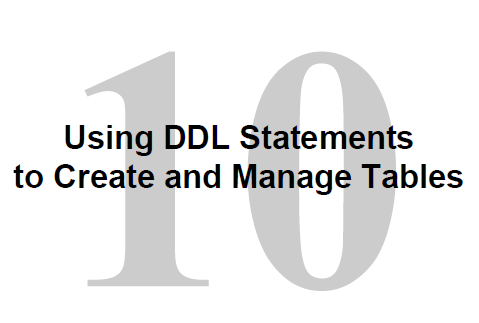
Les10-create manage tables

Chapter 10



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

Alter the table

Delete the table

# • Explain how constraints are created at the time of table creation

Inline constraints

Out of line constraints

Exception complaints

# • Describe how schema objects work

**Lesson Agenda**

• Database objects

– Naming rules

• CREATE TABLE statement:

– Access another user’s tables

– DEFAULT option

• Data types

• Overview of constraints: NOT NULL, UNIQUE, PRIMARY

KEY, FOREIGN KEY, CHECK constraints

• Creating a table using a subquery

• ALTER TABLE

– Read-only tables

## • DROP TABLE statement

Database Objects

|  |  |
| --- | --- |
| OBJECT | DESCRIPTION |
| Table | Basic storage unit composed of row of data divided into columns of fields |
| View | Logical representation of data from 1 or more tables. Not physical |
| Sequence | Generate numeric values. Example invoice numbers, employee id etc. |
| Index | Improves performance for some queries. Done later |
| Synonym | Gives an alternate name to an object. |

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* THERE ARE OTHER OBJECTS \*\*\*\*\*\*\*\*\*\*\*\*

A **sequence** is an object in **Oracle** that is used to generate a number **sequence**. This can be useful when you need to create a unique number to act as a primary key. 🡺autonumber

An **index** helps just like an index in a book or manual that speeds up the search through the manual. It will make queries faster. Not all tables require an index

A **synonym** is an alternative name for objects such as tables, views, sequences, stored procedures, and other database objects. You generally use **synonyms** when you are granting access to an object from another schema and you don't want the users to have to worry about knowing which schema owns the 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.

.

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

🡺 🡺 Names are NOT case sensitive

**Lesson Agenda**

• Database objects

– Naming rules

• CREATE TABLE statement:

– Access another user’s tables

– DEFAULT option

• Data types

• Overview of constraints: NOT NULL, UNIQUE, PRIMARY

KEY, FOREIGN KEY, CHECK constraints

• Creating a table using a subquery

• ALTER TABLE

– Read-only tables

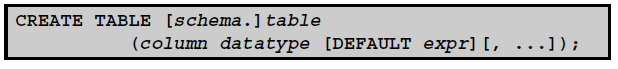
• DROP TABLE statement

Create table

You must have (a) privilege

(b) must be storage area

General Syntax



Basically, you provide

- Table name

- Column name

- Data type

- Size

NOTE: because as students we are working in one schema, the schema name can be left off

Referencing table of other users

Schema is a collection of logical structures of data known as a schema object

Schemas are owned by a user and has the same name as that user.

Each owner has a single schema.

To access table in another schema you need to qualify the name. Similar to iSeries in DBS201

SELECT \*

FROM PREMIER.employees;

Create table

# **Create a table with 4 columns**.

Specified:

1 Table name 🡺 dept

2 Column Names

3 Data type of each column

4 Size (date has a specific size built in)

**CREATE TABLE dept**

**( deptno NUMBER(2),**

**dname VARCHAR2(14),**

**loc VARCHAR2(13),**

**create\_date DATE);**

# **Confirm creation**

**DESCRIBE dept**

# **How to query what tables you own**

SELECT TABLE\_NAME

FROM USER\_TABLES;

Default Option

- You can specify a default option that effects data being inserted

- the default data type must match the column data type

EXAMPLE:

CREATE TABLE HIRE\_DATES

( ID NUMBER(8),

HIRE\_DATE DATE **DEFAULT SYSDATE**);

On the insert, if no value specified the system date will be inserted along with the ID

- You can specify NULL if you don't want any data

**Lesson Agenda**

• Database objects

– Naming rules

• CREATE TABLE statement:

– Access another user’s tables

– DEFAULT option

• Data types

• Overview of constraints: NOT NULL, UNIQUE, PRIMARY

KEY, FOREIGN KEY, CHECK constraints

• Creating a table using a subquery

• ALTER TABLE

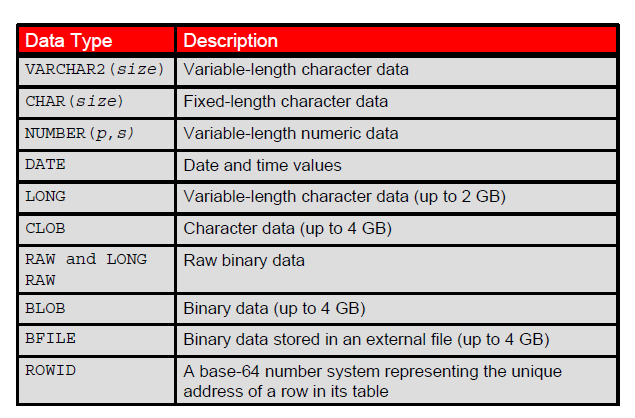
– Read-only tables

• DROP TABLE statement

Data types available

(Not all will be covered on this course)

The first 4 will be the most common



VARCHAR2 (size) Maximum size need to be specified (up to 4000)

CHAR Fixed Length size to maximum 2000

NUMBER (p, s) P is precision or total number of decimal digit and

S is scale or number of digits to the right of the decimal point

EX: NUMBER (5, 2) means 5 all together and 2 decimal places

The value 1000 will be rejected by the server as that is 6 wide

DATE Date and Time value to the nearest second

Range: Jan 1, 4712 BC and Dec 31, 9999

==========================================================================

LONG data type is variable length up to @GB

CLOB character data up to 4GB

Others you will encounter in later courses

**Data Type Description**

RAW(*size*) Raw binary data of length *size* (A maximum *size* must be specified: maximum *size* is 2,000.)

LONG RAW Raw binary data of variable length (up to 2 GB)

BLOB Binary data (up to 4 GB)

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

DATETIME data types

TIMESTAMP - By default this is in microsecond

- EX: 12-MAR-15 08:45.23.123456 🡸 6 decimals for microseconds

TIMESTAMP (0) - removes part seconds

TIMESTAMP (9) - can go to nanoseconds

On more advanced courses

INTERVAL to MONTH 15-4 is how it is store 15 years 4 months

- Instead of 5773 days

INTERVAL TO SECOND 3 12:42:21 3 days 12 hours etc…

- Use in biochemistry and other fields o mark time passed

Lesson Agenda

• Database objects

– Naming rules

• CREATE TABLE statement:

– Access another user’s tables

– DEFAULT option

• Data types

• Overview of constraints: NOT NULL, UNIQUE, PRIMARY KEY, FOREIGN KEY, CHECK constraints

• Creating a table using a subquery

• ALTER TABLE

– Read-only tables

• DROP TABLE

Constraints

Constraints enforce rules at the table level

- prevent adding data to a child table without the parent table data

- prevents deleting parent table or row when child data exists

# **Types of Constraints**

|  |  |
| --- | --- |
| NOT NULL | - SPECIFY DATA CANNOT BE null |
| UNIQUE | - PREVENTS DUPLICATION OF DATA INTHAT ROW |
| PRIMARY KEY | Unique identifier for each row in a table  Aside: It is both NT NULL and UNIQUE |
| 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 TRUE condition |

Constraint Guidelines

# • You can name a constraint,

# OR

# The Oracle server generates a name by using the SYS\_C*n* format.

# 🡺 best to name the constraint

# • 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 🡨 vis the ALTER statement

# • Define a constraint at the column or table level.

-- INLINE – done a the column is define

-- OUTLINE – done at end of column definitions

# • View a constraint in the data dictionary.

# NOTE:

One constraint can only be done at the table level

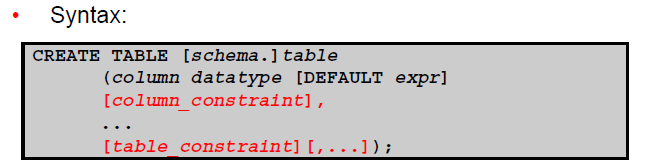
When more than 1 key is needed for the Primary Key

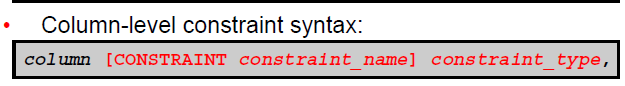
Composite key

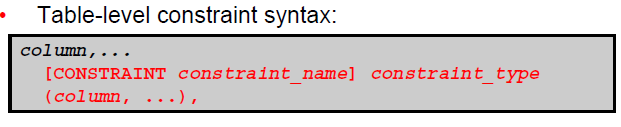
Concatenated key

Constraints – defined

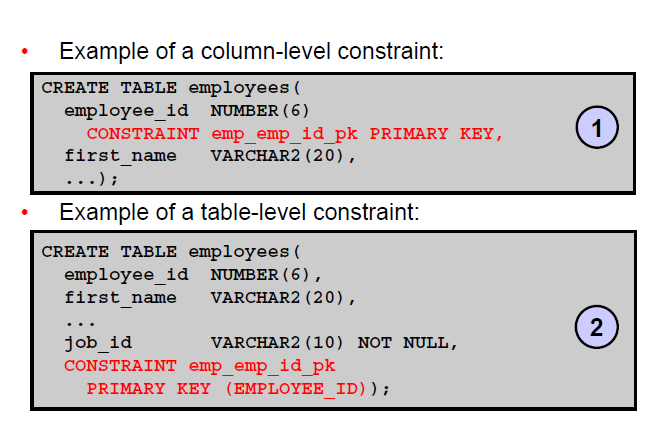
Generic syntax:







CONSTRAINT – Examples



NOTE:

**COLUMN LEVEL**

The comma goes AFTER the constraint on the column level

The column name is NOT specified because it is on the column definition

Naming convention – TABLE\_COLUMN\_TYPE

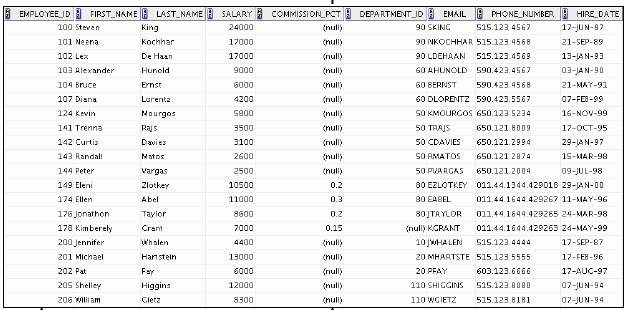
**TABLE LEVEL**

After the last comma

Column needs to be specified

NOT NULL -- What does it do?

Sample data in a table

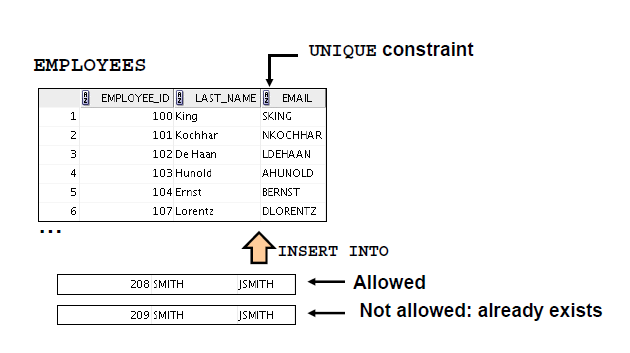


PK -- enforces a NOT NULL and UNQUE

This column has no constraint so can be NULL

The remainder have NOT NULL constraints

UNIQUE



NOTE:

1 Can describe a COMPOSITE UNIQUE KEY made of more than one column

2 Can contain NULL – Null satisfies the UNIQUE because null is not equal to anything

3 On a composite Unique you cannot have duplicates in the non-null part

UNIQUE CONSTRAINT

# Example: At the TABLE 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),**

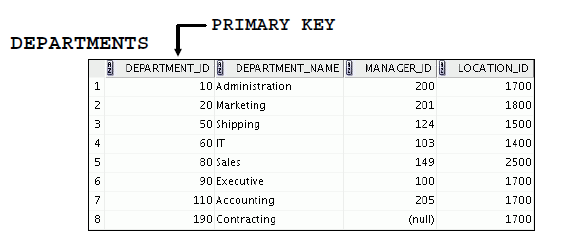
**hire\_date DATE NOT NULL,**

**...**

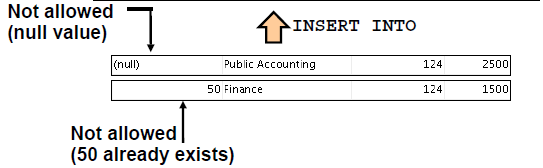
**CONSTRAINT emp\_email\_uk UNIQUE(email));**

PRIMARY KEY

**EXISTING TABLE:**



**Example of Insert data**

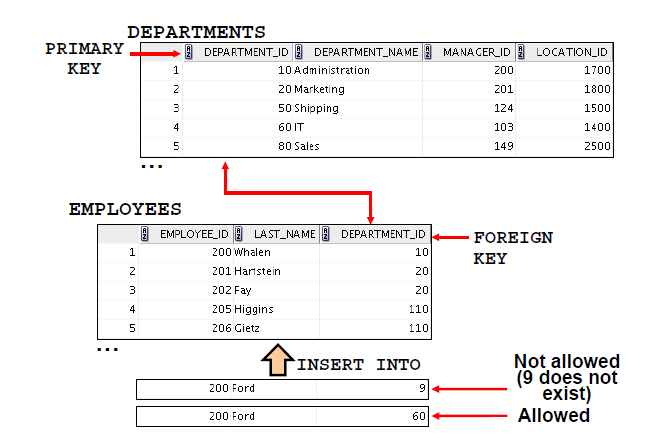


FOREIGN KEY

Established referential integrity.

Designate a column or group of columns as a foreign key.

Establishes a relationship with a primary key or a unique key in the same table or a different table.



**NOTE: Foreign key MUST match an existing value in the parent table**

**OR … it must be NULL**

Foreign Key – table 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));**

Foreign Key – column level

**CREATE TABLE employees**

**(...**

**department\_id NUMBER(4) CONSTRAINT emp\_deptid\_fk**

**REFERENCES departments(department\_id),**

**...**

**)**

**NOTE:**

**Foreign Key not stated as the reference handles that concept**

**Still need to name it and specify what if references**

FOREIGN KEY

Constraint: Keywords

• FOREIGN KEY: Defines the colum in the child table at the table-constraint level

• REFERENCES: Identifies the table and column in the parent table

2 more options- later in course

The parent record cannot be changed or deleted if there exists an FK in the child table that references it. This is done to maintain the integrity of the data

• 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

CHECK CONSTRAINT

# **Defines a condition that each row must satisfy in order to be added to the table**

**EXAMPLE:**

**CREATE TABLE EMPLOEES**

**( …. other columns**

**salary NUMBER(2)**

**CONSTRAINT employees\_salary\_min CHECK (salary > 0),**

**...**

**)**

FULL EXAMPLE

CREATE TABLE employees

( employee\_id NUMBER(6)

CONSTRAINT emp\_employee\_id PRIMARY KEY

, first\_name VARCHAR2(20)

, last\_name VARCHAR2(25)

CONSTRAINT emp\_last\_name\_nn NOT NULL

, email VARCHAR2(25)

Can have 2 constraints

CONSTRAINT emp\_email\_nn NOT NULL

CONSTRAINT emp\_email\_uk UNIQUE

, phone\_number VARCHAR2(20)

, hire\_date DATE

CONSTRAINT emp\_hire\_date\_nn NOT NULL

, job\_id VARCHAR2(10)

CONSTRAINT emp\_job\_nn NOT NULL

, salary NUMBER(8,2)

CONSTRAINT emp\_salary\_ck CHECK (salary>0)

, commission\_pct NUMBER(2,2)

, manager\_id NUMBER(6)

CONSTRAINT emp\_manager\_fk REFERENCES employees (employee\_id)

, department\_id NUMBER(4)

CONSTRAINT emp\_dept\_fk REFERENCES departments (department\_id)

);

Violating Constraints

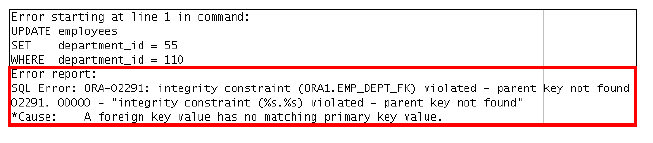
# **Changing department ID from 110 to 55**

**UPDATE employees**

**SET department\_id = 55**

**WHERE department\_id = 110;**

# **Here is the error**



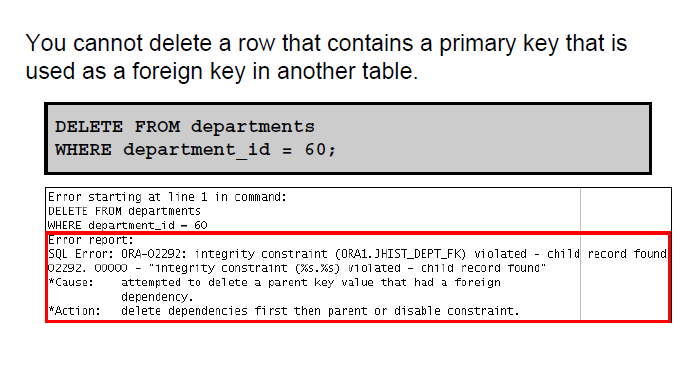
NOTE: The name of constraint is stated in the error message. From that message we know

Table: EMP or EMPLOYEES

Column: Department\_id

Cause: A foreign key value has no matching PK value

Violating Constraints – on a delete



# **EXAMPLE of a delete that works**

DELETE FROM departments

WHERE department\_id = 70; 🡸 because there are no employees in department 70

Lesson Agenda

• Database objects

– Naming rules

• CREATE TABLE statement:

– Access another user’s tables

– DEFAULT option

• Data types

• Overview of constraints: NOT NULL, UNIQUE, PRIMARY KEY, FOREIGN KEY, CHECK constraints

• Creating a table using a subquery

• ALTER TABLE

– Read-only tables

• DROP TABLE statement

Creating a table – via a subquery

• Create a table and insert rows by combining the

CREATE TABLE statement and the

AS *subquery* option.

Generic Syntax:

**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

-- see examples

CREATE TABLE using subquery

**CREATE TABLE dept80**

**AS**

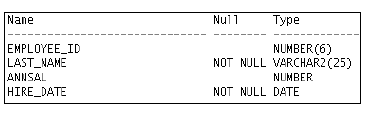
**SELECT employee\_id, last\_name, salary\*12 ANNSAL,hire\_date**

**FROM employees**

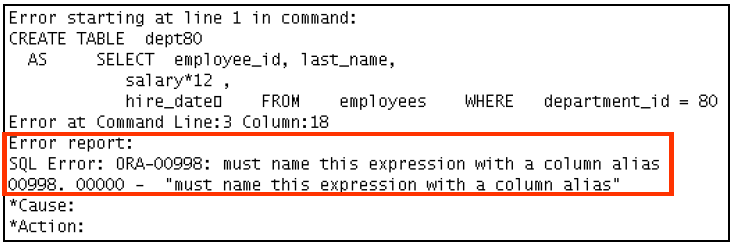
**WHERE department\_id = 80;**

DESCRIBE dept80

Notice an alias for a non-existing colummn



# **Alias not used 🡺 generates an error**



Lesson Agenda

• Database objects

– Naming rules

• CREATE TABLE statement:

– Access another user’s tables

– DEFAULT option

• Data types

• Overview of constraints: NOT NULL, UNIQUE, PRIMARY KEY, FOREIGN KEY, CHECK constraints

• Creating a table using a subquery

• ALTER TABLE

– Read-only tables

• DROP TABLE statement

ALTER TABLE Statement

Use the ALTER TABLE statement to:

• Add a new column

• Modify an existing column definition

• Define a default value for the new column

• Drop a column

• Rename a column

• Change table to read-only status

Read-Only Tables

# **When changes are being make to the table it is a good idea to put the table into read-only mode**

ALTER TABLE employees READ ONLY;

---- make the changes

ALTER TABLE employees READ WRITE;

In the environment of a student that wouldn't be done as you ar not updating data in a table at the same time as maintenance a you are the only user.

ASIDE:

You can drop a table that is in the READ ONLY mode. The DROP command is executed only in the data dictionary, so access to the table contents is not required. The space used by the table will not be reclaimed until the tablespace is made read/write again, and then the required changes can be made to the block segment headers, and so on.

Lesson Agenda

• Database objects

– Naming rules

• CREATE TABLE statement:

– Access another user’s tables

– DEFAULT option

• Data types

• Overview of constraints: NOT NULL, UNIQUE, PRIMARY KEY, FOREIGN KEY, CHECK constraints

• Creating a table using a subquery

• ALTER TABLE

– Read-only tables

• DROP TABLE statement

Finally at the end topic

Dropping a Table

DROP TABLE employees -- (don't test this)

Moves a table to the recycle bin

Removes the table and all its data entirely

- If PURGE specified it releases space for other objects to use

Invalidates dependent objects and removes object privileges on the table

- Views using the table remain, but are invalid

NOTE: There is a way of getting a table back from the recycle bin … see FLASBACK in a later course

**Quiz**

You can use constraints to do the following:

1. Enforce rules on the data in a table whenever a row is inserted, updated, or deleted.

2. Prevent the deletion of a table.

3. Prevent the creation of a table.

4. Prevent the creation of data in a table.

124