14-1 Intro to Constraints; NOT NULL and UNIQUE Constraints

This lesson covers the following objectives:
• Define the term "constraint" as it relates to data integrity
• State when it is possible to define a constraint at the column level, and when it is possible at the table level
• State why it is important to give meaningful names to constraints
• State which data integrity rules are enforced by NOT NULL and UNIQUE constraints
• Write a CREATE TABLE statement which includes NOT NULL and UNIQUE constraints at the table and column levels
• Explain how constraints are created at the time of table creation
• Constraints are used to prevent invalid data entry into tables.

Constraints in General

• So, what exactly is a constraint?
• Think of constraints as database rules.
• All constraint definitions are stored in the data dictionary.
• Constraints prevent the deletion of a table if there are dependencies from other tables.
• Constraints enforce rules on the data whenever a row is inserted, updated, or deleted from a table.
• Constraints are important and naming them is also important.
• Although you could name a constraint "constraint_1" or "constraint_2," you'd soon find it difficult to distinguish one constraint from another and would end up redoing a lot of work.

Creating Constraints

• Recall the SQL syntax for creating a table.
• In the CREATE TABLE statement shown on the next slide, each column and its data type is defined.
• You use the CREATE TABLE statement to establish constraints for each column in the table.
• There are two different places in the CREATE TABLE statement that you can specify the constraint details:
  – At the column level next to the name and data type
  – At the table level after all the column names are listed

```
CREATE TABLE clients
        (client_number NUMBER(4),
        first_name VARCHAR2(14),
        last_name VARCHAR2(13));
```

• Note that the column-level simply refers to the area in the CREATE TABLE statement where the columns are defined.
• The table level refers to the last line in the statement below the list of individual column names.

Constraints at the Column Level

• A column-level constraint references a single column.
• To establish a column-level constraint, the constraint must be defined in the CREATE TABLE statement as part of the column definition.
• Examine the following SQL statement that establishes a column-level constraint.

```
CREATE TABLE clients
        (client_number NUMBER(4) CONSTRAINT clients_client_num_pk PRIMARY KEY,
        first_name VARCHAR2(14),
        last_name VARCHAR2(13));
```

• CREATE TABLE clients

```
CREATE TABLE clients (client_number NUMBER(4) CONSTRAINT clients_client_num_pk PRIMARY KEY,
        first_name VARCHAR2(14),
        last_name VARCHAR2(13));
```

• The name of the constraint is clients_client_num_pk.
• It enforces the business rule that the client_number is the primary key of the clients table.

Naming Constraints

• Every constraint in the database has a name. When a constraint is created, it can be given a name, such as clients_client_num_pk, or given no name, in which case the system gives the constraint a name, such as SYS_C00585417.
• A naming convention can be the combination of the tablename abbreviated and a column name abbreviated followed by the constraint abbreviation: table-name_column-name_constraint-type

• If the reserved word CONSTRAINT is used in the CREATE TABLE definition, you must give the constraint a name. Constraint names are limited to 30 characters.

Naming Constraints at the Column Level

• It is best to name constraints yourself because system-generated names are not easy to recognize.
• Look at this table definition:

```
CREATE TABLE clients
        (client_number NUMBER(4),
        last_name VARCHAR2(13),
        email VARCHAR2(80));
```

• According to our naming convention:
  – A primary key constraint on client_number would be named clients_client_number_pk
  – A not null constraint on last_name would be named clients_last_name_nn
  – A unique constraint on email address would be named clients_email_uk

```
CREATE TABLE clients
        (client_number NUMBER(4) CONSTRAINT clients_cient_num_pk PRIMARY KEY,
        last_name VARCHAR2(13) CONSTRAINT clients_last_name_nn NOT NULL,
        email VARCHAR2(80) CONSTRAINT clients_emil_uk UNIQUE);
```

Constraint Naming Example

• This example shows both a user-named constraint and a system-named constraint:
• System-named constraint:

```
CREATE TABLE clients
        (client_number NUMBER(4) CONSTRAINT clients_client_num_pk PRIMARY KEY,
        last_name VARCHAR2(13) NOT NULL,
        email VARCHAR2(80));
```

•Two constraints have been created:
  –A user-named constraint named clients_client_num_pk, to enforce the rule that client_number is the primary key
  –A system-named constraint named SYS_Cn (where n is a unique integer) to enforce the rule that last_names cannot be null.

Constraints at the Table Level

• Table-level constraints are listed separately from the column definitions in the CREATE TABLE statement.
• Table-level constraint definitions are listed after all the table columns have been defined.
• In the example shown, the unique constraint is listed last in the CREATE TABLE statement.

```
CREATE TABLE clients (
        client_number NUMBER(6) NOT NULL,
        first_name VARCHAR2(20),
        last_name VARCHAR2(20),
        phone VARCHAR2(20),
        email VARCHAR2(10) NOT NULL,
CONSTRAINT clients_phone_email_uk UNIQUE (email,phone));
```

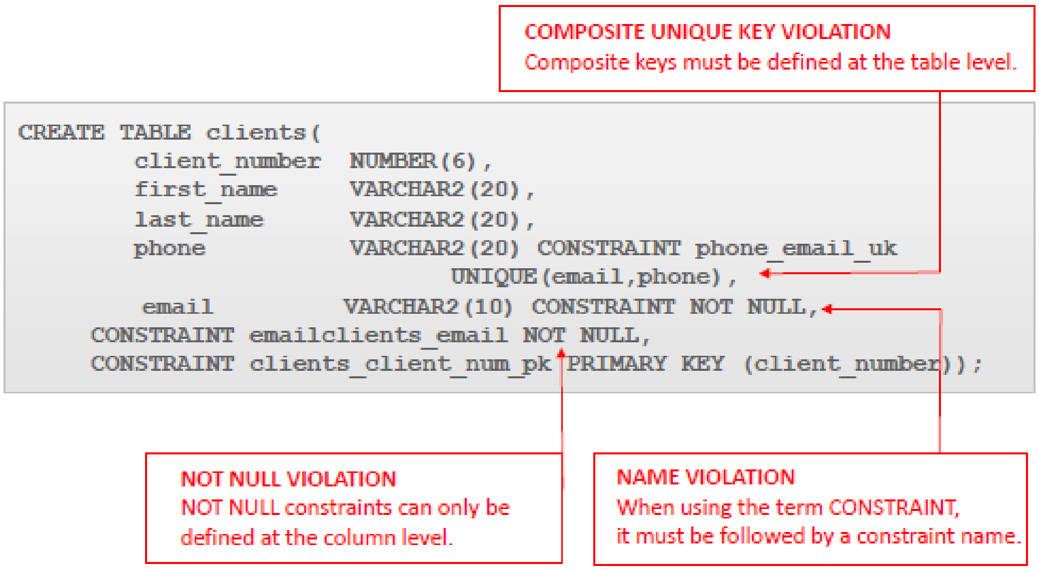
Basic Rules for Constraints

• Constraints that refer to more than one column (a composite key) must be defined at the table level
• The NOT NULL constraint can be specified only at the column level, not the table level
• UNIQUE, PRIMARY KEY, FOREIGN KEY, and CHECK constraints can be defined at either the column or table level
• If the word CONSTRAINT is used in a CREATE TABLE statement, you must give the constraint a name

Examine the Violations

```
CREATE TABLE clients(
        client_number NUMBER(6),
        first_name VARCHAR2(20),
        last_name VARCHAR2(20),
        phone VARCHAR2(20) CONSTRAINT phone_email_uk
        UNIQUE(email,phone),
        email VARCHAR2(10) CONSTRAINT NOT NULL,
CONSTRAINT emailclients_email NOT NULL,
```

CONSTRAINT clients_client_num_pk PRIMARY KEY (client_number));



```
CREATE TABLE clients(
        client_number  NUMBER(6),
        first_name     VARCHAR2(20),
        last_name      VARCHAR2(20),
        phone          VARCHAR2(20) CONSTRAINT phone_email_uk
                              UNIQUE(email,phone),
         email         VARCHAR2(10) CONSTRAINT NOT NULL,
     CONSTRAINT emailclients_email NOT NULL,
     CONSTRAINT clients_client_num_pk PRIMARY KEY (client_number));
```

COMPOSITE UNIQUE KEY VIOLATION
Composite keys must be defined at the table level.

NOT NULL VIOLATION
NOT NULL constraints can only be defined at the column level.

NAME VIOLATION
When using the term CONSTRAINT, it must be followed by a constraint name.

Five Types of Constraints
• Five types of constraints exist within an Oracle database.
• Each type enforces a different rule.
• The types are:
  – NOT NULL constraints
  – UNIQUE constraints
  – PRIMARY KEY constraints
  – FOREIGN KEY constraints
  – CHECK constraints
• In the remainder of this lesson, you will learn about NOT NULL and UNIQUE constraints.
• The next lesson will teach you about the other three types of constraints.

NOT NULL Constraint
• A column defined with a NOT NULL constraint requires that for every row entered into the table, a value must exist for that column.
• For example, if the email column in an employees table was defined as NOT NULL, every employee entered into the table MUST have a value in the email column.
• When defining NOT NULL columns, it is customary to use the suffix _nn in the constraint name.
• For example, the constraint name for the NOT NULL email column in the employees table could be emp_email_nn.

UNIQUE Constraint
• A UNIQUE constraint requires that every value in a column or set of columns (a composite key) be unique; that is, no two rows of a table can have duplicate values.
• For example, it may be important for a business to ensure that no two people have the same email address.
• The email column could be defined using a UNIQUE constraint.
• The column or set of columns that is defined as UNIQUE is called a unique key.
• If the combination of two or more columns must be unique for every entry, the constraint is said to be a composite unique key.
• Stating that all combinations of email and last name must be UNIQUE is an example of a composite unique key.
• The word "key" refers to the columns, not the constraint names.

Unique Constraint Example
• If the email column in the table is defined with a UNIQUE constraint, no other client entry can have an identical email.
• What if two clients live in the same household and share an email address?

CLIENT_NUMBER	FIRST_NAME	LAST_NAME	PHONE	EMAIL
5922	Hiram	Peters	3715832249	hpeters@yahoo.com
5857	Serena	Jones	7035335900	serena.jones@jones.com
6133	Lauren	Vigil	4072220090	lbv@lbv.net

```
INSERT INTO clients (client_number, first_name, Last_name, phone, email)
VALUES (7234, 'Lonny', 'Vigil', 4072220091, 'lbv@lbv.net');
ORA-00001: unique constraint
(USWA_SKHS_SQL01_T01.CLIENT_EMAIL_UK) violated
```

Defining UNIQUE Constraints
• When defining UNIQUE constraints, it is customary to use the suffix _uk in the constraint name.
• For example, the constraint name for the UNIQUE email column in the employees table could be emp_email_uk.
• To define a composite unique key, you must define the constraint at the table level rather than the column level.
• An example of a composite unique-key constraint name is:

```
CONSTRAINT clients_phone_email_uk UNIQUE(email,phone)
```

Composite Unique Key
• UNIQUE constraints allow the input of nulls unless the column also has a NOT NULL constraint defined.
• A null in a column (or in all columns of a composite unique key) always satisfies a UNIQUE constraint because nulls are not considered equal to anything.
CLIENT_NUMBER
FIRST_NAME
LAST_NAME
PHONE

CLIENT_NUMBER	FIRST_NAME	LAST_NAME	PHONE	EMAIL
5922	Hiram	Peters	3715832249	hpeters@yahoo.com
5857	Serena	Jones	7035335900	serena.jones@jones.com
6133	Lauren	Vigil	4072220090	lbv@lbv.net
7234	Lonny	Vigil	4072220091	lbv@lbv.net

$\uparrow$ $\uparrow$
This combination of columns
must be **UNIQUE**

• To satisfy a constraint that designates a composite unique key, no two rows in the table can have the same combination of values in the key columns.
• Also, any row that contains nulls in all key columns automatically satisfies the constraint.

Constraints Created at Table Creation
• When you add a NOT NULL constraint as part of a table creation statement, the Oracle database will create a Check Constraint in the database to enforce a value in the NOT NULL column.
• This constraint creation can be almost invisible to you when you create your table—Oracle just does it.
• At the end of your table creation statement, the message "Table created" displays, but no details are provided about the number or types of constraints that were also created.

14-2 PRIMARY KEY, FOREIGN KEY, and CHECK Constraints

• Ensuring data integrity is what constraints are all about. After all, you're unique!

PRIMARY KEY Constraints
• A PRIMARY KEY constraint is a rule that the values in one column or a combination of columns must uniquely identify each row in a table.
• No primary-key value can appear in more than one row in the table.
• To satisfy a PRIMARY KEY constraint, both of the following conditions must be true:
  – No column that is part of the primary key can contain a null.
  – A table can have only one primary key.
• PRIMARY KEY constraints can be defined at the column or the table level.
• However, if a composite PRIMARY KEY is created, it must be defined at the table level.
• When defining PRIMARY KEY columns, it is a good practice to use the suffix _pk in the constraint name.
• For example, the constraint name for the PRIMARY KEY column named client_number in table named CLIENTS could be clients_client_num_pk.
• In a CREATE TABLE statement, the column-level PRIMARY KEY constraint syntax is stated:

```
REATE TABLE clients
        (client_number NUMBER(4) CONSTRAINT clients_client_num_pk PRIMARY KEY,
        first_name VARCHAR2(14),
        last_name VARCHAR2(13));
```

• Note that the column-level simply refers to the area in the CREATE TABLE statement where the columns are defined.
•The table level refers to the last line in the statement below the list of individual column names.
• To create the PRIMARY KEY constraint at table-level the syntax is:

```
CREATE TABLE clients
        (client_number NUMBER(4),
        first_name VARCHAR2(14),
        last_name VARCHAR2(13),
CONSTRAINT clients_client_num_pk PRIMARY KEY (client_number));
```

• Note that the PRIMARY KEY column name follows the constraint type, and is enclosed in parenthesis.
• To define a composite PRIMARY KEY, you must define the constraint at the table level rather than the column level.
• An example of a composite primary key constraint is shown below:

```
CREATE TABLE copy_job_history
        (employee_id NUMBER(6,0),
        start_date DATE,
        job_id VARCHAR2(10),
        department_id NUMBER(4,0),
CONSTRAINT copy_jhist_id_st_date_pk PRIMARY KEY(employee_id, start_date));
```

FOREIGN KEY (REFERENTIAL INTEGRITY) Constraints
• FOREIGN KEY constraints are also called "referential integrity" constraints.
• Foreign Key constraints designate a column or combination of columns as a foreign key.
• A foreign keys links back to the primary key (or a unique key) in another table, and this link is the basis of the relationship between tables.

Viewing a Foreign Key
• The table containing the foreign key is called the "child" table and the table containing the referenced key is called the "parent" table.

DEPARTMENTS - Parent

DEPARTMENT_ID	DEPT_NAME	MANAGER_ID	LOCATION_ID
90	Executive	100	1700
110	Accounting	205	1700
190	Contracting	-	1700

EMPLOYEE - Child

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	DEPARTMENT_ID
100	Steven	King	90
101	Neena	Kochhar	90
102	Lex	De Haan	90
205	Shelley	Higgins	110
206	William	Gietz	110

• In the tables shown, the primary-key of the DEPARTMENTS table, department_id, also appears in the EMPLOYEES table as a foreign-key column.

Referential-integrity Constraint
• To satisfy a referential-integrity constraint, a foreign-key value must match an existing value in the parent table or be NULL.

DEPARTMENTS - Parent

DEPARTMENT_ID	DEPT_NAME	MANAGER_ID	LOCATION_ID
90	Executive	100	1700
110	Accounting	205	1700

190	Contracting	-	1700

EMPLOYEE - Child

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	DEPARTMENT_ID
100	Steven	King	90
101	Neena	Kochhar	90
102	Lex	De Haan	90
205	Shelley	Higgins	110
206	William	Gietz	110

• A primary-key value can exist without a corresponding foreign-key value; however, a foreign-key must have a corresponding primary key.
• The rule is: before you define a referential-integrity constraint in the child table, the referenced UNIQUE or PRIMARY KEY constraint on the parent table must already be defined.
• In other words, you must first have a parent primary key defined before you can create a foreign key in a child table.

FOREIGN KEY Constraint
• To define a FOREIGN KEY constraint, it is good practice to use the suffix _fk in the constraint name.
• For example, the constraint name for the FOREIGN KEY column department_id in the employees table could be named emps_dept_id_fk.

FOREIGN KEY Constraint Syntax
• The syntax for defining a FOREIGN KEY constraint requires a reference to the table and column in the parent table.
• A FOREIGN KEY constraint in a CREATE TABLE statement can be defined as follows.
• Column-level syntax example:

```
CREATE TABLE copy_employees
        (employee_id NUMBER(6,0) CONSTRAINT copy_emp_pk PRIMARY KEY,
        first_name VARCHAR2(20),
        last_name VARCHAR2(25),
        department_id NUMBER(4,0) CONSTRAINT c_emps_dept_id_fk REFERENCES
                        departments (department_id),
        email VARCHAR2(25));
```

• The syntax for defining a FOREIGN KEY constraint requires a reference to the table and column in the parent table.
• A FOREIGN KEY constraint in a CREATE TABLE statement can be defined as follows.
• Table-level syntax example:

```
CREATE TABLE copy_employees
        (employee_id NUMBER(6,0) CONSTRAINT copy_emp_pk PRIMARY KEY,
        first_name VARCHAR2(20),
        last_name VARCHAR2(25),
        department_id NUMBER(4,0),
        email VARCHAR2(25),
CONSTRAINT c_emps_dept_id_fk FOREIGN KEY (department_id) REFERENCES departments(department_id));
```

ON DELETE CASCADE - Maintaining Referential Integrity
• Using the ON DELETE CASCADE option when defining a foreign key enables the dependent rows in the child table to be deleted when a row in the parent table is deleted.
• If the foreign key does not have an ON DELETE CASCADE option, referenced rows in the parent table cannot be deleted.
• In other words, the child table FOREIGN KEY constraint includes the ON DELETE CASCADE permission allowing its parent to delete the rows that it refers to.

DEPARTMENTS - Parent

DEPARTMENT_ID	DEPT_NAME	MANAGER_ID	LOCATION_ID
90	Executive	100	1700
110	Accounting	205	1700
190	Contracting	-	1700

MPLOYEE - Child

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	DEPARTMENT_ID
100	Steven	King	90
101	Neena	Kochhar	90
102	Lex	De Haan	90
205	Shelley	Higgins	110
206	William	Gietz	110

• If the department_id column in employees was created with the ON DELETE CASCADE option specified, the DELETE statement issued on the departments table will execute.
• If the ON DELETE CASCADE option was not specified when the FOREIGN KEY was created, an attempt to delete a department from the departments table that has entries in the employees table will fail.

ON DELETE CASCADE Syntax

• Table created without ON DELETE CASCADE:

```
CREATE TABLE copy_employees
        (employee_id NUMBER(6,0) CONSTRAINT copy_emp_pk PRIMARY KEY,
        first_name VARCHAR2(20),
        last_name VARCHAR2(25),
        department_id NUMBER(4,0),
        email VARCHAR2(25),
CONSTRAINT cdept_dept_id_fk FOREIGN KEY (department_id)
REFERENCES copy_departments(department_id));
```

•An attempt to delete department_id 110 from the departments table fails as there are dependent rows in the employee table.

```
ORA-02292: integrity constraint (US_A009EMEA815_PLSQL_T01.CDEPT_DEPT_ID_FK)
violated - child record found
```

• Table created with ON DELETE CASCADE:

```
CREATE TABLE copy_employees
        (employee_id NUMBER(6,0) CONSTRAINT copy_emp_pk PRIMARY KEY,
        first_name VARCHAR2(20),
        last_name VARCHAR2(25),
        department_id NUMBER(4,0),
        email VARCHAR2(25),
CONSTRAINT cdept_dept_id_fk FOREIGN KEY (department_id)
REFERENCES  copy_departments(department_id) ON DELETE CASCADE);
```

•An attempt to delete department_id 110 from the departments table succeeds, and the dependent rows in the employee table are also deleted.
•1 row(s) deleted.

ON DELETE SET NULL

• Rather than having the rows in the child table deleted when using an ON DELETE CASCADE option, the child rows can be filled with null values using the ON DELETE SET NULL option.

```
CREATE TABLE copy_employees
        (employee_id NUMBER(6,0) CONSTRAINT copy_emp_pk PRIMARY KEY,
        first_name VARCHAR2(20),
        last_name VARCHAR2(25),
        department_id NUMBER(4,0),
        email VARCHAR2(25),
CONSTRAINT cdept_dept_id_fk FOREIGN KEY (department_id) REFERENCES
copy_departments(department_id) ON DELETE SET NULL);
```

• This could be useful when the parent table value is being changed to a new number such as converting inventory numbers to bar-code numbers.
• You would not want to delete the rows in the child table.
• When the new bar-code numbers are entered into the parent table, they would then be able to be inserted into the child table without having to totally re-create each child table row.

CHECK Constraints
• The CHECK constraint explicitly defines a condition that must be met.
• To satisfy the constraint, each row in the table must make the condition either True or unknown (due to a null).
• The condition of a CHECK constraint can refer to any column in the specified table, but not to columns of other tables.

CHECK Constraint Example
• This CHECK constraint ensures that a value entered for end_date is later than start_date.

```
CREATE TABLE copy_job_history
        (employee_id NUMBER(6,0),
        start_date DATE,
        end_date DATE,
        job_id VARCHAR2(10),
        department_id NUMBER(4,0),
CONSTRAINT cjhist_emp_id_st_date_pk PRIMARY KEY(employee_id, start_date),
CONSTRAINT cjhist_end_ck CHECK (end_date > start_date));
```

•As this CHECK CONSTRAINT is referencing two columns in the table, it MUST be defined at table level.

CHECK Constraint Conditions
• A CHECK constraint must only be on the row where the constraint is defined.
• A CHECK constraint cannot be used in queries that refer to values in other rows.
• The CHECK constraint cannot contain calls to the functions SYSDATE, UID, USER, or USERENV.
• The statement CHECK(SYSDATE >'05-May-1999') is not allowed.
• The CHECK constraint cannot use the pseudocolumns CURRVAL, NEXTVAL, LEVEL, or ROWNUM.
• The statement CHECK(NEXTVAL > 0) is not allowed.
• A single column can have multiple CHECK constraints that reference the column in its definition.
• There is no limit to the number of CHECK constraints that you can define on a column.

CHECK Constraint Syntax
• CHECK constraints can be defined at the column level or the table level.
• The syntax to define a CHECK constraint is:
  – Column-level syntax:

```
salary NUMBER(8,2) CONSTRAINT employees_min_sal_ck CHECK (salary > 0)
```

  –Table-level syntax:

```
CONSTRAINT employees_min_sal_ck CHECK (salary > 0)
```

14-3 Managing Constraints

This lesson covers the following objectives:
• List four different functions that the ALTER statement can perform on constraints
• Write ALTER TABLE statements to add, drop, disable, and enable constraints
• Name a business function that would require a DBA to drop, enable, and/or disable a constraint or use the CASCADE syntax
• Query the data dictionary for USER_CONSTRAINTS and interpret the information returned

A database system needs to be able to enforce business rules and, at the same time, prevent adding, modifying, or deleting data that might result in a violation of the referential integrity of the database.

• The ALTER TABLE statement is used to make changes to constraints in existing tables.
• These changes can include adding or dropping constraints, enabling or disabling constraints, and adding a NOT NULL constraint to a column.

The guidelines for making changes to constraints are:
– You can add, drop, enable, or disable a constraint, but you cannot modify its structure.
– You can add a NOT NULL constraint to an existing column by using the MODIFY clause of the ALTER TABLE statement.
– MODIFY is used because NOT NULL is a column-level change.
– You can define a NOT NULL constraint only if the table is empty or if the column contains a value for every row.

The ALTER Statement

• The ALTER statement requires:
  – name of the table
  – name of the constraint
  – type of constraint
  – name of the column affected by the constraint

• In the code example shown below, using the employees table, the primary-key constraint could have been added after the table was originally created.

```
ALTER TABLE employees
ADD CONSTRAINT emp_id_pk PRIMARY KEY (employee_id);
```

Adding Constraints

• To add a constraint to an existing table, use the following SQL syntax:

```
ALTER TABLE table_name
ADD [CONSTRAINT constraint_name] type of constraint (column_name);
```

• If the constraint is a FOREIGN KEY constraint, the REFERENCES keyword must be included in the statement.

• Syntax:

```
ALTER TABLE tablename
ADD CONSTRAINT constraint_name FOREIGN KEY(column_name) REFERENCES tablename(column_name);
```

Adding Constraints Example

• Consider the employees database.

• The primary key from the DEPARTMENTS table is entered in the EMPLOYEES table as a foreign key.

DEPARTMENTS - Parent

DEPARTMENT_ID	DEPT_NAME	MANAGER_ID	LOCATION_ID
90	Executive	100	1700
110	Accounting	205	1700
190	Contracting	-	1700

EMPLOYEE - Child

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	DEPARTMENT_ID
100	Steven	King	90
101	Neena	Kochhar	90
102	Lex	De Haan	90
205	Shelley	Higgins	110
206	William	Gietz	110

• The following example demonstrates the syntax to add this foreign key to the EMPLOYEES table:

```
ALTER TABLE employees
ADD CONSTRAINT emp_dept_fk FOREIGN KEY (department_id)
REFERENCES departments (department_id)
ON DELETE CASCADE;
```

DEPARTMENTS - Parent

DEPARTMENT_ID	DEPT_NAME	MANAGER_ID	LOCATION_ID
90	Executive	100	1700
110	Accounting	205	1700
190	Contracting	-	1700

EMPLOYEE - Child

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	DEPARTMENT_ID
100	Steven	King	90
101	Neena	Kochhar	90
102	Lex	De Haan	90
205	Shelley	Higgins	110
206	William	Gietz	110

• If the constraint is a NOT NULL constraint, the ALTER TABLE statement uses MODIFY in place of ADD.
• NOT NULL constraints can be added only if the table is empty or if the column contains a value for every row:
ALTER TABLE table_name MODIFY (column_name CONSTRAINT constraint_name NOT NULL);
ALTER TABLE employees MODIFY (email CONSTRAINT emp_email_nn NOT NULL);

Why Enable and Disable Constraints?
• To enforce the rules defined by integrity constraints, the constraints should always be enabled.
• In certain situations, however, it is desirable to temporarily disable the integrity constraints of a table for performance reasons, such as:
– When loading large amounts of data into a table
– When performing batch operations that make massive changes to a table (such as changing everyone's employee number by adding 1,000 to the existing number)

Dropping Constraints
• To drop a constraint, you need to know the name of the constraint.
• If you do not know it, you can find the constraint name from the USER_CONSTRAINTS and USER_CONS_COLUMNS in the data dictionary.
• The CASCADE option of the DROP clause causes any dependent constraints also to be dropped.
• Note that when you drop an integrity constraint, that constraint is no longer enforced by the Oracle Server and is no longer available in the data dictionary.
• No rows or any data in any of the affected tables are deleted when you drop a constraint.
ALTER TABLE table_name DROP CONSTRAINT name [CASCADE}
ALTER TABLE copy_departments DROP CONSTRAINT c_dept_dept_id_pk CASCADE;

Disabling Constraints
• By default, whenever an integrity constraint is defined in a CREATE or ALTER TABLE statement, the constraint is automatically enabled (enforced) by Oracle unless it is specifically created in a disabled state using the DISABLE clause.
• You can disable a constraint without dropping it or re-creating it by using the ALTER TABLE option DISABLE.
• DISABLE allows incoming data, whether or not it conforms to the constraint.
• This function allows data to be added to a child table without having corresponding values in the parent table.
• DISABLE simply switches off the constraint.
• You can use the DISABLE clause in both the ALTER TABLE statement and the CREATE TABLE statement.

```
CREATE TABLE copy_employees
        ( employee_id NUMBER(6,0) PRIMARY KEY DISABLE,
        ...
        ...);

ALTER TABLE copy_employees
DISABLE CONSTRAINT c_emp_dept_id_fk;
```

•Disabling a unique or primary-key constraint removes the unique index.

Using the CASCADE Clause
• The CASCADE clause disables dependent integrity constraints. If the constraint is later enabled, the dependent constraints are not automatically enabled.
• Syntax and example:
ALTER TABLE copy_departments DISABLE CONSTRAINT c_dept_dept_id_pk CASCADE;;
ALTER TABLE table_name DISABLE CONSTRAINT constraint_name [CASCADE];

Enabling Constraints
• To activate an integrity constraint currently disabled, use the ENABLE clause in the ALTER TABLE statement.
• ENABLE ensures that all incoming data conforms to the constraint.

• Syntax and example:

```
ALTER TABLE table_name
ENABLE CONSTRAINT constraint_name;
ALTER TABLE copy_departments
ENABLE CONSTRAINT c_dept_dept_id_pk;
```

•You can use the ENABLE clause in both the CREATE TABLE statement and the ALTER TABLE statement.

Enabling Constraint Considerations
• If you enable a constraint, that constraint applies to all the data in the table.
• All the data in the table must fit the constraint.
• If you enable a UNIQUE KEY or PRIMARY KEY constraint, a UNIQUE or PRIMARY KEY index is created automatically.
• Enabling a PRIMARY KEY constraint that was disabled with the CASCADE option does not enable any foreign keys that are dependent on the primary key.
• ENABLE switches the constraint back on after you switched it off.

Cascading Constraints
• Cascading referential-integrity constraints allow you to define the actions the database server takes when a user attempts to delete or update a key to which existing foreign keys point.
• The CASCADE CONSTRAINTS clause is used along with the DROP COLUMN clause.
• It drops all referential-integrity constraints that refer to the primary and unique keys defined on the dropped columns.
• It also drops all multicolumn constraints defined on the dropped columns.

If an ALTER TABLE statement does not include the CASCADE CONSTRAINTS option, any attempt to drop a primary key or multicolumn constraint will fail.
• Remember, you can't delete a parent value if child values exist in other tables.

```
ALTER TABLE table_name DROP(column name(s)) CASCADE CONSTRAINTS;
```

When CASCADE is Not Required
• If all columns referenced by the constraints defined on the dropped columns are also dropped, then CASCADE CONSTRAINTS is not required.
• For example, assuming that no other referential constraints from other tables refer to column PK, it is valid to submit the following statement without the CASCADE CONSTRAINTS clause:

```
ALTER TABLE tablename DROP(pk_column_name(s));
```

•However, if any constraint is referenced by columns from other tables or remaining columns in the target table, you must specify CASCADE CONSTRAINTS to avoid an error.

Viewing Constraints
• After creating a table, you can confirm its existence by issuing a DESCRIBE command.
• The only constraint that you can verify using DESCRIBE is the NOT NULL constraint.
• The NOT NULL constraint will also appear in the data dictionary as a CHECK constraint.
• To view all constraints on your table, query the USER_CONSTRAINTS table.

```
SELECT constraint_name, table_name, constraint_type, status
FROM USER_CONSTRAINTS
WHERE table_name ='COPY_EMPLOYEES';
```

CONSTRAINT_NAME	TABLE_NAME	CONSTRAINT_TYPE	STATUS
COPY_EMP_PK	COPY_EMPLOYEES	P	ENABLED
CDEPT_DEPT_ID_FK	COPY_EMPLOYEES	R	ENABLED

Query USER_CONSTRAINTS
• The constraint types listed in the Data Dictionary are:
– P – PRIMARY KEY; R – REFERENCES (foreign key);
– C – CHECK constraint (including NOT NULL);
– U – UNIQUE.