Created by Ahsan Arif

Foreign Keys with "set null on delete"

A **foreign key** means that values in one table must also appear in another table.

The referenced table is called the **parent table** while the table with the foreign key is called the **child table**. The foreign key in the child table will generally reference a primary key in the parent table.

A foreign key with a "set null on delete" means that if a record in the parent table is deleted, then the corresponding records in the child table will have the foreign key fields set to null. The records in the child table will **not** be deleted.

A foreign key with a "set null on delete" can be defined in either a CREATE TABLE statement or an ALTER TABLE statement.

Using a CREATE TABLE statement

The syntax for creating a foreign key using a CREATE TABLE statement is:

```
CREATE TABLE table_name
(column1 datatype null/not null,
column2 datatype null/not null,
...
CONSTRAINT fk_column
  FOREIGN KEY (column1, column2, ... column_n)
  REFERENCES parent_table (column1, column2, ... column_n)
  ON DELETE SET NULL
);
```

For example:

```
CREATE TABLE supplier
( supplier_id      numeric(10)  not null,
  supplier_name varchar2(50)  not null,
  contact_name  varchar2(50),
  CONSTRAINT supplier_pk PRIMARY KEY
  (supplier_id)
);
```

```
CREATE TABLE products
( product_id      numeric(10)  not null,
  supplier_id     numeric(10),
  CONSTRAINT fk_supplier
    FOREIGN KEY (supplier_id)
   REFERENCES supplier(supplier_id)
   ON DELETE SET NULL
);
```

In this example, we've created a primary key on the supplier table called *supplier_pk*. It consists of only one field - the supplier_id field. Then we've created a foreign key called *fk_supplier* on the products table that references the supplier table based on the supplier_id field.

Because of the set null on delete, when a record in the supplier table is deleted, all corresponding records in the products table will have the supplier_id values set to null.

We could also create a foreign key "set null on delete" with more than one field as in the example below:

```
CREATE TABLE supplier
( supplier_id      numeric(10)  not null,
  supplier_name varchar2(50)  not null,
  contact_name  varchar2(50),
  CONSTRAINT supplier_pk PRIMARY KEY (supplier_id,
  supplier_name)
);

CREATE TABLE products
( product_id      numeric(10)  not null,
  supplier_id     numeric(10),
  supplier_name varchar2(50),
  CONSTRAINT fk_supplier_comp
    FOREIGN KEY (supplier_id, supplier_name)
   REFERENCES supplier(supplier_id, supplier_name)
   ON DELETE SET NULL
);
```

In this example, our foreign key called *fk_foreign_comp* references the supplier table based on two fields - the supplier_id and supplier_name fields.

The delete on the foreign key called *fk_foreign_comp* causes all corresponding records in the products table to have the supplier_id and supplier_name fields set to null when a record in the supplier table is deleted, based on supplier_id and supplier_name.

Using an ALTER TABLE statement

The syntax for creating a foreign key in an ALTER TABLE statement is:

```
ALTER TABLE table_name
add CONSTRAINT constraint_name
  FOREIGN KEY (column1, column2, ... column_n)
  REFERENCES parent_table (column1, column2, ... column_n)
  ON DELETE SET NULL;
```

For example:

```
ALTER TABLE products
add CONSTRAINT fk_supplier
  FOREIGN KEY (supplier_id)
  REFERENCES supplier(supplier_id)
  ON DELETE SET NULL;
```

In this example, we've created a foreign key "with a set null on delete" called *fk_supplier* that references the supplier table based on the supplier_id field.

We could also create a foreign key "with a set null on delete" with more than one field as in the example below:

```
ALTER TABLE products
add CONSTRAINT fk_supplier
  FOREIGN KEY (supplier_id, supplier_name)
  REFERENCES supplier(supplier_id, supplier_name)
  ON DELETE SET NULL;
```