MANAGING REFERENTIAL INTEGRITY CONSTRAINTS

Elizabeth Boss

INTRODUCTION

AGENDA

PRIMARY KEY CONSTRAINTS
FOREIGN KEY CONSTRAINTS
ENABLING CONSTRAINTS
DISABLING CONSTRAINTS
DROPPING CONSTRAINTS
DATA DICTIONARY VIEWS
Q&A

OBJECTIVES

- To provide an understanding of the relationships that ORACLE enforces with regards to primary key/foreign key constraints.
- Describe script files that will provide information concerning constraints that are currently being enforced.
- Provide sample script files for enabling, disabling, and dropping parent and referential integrity constraints.

PRIMARY KEYS

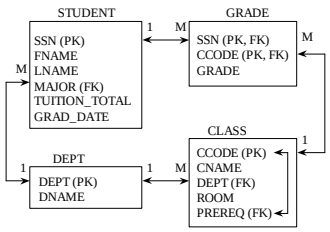
- A primary key is defined as the column, or columns, that serve to uniquely identify a row in a table.
- By definition, a primary key cannot have null values or duplicate values.
- When selecting a primary key:
  - Make sure the column values will never be null.
  - Remember that values must remain unique.
  - Do not use columns whose values can change.
  - Use short numeric or alphanumeric values (sequences are a great idea).
  - Try to avoid composite primary keys.

FOREIGN KEYS

- Foreign key constraints are used to enforce referential integrity rules.
- A foreign key is a column in a table that is a primary or unique key in some other table.
- Foreign keys are used to show the relationships between tables in a relational database, normally in a parent-child or master-detail fashion.

REFERENTIAL RELATIONSHIPS

- The following diagram depicts referential relationships between several sets of tables.



FOREIGN KEY RULES

- For each unique value in a column in the parent table (parent key), there could be one or more corresponding values in the child table column (foreign key).
- Referential integrity says that every value in a foreign key column must exist as a value in the parent key or be null.
- When the parent key exists in the same table as the foreign key, this is called self-referential integrity.
- To enforce the disallowing of partially null foreign keys, a constraint would also have to be defined allowing all nulls or no nulls.

FOREIGN KEY RULES Contd.

- ORACLE7 supports the following referential actions:
  - Update and Delete Restrict will not permit a parent key value to be deleted or changed if that value is referenced by a foreign key.
  - Updates and deletes to child values (foreign key) are permitted as long as the value exists in the parent key.
  - Delete Cascade deletes all child records when the corresponding parent value is deleted.
  - Inserts into the child table are only permitted if the foreign key value exists in the parent table.

PRIMARY KEY CONSTRAINTS

- A primary key constraint can be specified when a table is created, as the following example shows:

```
CREATE TABLE dept (dept   varchar2(4) CONSTRAINT dept_pk
                                      PRIMARY KEY,
                   dname  varchar2(30));
```

- Composite primary keys must always be defined as a table level constraint:

```
CREATE TABLE grade (ssn   char(9),
                    ccode char(4),
                    grade varchar2(2),
                    CONSTRAINT grade_pk
                    PRIMARY KEY (ssn,ccode));
```

PRIMARY KEY CONSTRAINTS Contd.

- The Primary Key constraint creates an implicit index, and must adhere to the same limitations as an index.
  - There is a 16-column limit for a composite primary key.
  - Total byte size of the length of the defined column(s) cannot exceed half the block size defined for the database.
- The name of the index will be the same as the constraint name.
  - If an index exists and the CONSTRAINT keyword is omitted, the existing index name is used.
- Tablespace and storage parameter assignment is accomplished through the USING INDEX option of the ENABLE clause.

```
CREATE TABLE class (ccode char(4) CONSTRAINT class_pk
                                  PRIMARY KEY,
                    cname varchar2(30),
                    dept  varchar2(4),
                    prereqchar(4))
                    ENABLE PRIMARY KEY USING
                    INDEX TABLESPACE index_ts;
```


ADDING PRIMARY KEY CONSTRAINTS

- Primary Key constraints can also be added to a table after the table has been created:

```
ALTER TABLE student ADD CONSTRAINT student_pk
  PRIMARY KEY (ssn);
```

- Whenever a primary key constraint is defined, it must be enabled; if there are no exceptions found, it is enabled automatically at creation.
- If an exception is found, an error is returned, and the constraint will not be enabled.
  - The statement will also be rolled back.
- To locate rows causing exceptions, a table for logging errors must be created.
  - All exceptions must be updated or removed for the ALTER TABLE statement to complete successfully and add the constraint.

IDENTIFYING EXCEPTIONS

- A script, utlexcpt.sql, is supplied by ORACLE to create a table called exceptions.

```
CREATE TABLE exceptions(row_id rowid,
                        owner varchar2(30),
                        table_name varchar2(30),
                        constraint varchar2(30));
```

  The script can be copied, and modified, changing the table name to a user-defined name.
  - The columns must remain the same, as the EXCEPTIONS statement inserts values into those pre-defined columns.
- Issue the EXCEPTIONS option and indicate the name of the exception table.

```
ALTER TABLE dept ADD CONSTRAINT dept_pk PRIMARY KEY
  EXCEPTIONS INTO exceptions;
```

IDENTIFYING EXCEPTIONS Contd.

- A query that finds the actual rows causing the problem can be created by joining the exceptions table to the table in question.
  - Include a WHERE clause comparing the rowids.
  - If multiple constraints are being enabled, supply the constraint name in the WHERE clause.

```
SELECT dept, dname
  FROM dept, exceptions
  WHERE dept.rowid = exceptions.rowid
  AND exceptions.constraint = 'DEPT_PK';
```


IDENTIFYING PRIMARY KEY CONSTRAINTS

- The data dictionary tables dba/all/user_constraints, and dba/all/user_cons_columns can be used to identify primary key constraints that have been created for a specific table or group of tables.
-

```
column constraint_name format a16
column table_name format a16

select cons.table_name,
       cols.column_name,
       cons.constraint_name,
       decode(constraint_type,'P','Primary Key'),
       status
from user_constraints cons,
user_cons_columns cols
where cons.constraint_name = cols.constraint_name
and   constraint_type = 'P'
order by table_name;
```


IDENTIFYING PRIMARY KEY INDEXES

- When a primary key constraint is created, a unique index is created using the same name as the constraint.
- The following query displays indexes created on behalf of primary key constraints:

```
select indx.table_name,
       indx.index_name
  from user_indexes indx,
       user_constraints cons
  where indx.index_name = cons.constraint_name
  order by table_name,
           index_name;
```

FOREIGN KEY CONSTRAINTS

- Foreign Key constraints can be created at the time the table is created, as the following example shows:

```
CREATE TABLE grade (ssn       char(9),
                    ccode     char(4),
                    grade     varchar2(2),
                    CONSTRAINT grade_pk
                    PRIMARY KEY (ssn,ccode),
                    CONSTRAINT ssn_fk
                    FOREIGN KEY (ssn) REFERENCES student(ssn));
```

- ORACLE assumes the parent key is the primary key for the specified table unless otherwise indicated.
- When a column other than a primary key is the parent key, the name of the column must be specified in parentheses following the name of the parent table.
- ORACLE will check to make sure that the parent key is defined as UNIQUE or as a PRIMARY KEY; if not an error will be returned.

FOREIGN KEY CONSTRAINTS Contd.

- Foreign Key constraints can also be defined at the column level:

```
CREATE TABLE class (ccode char(4),
                    cname varchar2(30),
                    dept  varchar2(4) CONSTRAINT dept_fk
                    REFERENCES dept(dept),
                    prereq char(4));
```

- When the constraint is specified at the column level the key words FOREIGN KEY are not required.
- The ON DELETE CASCADE clause specifies that if a value is removed from the parent table, all matching rows in the child table should also be removed.
-

```
CREATE TABLE grade (ssn char(9),
                    ccode char(4),
                    grade varchar2(2),
                    CONSTRAINT ssn_fk FOREIGN KEY (ssn)
                    REFERENCES student(ssn)
                    ON DELETE CASCADE);
```


IDENTIFYING FOREIGN KEY CONSTRAINTS

- The following query displays a list of all foreign key constraints that have been defined for tables.

```
column constraint_name format a16
  column table_name format a16
  column column_name format a16
  column r_owner format a16
  column r_constraint_name format a16

select cons.table_name,
       cols.column_name,
       cons.constraint_name,
       decode(constraint_type,'R','Foreign Key'),
       r_owner,
       r_constraint_name,
       delete_rule,
       status
  from user_constraints cons,
       user_cons_columns cols
  where cons.constraint_name = cols.constraint_name(+)
  and   constraint_type = 'R'
  order by table_name;
```

NON-INDEXED FOREIGN KEY CONSTRAINTS

- The following query displays foreign key constraints that do not have an index created for them.

```
column table_column format a20 heading 'Needs Index'
  column constraint_name heading 'Constraint'
  column ref_table format a22 heading 'Referencing This Table'

select cons.constraint_name,
       cons.table_name||'.'||cols.column_name table_column,
       ref.table_name ref_table
  from user_constraints cons,
       user_cons_columns cols,
       user_constraints ref
  where cons.r_constraint_name = ref.constraint_name
  and   cons.table_name = cols.table_name
  and   cons.constraint_name = cols.constraint_name
  and not exists (select 'x' from user_ind_columns ind
                  where ind.table_name = cols.table_name
                  and   ind.column_name = cols.column_name
                  and   ind.column_position <> 1)
  order by table_column;
```


DISABLING CONSTRAINTS

- For the most part, constraints should remain enabled, but some operations can experience performance gains by temporarily turning them off.
- Disabling constraints can be particularly useful when:
  - Using SQL*Loader for large table loads.
  - Performing batch operations with many DML commands.
  - Exporting or importing one table.
- Constraints can be disabled using an ALTER TABLE command.

```
ALTER TABLE student DISABLE CONSTRAINT major_fk;
  ALTER TABLE dept DISABLE PRIMARY KEY;
```

- Any column with a PRIMARY KEY constraint cannot be disabled unless all dependent FOREIGN KEY constraints have also been disabled.
- When a Primary Key constraint is disabled, the associated index is dropped.

DISABLING CONSTRAINTS Contd.

- To determine constraints that refer to other constraints the following query can be used:

```
column constraint_name format a16
  column table_name format a16
  column r_owner format a16
  column r_constraint_name format a16
  column column_name format a16

  select cons.table_name,
         cols.column_name,
         cons.constraint_name,
         cons.r_owner,
         cons.r_constraint_name,
         refer.table_name
  from user_constraints cons,
       user_cons_columns cols,
       user_constraints refer
  where cons.constraint_name = cols.constraint_name
  and   cons.r_constraint_name = refer.constraint_name
  and   cons.constraint_type <> 'C'
  order by cons.table_name;
```

DISABLING CONSTRAINTS Contd.

- The following PL/SQL script provides an easy method for disabling all constraints for a specific table.

```
drop table disable_table;
  create table disable_table(disable_command varchar2(123));
  drop table original_table;
  create table original_table(disable_command varchar2(123));
  begin
  declare
     current_table user_constraints.table_name%type := '&tname';
     cursor get_constraint_cursor is
        select constraint_name
        from user_constraints
        where table_name = current_table;
     cursor disable_cursor (current_constraint char) is
        select constraint_name, table_name,owner
        from user_constraints
        where r_constraint_name = current_constraint;
```


DISABLING CONSTRAINTS Contd.

```
begin
      for get_constraint_rec in get_constraint_cursor loop
         insert into original_table values
         ('ALTER TABLE ' ||current_table
          ||' DISABLE CONSTRAINT '
          ||get_constraint_rec.constraint_name||';');
         for disable_rec in disable_cursor
            (get_constraint_rec.constraint_name) loop
            insert into disable_table values
            ('ALTER TABLE '||disable_rec.owner||'.'
             ||disable_rec.table_name||' DISABLE CONSTRAINT '
             ||disable_rec.constraint_name||';');
         end loop;
      end loop;
   end;
end;
.
/
```


DISABLING CONSTRAINTS Contd.

```
set heading off
set feedback off
spool disable_it.sql
select * from disable_table;
select * from original_table;
spool off
set heading on
set feedback on
@disable_it
```

- The advantage to disabling a constraint is that the definition for the constraint still exists in the data dictionary, and therefore, the constraint can be re-enabled.
- When a foreign key constraint is dropped, the associated index remains intact, and does not need to be re-built when the constraint is re-enabled.

DROPPING CONSTRAINTS

- If a constraint is no longer required it can be dropped using the DROP CONSTRAINT clause of the ALTER TABLE command.

```
ALTER TABLE student DROP CONSTRAINT major_fk;
ALTER TABLE dept DROP PRIMARY KEY;
```

- When a primary key constraint is dropped, the associated index is also dropped.
- Dropping Foreign key constraints, on the other hand, does not cause the associated index to be dropped.
- The disadvantage of dropping a constraint, is that the definition for the constraint is removed from the data dictionary and it cannot be enabled at a later time.
- To enforce the constraint again, the constraint would have to be added to the table using the ADD CONSTRAINT clause of the ALTER TABLE command.

IDENTIFYING FOREIGN KEY INDEXES

- The following query can be used to obtain a list of indexes that have been created (explicitly) on behalf of foreign key constraints.

```
column table_column format a20 heading 'Available Index'
column constraint_name heading 'Constraint'

select cons.constraint_name,
       cols.table_name||'.'||cols.column_name table_column
from user_constraints cons,
     user_cons_columns cols,
     user_constraints ref
where cons.r_constraint_name = ref.constraint_name
and   cons.table_name = cols.table_name
and   cons.constraint_name = cols.constraint_name
and exists (select 'x' from user_ind_columns ind
            where ind.table_name = cols.table_name
            and   ind.column_name = cols.column_name
            and   ind.column_position = 1)
order by table_column;
```


ENABLING CONSTRAINTS

- A constraint that has been disabled can be re-enabled using the ENABLE CONSTRAINT clause of the ALTER TABLE command.

```
ALTER TABLE student ENABLE CONSTRAINT major_fk;
ALTER TABLE dept ENABLE PRIMARY KEY;
```

- This command requires that the constraint names (other than primary key constraints) be known.
- The following script provides the names of primary/foreign key constraints that have been disabled.

```
select table_name,
       constraint_name,
       decode(constraint_type,'P','Primary Key',
                              'R','Foreign Key')TYPE,
       status
  from user_constraints
  where status = 'DISABLED'
  and   constraint_type <> 'C';
```

ENABLING CONSTRAINTS Contd.

- The following script file enables the primary key constraint and referential integrity constraints for the table name provided.
  - Prior to enabling any foreign key constraints, a check is performed to ensure that the associated primary key is enabled.
  - If the parent key is not enabled, it will be enabled prior to enabling the foreign key.

```
set serveroutput on
  set feedback off
  drop table pk_table;
  create table pk_table(enable_command varchar2(200));
  drop table enable_table;
  create table enable_table(enable_command varchar2(200));

  declare
     current_table user_constraints.table_name%type := '&TABLE';
     pk_status     user_constraints.status%type;
```


ENABLING CONSTRAINTS Contd.

```
cursor enable_cons is
  select cons.constraint_name cons_name,
         cons.table_name table_name,
         cons.owner owner,
         cons.r_owner r_owner,
         cons.r_constraint_name r_constraint_name,
         ref.owner ref_owner,
         ref.table_name ref_table,
         ref.constraint_type ref_type,
         ref.status status
  from user_constraints cons,
       user_constraints ref
  where cons.r_constraint_name = ref.constraint_name
  and   cons.table_name = current_table
  and   cons.status = 'DISABLED'
  and   cons.constraint_type = 'R';
```


ENABLING CONSTRAINTS Contd.

```
begin
  select status into pk_status
  from user_constraints
  where table_name = current_table
  and constraint_type = 'P';

  if pk_status = 'DISABLED' then
     insert into pk_table values
     ('ALTER TABLE '||current_table||' ENABLE PRIMARY KEY'
      ||';');
  else dbms_output.put_line('Primary key is already enabled');
  end if;

  for enable_rec in enable_cons loop
     if enable_rec.ref_type = 'P'
      and enable_rec.status = 'DISABLED' then
        insert into pk_table values
           ('ALTER TABLE '||enable_rec.r_owner||'.'
            ||enable_rec.ref_table
            ||' ENABLE PRIMARY KEY'||';');
     end if;
```

ENABLING CONSTRAINTS Contd.

```
          insert into enable_table values
            ('ALTER TABLE '||enable_rec.owner||'.'
            ||enable_rec.table_name
            ||' ENABLE CONSTRAINT '||enable_rec.cons_name||';');
         end loop;
      exception when no_data_found then
         dbms_output.put_line('The table name entered is invalid');
      end;
      .
      /
      set heading off
      set feedback off
      spool enableit.sql
      select * from pk_table;
      select * from enable_table;
      spool off
      set heading on
      set feedback on
      @enableit
```


SUMMARY

- Referential integrity allows the enforcement of master/detail or parent/child relationships between tables.
- There will be occasions when it is more efficient to disable those referential integrity constraints, and not impose the overhead of checking the constraints.
- The ORACLE data dictionary provides several tables that are useful in managing referential integrity constraints including:
  - user_constraints
  - user_cons_columns
  - user_indexes
  - user_ind_columns
- The script files discussed in this presentation provide a basic tool-set for managing referential integrity constraints.