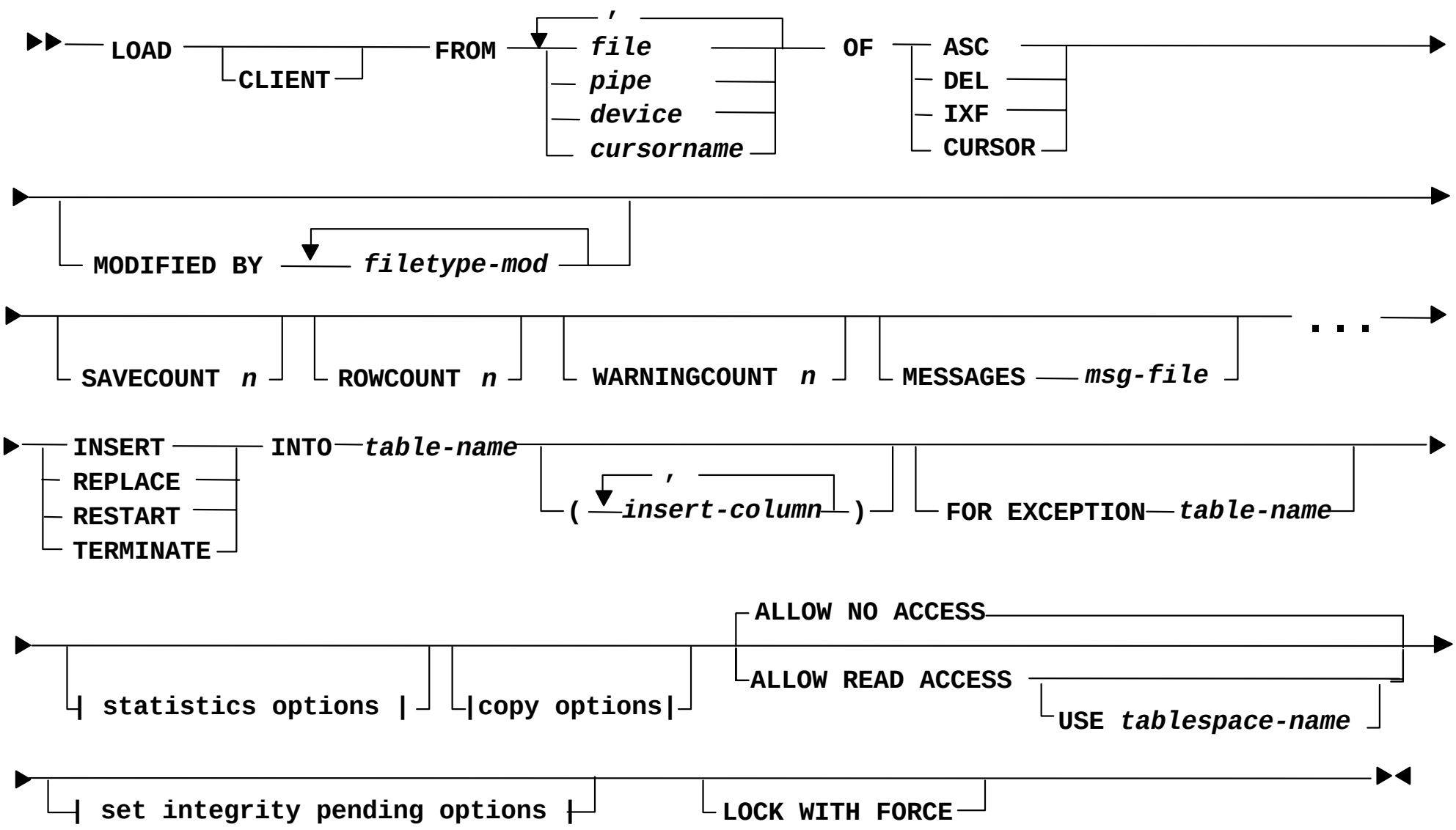
- The index data is copied from a system temporary table space to the original table space.
- This will only occur if a system temporary table space was specified for index creation during a load operation with the READ ACCESS option specified.



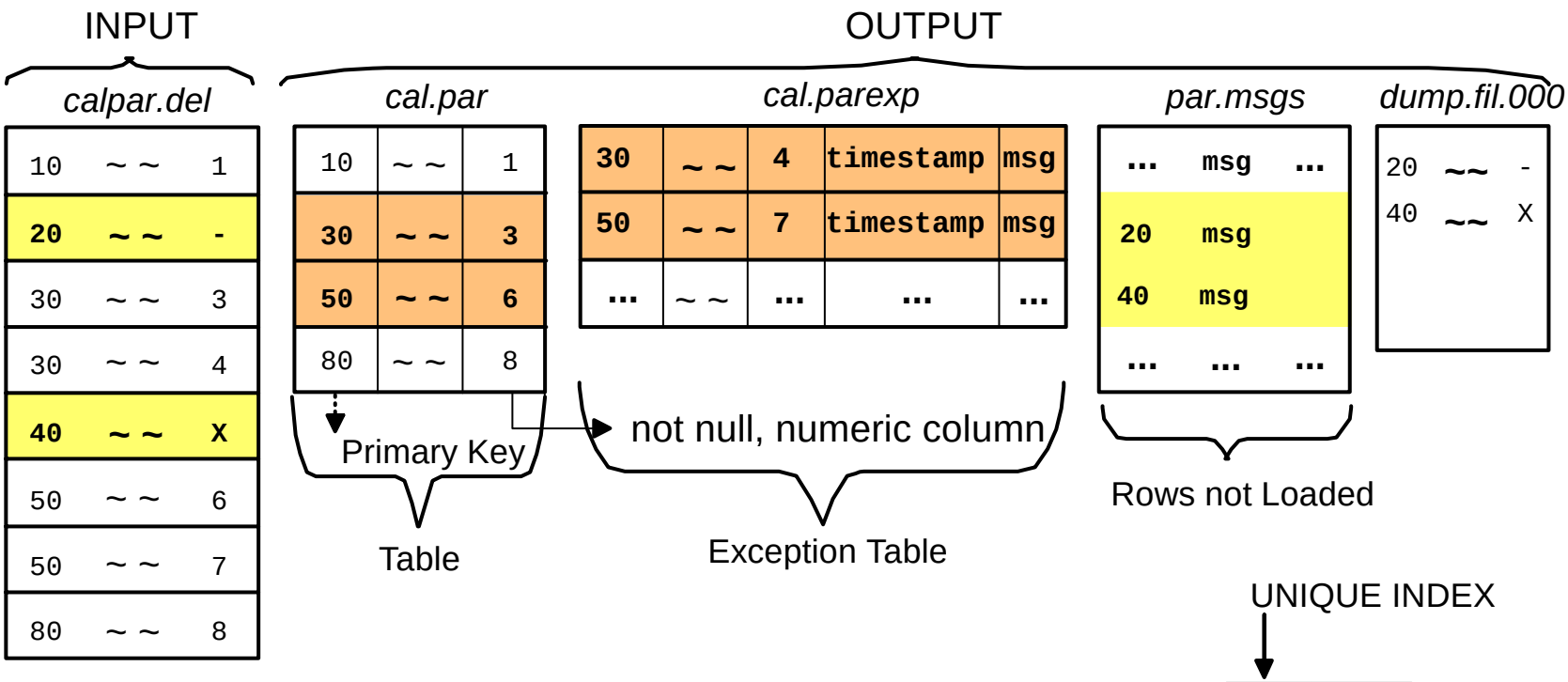


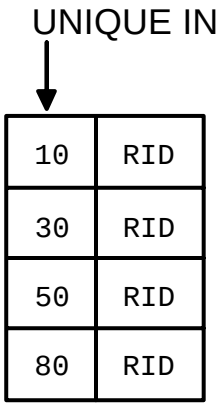


LOAD command syntax (Basic)



LOAD scenario






```
create tables/indexes
obtain delimited input file in sorted format
create exception table
```

```
db2 load from calpar.del  of del
   modified by dumpfile=<path>/dump.fil
   warningcount 100 messages par.msgs
   insert into cal.par for exception cal.parexp
db2 load query table cal.par
```

```
examine par.msgs file
```

```
examine cal.parexp exception table
```

Rules and methods for creating Exception Tables

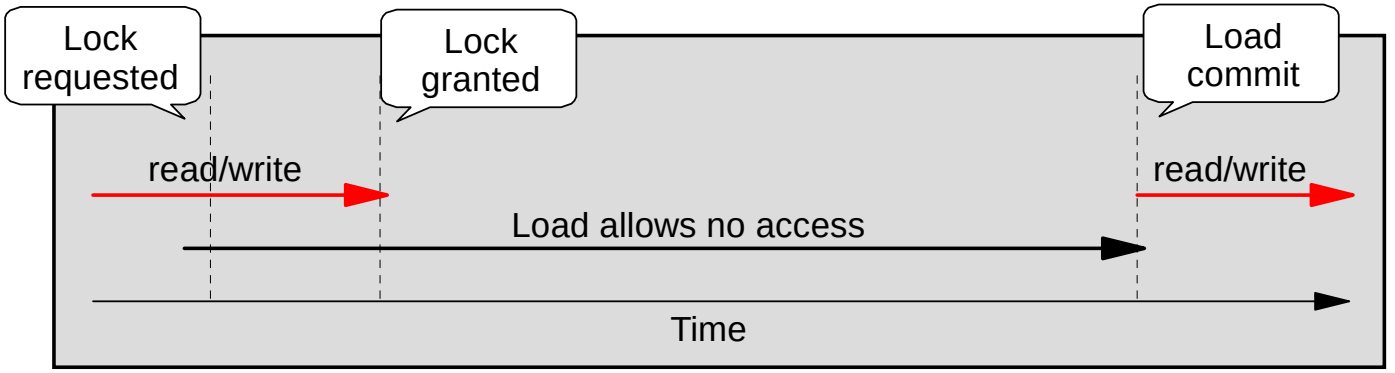
- The first *n* columns are the same
- No constraints and no trigger definitions
- *n* + 1 column TIMESTAMP
- *n* + 2 column CLOB (32 KB)
- user INSERT privilege

```
CREATE TABLE T1EXC LIKE T1

ALTER TABLE T1EXC
    ADD COLUMN TS TIMESTAMP
    ADD COLUMN MSG CLOB(32K)
```

```
CREATE TABLE T1EXC AS
   (SELECT T1.*,
       CURRENT TIMESTAMP AS TS,
       CLOB('', 32767) AS MSG
    FROM T1)
   DEFINITION ONLY
```

Offline versus Online Load

- ALLOW NO ACCESS



- ALLOW READ ACCESS

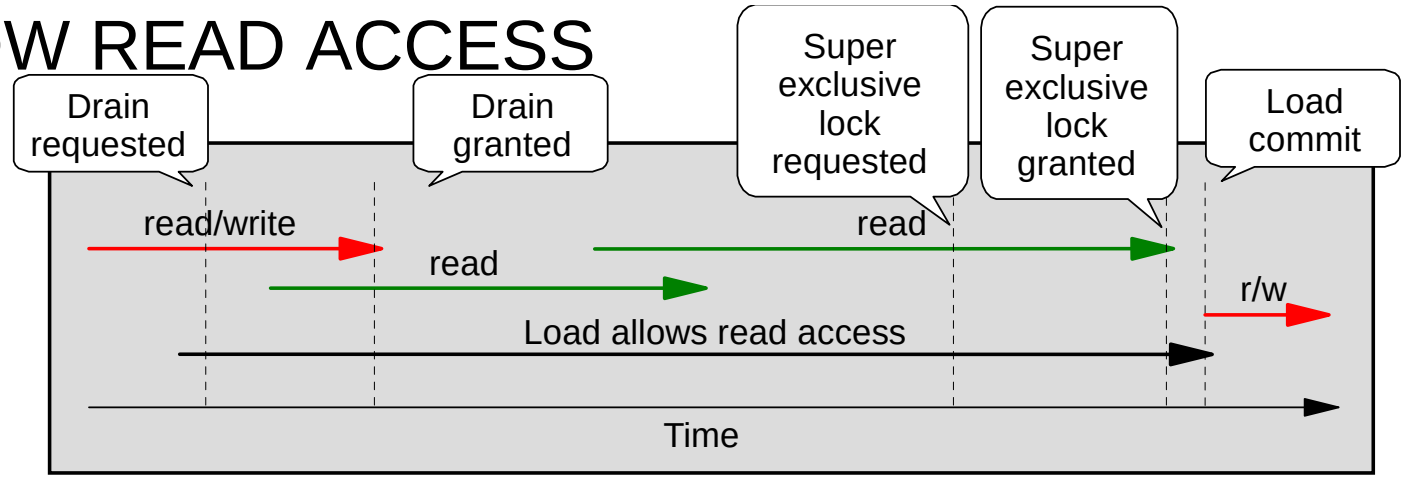


Table states

- (Load pending, Set Integrity Pending)
- LOAD QUERY TABLE <table-name>
- Tablestate:
  - Normal
  - Set Integrity Pending
  - Load in Progress
  - Load Pending
  - Reorg Pending
  - Read Access Only
  - Unavailable
  - Not Load Restartable
  - Unknown
- Table can be in several states at same time

```
Tablestate:
   Set Integrity Pending
   Load in Progress
   Read Access Only
```

Checking Load status: Load query

db2 load query table inst481.loadhist1

```
SQL3501W  The table space(s) in which the table resides will not be placed in
backup pending state since forward recovery is disabled for the database.
SQL3109N  The utility is beginning to load data from file
"/home/inst481/datamove/savehist.del".
SQL3500W  The utility is beginning the "LOAD" phase at time "05/12/2012
02:44:13.967160".
SQL3519W  Begin Load Consistency Point. Input record count = "0".
SQL3520W  Load Consistency Point was successful.
SQL3519W  Begin Load Consistency Point. Input record count = "10248".
...
SQL3519W  Begin Load Consistency Point. Input record count = "51450".
SQL3520W  Load Consistency Point was successful.
SQL0289N  Unable to allocate new pages in table space "LOADTSPD".
SQLSTATE=57011
SQL3532I  The Load utility is currently in the "LOAD" phase.

Number of rows read         = 51450
Number of rows skipped      = 0
Number of rows loaded       = 51450
Number of rows rejected     = 0
Number of rows deleted      = 0
Number of rows committed    = 51450
Number of warnings          = 0

Tablestate:
  Load Pending
```

Load monitoring: LIST UTILITIES

db2 LIST UTILITIES SHOW DETAIL

```
ID                                   = 4
Type                                 = LOAD
Database Name                        = MUSICDB
Member Number                        = 0
Description                          = [LOADID: 18.2012-05-12-02.48.55.850877.0 (11;4)]
[*LOCAL.inst481.120512063958] ONLINE LOAD DEL AUTOMATIC INDEXING INSERT NON-RECOVERABLE
INST481 .LOADHIST1
Start Time                           = 05/12/2012 02:48:55.869016
State                                = Executing
Invocation Type                      = User
Progress Monitoring:
   Phase Number                      = 1
      Description                    = SETUP
      Total Work                     = 0 bytes
      Completed Work                 = 0 bytes
      Start Time                     = 05/12/2012 02:48:55.869085

   Phase Number                      = 2
      Description                    = LOAD
      Total Work                     = 10000 rows
      Completed Work                 = 10000 rows
      Start Time                     = 05/12/2012 02:49:07.057958

   Phase Number [Current]            = 3
      Description                    = BUILD
      Total Work                     = 2 indexes
      Completed Work                 = 2 indexes
      Start Time                     = 05/12/2012 02:49:07.36690
```

Load Pending state: Recovering from LOAD failure

- Restart the Load:
  - Check Messages files
  - Use Restart option
  - Load operation automatically continues from last consistency point in Load or Build phase
    - or Delete phase if ALLOW NO ACCESS
- Replace the whole table
  - LOAD ... REPLACE
- Terminate the Load:
  - If LOAD ... INSERT, returns table to state preceding Load
  - If LOAD ... REPLACE, table will be truncated to an empty state
    - Create backup copy prior to Load to return to state preceding Load

Do not tamper with Load temporary files

Backup Pending state: COPY options

- When loading data and forward recovery is enabled:
  - COPY NO (default)
    - During load, *Backup pending* and *Load in progress*
    - After load, *Backup Pending*
  - COPY YES
    - Load has made Copy not *Backup pending*
  - NONRECOVERABLE
    - No copy made and no backup required

Load Copy file naming conventions

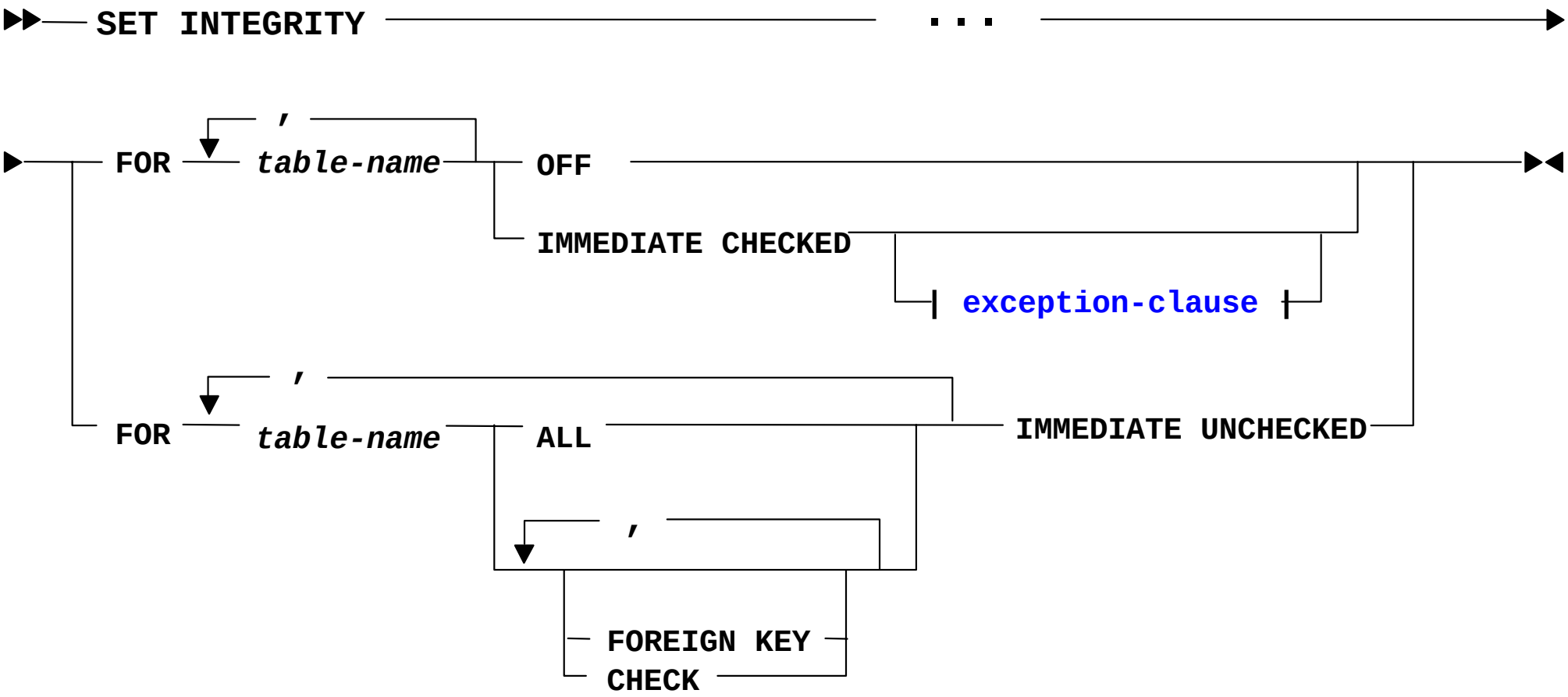
`Databasealias.Type.Instancename.DBPART000.Timestamp.number`

Type: 0=Full Backup
3=Table Space Backup
4 =Copy from Table Load

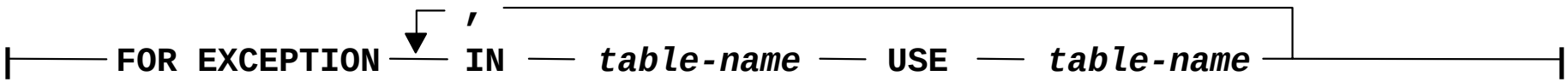
Set Integrity Pending table state

- Load turns OFF constraint checking:
  - Leaves table in Set Integrity Pending state
  - If parent table is in Set Integrity Pending, then dependents may also be in Set Integrity Pending
  - LOAD INSERT, ALLOW READ ACCESS
    - Loaded table in Set Integrity Pending with read access
  - LOAD INSERT, ALLOW NO ACCESS
    - Loaded table in Set Integrity Pending with no access
  - LOAD REPLACE, SET INTEGRITY PENDING CASCADE DEFERRED
    - Loaded table in Set Integrity Pending with no access
  - LOAD REPLACE, SET INTEGRITY PENDING CASCADE IMMEDIATE
    - Loaded table and descendant Foreign Key tables are in Set Integrity Pending with no access

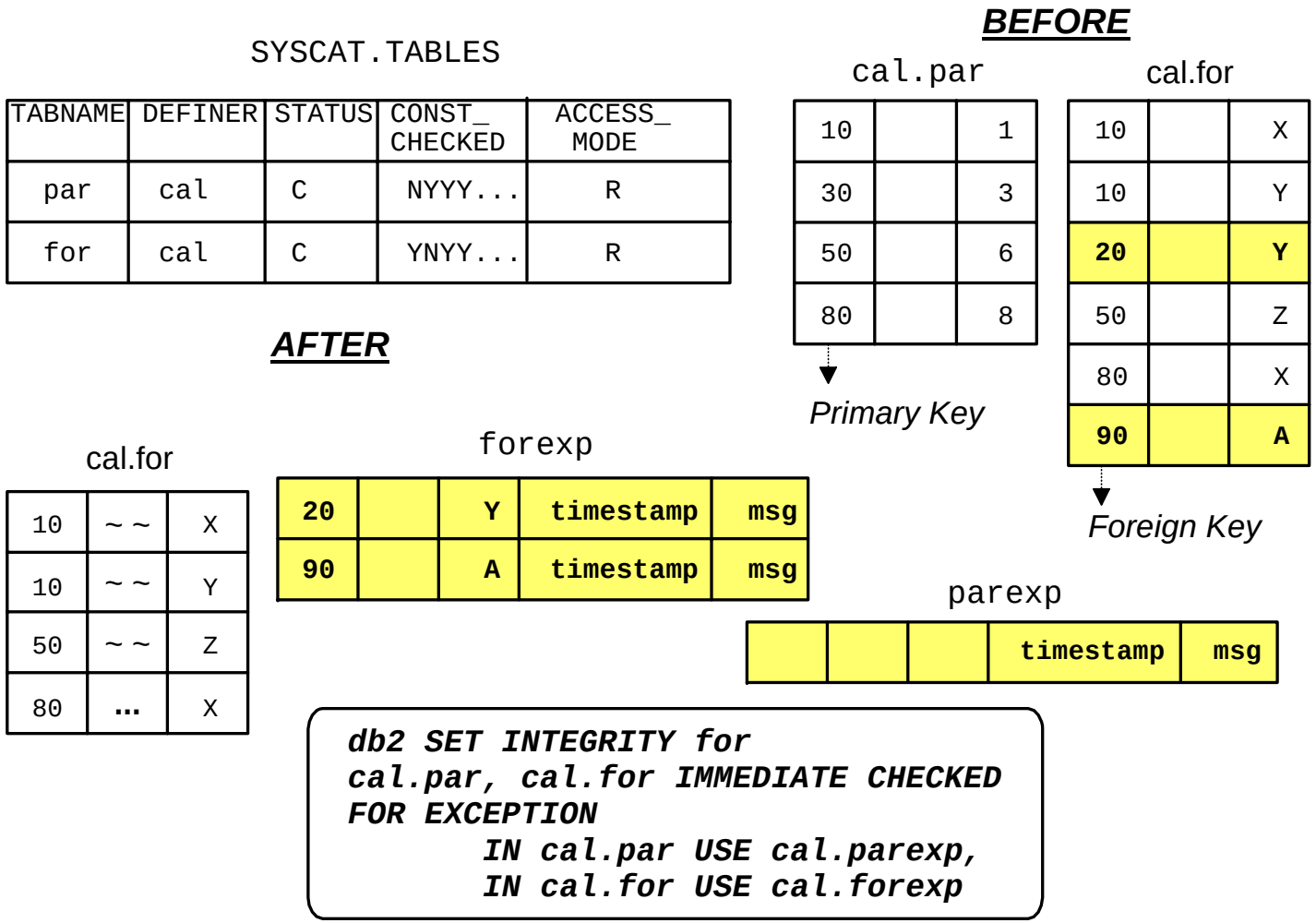
SET INTEGRITY command syntax (Basic)



exception-clause



Example – Running Set Integrity



Meaningful steps for LOAD

- Create tables and indexes
- Create exception tables
- Sort data
- Back up TS/DB (if using REPLACE)
- Consider freespace
- Load .... for Exception ... Savecount ... Warningcount...
- Examine xx.msg and dumpfile (after LOAD completes)
- Examine exception tables (after LOAD completes)
- Back up table space if log retain=recovery and COPY NO
- Set Integrity for .... (only if table in Set Integrity Pending state)
- Update statistics (if necessary)

db2move utility

- Facilitates the moving/copying of large numbers of tables between databases
- Can be used with **db2look** to move/copy a database between different platforms (for example, AIX to Windows).

- **Usage:** `db2move <dbName> EXPORT/IMPORT/LOAD/COPY [options]`
- **EXPORT:** The system catalogs are queried, a list of tables is compiled (based on the options), and the tables are exported in IXF format -- additionally, a file called **db2move.lst** is created
- **IMPORT:** The db2move.lst file is used to import the IXF files created in the EXPORT step
- **LOAD:** The db2move.lst file is used to load the PC/IXF data files created in the EXPORT step
- **COPY:** Duplicates schema(s) into a target database

db2move COPY option

Copy one or more schemas between DB2 databases

Uses a -co option to specify:

- Target Database:

  `"TARGET_DB <db name> [USER <userid> USING <password>]"`

- MODE:
  - DDL_AND_LOAD - Creates all supported objects from the source schema, and populates the tables with the source table data. Default
  - DDL_ONLY -Creates all supported objects from the source schema, but does not repopulate the tables.
  - LOAD_ONLY- Loads all specified tables from the source database to the target database. The tables must already exist on the target.
- SCHEMA_MAP: Allows user to rename schema when copying to target
- TABLESPACE_MAP: Table space name mappings to be used
- Load Utility option: COPY NO or Nonrecoverable
- Owner: Change the owner of each new object created in the target schema

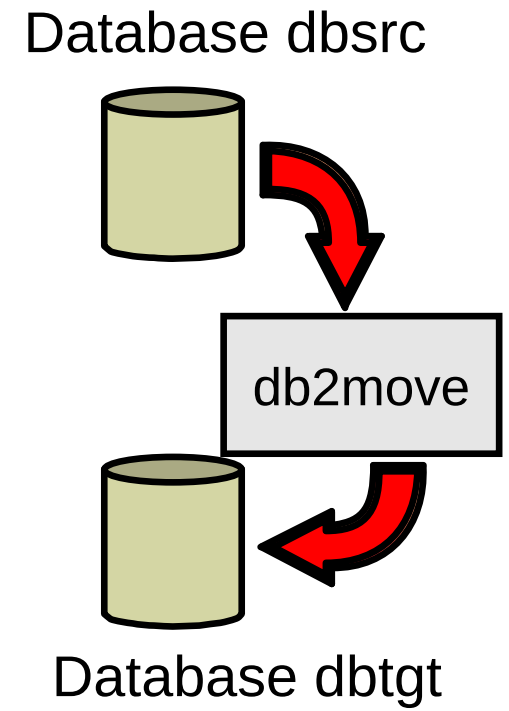
db2move COPY schema examples

- To duplicate schema schema1
  from source database dbsrc
  to target database dbtgt, issue:

```
db2move dbsrc COPY -sn schema1 -co TARGET_DB dbtgt
    USER myuser1 USING mypass1
```

- To duplicate schema schema1
  from source database dbsrc
  to target database dbtgt and
  rename the schema to newschema1 on
  the target and map source table space ts1
  to ts2 on the target, issue:

```
db2move dbsrc COPY -sn schema1 -co TARGET_DB dbtgt
   USER myuser1 USING mypass1
    SCHEMA_MAP ((schema1,newschema1))
   TABLESPACE_MAP ((ts1,ts2), SYS_ANY))
```



Output files generated:

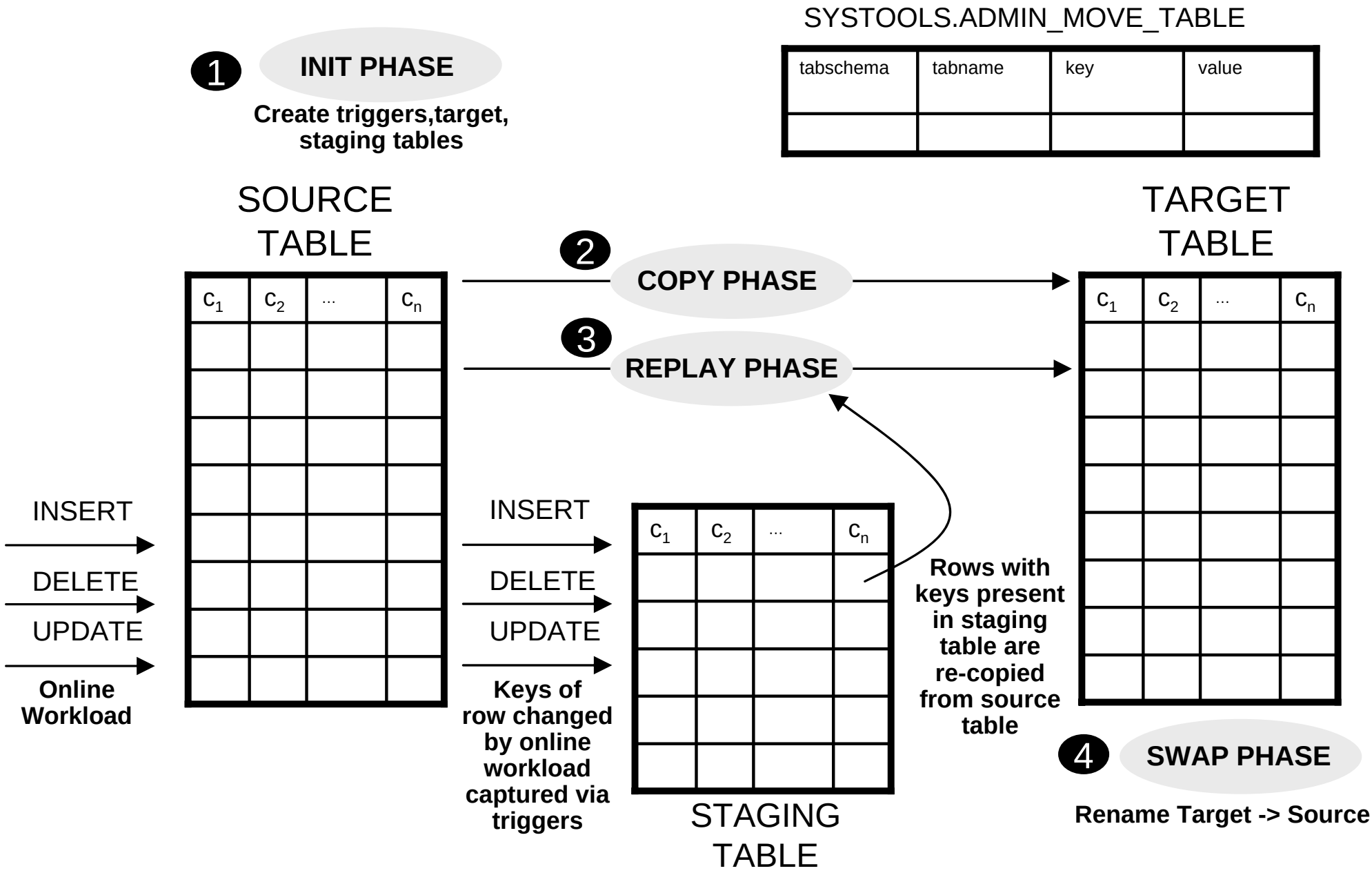
COPYSCHEMA.msg
COPYSCHEMA.err
LOADTABLE.msg
LOADTABLE.err

These files are timestamped.

Online Table Move stored procedure

- The ADMIN_MOVE_TABLE procedure is designed to move data from a source table to a target table with a minimal impact to application access
  - Changes that can be made using ADMIN_MOVE_TABLE:
    - New Data, Index or Long table spaces, which could have a different page size, extent size or type of table space management (like moving from SMS to Automatic Storage)
    - Data compression could be implemented during the move
    - MDC clustering can be added or changed
    - Range partitions can be added or changed
    - Distribution keys can be changed for DPF tables
    - Columns can be added, removed or changed
  - Multiple phased processing allows write access to the source table except for a short outage required to swap access to the target table

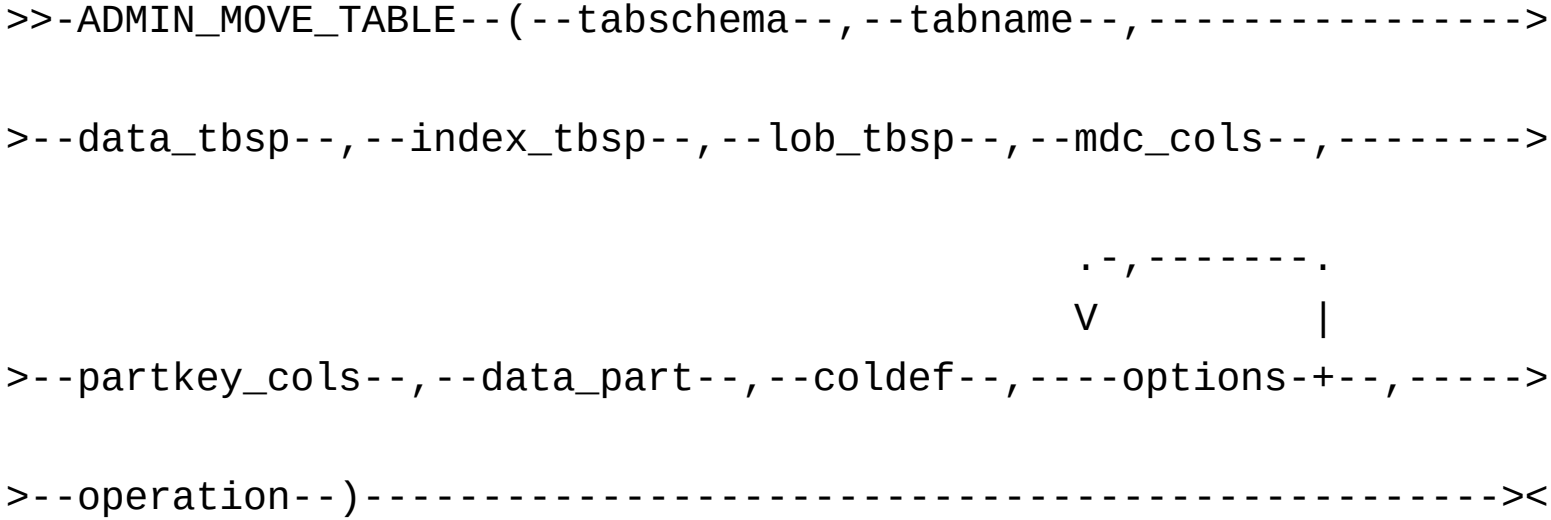
ADMIN_MOVE_TABLE: Processing phases



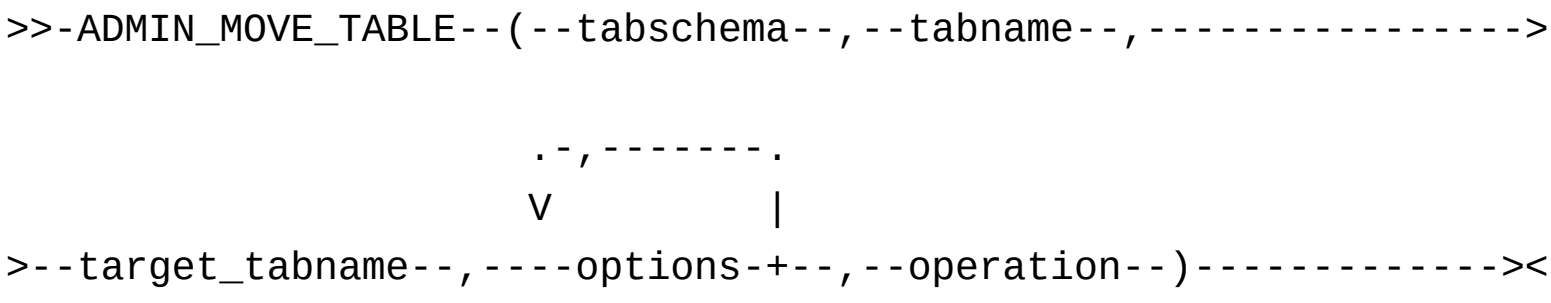
ADMIN_MOVE_TABLE procedure methods

- There are two methods of calling ADMIN_MOVE_TABLE:

  One method specifies the how to define the target table.



  The second method allows a predefined table to be specified as the target for the move.



Example: Move a table to new table space

```
CALL  SYSPROC.ADMIN_MOVE_TABLE ( 'INST481', 'LOADHIST1',
  'TSHISTM1','TSHISTM2','TSHISTM2',
  NULL,NULL,NULL,NULL, 'COPY_USE_LOAD,FORCE','MOVE')
```

```
  KEY                                VALUE
 ---------------------------------- --------------------------------
 AUTHID                             INST481
 CLEANUP_END                        2012-05-12-02.58.44.712855
 CLEANUP_START                      2012-05-12-02.58.44.464132
 COPY_END                           2012-05-12-02.58.35.933404
 COPY_OPTS                          LOAD,WITH_INDEXES,NON_CLUSTER
 COPY_START                         2012-05-12-02.58.29.844891
 COPY_TOTAL_ROWS                    110000
 INDEXNAME                          LHIST1IX1
 INDEXSCHEMA                        INST481
 INDEX_CREATION_TOTAL_TIME          0
 INIT_END                           2012-05-12-02.58.28.012105
 INIT_START                         2012-05-12-02.58.24.682107
 REPLAY_END                         2012-05-12-02.58.43.853669
 REPLAY_START                       2012-05-12-02.58.35.939140
 REPLAY_TOTAL_ROWS                  0
 REPLAY_TOTAL_TIME                  3
 STATUS                             COMPLETE
 SWAP_END                           2012-05-12-02.58.44.322158
 SWAP_RETRIES                       0
 SWAP_START                         2012-05-12-02.58.43.861629
 UTILITY_INVOCATION_ID
  0100000009000000080000000000000000020120512025828024814000000000
 VERSION                            10.01.0000
```

Ingest Utility – for Continuous Data Ingest

- The ingest utility is a high-speed client-side DB2 utility that streams data from files and pipes into DB2 target tables.
- The ingest utility ingests pre-processed data directly or from files output by ETL tools or other means
- It can run continually and thus it can process a continuous data stream through pipes.
- The data is ingested at speeds that are high enough to populate even large databases in partitioned database environments
- An INGEST command updates the target table with low latency in a single step
- Uses row locking, so it has minimal interference with other user activities on the same table
- These ingest operations support the following SQL statements: INSERT, UPDATE, MERGE, REPLACE, and DELETE
- Other important features of the ingest utility include:
  - Commit by time or number of rows.
  - Support for copying rejected records to a file or table, or discarding them
  - Support for restart and recovery.
- The INGEST command supports the following input data formats:
  - Delimited text
  - Positional text and binary
  - Columns in various orders and formats

Ingest command examples – Insert

The following example inserts data from a delimited text file with fields separated by a comma (the default).

The fields in the file correspond to the table columns.

```
INGEST FROM FILE my_file.txt
   FORMAT DELIMITED
   (
      $field1 INTEGER EXTERNAL,
      $field2 DATE 'mm/dd/yyyy',
      $field3 CHAR(32)
   )
   INSERT INTO my_table
      VALUES($field1, $field2, $field3);
```

Ingest command examples – Update

The following examples update the table rows whose primary key matches the corresponding fields in the input file.

```
INGEST FROM FILE my_file.txt
  FORMAT DELIMITED
  (
       $key1  INTEGER EXTERNAL,
       $key2  INTEGER EXTERNAL,
       $data1 CHAR(8),
       $data2 CHAR(32),
       $data3 DECIMAL(5,2) EXTERNAL
  )
  UPDATE my_table
       SET (data1, data2, data3) = ($data1, $data2, $data3)
       WHERE (key1 = $key1) AND (key2 = $key2);
```

Monitoring INGEST using INGEST LIST and INGEST GET STATS

```
=> INGEST LIST
Ingest job ID       = DB2100000:20101116.123456.234567:34567:45678
Ingest temp job ID  = 4
Database Name       = MYDB
Input type          = FILE
Target table        = MY_SCHEMA.MY_TABLE
Start Time          = 01/10/2012 11:54:45.773215
Running Time        = 00:02:03
Number of records processed = 30,000
```

```
=> INGEST GET STATS FOR 4 EVERY 3 SECONDS

Ingest job ID = DB2100000:20101116.123456.234567:34567:45678
Database      = MYDB
Target table  = MY_SCHEMA.MY_TABLE

Overall          Overall         Current          Current
ingest rate      write rate      ingest rate      write rate       Total records
(records/second) (writes/second) (records/second) (writes/second)
---------------- --------------- ---------------  ---------------- ----------------
            3333           65432           76543             87654           108765
            3334           75432           86543             97654           118765
            3335           85432           96543            107654           128765
     etc (new row every 3 seconds until INGEST command ends)
```

When to use INGEST rather than LOAD

- Use INGEST when any of the following is true
  - You need other applications to update the table while it is being loaded
  - The input file contains fields you want to skip over
  - You need to specify an SQL statement other than INSERT
  - You need to specify an SQL expression (to construct a column value from field values)
  - You need to recover and continue on when the utility gets a recoverable error

When to use LOAD rather than INGEST

- Use LOAD when any of the following is true
  - You don't need other applications to update the table while it is being loaded
  - You need to load a table that contains XML or LOB columns
  - You need to load from cursor or load from a device
  - You need to load from a file in IXF format
  - You need to load a GENERATED ALWAYS column or SYSTEM_TIME column with the data specified in the input file

Unit summary

Having completed this unit, you should be able to:

- Discuss using the INSERT SQL statement to populate tables
- Explain the differences between IMPORT and LOAD processing
- Explain the EXPORT, IMPORT, and LOAD command options
- Create and use Exception Tables and Dump-Files
- Check table status using LOAD QUERY
- Describe Load Pending and Set Integrity Pending status for a table
- Use the SET INTEGRITY command
- Discuss the db2move and db2look commands
- Use the ADMIN_MOVE_TABLE procedure to move a table to different table spaces
- List some of the features of the Ingest utility for continuous data ingest

Student exercise