Import Utility

What Is the Import Utility?

- The Import utility reads the object definitions and table data from an Export dump file.
- It inserts the data objects into an Oracle database.
- Export dump files can only be read by the Oracle Import utility.
- The version of the
- Import utility cannot be earlier than the version of the Export utility used to create the dump file.

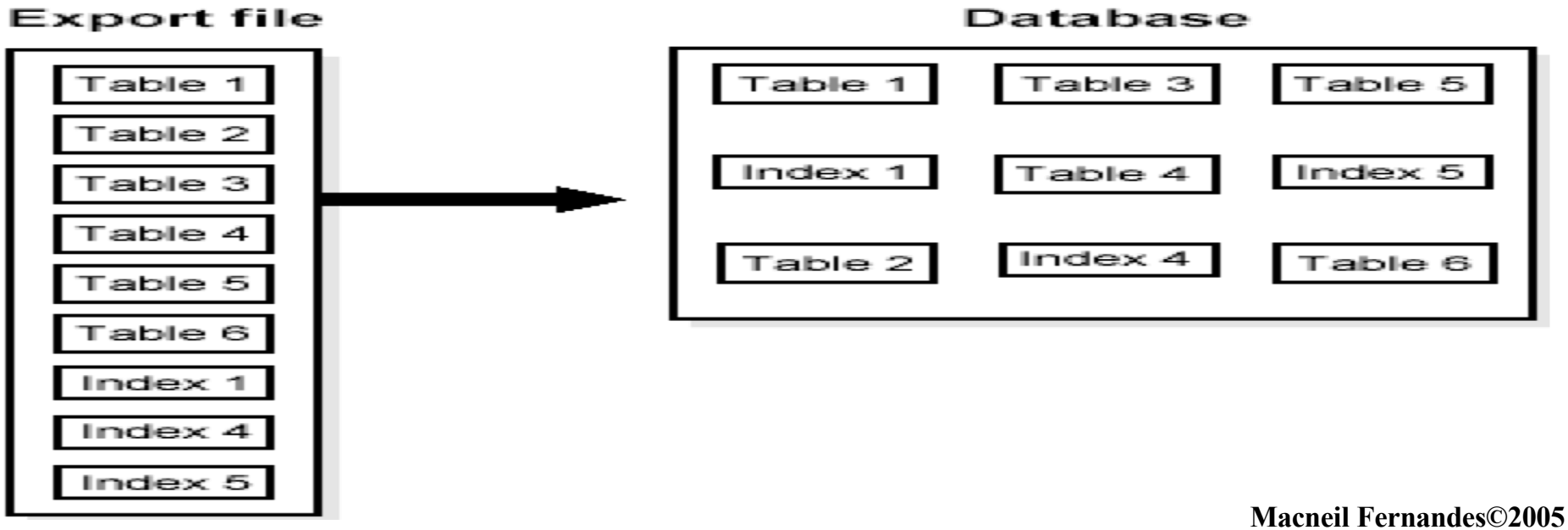


Table Objects: Order of Import

- Table objects are imported as they are read from the export file.
- The export file contains objects in the following order:
  1. Type definitions
  2. Table definitions
  3. Table data
  4. Table indexes
  5. Integrity constraints, views, procedures, and triggers
  6. Bitmap, functional, and domain indexes
- This sequence prevents data from being rejected due to the order in which tables are imported.
- This sequence also prevents redundant triggers from firing twice on the same data (once when it is originally inserted and again during the import).

Before Using Import

- Before you begin using Import, be sure you take care of the following items:
  - Run the `catexp.sql` or `catalog.sql` script
  - Verify that you have the required access privileges

Running catexp.sql or catalog.sql

- `The catexp.sql` or `catalog.sql` script needs to be run only **once** on a database. You do not need to run either script again before performing future import operations. Both scripts perform the following tasks to *prepare the database for Import*:
  - Assign all necessary privileges to the `IMP_FULL_DATABASE` role.
  - Assign `IMP_FULL_DATABASE` to the `DBA` role.
  - Create required views of the data dictionary.

Before Using Import

Verifying Access Privileges

- To use Import, you need the privilege `CREATE SESSION` to log on to the Oracle database server. This privilege belongs to the `CONNECT` role established during database creation.
- You can do an import even if you did not create the export file. However, if the export file was created by someone other than you, you can import that file only if you have the `IMP_FULL_DATABASE` role.

Importing into Existing Tables

Manually Creating Tables Before Importing Data

- You should use either the same table definition previously used or a compatible format.
- Although you can
  - Increase the width of columns and
  - Change their order,
- You cannot do the following:
  - Add `NOT NULL` columns
  - Change the datatype of a column to an incompatible datatype (`LONG` to `NUMBER`, for example)
  - Change the definition of object types used in a table
  - Change `DEFAULT` column values

Importing into Existing Tables

Disabling Referential Constraints

- In the normal import order, referential constraints are imported only after all tables are imported.
- Referential constraints between tables can also cause problems. For example, if the `emp` table appears before the `dept` table in the export file, but a referential check exists from the `emp` table into the `dept` table, some of the rows from the `emp` table may not be imported due to a referential constraint violation.
- When an error occurs, Import generates an error message, bypasses the failed row, and continues importing other rows in the table. You can disable constraints manually to avoid this.

Importing into Existing Tables

Manually Ordering the Import

- When the constraints are reenabled after importing, the entire table is checked, which may take a long time for a large table. If the time required for that check is too long, it may be beneficial to order the import manually.
- To do so, perform several imports from an export file instead of one. First, import tables that are the targets of referential checks. Then, import the tables that reference them. This option works if tables do not reference each other in a circular fashion, and if a table does not reference itself.

Invoking Import

Command-Line Entries

- You can specify all valid parameters and their values from the command line using the following syntax:

```
imp username/password PARAMETER=value or
imp username/password
PARAMETER=(value1,value2,...,value n)
```

- The number of parameters cannot exceed the maximum length of a command line on the system.

Interactive Import Prompts

- You can use the following syntax to start Import in interactive mode:

```
imp username/password
```

Invoking Import

- Import will display each parameter with a request for you to enter a value.
- This method exists for backward compatibility and is not recommended because it provides less functionality than the other methods.

Parameter Files

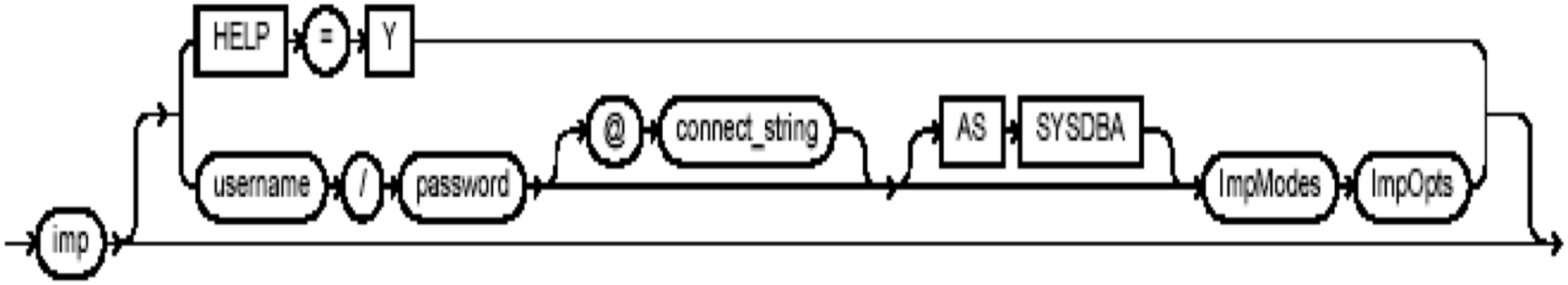
- You can specify all valid parameters and their values in a parameter file.
- Storing the parameters in a file allows them to be easily modified or reused, and is the recommended method for invoking Import.
- Create the parameter file using any flat file text editor. The command-line option `PARFILE=filename` tells Import to read the parameters from the specified file rather than from the command line.

Import Modes

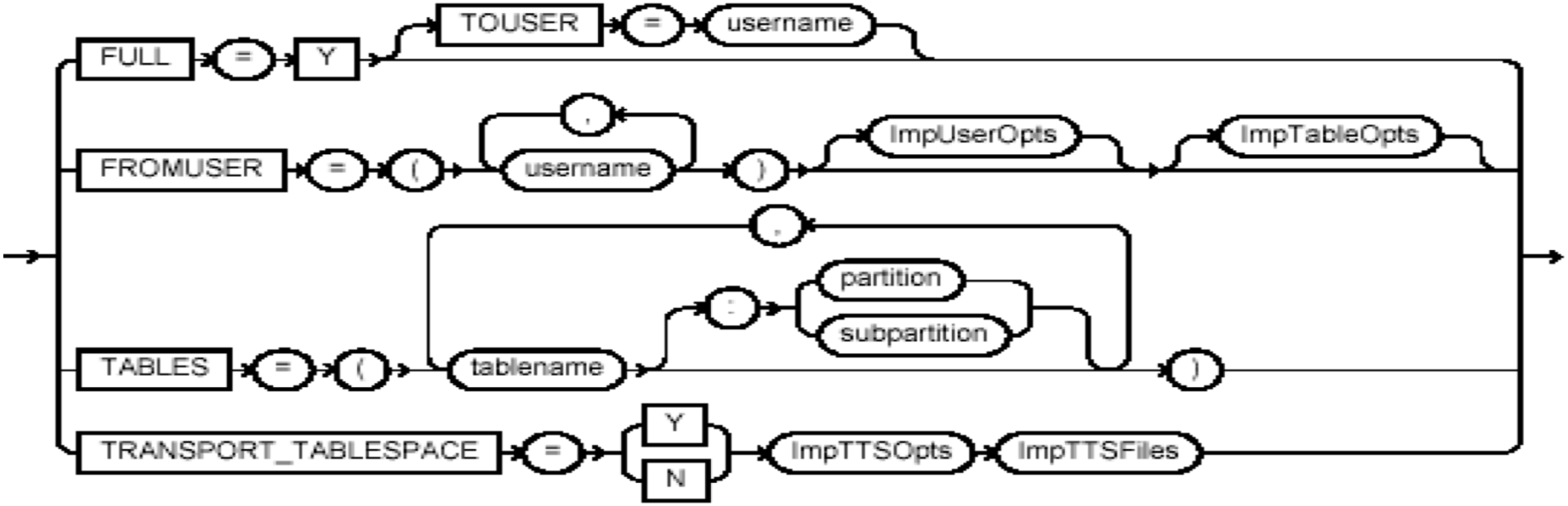
1. Full
   - Only users with the `IMP_FULL_DATABASE` role can import in this mode, which imports a full database export dump file.
   - Use the `FULL` parameter to specify this mode.
2. Tablespace
   - Allows a privileged user to move a set of tablespaces from one Oracle database to another.
   - Use the `TRANSPORT_TABLESPACE` parameter to specify this mode.
3. User (Owner)
   - Allows you to import all objects that belong to you
   - Use the `FROMUSER` parameter to specify this mode.
4. Table
   - Allows you to import specific tables and partitions.
   - Use the `TABLES` parameter to specify this mode.

Import Parameters

Import_start

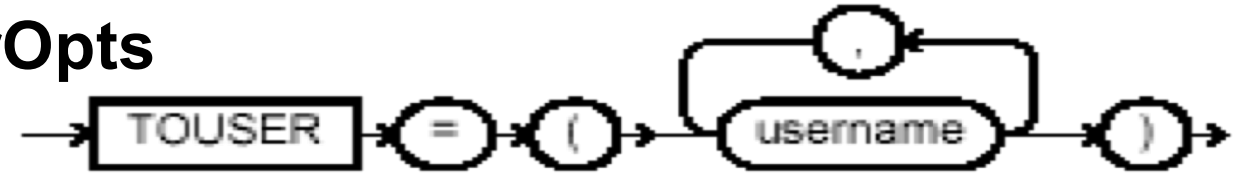


ImpModes

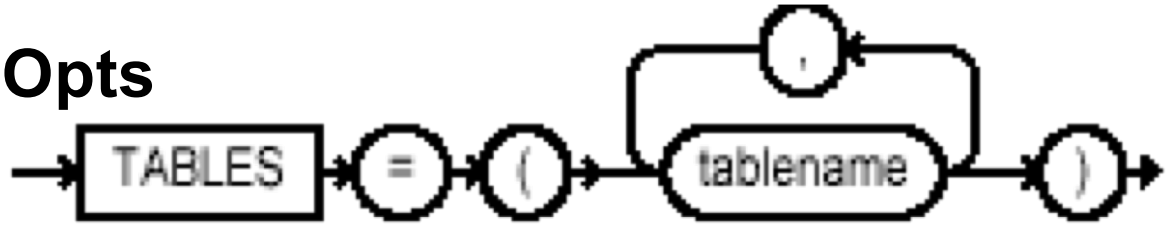


Import Parameters

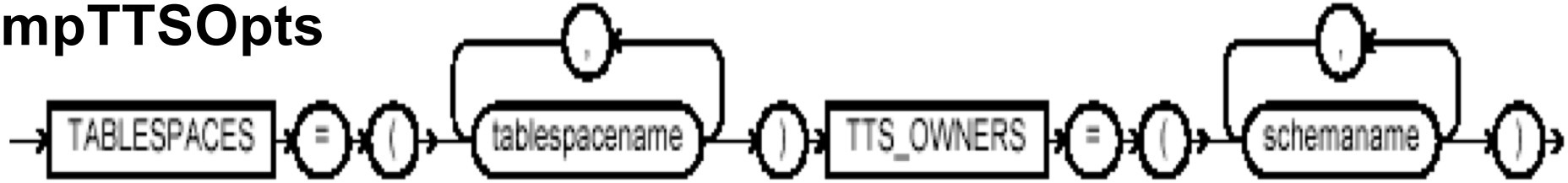
ImpUserOpts



ImpTableOpts



ImpTTSOpts



ImpTTSFiles

Import Parameters

CONSTRAINTS

- Default: `y`
- Specifies whether or not table constraints are to be imported.
- Primary key constraints for index-organized tables (IOTs) and object tables are always imported.

FILE

- Default: `expdat.dmp`
- Specifies the names of the export files to import.
- You need not be the user who exported the export files; however, you must have read access to the files.

Import Parameters

FROMUSER

- Default: none
- A comma-separated list of schemas to import.

GRANTS

- Default: `y`
- Specifies whether to import object grants.
- By default, the Import utility imports any object grants that were exported. If the export was a user-mode Export, the export file contains only first-level object grants (those granted by the owner).
- If the export was a full database mode Export, the export file contains all object grants, including lower-level grants

Import Parameters

FULL

- Default: `n`
- Specifies whether to import the entire export file.

IGNORE

- Default: `n`
- Specifies how object creation errors should be handled.
- If you specify `IGNORE=y`, Import overlooks object creation errors when it attempts to create database objects, and continues without reporting the errors. Even if `IGNORE=y`, Import will not replace an existing object; instead, it will skip the object.
- If `IGNORE=n`, Import logs and/or displays the object creation error before continuing.

Import Parameters

PARFILE

- Default: none
- Specifies a filename for a file that contains a list of Import parameters.

ROWS

- Default: `y`
- Specifies whether or not to import the rows of table data.

TABLES

- Default: none
- Specifies that the Import is a table-mode import and lists the table names and partition and subpartition names to import.

Import Parameters

TOUSER

- Default: none
- Specifies a list of usernames whose schemas will be targets for Import.

STATISTICS

- Default: `ALWAYS`
- Specifies what is done with the database optimizer statistics at import time.
- The options are:
  - `ALWAYS`
  - `NONE`
  - `SAFE`
  - `RECALCULATE`

Example Import Sessions

In this example, using a full database export file, an administrator imports the `dept` and `emp` tables into the `scott` schema.

- Parameter File Method

```
> imp SYSTEM/password PARFILE=params.dat
```

The params.dat file contains the following information:

```
FILE=dba.dmp
SHOW=n
IGNORE=n
GRANTS=y
FROMUSER=scott
TABLES=(dept,emp)
```

- Command-Line Method

```
> imp SYSTEM/password FILE=dba.dmp FROMUSER=scott
  TABLES=(dept,emp)
```

Example Import Sessions

This example illustrates importing the **unit** and **manager** tables from a file exported by **blake** into the **scott** schema.

- Parameter File Method

```
> imp SYSTEM/password PARFILE=params.dat
```

The **params.dat** file contains the following information:

```
FILE=blake.dmp
SHOW=n
IGNORE=n
GRANTS=y
ROWS=y
FROMUSER=blake
TOUSER=scott
TABLES=(unit,manager)
```

- Command-Line Method

```
> imp SYSTEM/password FROMUSER=blake TOUSER=scott
  FILE=blake.dmp -
TABLES=(unit,manager)
```

Example Import Sessions

In this example, a DBA imports all tables belonging to `scott` into user `blake's` account.

- Parameter File Method

```
> imp SYSTEM/password PARFILE=params.dat
```

The params.dat file contains the following information:

```
FILE=scott.dmp
FROMUSER=scott
TOUSER=blake
TABLES=(*)
```

- Command-Line Method

```
> imp SYSTEM/password FILE=scott.dmp FROMUSER=scott
  TOUSER=blake TABLES=(*)
```

Example Import Sessions

In this example, pattern matching is used to import various tables for user `scott`.

- Parameter File Method

```
imp SYSTEM/password PARFILE=params.dat
```

The params.dat file contains the following information:

```
FILE=scott.dmp
IGNORE=n
GRANTS=y
ROWS=y
FROMUSER=scott
TABLES=(%d%,b%s)
```

- Command-Line Method

```
imp SYSTEM/password FROMUSER=scott FILE=scott.dmp
  TABLES=(%d%,b%s)
```