

# 11

## Introduction to Data Definition Language

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

# Objectives

After completing this lesson, you should be able to do the following:

- Categorize the main database objects
- Review the table structure
- List the data types that are available for columns
- Create a simple table
- Explain how constraints are created at the time of table creation
- Describe how schema objects work

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

In this lesson, you are introduced to the data definition language (DDL) statements. You learn the basics of how to create simple tables, alter them, and remove them. The data types available in DDL are shown and schema concepts are introduced. Constraints are discussed in this lesson. Exception messages that are generated from violating constraints during DML operations are shown and explained.

# Lesson Agenda

- Database objects
  - Naming rules
- Data types
- CREATE TABLE statement
- Overview of constraints: NOT NULL, UNIQUE, PRIMARY KEY, FOREIGN KEY, CHECK constraints
- Creating a table using a subquery
- ALTER TABLE statement
- DROP TABLE statement

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

# Database Objects

Object	Description
Table	Is the basic unit of storage; composed of rows
View	Logically represents subsets of data from one or more tables
Sequence	Generates numeric values
Index	Improves the performance of some queries
Synonym	Gives alternative name to an object

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The Oracle database can contain multiple data structures. Each structure should be outlined in the database design so that it can be created during the build stage of database development.

- **Table:** Stores data
- **View:** Is a subset of data from one or more tables
- **Sequence:** Generates numeric values
- **Index:** Improves the performance of some queries
- **Synonym:** Gives alternative name to an object

## Oracle Table Structures

- Tables can be created at any time, even when users are using the database.
- You do not need to specify the size of a table. The size is ultimately defined by the amount of space allocated to the database as a whole. It is important, however, to estimate how much space a table will use over time.
- Table structure can be modified online.

**Note:** More database objects are available, but are not covered in this course.

# Naming Rules

Table names and column names must:

- Begin with a letter
- Be 1–30 characters long
- Contain only A–Z, a–z, 0–9, \_, \$, and #
- Not duplicate the name of another object owned by the same user
- Not be an Oracle server–reserved word

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

You name database tables and columns according to the standard rules for naming any Oracle database object.

- Table names and column names must begin with a letter and be 1–30 characters long.
- Names must contain only the characters A–Z, a–z, 0–9, \_ (underscore), \$, and # (legal characters, but their use is discouraged).
- Names must not duplicate the name of another object owned by the same Oracle server user.
- Names must not be an Oracle server–reserved word.
  - You may also use quoted identifiers to represent the name of an object. A quoted identifier begins and ends with double quotation marks (""). If you name a schema object using a quoted identifier, you must use the double quotation marks whenever you refer to that object. Quoted identifiers can be reserved words, although this is not recommended.

## Naming Guidelines

Use descriptive names for tables and other database objects.

**Note:** Names are not case-sensitive. For example, `EMPLOYEES` is treated to be the same name as `eMPLOYEES` or `eMpLOYEES`. However, quoted identifiers are case-sensitive.

For more information, see the “Schema Object Names and Qualifiers” section in the *Oracle Database SQL Language Reference* for 10g or 11g database.

## Lesson Agenda

- Database objects
  - Naming rules
- **Data types**
- CREATE TABLE statement
- Overview of constraints: NOT NULL, UNIQUE, PRIMARY KEY, FOREIGN KEY, CHECK constraints
- Creating a table using a subquery
- ALTER TABLE statement
- DROP TABLE statement

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

# Data Types

Data Type	Description
VARCHAR2 ( <i>size</i> )	Variable-length character data
CHAR ( <i>size</i> )	Fixed-length character data
NUMBER ( <i>p</i> , <i>s</i> )	Variable-length numeric data
DATE	Date and time values
LONG	Variable-length character data (up to 2 GB)
CLOB	Maximum size is (4 gigabytes - 1) * (DB_BLOCK_SIZE).
RAW and LONG RAW	Raw binary data
BLOB	Maximum size is (4 gigabytes - 1) * (DB_BLOCK_SIZE initialization parameter (8 TB to 128 TB)).
BFILE	Binary data stored in an external file (up to 4 GB)
ROWID	A base-64 number system representing the unique address of a row in its table

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

When you identify a column for a table, you need to provide a data type for the column. There are several data types available:

Data Type	Description
VARCHAR2 ( <i>size</i> )	Variable-length character data (A maximum <i>size</i> must be specified: minimum <i>size</i> is 1.) Maximum size is: <ul style="list-style-type: none"> <li>32767 bytes if MAX_SQL_STRING_SIZE = EXTENDED</li> <li>4000 bytes if MAX_SQL_STRING_SIZE = LEGACY</li> </ul>
CHAR [( <i>size</i> )]	Fixed-length character data of length <i>size</i> bytes (Default and minimum <i>size</i> is 1; maximum <i>size</i> is 2,000.)
NUMBER [( <i>p</i> , <i>s</i> )]	Number having precision <i>p</i> and scale <i>s</i> (Precision is the total number of decimal digits and scale is the number of digits to the right of the decimal point; precision can range from 1 to 38, and scale can range from -84 to 127.)
DATE	Date and time values to the nearest second between January 1, 4712 B.C., and December 31, 9999 A.D.

Data Type	Description
LONG	Variable-length character data (up to 2 GB)
CLOB	A character large object containing single-byte or multibyte characters. Maximum size is (4 gigabytes - 1) * (DB_BLOCK_SIZE); stores national character set data.
NCLOB	A character large object containing Unicode characters. Both fixed-width and variable-width character sets are supported, both using the database national character set. Maximum size is (4 gigabytes - 1) * (database block size); stores national character set data.
RAW(size)	Raw binary data of length <i>size</i> bytes. You must specify <i>size</i> for a RAW value. Maximum <i>size</i> is: 32767 bytes if MAX_SQL_STRING_SIZE = EXTENDED 4000 bytes if MAX_SQL_STRING_SIZE = LEGACY
LONG RAW	Raw binary data of variable length up to 2 gigabytes
BLOB	A binary large object. Maximum size is (4 gigabytes - 1) * (DB_BLOCK_SIZE initialization parameter (8 TB to 128 TB)).
BFILE	Binary data stored in an external file (up to 4 GB)
ROWID	Base 64 string representing the unique address of a row in its table. This data type is primarily for values returned by the ROWID pseudocolumn

### Guidelines

- A LONG column is not copied when a table is created using a subquery.
- A LONG column cannot be included in a GROUP BY or an ORDER BY clause.
- Only one LONG column can be used per table.
- No constraints can be defined on a LONG column.
- You might want to use a CLOB column rather than a LONG column.



# Datetime Data Types

You can use several datetime data types:

Data Type	Description
TIMESTAMP	Date with fractional seconds
INTERVAL YEAR TO MONTH	Stored as an interval of years and months
INTERVAL DAY TO SECOND	Stored as an interval of days, hours, minutes, and seconds



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Data Type	Description
TIMESTAMP	Enables storage of time as a date with fractional seconds. It stores the year, month, day, hour, minute, and the second value of the DATE data type, as well as the fractional seconds value. There are several variations of this data type such as WITH TIMEZONE and WITH LOCALTIMEZONE.
INTERVAL YEAR TO MONTH	Enables storage of time as an interval of years and months; used to represent the difference between two datetime values in which the only significant portions are the year and month
INTERVAL DAY TO SECOND	Enables storage of time as an interval of days, hours, minutes, and seconds; used to represent the precise difference between two datetime values

**Note:** These datetime data types are available with Oracle9i and later releases. The datetime data types are discussed in detail in the lesson titled “Managing Data in Different Time Zones” in the *Oracle Database: SQL Workshop II* course.

Also, for more information about the datetime data types, see the sections on “TIMESTAMP Datatype,” “INTERVAL YEAR TO MONTH Datatype,” and “INTERVAL DAY TO SECOND Datatype” in *Oracle Database SQL Language Reference* for 12c database.

## DEFAULT Option

- Specify a default value for a column during the CREATE table.

```
... hire_date DATE DEFAULT SYSDATE, ...
```

- Literal values, expressions, or SQL functions are legal values.
- Another column's name or a pseudocolumn are illegal values.
- The default data type must match the column data type.

```
CREATE TABLE hire_dates
  (id          NUMBER(8) ,
   hire date DATE DEFAULT SYSDATE);
table HIRE_DATES created.
```

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

When you define a table, you can specify that a column should be given a default value by using the `DEFAULT` option. This option prevents null values from entering the columns when a row is inserted without a value for the column. The default value can be a literal, an expression, or a SQL function (such as `SYSDATE` or `USER`), but the value cannot be the name of another column or a pseudocolumn (such as `NEXTVAL` or `CURRVAL`). The default expression must match the data type of the column.

Consider the following examples:

```
INSERT INTO hire_dates values(45, NULL);
```

The preceding statement will insert the null value rather than the default value.

```
INSERT INTO hire_dates(id) values(35);
```

The preceding statement will insert `SYSDATE` for the `HIRE_DATE` column.

**Note:** In SQL Developer, click the Run Script icon or press F5 to run the DDL statements. The feedback messages will be shown on the Script Output tabbed page.

## Lesson Agenda

- Database objects
  - Naming rules
- Data types
- **CREATE TABLE statement**
- Overview of constraints: NOT NULL, UNIQUE, PRIMARY KEY, FOREIGN KEY, CHECK constraints
- Creating a table using a subquery
- ALTER TABLE statement
- DROP TABLE statement

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## CREATE TABLE Statement

- You must have:
  - The CREATE TABLE privilege
  - A storage area

```
CREATE TABLE [schema.]table
      (column datatype [DEFAULT expr] [, ...]);
```

- You specify:
  - The table name
  - The column name, column data type, and column size



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

You create tables to store data by executing the SQL CREATE TABLE statement. This statement is one of the DDL statements that are a subset of the SQL statements used to create, modify, or remove Oracle Database structures. These statements have an immediate effect on the database and they also record information in the data dictionary.

To create a table, a user must have the CREATE TABLE privilege and a storage area in which to create objects. The database administrator (DBA) uses data control language (DCL) statements to grant privileges to users.

In the syntax:

<i>schema</i>	Is the same as the owner's name
<i>table</i>	Is the name of the table
DEFAULT <i>expr</i>	Specifies a default value if a value is omitted in the INSERT statement
<i>column</i>	Is the name of the column
<i>datatype</i>	Is the column's data type and length

**Note:** The CREATE ANY TABLE privilege is needed to create a table in any schema other than the user's schema.

# Creating Tables

- Create the table:

```
CREATE TABLE dept
  (deptno      NUMBER(2) ,
   dname       VARCHAR2(14) ,
   loc         VARCHAR2(13) ,
   create_date DATE DEFAULT SYSDATE) ;
```

table DEPT created.

- Confirm table creation:

```
DESCRIBE dept
```

```
DESCRIBE dept
Name          Null Type
-----
DEPTNO        NUMBER(2)
DNAME         VARCHAR2(14)
LOC           VARCHAR2(13)
CREATE_DATE    DATE
```

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The example in the slide creates the DEPT table with four columns: DEPTNO, DNAME, LOC, and CREATE\_DATE. The CREATE\_DATE column has a default value. If a value is not provided for an INSERT statement, the system date is automatically inserted.

To confirm that the table was created, run the DESCRIBE command.

Because creating a table is a DDL statement, an automatic commit takes place when this statement is executed.

**Note:** You can view the list of tables that you own by querying the data dictionary. For example:

```
select table_name from user_tables;
```

Using data dictionary views, you can also find information about other database objects such as views, indexes, and so on. You will learn about data dictionaries in detail in the *Oracle Database: SQL Fundamentals II* course.

## Lesson Agenda

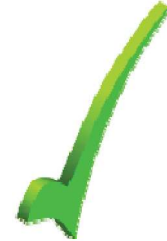
- Database objects
  - Naming rules
- Data types
- CREATE TABLE statement
- **Overview of constraints: NOT NULL, UNIQUE, PRIMARY KEY, FOREIGN KEY, CHECK constraints**
- Creating a table using a subquery
- ALTER TABLE statement
- DROP TABLE statement

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Including Constraints

- Constraints enforce rules at the table level.
- Constraints ensure the consistency and integrity of the database.
- The following constraint types are valid:
  - NOT NULL
  - UNIQUE
  - PRIMARY KEY
  - FOREIGN KEY
  - CHECK



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The Oracle server uses constraints to prevent invalid data entry into tables.

You can use constraints to do the following:

- Enforce rules on the data in a table whenever a row is inserted, updated, or deleted from that table. The constraint must be satisfied for the operation to succeed.
- Prevent the dropping of a table if there are dependencies from other tables.
- Provide rules for Oracle tools, such as Oracle Developer.

### Data Integrity Constraints

Constraint	Description
NOT NULL	Specifies that the column cannot contain a null value
UNIQUE	Specifies a column or combination of columns whose values must be unique for all rows in the table
PRIMARY KEY	Uniquely identifies each row of the table
FOREIGN KEY	Establishes and enforces a referential integrity between the column and a column of the referenced table such that values in one table match values in another table.
CHECK	Specifies a condition that must be true

## Constraint Guidelines

- You can name a constraint or the Oracle server generates a name by using the `SYS_Cn` format.
- Create a constraint at either of the following times:
  - At the same time as the creation of the table
  - After the creation of the table
- Define a constraint at the column or table level.
- View a constraint in the data dictionary.

The Oracle logo, consisting of the word "ORACLE" in a white, sans-serif font, centered within a red rectangular background.

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

All constraints are stored in the data dictionary. Constraints are easy to reference if you give them a meaningful name. Constraint names must follow the standard object-naming rules, except that the name cannot be the same as another object owned by the same user. If you do not name your constraint, the Oracle server generates a name with the format `SYS_Cn`, where *n* is an integer so that the constraint name is unique.

Constraints can be defined at the time of table creation or after the creation of the table. You can define a constraint at the column or table level. Functionally, a table-level constraint is the same as a column-level constraint.

For more information, see the section on “Constraints” in *Oracle Database SQL Language Reference* for 12c database.



# Defining Constraints

- Syntax:

```
CREATE TABLE [schema.]table
  (column datatype [DEFAULT expr]
   [column_constraint],
   ...
   [table_constraint] [, ...] );
```

- Column-level constraint syntax:

```
column [CONSTRAINT constraint_name] constraint_type,
```

- Table-level constraint syntax:

```
column, ...
  [CONSTRAINT constraint_name] constraint_type
  (column, ...),
```

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The slide gives the syntax for defining constraints when creating a table. You can create constraints at either the column level or table level. Constraints defined at the column level are included when the column is defined. Table-level constraints are defined at the end of the table definition, and must refer to the column or columns on which the constraint pertains in a set of parentheses. It is mainly the syntax that differentiates the two; otherwise, functionally, a column-level constraint is the same as a table-level constraint.

NOT NULL constraints must be defined at the column level.

Constraints that apply to more than one column must be defined at the table level.

In the syntax:

<i>schema</i>	Is the same as the owner's name
<i>table</i>	Is the name of the table
DEFAULT <i>expr</i>	Specifies a default value to be used if a value is omitted in the statement
INSERT	
<i>column</i>	Is the name of the column
<i>datatype</i>	Is the column's data type and length
<i>column_constraint</i>	Is an integrity constraint as part of the column definition
<i>table_constraint</i>	Is an integrity constraint as part of the table definition

# Defining Constraints

- Example of a column-level constraint:

```
CREATE TABLE employees(  
  employee_id  NUMBER(6)  
    CONSTRAINT emp_emp_id_pk PRIMARY KEY,  
  first_name   VARCHAR2(20),  
  ...);
```

**1**

- Example of a table-level constraint:

```
CREATE TABLE employees(  
  employee_id  NUMBER(6),  
  first_name   VARCHAR2(20),  
  ...  
  job_id       VARCHAR2(10) NOT NULL,  
  CONSTRAINT emp_emp_id_pk  
    PRIMARY KEY (EMPLOYEE_ID));
```

**2****ORACLE**

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Constraints are usually created at the same time as the table. Constraints can be added to a table after its creation and also be temporarily disabled.

Both examples in the slide create a primary key constraint on the `EMPLOYEE_ID` column of the `EMPLOYEES` table.

1. The first example uses the column-level syntax to define the constraint.
2. The second example uses the table-level syntax to define the constraint.

More details about the primary key constraint are provided later in this lesson.

## NOT NULL Constraint

Ensures that null values are not permitted for the column:

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY	COMMISSION_PCT	DEPARTMENT_ID	EMAIL	PHONE_NUMBER	HIRE_DATE
100	Steven	King	24000	(null)	90	SKING	515.123.4567	17-JUN-87
101	Neena	Kochhar	17000	(null)	90	NKOCHHAR	515.123.4568	21-SEP-89
102	Lex	De Haan	17000	(null)	90	LDEHAAN	515.123.4569	13-JAN-93
103	Alexander	Hunold	9000	(null)	60	AHUNOLD	590.423.4567	03-JAN-90
104	Bruce	Ernst	6000	(null)	60	BERNST	590.423.4568	21-MAY-91
107	Diana	Lorentz	4200	(null)	60	DLORENTZ	590.423.5567	07-FEB-99
124	Kevin	Mourgos	5800	(null)	50	KMOURGOS	650.123.5234	16-NOV-99
141	Trenna	Rajs	3500	(null)	50	TRAJS	650.121.8009	17-OCT-95
142	Curtis	Davies	3100	(null)	50	CDAVIES	650.121.2994	29-JAN-97
143	Randall	Matos	2600	(null)	50	RMATOS	650.121.2874	15-MAR-98
144	Peter	Vargas	2500	(null)	50	PVARGAS	650.121.2004	09-JUL-98
149	Eleni	Zlotkey	10500	0.2	80	EZLOTKEY	011.44.1344.429016	29-JAN-00
174	Ellen	Abel	11000	0.3	80	EABEL	011.44.1644.429267	11-MAY-96
176	Jonathon	Taylor	8600	0.2	80	JTAYLOR	011.44.1644.429265	24-MAR-98
178	Kimberely	Grant	7000	0.15	(null)	KGRANT	011.44.1644.429263	24-MAY-99
200	Jennifer	Whalen	4400	(null)	10	JWHALEN	515.123.4444	17-SEP-87
201	Michael	Hartstein	13000	(null)	20	MHARTSTE	515.123.5555	17-FEB-96
202	Pat	Fay	6000	(null)	20	PFAY	603.123.6666	17-AUG-97
205	Shelley	Higgins	12000	(null)	110	SHIGGINS	515.123.8080	07-JUN-94
206	William	Gietz	8300	(null)	110	WGIEZT	515.123.8181	07-JUN-94

↑  
NOT NULL constraint  
(Primary Key enforces NOT  
NULL constraint.)

↑  
NOT NULL  
constraint

↑  
Absence of NOT NULL constraint  
(Any row can contain a null value  
for this column.)

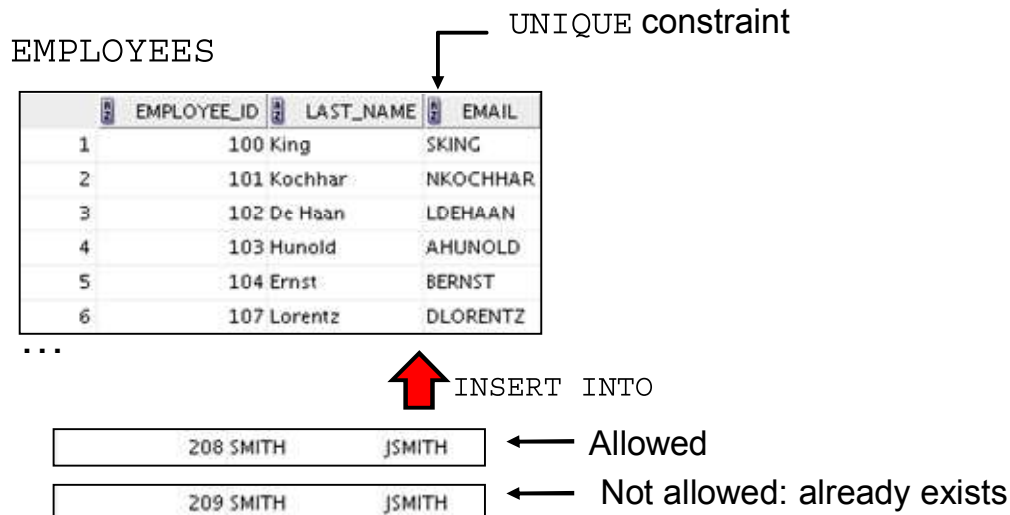
ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The NOT NULL constraint ensures that the column contains no null values. Columns without the NOT NULL constraint can contain null values by default. NOT NULL constraints must be defined at the column level. In the EMPLOYEES table, the EMPLOYEE\_ID column inherits a NOT NULL constraint because it is defined as a primary key. Otherwise, the LAST\_NAME, EMAIL, HIRE\_DATE, and JOB\_ID columns have the NOT NULL constraint enforced on them.

**Note:** Primary key constraint is discussed in detail later in this lesson.

# UNIQUE Constraint



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

A **UNIQUE** key integrity constraint requires that every value in a column or a set of columns (key) be unique—that is, no two rows of a table can have duplicate values in a specified column or a set of columns. The column (or set of columns) included in the definition of the **UNIQUE** key constraint is called the *unique key*. If the **UNIQUE** constraint comprises more than one column, that group of columns is called a *composite unique key*.

**UNIQUE** constraints enable the input of nulls unless you also define **NOT NULL** constraints for the same columns. In fact, any number of rows can include nulls for columns without the **NOT NULL** constraints because nulls are not considered equal to anything. A null in a column (or in all columns of a composite **UNIQUE** key) always satisfies a **UNIQUE** constraint.

**Note:** Because of the search mechanism for the **UNIQUE** constraints on more than one column, you cannot have identical values in the non-null columns of a partially null composite **UNIQUE** key constraint.

# UNIQUE Constraint

Defined at either the table level or the column level:

```
CREATE TABLE employees(  
    employee_id      NUMBER(6),  
    last_name        VARCHAR2(25) NOT NULL,  
    email            VARCHAR2(25),  
    salary           NUMBER(8,2),  
    commission_pct   NUMBER(2,2),  
    hire_date        DATE NOT NULL,  
    ...  
    CONSTRAINT emp_email_uk UNIQUE(email));
```

ORACLE

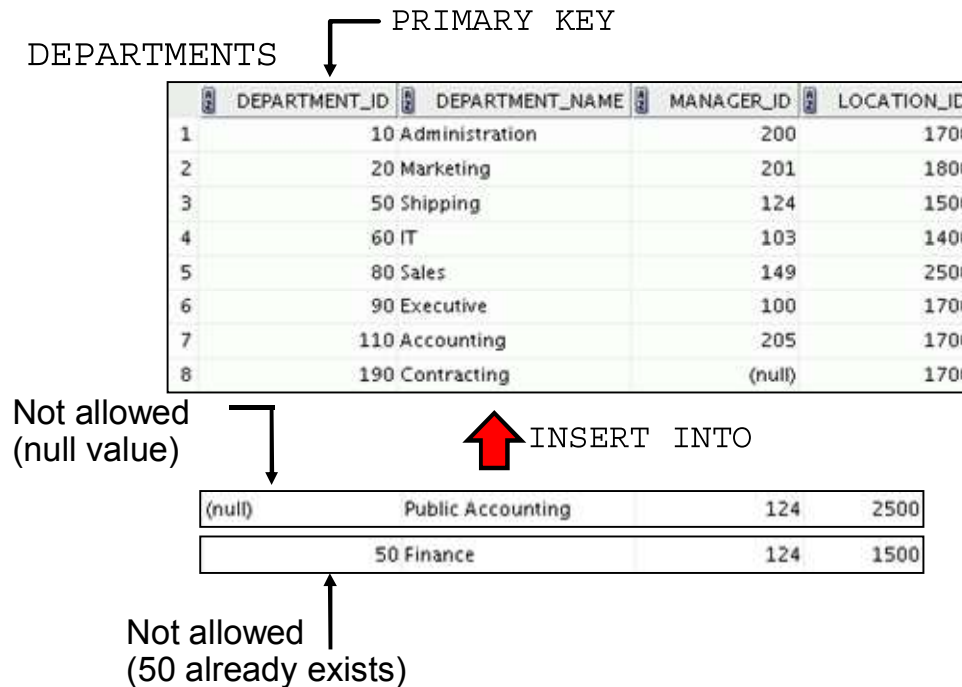
Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

UNIQUE constraints can be defined at the column level or table level. You define the constraint at the table level when you want to create a composite unique key. A composite key is defined when there is not a single attribute that can uniquely identify a row. In that case, you can have a unique key that is composed of two or more columns, the combined value of which is always unique and can identify rows.

The example in the slide applies the UNIQUE constraint to the EMAIL column of the EMPLOYEES table. The name of the constraint is EMP\_EMAIL\_UK.

**Note:** The Oracle server enforces the UNIQUE constraint by implicitly creating a unique index on the unique key column or columns.

# PRIMARY KEY Constraint



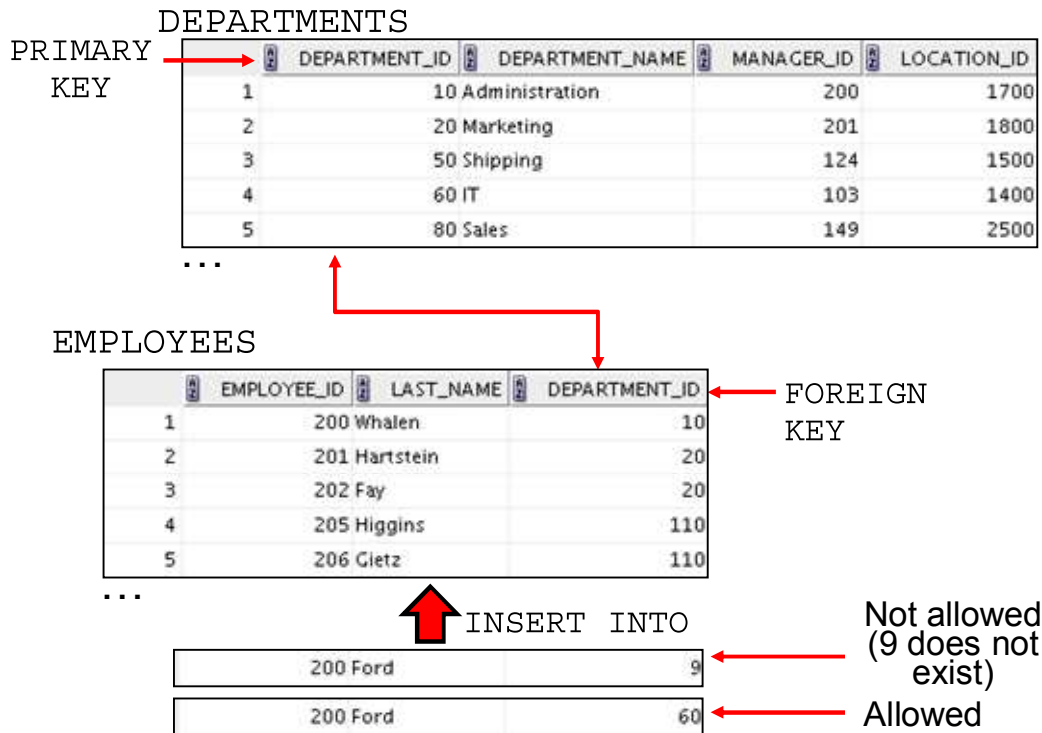
ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

A **PRIMARY KEY** constraint creates a primary key for the table. Only one primary key can be created for each table. The **PRIMARY KEY** constraint is a column or a set of columns that uniquely identifies each row in a table. This constraint enforces the uniqueness of the column or column combination, and ensures that no column that is part of the primary key can contain a null value.

**Note:** Because uniqueness is part of the primary key constraint definition, the Oracle server enforces the uniqueness by implicitly creating a unique index on the primary key column or columns.

# FOREIGN KEY Constraint



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The FOREIGN KEY (or referential integrity) constraint designates a column or a combination of columns as a foreign key, and establishes a relationship with a primary key or a unique key in the same table or a different table.

In the example in the slide, DEPARTMENT\_ID has been defined as the foreign key in the EMPLOYEES table (dependent or child table); it references the DEPARTMENT\_ID column of the DEPARTMENTS table (the referenced or parent table).

## Guidelines

- A foreign key value must match an existing value in the parent table or be NULL.
- Foreign keys are based on data values and are purely logical, rather than physical, pointers.

## FOREIGN KEY Constraint

Defined at either the table level or the column level:

```
CREATE TABLE employees(
  employee_id      NUMBER(6),
  last_name        VARCHAR2(25) NOT NULL,
  email            VARCHAR2(25),
  salary           NUMBER(8,2),
  commission_pct   NUMBER(2,2),
  hire_date        DATE NOT NULL,
  ...
  department_id    NUMBER(4),
  CONSTRAINT emp_dept_fk FOREIGN KEY (department_id)
    REFERENCES departments(department_id),
  CONSTRAINT emp_email_uk UNIQUE(email));
```

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

FOREIGN KEY constraints can be defined at the column or table constraint level. A composite foreign key must be created by using the table-level definition.

The example in the slide defines a FOREIGN KEY constraint on the DEPARTMENT\_ID column of the EMPLOYEES table, using table-level syntax. The name of the constraint is EMP\_DEPT\_FK.

The foreign key can also be defined at the column level, provided that the constraint is based on a single column. The syntax differs in that the keywords FOREIGN KEY do not appear. For example:

```
CREATE TABLE employees
(
  ...
  department_id NUMBER(4) CONSTRAINT emp_deptid_fk
    REFERENCES departments(department_id),
  ...
)
```



## FOREIGN KEY Constraint: Keywords

- **FOREIGN KEY:** Defines the column in the child table at the table-constraint level
- **REFERENCES:** Identifies the table and column in the parent table
- **ON DELETE CASCADE:** Deletes the dependent rows in the child table when a row in the parent table is deleted
- **ON DELETE SET NULL:** Converts dependent foreign key values to null

The Oracle logo, consisting of the word "ORACLE" in a white, sans-serif font, centered within a red rectangular background.

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The foreign key is defined in the child table and the table containing the referenced column is the parent table. The foreign key is defined using a combination of the following keywords:

- **FOREIGN KEY** is used to define the column in the child table at the table-constraint level.
- **REFERENCES** identifies the table and the column in the parent table.
- **ON DELETE CASCADE** indicates that when a row in the parent table is deleted, the dependent rows in the child table are also deleted.
- **ON DELETE SET NULL** indicates that when a row in the parent table is deleted, the foreign key values are set to null.

The default behavior is called the *restrict rule*, which disallows the update or deletion of referenced data.

Without the **ON DELETE CASCADE** or the **ON DELETE SET NULL** options, the row in the parent table cannot be deleted if it is referenced in the child table. And these keyword cannot be used in column-level syntax.

## CHECK Constraint

- It defines a condition that each row must satisfy.
- The following expressions are not allowed:
  - References to CURRVAL, NEXTVAL, LEVEL, and ROWNUM pseudocolumns
  - Calls to SYSDATE, UID, USER, and USERENV functions
  - Queries that refer to other values in other rows

```
..., salary NUMBER(2)
      CONSTRAINT emp_salary_min
      CHECK (salary > 0),...
```

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The **CHECK** constraint defines a condition that each row must satisfy. The condition can use the same constructs as the query conditions, with the following exceptions:

- References to the CURRVAL, NEXTVAL, LEVEL, and ROWNUM pseudocolumns
- Calls to SYSDATE, UID, USER, and USERENV functions
- Queries that refer to other values in other rows

A single column can have multiple **CHECK** constraints that refer to the column in its definition. There is no limit to the number of **CHECK** constraints that you can define on a column.

**CHECK** constraints can be defined at the column level or table level.

```
CREATE TABLE employees
(
  ...
  salary NUMBER(8,2) CONSTRAINT emp_salary_min
                        CHECK (salary > 0),
  ...
)
```

## CREATE TABLE: Example

```
CREATE TABLE teach_emp (  
    empno      NUMBER(5) PRIMARY KEY,  
    ename      VARCHAR2(15) NOT NULL,  
    job        VARCHAR2(10),  
    mgr        NUMBER(5),  
    hiredate   DATE DEFAULT (sysdate),  
    photo      BLOB,  
    sal        NUMBER(7,2),  
    deptno     NUMBER(3) NOT NULL  
    CONSTRAINT admin_dept_fkey REFERENCES  
        departments(department_id));
```

The Oracle logo, consisting of the word "ORACLE" in a white, sans-serif font on a red rectangular background.

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The example in the slide shows the statement that is used to create the `TEACH_EMP` table.

## Violating Constraints

```
UPDATE employees  
SET    department_id = 55  
WHERE  department_id = 110;
```

```
Error starting at line 1 in command:  
UPDATE employees  
SET    department_id = 55  
WHERE  department_id = 110  
Error report:  
SQL Error: ORA-02291: integrity constraint (ORA1.EMP_DEPT_FK) violated - parent key not found  
02291. 00000 - "integrity constraint (%s.%s) violated - parent key not found"  
*Cause:      A foreign key value has no matching primary key value.  
*Action:     Delete the foreign key or add a matching primary key.
```

Department 55 does not exist.

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

When you have constraints in place on columns, an error is returned if you try to violate the constraint rule. For example, if you try to update a record with a value that is tied to an integrity constraint, an error is returned.

In the example in the slide, department 55 does not exist in the parent table, DEPARTMENTS, and so you receive the “parent key not found” violation ORA-02291.

## Violating Constraints

You cannot delete a row that contains a primary key that is used as a foreign key in another table.

```
DELETE FROM departments
WHERE department_id = 60;
```

```
Error starting at line 1 in command:
DELETE FROM departments
WHERE department_id = 60
Error report:
SQL Error: ORA-02292: integrity constraint (ORA1.JHIST_DEPT_FK) violated - child record found
02292. 00000 - "integrity constraint (%s.%s) violated - child record found"
*Cause:      attempted to delete a parent key value that had a foreign
              dependency.
*Action:     delete dependencies first then parent or disable constraint.
```

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

If you attempt to delete a record with a value that is tied to an integrity constraint, an error is returned.

The example in the slide tries to delete department 60 from the `DEPARTMENTS` table, but it results in an error because that department number is used as a foreign key in the `EMPLOYEES` table. If the parent record that you attempt to delete has child records, you receive the “child record found” violation `ORA-02292`.

The following statement works because there are no employees in department 70:

```
DELETE FROM departments
WHERE department_id = 70;
```

```
1 rows deleted
```

## Lesson Agenda

- Database objects
  - Naming rules
- Data types
- CREATE TABLE statement
- Overview of constraints: NOT NULL, UNIQUE, PRIMARY KEY, FOREIGN KEY, CHECK constraints
- **Creating a table using a subquery**
- ALTER TABLE statement
- DROP TABLE statement

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Creating a Table Using a Subquery

- Create a table and insert rows by combining the `CREATE TABLE` statement and the `AS subquery` option.

```
CREATE TABLE table
      [(column, column...)]
AS subquery;
```

- Match the number of specified columns to the number of subquery columns.
- Define columns with column names and default values.

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

A second method for creating a table is to apply the `AS subquery` clause, which both creates the table and inserts rows returned from the subquery.

In the syntax:

<i>table</i>	Is the name of the table
<i>column</i>	Is the name of the column, default value, and integrity constraint
<i>subquery</i>	Is the <code>SELECT</code> statement that defines the set of rows to be inserted into the new table

### Guidelines

- The table is created with the specified column names, and the rows retrieved by the `SELECT` statement are inserted into the table.
- The column definition can contain only the column name and default value.
- If column specifications are given, the number of columns must equal the number of columns in the subquery `SELECT` list.
- If no column specifications are given, the column names of the table are the same as the column names in the subquery.
- The column data type definitions and the `NOT NULL` constraint are passed to the new table. Note that only the explicit `NOT NULL` constraint will be inherited. The `PRIMARY KEY` column will not pass the `NOT NULL` feature to the new column. Any other constraint rules are not passed to the new table. However, you can add constraints in the column definition.