impdp exclude用法

参考 ： http://www.itpub.net/thread-602958-1-1.html

导入时指定了exclude=index, constraint
发现结果逐渐约束（索引？）没有导入，但是非空约束还是导入进来了
原因 ：
非空约束不能备excluded.

The examples below are based on:
- the demo schema SCOTT that is created with script: $ORACLE_HOME/rdbms/admin/scott.sql
- the directory object my_dir that refers to an existing directory on the server where the Oracle RDBMS is installed. Example:

Windows:
CONNECT system/manager
CREATE OR REPLACE DIRECTORY my_dir AS 'D:\export';
GRANT read,write ON DIRECTORY my_dir TO public;
Unix:
CONNECT system/manager
CREATE OR REPLACE DIRECTORY my_dir AS '/home/users/export';
GRANT read,write ON DIRECTORY my_dir TO public;
Note that when creating an export DataPump dumpfile, you have to ensure that the filename does not already exist in that directory.

The following examples show how metadata can be filtered with the EXCLUDE and INCLUDE parameters.

1. Syntax of the INCLUDE and EXCLUDE DataPump parameters.
With Metadata filters you can specify a set of objects to be included or excluded from an Export or Import operation, such as tables, indexes, grants, procedures.

EXCLUDE = object_type[:name_clause] [, ...]
INCLUDE = object_type[:name_clause] [, ...]
Examples:

expdp <other_parameters> SCHEMAS=scott EXCLUDE=SEQUENCE, TABLE:"IN ('EMP', 'DEPT')"
impdp <other_parameters> SCHEMAS=scott INCLUDE=FUNCTION, PACKAGE, PROCEDURE, TABLE:"= 'EMP'"
The name_clause (specified after the colon separator) is optional. It allows a selection of specific objects within an object type. The EXCLUDE example above will export the complete SCOTT schema, except (1) the sequences owned by SCOTT and (2) the tables EMP and DEPT (with their dependent objects).
The INCLUDE example above will only import the functions, procedures, and packages that are owned by SCOTT, and will also import the table EMP (with its dependent objects).

A different kind of filtering is Data filtering. Data filtering is implemented through the QUERY and SAMPLE parameters, which specify restrictions on the table rows that are to be exported. For details, see also:
Note 277010.1 "Export/Import DataPump Parameter QUERY - How to Specify a Query"

2. SQL Operator usage.
The name_clause is a SQL expression that is used as a filter on the object names of the object. It consists of a SQL operator and the values against which the object names of the specified type are to be compared. If no name_clause is provided, all objects of the specified type are excluded/included. The name clause must be separated from the object type with a colon. Examples of operator-usage:

EXCLUDE=SEQUENCE
or:

EXCLUDE=TABLE:"IN ('EMP', 'DEPT')"
or:
EXCLUDE=INDEX:"= 'MY_INDX'"
or:
INCLUDE=PROCEDURE:"LIKE 'MY_PROC_%'"
or:
INCLUDE=TABLE:"> 'E'"

3. Double quotes and single quotes usage.
The name clause is separated from the object type with a colon. The name clause must be enclosed in double quotation marks. The single-quotation marks are required to delimit the name strings. Using the INCLUDE or EXCLUDE parameter in a parameter file is the preferred method.

Parameter file: exp.par

DIRECTORY = my_dir
DUMPFILE = exp_tab.dmp
LOGFILE   = exp_tab.log
SCHEMAS   = scott
INCLUDE   = TABLE:"IN ('EMP', 'DEPT')"

expdp system/manager parfile=exp.par
To run this job without a parameter file, you need to escape the special characters. Incorrect escaping can result in errors such as: ksh: syntax error: '(' unexpected.
Command line examples (for Windows: type parameters on one single line) :

Windows:
D:\> expdp system/manager DIRECTORY=my_dir DUMPFILE=exp_tab.dmp LOGFILE=exp_tab.log
SCHEMAS=scott INCLUDE=TABLE:\"IN ('EMP', 'DEP')\"
Unix:
% expdp system/manager DIRECTORY=my_dir DUMPFILE=exp_tab.dmp LOGFILE=exp_tab.log \
SCHEMAS=scott INCLUDE=TABLE:\"IN \(\'EMP\', \'DEP\'\)\"

4. Pay special attention when the same filter name for an object type is used more than once.
If multiple filters are specified for an object type, an implicit AND operation is applied to them. That is, the objects that are exported or imported during the job have passed all of the filters applied to their object types.

Incorrect syntax (no tables are exported; error: ORA-31655):
INCLUDE=TABLE:"= 'EMP'"
INCLUDE=TABLE:"= 'DEPT'"

Correct syntax:
INCLUDE=TABLE:"IN ('EMP', 'DEPT')"

or (all tables that have an 'E' and a 'P' in their name):
INCLUDE=TABLE:"LIKE '%E%'"
INCLUDE=TABLE:"LIKE '%P%'"

5. The EXCLUDE and INCLUDE parameters are mutually exclusive.
It is not possible to specify both the INCLUDE parameter and the EXCLUDE parameter in the same job.

Incorrect syntax (error: UDE-00011):
INCLUDE=TABLE:"IN ('EMP', 'DEPT')"
EXCLUDE=INDEX:"= 'PK_EMP'"

Correct syntax:
INCLUDE=TABLE:"IN ('EMP', 'DEPT')"

6. The object types that can be specified, depend on the export/import DataPump mode.
During a TABLE level export/import, certain object types that are directly related to SCHEMA or DATABASE level jobs, cannot be specified. The same applies to a SCHEMA level export/import where no DATABASE level object types can be specified.

Example (incorrect spelling of object type USERS (should be: USER); error: ORA-39041):

```
DIRECTORY = my_dir
DUMPFILE  = exp_tab.dmp
LOGFILE   = exp_tab.log
TABLES    = scott.emp
INCLUDE   = USERS:"= 'SCOTT'", TABLESPACE_QUOTA, SYSTEM_GRANT, ROLE_GRANT, DEFAULT_ROLE
```

Example (incorrect usage of object types in INCLUDE parameter for a TABLE level export; error: ORA-39038):

```
DIRECTORY = my_dir
DUMPFILE  = exp_tab.dmp
LOGFILE   = exp_tab.log
TABLES    = scott.emp
INCLUDE   = USER:"= 'SCOTT'", TABLESPACE_QUOTA, SYSTEM_GRANT, ROLE_GRANT, DEFAULT_ROLE
```

Corrected parameters (run job in schema mode):

```
DIRECTORY = my_dir
DUMPFILE  = exp_tab.dmp
LOGFILE   = exp_tab.log
SCHEMAS   = scott
INCLUDE   = USER:"= 'SCOTT'", TABLESPACE_QUOTA, SYSTEM_GRANT, ROLE_GRANT, DEFAULT_ROLE
INCLUDE   = TABLE:"= 'EMP'"
```

To determine the name of the object types can be specified with EXCLUDE and INCLUDE, you can run the following query:

```
SET lines 200 pages 20000
COL object_path FOR a60
COL comments FOR a110

-- for database level export/import:
SELECT named, object_path, comments
  FROM database_export_objects
WHERE object_path NOT LIKE '%/%';

-- for table schema export/import:
SELECT named, object_path, comments
  FROM schema_export_objects
WHERE object_path NOT LIKE '%/%';

-- for table level export/import:
SELECT named, object_path, comments
  FROM table_export_objects
WHERE object_path NOT LIKE '%/%';
```

7. Only specific object types can be named with a Name clause.
The name clause applies only to object types whose instances have names (for example, it is applicable to TABLE, but not to GRANT).
To determine which object types can be named, you can run the following query:

```
SET lines 150 PAGES 20000
COL object_path FOR a30
COL comments FOR a110

-- for database level export/import:
SELECT named, object_path, comments
```

```
  FROM database_export_objects
WHERE named='Y';

-- for table schema export/import:
SELECT named, object_path, comments
  FROM schema_export_objects
WHERE named='Y';

-- for table level export/import:
SELECT named, object_path, comments
  FROM table_export_objects
WHERE named='Y';

N OBJECT_PATH                  COMMENTS
- ----------------------------- ------------------------------------------------
Y CONSTRAINT                   Constraints (including referential constraints)
Y INDEX                        Indexes
Y PROCDEPOBJ                   Instance procedural objects
Y REF_CONSTRAINT               Referential constraints
Y TRIGGER                      Triggers on the selected tables
```

Note that the object type TABLE is not listed here because this is the query output of the TABLE_EXPORT_OBJECTS view: the tables are already specified with the TABLES parameter in the DataPump job.

Import DataPump example:

```
DIRECTORY = my_dir
DUMPFILE  = exp_tab.dmp
LOGFILE   = exp_tab.log
TABLES    = scott.emp
EXCLUDE   = TRIGGER:"IN ('TRIG1', 'TRIG2')", INDEX:"= 'INDX1'", REF_CONSTRAINT
```

8. Excluding/Including an object, will also exclude/include it's dependent objects.
Dependent objects of an identified object are processed along with the identified object. For example, if a filter specifies that an index is to be included in an operation, then statistics from that index will also be included. Likewise, if a table is excluded by a filter, then indexes, constraints, grants, and triggers upon the table will also be excluded by the filter.

To determine which objects are dependent, e.g. for a TABLE, you can run the following query (in Oracle10g Release 2 and higher):

```
SET lines 200 pages 20000
COL object_path FOR a60
COL comments FOR a110

-- for TABLE dependent object types (10.2.0.x only):
SELECT named, object_path, comments
  FROM database_export_objects
WHERE object_path LIKE 'TABLE/%';

N OBJECT_PATH                              COMMENTS
- ----------------------------------------- ------------------------------------------------
  TABLE/AUDIT_OBJ                           Object audits on the selected tables
  TABLE/COMMENT                             Table and column comments on the selected tables
  TABLE/CONSTRAINT                          Constraints (including referential constraints)
  TABLE/CONSTRAINT/REF_CONSTRAINT           Referential constraints
  TABLE/FGA_POLICY                          Fine-grained auditing policies
  TABLE/GRANT                               Object grants on the selected tables
  TABLE/GRANT/OWNER_GRANT/OBJECT_GRANT      Object grants on the selected tables
  TABLE/INDEX                               Indexes
  TABLE/INDEX/STATISTICS                    Precomputed statistics
```

TABLE/INSTANCE_CALLOUT	Instance callouts
TABLE/MATERIALIZED_VIEW_LOG	Materialized view logs
TABLE/POST_INSTANCE/GRANT/PROCDEPOBJ_GRANT	Grants on instance procedural objects
TABLE/POST_INSTANCE/PROCDEPOBJ	Instance procedural objects
TABLE/POST_INSTANCE/PROCDEPOBJ_AUDIT	Audits on instance procedural objects
TABLE/POST_TABLE_ACTION	Post-table actions
TABLE/PRE_TABLE_ACTION	Pre-table actions
TABLE/PROCACT_INSTANCE	Instance procedural actions
TABLE/RLS_CONTEXT	Fine-grained access control contexts
TABLE/RLS_GROUP	Fine-grained access control policy groups
TABLE/RLS_POLICY	Fine-grained access control policies
TABLE/TRIGGER	Triggers

9. Excluding objects during an Export or Import DataPump job.
When specifying the EXCLUDE parameter for en Export DataPump or Import DataPump job, all object types for the given mode of export/import (like schema mode) will be included, except those specified in an EXCLUDE statement. If an object is excluded, all of its dependent objects are also excluded. For example, excluding a table will also exclude all indexes and triggers on the table.

9.1. Excluding Constraints.
The following constraints cannot be excluded:
- NOT NULL constraints.
- Constraints needed for the table to be created and loaded successfully (for example, primary key constraints for index-organized tables or REF SCOPE and WITH ROWID constraints for tables with REF columns).

This means that the following EXCLUDE statements will be interpreted as follows:
- EXCLUDE=CONSTRAINT will exclude all nonreferential constraints, except for NOT NULL constraints and any constraints needed for successful table creation and loading.
- EXCLUDE=REF_CONSTRAINT will exclude referential integrity (foreign key) constraints.

9.2. Excluding Grants.
Specifying EXCLUDE=GRANT excludes object grants on all object types and system privilege grants.

9.3. Excluding Users.
Specifying EXCLUDE=USER excludes only the definitions of users, not the objects contained within users' schemas. To exclude a specific user and all objects of that user, specify a filter such as the following (where SCOTT is the schema name of the user you want to exclude):

EXCLUDE=SCHEMA:"= 'SCOTT'"
If you try to exclude a user by using a statement such as EXCLUDE=USER:"= 'SCOTT'", only the CREATE USER scott DDL statement will be excluded, and you may not get the results you expect.

10. Including objects during an Export or Import DataPump job.
When specifying the INCLUDE parameter for en Export DataPump or Import DataPump job, only object types explicitly specified in INCLUDE statements (and their dependent objects) are exported/imported. No other object types, such as the schema definition information that is normally part of a schema-mode export when you have the EXP_FULL_DATABASE role, are exported/imported.