CS2312: Integrity Constraints in Oracle 8i

Carole Goble

1. Integrity Constraints

Integrity constraints can take the form of primary keys, foreign keys, check constraints and null constraints. Primary keys are automatically defined to be NOT NULL.

Constraints are of the form:

```
        constraint constraint_name primary key (attribute_name, ...)

        constraint constraint_name foreign key (child_attribute, ...)
              references tablename(parent_attribute, ...)

        constraint constraint_name not null
```

Note:

- The ‘constraint *constraint_name*’ part is optional but it is a good idea to name constraints (see below).
- When using column constraints (rather than table constraints) for foreign keys, the ‘foreign key (*child_attribute*)’ keywords should not be used (see below for Table and Column constraints).
- When a foreign key references a table’s primary key the (*parent_attribute* ...) part is optional after the tablename. It is a good idea to include the attributes referenced for clarity.

See the Oracle8 server manual entry for CONSTRAINT clause for a complete syntax summary. The page is at:

http://aardvark.cs.man.ac.uk:8899/ora_803_doc/DOC/dcommon/oin/index.htm

Here are some rules for defining a table’s primary and foreign key constraints:

1.1 Naming Constraints

1.1.1 Choosing Names

Constraints may or may not be named. It is a good idea to name at least primary and foreign keys. Oracle maintains a list of constraints defined on your tables in the Data Dictionary relations USER_CONS_COLUMNS and USER_CONSTRAINTS. If a constraint is not named then a system name is created. When a constraint violation occurs, Oracle reports the name of the constraint that was violated. User named constraints are much easier to identify than the obscure system names provided.

A good naming scheme would be to start all primary key constraint names with ‘pk_’, all foreign key constraint names with ‘fk_’ and all Not Null constraint names with ‘nn_’. The name of the attribute being used or the name of the table can then make up the complete constraint name. This makes identifying constraints in USER_CONS_COLUMNS much easier.

1.1.2 Data Dictionary Relations

USER_CONS_COLUMNS provides information about the names of the constraints and the attributes they are defined on. USER_CONSTRAINTS provides more detailed information including the type of constraint and the deletion method used, where appropriate. When using the SQL*Loader, if the test data does not satisfy a particular constraint, the constraint name is reported.

1.1.3 Unique Names

Constraint names must be unique throughout all the tables in a database. A constraint name used in one table may not be used in any other table. It is often easy to forget this when using identical foreign keys in several tables.
For example, suppose a table named COURSE is the target of foreign keys in tables named STAFF and STUDENT. The foreign key constraints must have different names even though they may refer to the same attribute in COURSE. If no constraint name is given Oracle will assign a unique system name.

1.2 Table Constraints and Column Constraints

1.2.1 Choosing Table or Column Constraints

Constraints may be either table constraints or column constraints. A column constraint is defined for an attribute along with the attribute's data type. A table constraint is defined for a named column before or after the column has been defined. There is not much difference between them. However, when a primary key or foreign key is made up of more than one attribute, a table constraint must be used. Since column constraints only refer to the current column they cannot be used for multi-attribute keys.

For example, the following two table definitions will give the same results.

```
create table year
 (yearno number(8) constraint year_pk1 primary key,
  yeartutorid number(4),
      constraint fk_tut foreign key (yeartutorid) references
      STAFF(staffid));

create table year
 (yearno number(8),
  constraint year_pk1 primary key (yearno),
  yeartutorid number(4) constraint fk_tut references
      STAFF(staffid));
```

The first definition uses a column constraint to specify the primary key and a table constraint to specify the foreign key.

The second definition uses a column constraint to specify the foreign key and a table constraint to specify the primary key. Notice that when a column constraint is used to define a foreign key, the keywords 'foreign key' are not required since Oracle knows that a foreign key is being defined. It is often only the placing of commas (,) that determine whether a constraint is a table constraint or a column constraint.

1.2.2 Using only Table Constraints or only Column Constraints

You could use only column constraints when specifying a table **provided the constraints are defined for single attributes only.** You could also use only table constraints. The next example can only be achieved using a table constraint for the primary key:

```
create table TEACH
 (courseno char(5),
  lecturid number(4),
  constraint pk_teach primary key (courseno, lecturid),
  constraint fk_teach foreign key (courseno) references
      COURSE,
  constraint fk_lect foreign key (lecturid) references
      STAFF(staffid));
```

Notice:

- The primary key is made up of two attributes. The only way to specify the two attributes to be the primary key is with a table constraint. The foreign keys have also been specified by table constraints in this case but since they are only made up of single attributes they could, in fact, have been defined with column constraints.

- The 'fk_teach' foreign key does not specify which attribute in the target table (course) is to be referenced. When no atribute is specified Oracle assumes the primary key is to be referenced. The 'fk_lect' foreign key does specify an attribute. In this case it also happens to be the primary key. It is a good idea to include the fields to be referenced, even though they may not strictly be need, for clarity.

- Indexed attributes as well as primary key atributes may be referenced by a foreign key. If an index attribute is used then its name will have to be specified.

- The referenced attribute must be of the same data type as the attribute which defines the foreign key, but not necessarily having the same name. It must also be indexed, either explicitly or by it being a primary key.

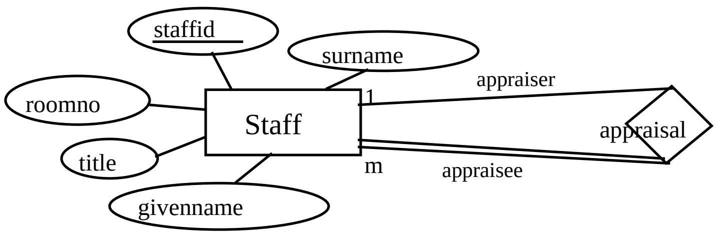
1.2.3 Which style to use

A good style to use when defining constraints is:

- Use column constraints for primary keys and 'not nulls' when each constraint is made up of a single attribute. A table constraint must be used otherwise.
- Use table constraints for all foreign keys after all the attributes have been defined. This makes the definition easier to read and edit.
- Name primary key constraints and foreign key constraints. You could also name 'not null' constraints.

1.3 Altering, Disabling and Enabling Constraints

A constraint may be disabled at the time the table is created and then explicitly activated by the owner at a later time. This is useful for recursive relationships where a foreign key refers to the primary key of its own relation. Loading data would be impossible if the foreign key could not be 'switched off' initially and then 'switched on' after the table has been populated. For example, consider:



The following table definition has the recursive relationship implemented as a foreign key defined on the 'appraiserid' attribute:

```
create table staff
 (staffid number(4) primary key,
  givenname char(20),
  surname char(20),
  title char(4)
      check (title in ('mrs', 'mr', 'ms','prof','rdr','dr')),
  roomno char(6),
  appraiserid number(4),
      constraint app_fk foreign key (appraiserid) references
      staff(staffid) disable);
```

Notice the ‘disable’ keyword in the foreign key definition. The foreign key on ‘appraiserid’ requires that an appraiser’s id must be a staff id of some other member of staff, assuming that nobody is their own appraiser.

Consider the contents of the table:

```
SQL> select * from staff;

STAFFID GIVENNAME      SURNAME         TITL ROOMNO APPRAISERID
------- -------------- --------------- ---- ------ -----------
      1 john           latham            dr   2.99           4
      2 carole         goble             ms   2.82          11
      3 graham         gough             dr   2.105          1
      4 vicky          bush              mrs  2.46          11
      5 dick           zobel             dr   2.36           6
      6 ian            watson            prof IT417          1
      7 margaret       clarke            dr   IT202          6
      8 alan           knowles           dr   A1.12          9
      9 roger          hubbold           dr   2.103          6
     10 peter          jinks             mr   A1.11          5
     11 john           gurd              prof 2.127
```

Look at the first tuple. How could this be loaded into the table? The appraiserid foreign key will try to locate a tuple with staffid ‘4’. Since the first tuple is the only tuple in table when the loading of the test data begins, the SQL*Loader will report that the foreign key has been violated and that it cannot find a tuple to match. The only way to get any data into the table is to disable the foreign foreign key, populate the table and then activate the foreign key using:

```
      alter table staff enable constraint app_fk;
```

To disable a constraint use

```
      alter table staff disable constraint app_fk;
```

If you want to change a constraint definition to be on delete cascade instead of delete restrict or vice versa, then you cannot just modify the constraint. Instead you must:

a) alter the table to delete the current constraint

```
      alter table staff drop constraint app_fk;
```

b) alter the table to add a new constraint

```
      alter table staff add constraint app_fk foreign key
      (appraiserid) references staff(staffid) on delete cascade;
```

To drop a primary key and drop any foreign key reference to the primary key then use the command

```
        alter table staff primary key cascade;
```

If you don’t include the cascade the primary key won’t be dropped if a foreign key points to it.

1.4 Maintaining Referential Integrity

1.4.1 Methods of Deletion

Foreign key constraints can specify the method of deletion. If the target primary key of a foreign key is to be deleted then either cascade deletion or restricted deletion can be used (see the lecture notes on Referential Integrity).

Oracle uses restricted deletion by default. To specify cascade deletion add `cascade delete` to the foreign key definition. For example, the following table will use cascade deletion:

```
create table year
 (yearno number(8),
  yeartutorid number(4) constraint fk_tut references
      STAFF(staffid) on delete cascade),
  constraint year_pk1 primary key (yearno));
```

When a tuple is removed from the STAFF table any tuples in the YEAR table whose yeartutorid matches the removed member of staff will be deleted automatically. If the ‘cascade delete’ option was omitted then Oracle would not permit any deletion of tuples from the STAFF table whose staffid was referenced by a yeartutorid in the YEAR table.

1.5 Multiple Constraints

An attribute can have more than one constraint defined for it. Any data inserted into the table must satisfy **all** the constraints defined for an attribute. Typical examples are that an attribute is a foreign key and it cannot be null, or an attribute has two foreign keys defined for it. For example, the following table has a ‘not null’ constraint and a foreign key constraint defined on the ‘country’ attribute:

```
create table Region
 (region char(13) constraint region_pk primary key,
  landtype char(4),
  page number(3),
  scenery char(9),
  country char(8) not null,
      constraint country_fk foreign key (country) references
      Country(country));
```

The ‘not null’ constraint has not been named but it is still enforced. Any tuples inserted into this table must have a non-null country that can be found in the Country relation.

2. Creating Tables

Tables in Oracle are created with the SQL*Plus CREATE TABLE command. This allows you to specify the table name, attribute name, attribute data types and any integrity constraints you may want to be enforced.

2.1 Order of Creation, Population and Deletion

2.1.1 Creation

When defining tables with foreign key integrity constraints the order in which the tables are created is important. A foreign key will usually refer to a target primary key of *another* table (see **1.3** where this is not the case). Oracle requires that the parent table containing the target primary key already exists when the child table containing the foreign key is created. This requirement dictates the order in which the tables must be created.

You could overcome this requirement by including the 'disable' keyword in all foreign key definitions so that the foreign keys are only defined, not enforced. Once all the tables have been created you would then have to enable all the constraints (see **1.3**). This would be too painful if a large number of tables and foreign keys were to be used.

The best method is to examine the tables and work out which ones do not refer to any other tables. These should then be created first. The remaining tables can then be created, ensuring that all parent tables exist before a child table is created. For example, the Staff/Student database example can be summarised in the following way:

Student references Year and Staff.
Enrol references Student and Course.
Teach references Course and Staff.
Staff references Staff.
Course does not reference any other relation.
Year reference Staff.

So the order of creation is

> Staff and Course (either one first),
> Year and Teach (either one first),
> Student,
> Enrol

2.1.2 Population

The same order should be used when populating the database. As tuples are inserted into the tables, Oracle enforces the integrity constraints for each tuple. A parent table must be populated before the child table is populated so that Oracle can look up target values of foreign keys.

2.1.3 Deletion

Whether a table can be deleted or not depends on the method of deletion defined in the foreign key constraint definitions (see **1.4**). If restricted deletion is used (the method used by Oracle unless you explicitly use 'cascade delete') then you will not be able to delete a table that is referenced by a foreign key using the usual '`drop` *`tablename`*'. Oracle will complain about the foreign keys referencing the table. The table can be dropped, however, using a 'cascade constraint' option, i.e.

```
drop table_name cascade constraint
```

This will drop the table 'table_name' and also drops any referential integrity constraints (such as foreign keys) that were referencing *table_name*.

2.2 Using Command Files

Command files are a useful way to store the SQL commands used to create the database and the order in which the tables should be built. For example the first command below creates all the tables with their integrity constraints. The second inserts tuples into the tables. The third removes the entire database by dropping all the tables. The ‘cascade constraint’ option is used just in case there are any foreign keys present.

build.sql

```
create table Staff
  (staffid number(4) constraint pk_staffid primary key,
          ...
  )

create table Course
  (courseno char(5) constraint .....
          ...
  )
```

populate.sql

```
insert into Staff values (1,’john’,’latham’,’dr’,2.99,4);
insert into Staff values (11,’john’,’gurd’,’prof’,2.127,null);
…
```

dropdb.sql

```
drop table staff cascade constraint
drop table course cascade constraint
drop table year cascade constraint
drop table student cascade constraint
drop table enrol cascade constraint
drop table teach cascade constraint
```

These files can be used to rebuild the database from scratch if something goes horribly wrong.