CE1

Guide to Oracle 10*g*

Chapters 2 & 3: Creating, Modifying Database Tables and Manipulating Data into them

Edited by:

Dr. Emad Al-Shawakfa
For the CIS 360Course
Department of Computer Information Systems
Yarmouk University
2020

Objectives

After completing these chapters' review, you should be able to do the following:

- Use structured query language (SQL*Plus) commands to create, modify, and drop database tables.
- Use SQL*Plus commands to manipulating data in the database tables.
- Explain Oracle 10*g* user schemas
- Create database tables using SQL*Plus
- Debug Oracle 10*g* SQL*Plus commands.

Slide 1

CE1 Global to this PPT: Please note that the titles of figure slides are inconsistent. At times the heading is used as the title, and other the figure caption is. I've commented the individual slides.
CE, 8/1/2005

Introduction to SQL

- Structured query language (SQL)
  - Standard query language for relational databases
  - Consists of about 30 commands
  - Enables users to create database objects and manipulate and view data
  - Two major categories of commands; Data Definition (DDL) and Data Manipulation (DML).
  - SQL Commands are reserved words.
  - Standard query language for relational databases

Database Objects

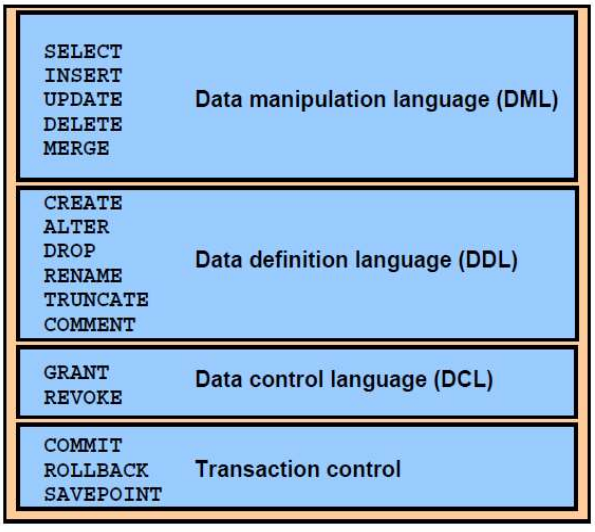
Object	Description
Table	Basic unit of storage; composed of rows
View	Logically represents subsets of data from one or more tables
Sequence	Generates numeric values
Index	Improves the performance of some queries
Synonym	Gives alternative names to objects

Oracle Naming Standard

- Oracle database objects must adhere to the *Oracle Naming Standard*
  - 1 to 30 characters long
  - Must begin with a character
  - Can contain characters, numbers, and the symbols $, _, and #
  - Must not duplicate the name of another object owned by the same user
  - Must not be an Oracle server–reserved word

5

SQL Statements



Guide to Oracle 10g

6

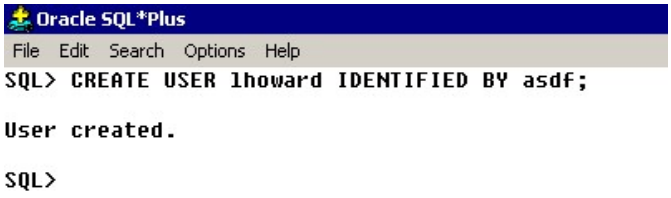
Oracle 10*g* User Accounts

- User account
  - Created for each user
  - Identified using unique username and password
- User schema
  - Area of database belonging to user
- Database objects
  - Also called schema objects
  - Objects in user schema

Creating New User Accounts

- Done by a DataBase Administrator (*DBA*)
- Syntax:

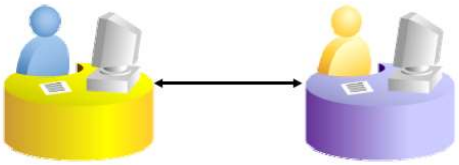
```
CREATE username IDENTIFIED BY password;
```



8

User Schema

- Tables belonging to other users are not in the user's schema.
- You should use the owner's name as a prefix to those tables.



USERA

```
SELECT *
FROM userB.employees;
```

USERB

```
SELECT *
FROM userA.employees;
```


Creating and Editing SQL Commands Using a Text Editor

- Good approach for entering commands:
  - Type commands into text editor such as Notepad
  - Copy commands, then paste into SQL*Plus
  - Execute commands

Using Scripts to Create or Manipulate Database Tables

- Script
  - Text file that contains several related SQL commands
- Run a script
  - Type `start` at SQL prompt
  - Blank space
  - Full path and filename of script file

Data Definition Language Commands

Create Table *Statement*

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

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

- **You need to specify:**
  - ***Table name***
  - ***Column name, column data type, column size, and any constraints needed.***

DEFAULT *Option*

- Specify a default value for a column during an insert.

```
... 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 created.
```

Oracle 10*g* Data Types

Data Type	Description
VARCHAR2(*size*)	Variable-length character data
CHAR(*size*)	Fixed-length character data
NUMBER(*p*,*s*)	Variable-length numeric data
DATE	Date and time values
LONG	Variable-length character data (up to 2 GB)
CLOB	Character data (up to 4 GB)
RAW and LONG RAW	Raw binary data
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

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

Datetime data types

- **The TIMESTAMP data type is an extension of the DATE data type.**
- **It stores the year, month, and day of the DATE data type plus hour, minute, and second values as well as the fractional second value.**
- **You can optionally specify the time zone.**

Datetime data types

- The `INTERVAL YEAR TO MONTH` data type stores a period of time using the `YEAR` and `MONTH` datetime fields:

```
INTERVAL YEAR [(year_precision)] TO MONTH
```

- The `INTERVAL DAY TO SECOND` data type stores a period of time in terms of days, hours, minutes, and seconds:

```
INTERVAL DAY [(day_precision)]
   TO SECOND [(fractional_seconds_precision)]
```

Date And Time Data Types (continued)

- DATE
  - Stores dates from December 31, 4712 BC to December 31, AD 4712
  - Default date format
    - DD-MON-YY
  - Default time format
    - HH:MI:SS AM
  - Syntax:
    - *columnname* DATE

Large Object (LOB) Data Types

- Store binary data such as:
  - Digitized sounds or images
  - References to binary files from word processor or spreadsheet
- General syntax
  - *columnname Lob_data_type*

Large Object (LOB) Data Types (continued)

Large Object (LOB) Data Type	Description
BLOB	Binary LOB, storing up to 4 GB of binary data in the database
BFILE	Binary file, storing a reference to a binary file located outside the database in a file maintained by the operating system
CLOB	Character LOB, storing up to 4 GB of character data in the database
NCLOB	Character LOB that supports 2-byte character codes, stored in the database—up to a maximum of 4 GB

Table 2-1 Large object (LOB) data types

Constraints

- Constraints enforce rules at the table level.
- Constraints prevent the deletion of a table if there are dependencies.
- The following constraint types are valid:
  - NOT NULL
  - UNIQUE
  - PRIMARY KEY
  - FOREIGN KEY
  - CHECK

Defining Constraints

- Syntax:

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

- Column-level constraint:

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

- Table-level constraint:

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


Defining Constraints Example

- Column-level constraint:

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

1

- 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

Constraints 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 table is created
  - After the table has been created
- Define a constraint at the column or table level.
- View a constraint in the data dictionary.

Constraints (continued)

- Constraint naming convention
  - *tablename_columnname_constraintid*

CE2

Common ConstraintID Abbreviations

Constraint Type	ConstraintID Abbreviation	Oracle10g Constraint Type Identifier
PRIMARY KEY	pk	P
FOREIGN KEY	fk	R
CHECK CONDITION	cc	C
NOT NULL	nn	N
UNIQUE	uk	U

Table 2-2 Common constraintID abbreviations

Integrity Constraints

- Primary key
  - Syntax (within table definition)
    - `CONSTRAINT` *`constraint_name`* `PRIMARY KEY`
  - Syntax (at end of table definition)
    - `CONSTRAINT` *`constraint_name`* `PRIMARY KEY (`*`columnname`*`)`

Slide 27

CE2 Title of this slide is figure caption, rather than A-head ("Constraints (continued)") - is this okay given other slides with figures (slide 21, for example) use the A-head as the title?
CE, 7/29/2005

Example

```
CREATE TABLE location
(loc_id NUMBER(6),
bldg_code VARCHAR2(10),
room VARCHAR2(6),
capacity NUMBER(5),
CONSTRAINT location_loc_id_pk PRIMARY KEY (loc_id));
```

OR

```
CREATE TABLE location
(loc_id NUMBER(6)
CONSTRAINT location_loc_id_pk PRIMARY KEY,
bldg_code VARCHAR2(10),
room VARCHAR2(6),
capacity NUMBER(5));
```


Integrity Constraints (continued)

- Foreign key
  - Column constraint
  - Specifies that value user inserts in column must exist as primary key in referenced table
  - Syntax (placed at end of table definition)

    CONSTRAINT *constraint_name*

    FOREIGN KEY (*columnname*)

    REFERENCES *primary_key_tablename* (*primary_key_columnname*)

Integrity Constraints (continued)

- Foreign key (continued)
  - Syntax (placed within table definition)

    CONSTRAINT *constraint_name*
    REFERENCES *primary_key_tablename*
    (*primary_key_columnname*)
- Composite key
  - Syntax
    - CONSTRAINT *constraint_name*
    - PRIMARY KEY (*columnname1*, *columnname2* …)

Example

In the Northwoods ENROLLMENT table, the primary key consists of both the S_ID and the C_SEC_ID columns. The command to create the ENROLLMENT table and specify that the S_ID and C_SEC_ID columns make up the composite primary key is:

```
CREATE TABLE enrollment
(s_id NUMBER(5) CONSTRAINT enrollment_s_id_fk
REFERENCES student(s_id),
c_sec_id NUMBER(8) CONSTRAINT enrollment_c_sec_id_fk
REFERENCES course_section(c_sec_id),
CONSTRAINT enrollment_s_id_c_sec_id_pk
PRIMARY KEY (s_id, c_sec_id));
```

Integrity Constraints (continued)

- Value constraints
  - Column-level constraints
  - Restrict data values that users can enter
  - Commonly used value constraints
    - CHECK conditions
    - NOT NULL constraint
    - DEFAULT constraint
    - UNIQUE constraint

Examples

Ex1:

As an example of a check condition, consider the S_CLASS column in the Northwoods University STUDENT table, in which the values are restricted to FR, SO, JR, or SR (freshman, sophomore, junior, or senior). The syntax to define this check condition is:

```
CONSTRAINT student_s_class_cc CHECK
((s_class = 'FR') OR (s_class = 'SO')
OR (s_class = 'JR') OR (s_class ='SR'))
```

Ex2:

You can also use a check condition to validate a range of allowable values. An example of a range check condition is in the CREDITS column in the Northwoods COURSE table, where the allowable values must be greater than 0 and less than 12. The constraint definition is:

```
CONSTRAINT course_credits_cc
CHECK((credits > 0) AND (credits < 12))
```

Using Oracle Online Help Resources to Debug SQL Commands

- Syntax error
  - SQL*Plus interpreter displays error information
    - Line number within command that caused error
    - Position of error within line
    - Error code and description of error

Using Oracle Online Help Resources To Debug SQL Commands (continued)

- Oracle 10*g* error codes have:
  - 3-character prefix (such as ORA)
  - 5-digits
- Causes of SQL command errors are not always readily apparent
  - Need to retrieve more information about error
  - Connect to Oracle Technology Network (OTN) Web Site and search for error code

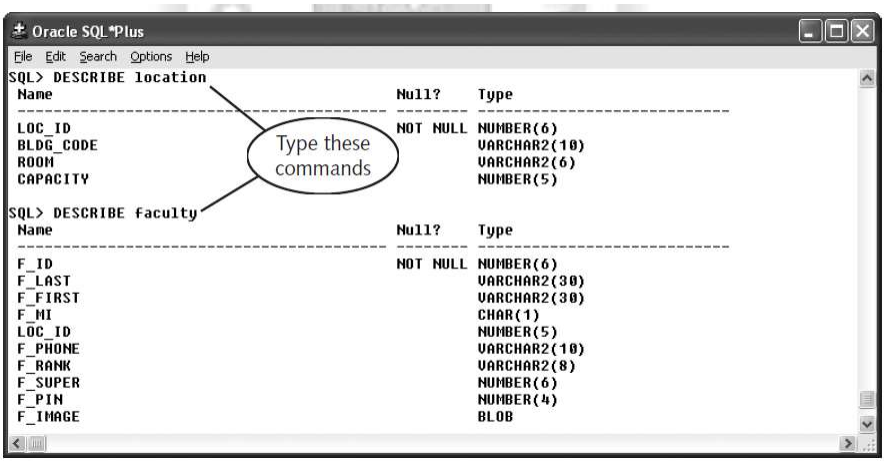
Using Oracle Online Help Resources To Debug SQL Commands (continued)

- Last resort debugging technique
  - Create table multiple times
  - Each time adding column declaration
  - Repeat process until you find declaration causing error

Viewing Information about Tables

- DESCRIBE command
  - View column names and data types of table
  - Syntax
    - `DESCRIBE` *`tablename`*
- Oracle 10*g* data dictionary
  - Consists of tables that contain information about structure of database

Example



Viewing Information about Tables (continued)

- Oracle10g data dictionary (continued)
  - System creates data dictionary in user schema named SYS
  - Users do not directly manipulate data dictionary
- View
  - Database object DBMS bases on actual database table
  - Enables DBMS to present table data in different format based on needs of users

Viewing Information about Tables (continued)

- Data dictionary views categories
  - USER
  - ALL
  - DBA
- Syntax
  - `SELECT view_columnname1, view_columnname2 … FROM prefix_object;`

CE4

Database Objects with Data Dictionary Views

Object Name	Object Type
OBJECTS	All database objects
TABLES	Database tables
INDEXES	Table indexes created to improve query retrieval performance
VIEWS	Database views
SEQUENCES	Sequences used to generate surrogate key values automatically
USERS	Database users
CONSTRAINTS	Table constraints
CONS_COLUMNS	Table columns that have constraints
IND_COLUMNS	Table columns that have indexes
TAB_COLUMNS	All table columns

Table 2-3 Database objects with data dictionary views

Slide 42

CE4 Again, this slide uses figure caption as title - is this okay given other slides with figures use the A-head as the title?

CE, 7/29/2005

Example

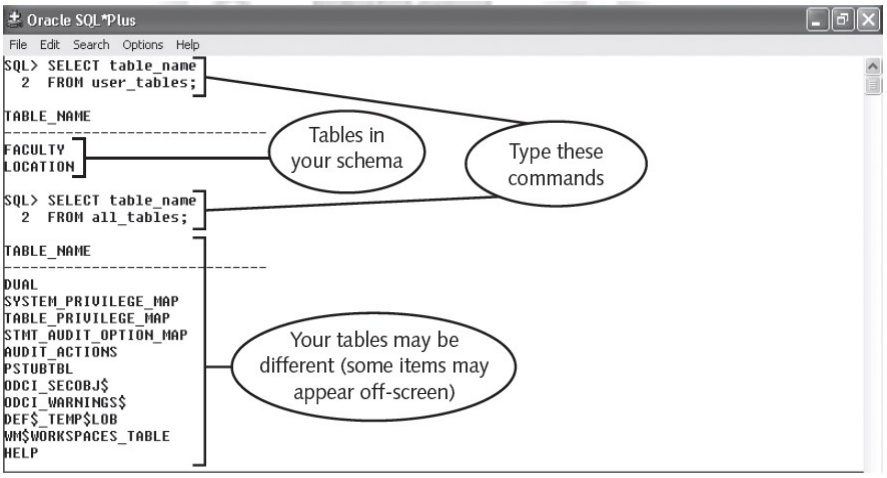


Figure 2-8 Retrieving table names from the USER_TABLES data dictionary view

Example

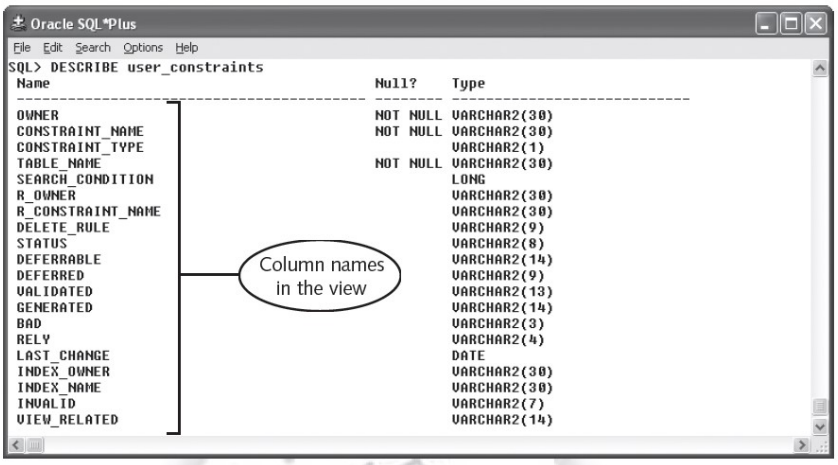


Figure 2-9 Column names in the USER_CONSTRAINTS data dictionary view

Example

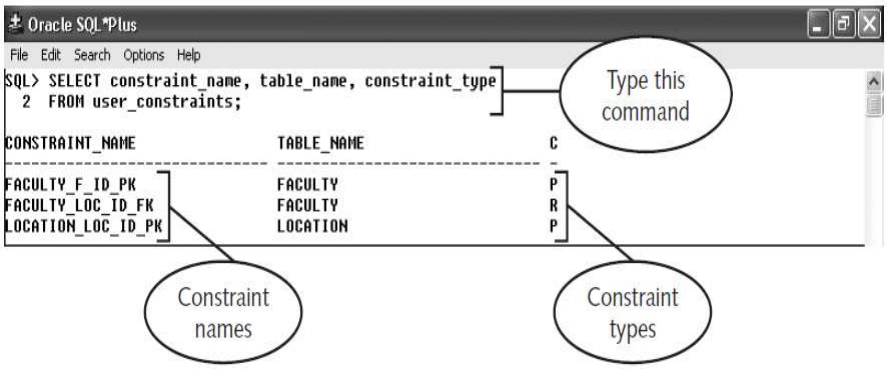


Figure 2-10 Viewing table constraint information

Example

To retrieve a list of all of the constraints for a specific database table in your user schema, you use the following command:

```
SELECT constraint_name, constraint_type
FROM user_constraints
WHERE table_name = 'DATABASE_TABLENAME';
```

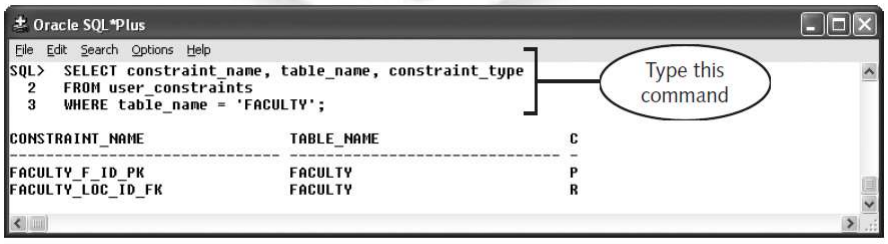


Figure 2-11 Viewing information about constraints in a specific table

Modifying and Deleting Database Tables

- Plan tables carefully to avoid having to change structure of database tables later
- Unrestricted action
  - Some specifications of tables can always be modified
- Restricted action
  - Table specifications that can be modified only in certain situations

CE5

Unrestricted Actions when Modifying Database Tables

Unrestricted Actions
Renaming a table
Adding new fields
Deleting fields
Increasing the *maximum_size* value of a field
Deleting constraints

Table 2-4 Unrestricted actions when modifying database tables

Slide 48

CE5 Again, this slide uses the figure caption as the title - is this okay given other slides with figures use the A-head as the title?

CE, 7/29/2005

Restricted Actions

Restricted Actions	Restriction
Deleting a table from a user schema	Allowed only if the table does not contain any columns that other tables reference as foreign keys
Changing an existing column's data type	Allowed only if existing data in the column is compatible with the new data type. For example, you could change a VARCHAR2 data type to a CHAR data type, but you could not change a VARCHAR2 data type to a NUMBER data type.
Decreasing the width of an existing column	Allowed only if existing column values are NULL
Adding a primary key constraint to an existing column	Allowed only if current column values are unique (no duplicate values) and NOT NULL
Adding a foreign key constraint	Allowed only if current column values are NULL, or exist in the referenced table
Adding a UNIQUE constraint to a column	Allowed only if current column values are all unique
Adding a CHECK constraint	Allowed, but the constraint applies only to values users insert after the constraint is added
Changing a column's default value	Allowed, but the default value is only inserted for rows that are added after the change

Table 2-5 Restricted actions when modifying database tables

Deleting and Renaming Existing Tables

- DROP TABLE command
  - Delete table
  - Syntax
    - `DROP TABLE tablename;`
    - `DROP TABLE tablename CASCADE CONSTRAINTS;`

Example

To drop the LOCATION table, along with all of the constraints that reference the table, follow these steps:

1. Type **DROP TABLE location;** at the SQL prompt, and then press **Enter**. The error message "ORA-02449: unique/primary keys in table referenced by foreign keys" appears as shown in Figure 2-12, which indicates that one or more columns in the LOCATION table are referenced as foreign keys in other tables.

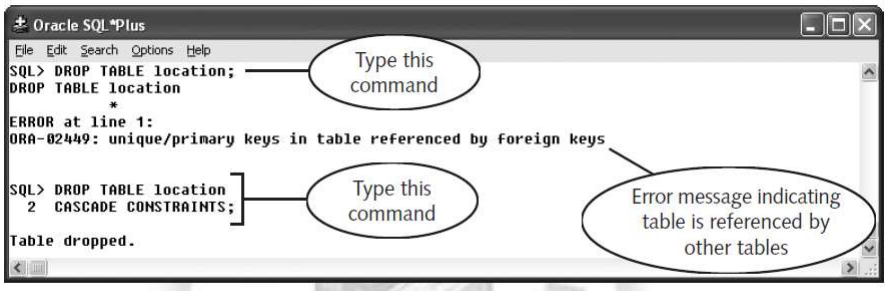


Figure 2-12 Commands to delete a database table

Deleting and Renaming Existing Tables (continued)

- RENAME TO command
  - Syntax
    - `RENAME old_tablename TO new_tablename;`

ALTER TABLE *Statement*

- **Use the ALTER TABLE statement to:**
  - **Add a new column**
  - **Modify an existing column**
  - **Define a default value for the new column**
  - **Drop a column**

Adding Columns to Existing Tables

- Add new column to table
  - Syntax

```
ALTER TABLE tablename
ADD(columnname data_declaration
  constraints);
```

To add the START_DATE column to the FACULTY table:

1. Type the following command at the SQL prompt:

```
ALTER TABLE faculty
ADD (start_date DATE);
```

Modifying Existing Column Data Definitions

- Modify existing column's data declaration
  - Syntax

```
ALTER TABLE tablename
MODIFY(columnname
  new_data_declaration);
```

To modify the data type and size of the F_RANK column:

1. At the SQL*Plus prompt, type the following command to modify the F_RANK column:

```
ALTER TABLE faculty
MODIFY (f_rank CHAR(4));
```


Deleting a Column

- Data stored in deleted column removed from database
- Syntax

```
ALTER TABLE tablename
DROP COLUMN columnname;
```

Renaming a Column

- Syntax

```
ALTER TABLE tablename
RENAME COLUMN old_columnname TO
  new_columnname;
```


Adding and Deleting Constraints

- Add constraint to existing table
  - Syntax

```
ALTER TABLE tablename
ADD CONSTRAINT constraint_name
  constraint_definition;
```

```
ALTER TABLE faculty
ADD CONSTRAINT faculty_f_pin_uk UNIQUE (f_pin);
```

- Remove existing constraint
  - Syntax

```
ALTER TABLE tablename
DROP CONSTRAINT constraint_name;
```

```
ALTER TABLE faculty
DROP CONSTRAINT faculty_f_pin_uk;
```

Enabling and Disabling Constraints

- Constraint enabled
  - DBMS enforces constraint when users attempt to add new data to database
- Disable existing constraint syntax:

```
ALTER TABLE tablename
DISABLE CONSTRAINT constraint_name;
```

```
ALTER TABLE faculty
DISABLE CONSTRAINT faculty_loc_id_fk;
```

- Enable existing constraint syntax:

```
ALTER TABLE tablename
ENABLE CONSTRAINT constraint_name;
```

```
ALTER TABLE faculty
ENABLE CONSTRAINT faculty_loc_id_fk;
```


Dropping Tables

- All data and structure in the table are deleted.
- Any pending transactions are committed.
- All indexes are dropped.
- All constraints are dropped.
- You *cannot* roll back the `DROP TABLE` statement.

```
DROP TABLE dept80;
Table dropped.
```

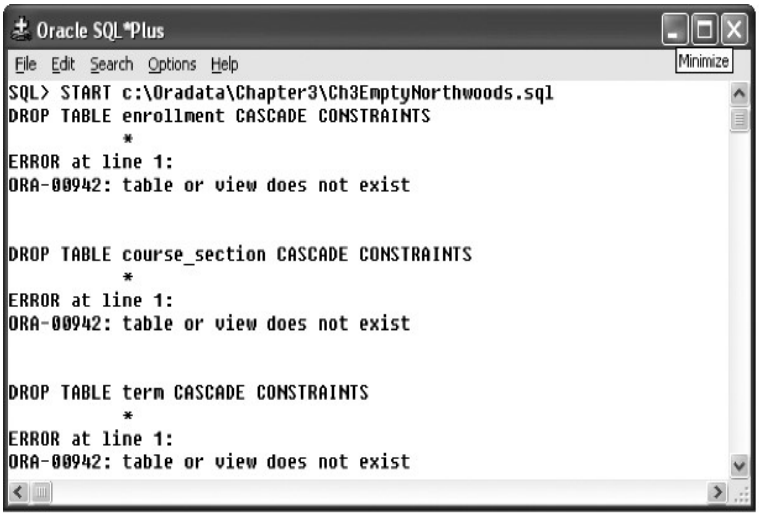


Figure 3-1 Error message that occurs while typing — script attempts to drop a nonexistent table

Data Manipulation Language Commands

DML Commands

- A DML statement is executed when you:
  - Add new rows to a table
  - Modify existing rows in a table
  - Remove existing rows from a table
- A *transaction* consists of a collection of DML statements that form a logical unit of work.

Creating Transactions and Committing New Data

- Transaction
  - Represents logical unit of work
  - All of action queries must succeed or no transactions can succeed
- Commit
  - Save changes in transaction
- Rollback
  - Discard changes in transaction

Creating Transactions and Committing New Data (continued)

- Purpose of transaction processing
  - Enable users to see consistent view of database
- New transaction begins when SQL*Plus started and command executed
- Transaction ends when current transaction committed
- COMMIT command commits transaction
- ROLLBACK command restores database to point before last commit

CE6

Using the INSERT Command

- Basic syntax for inserting into every column:

```
INSERT into tablename
VALUES (column1_value,
  column2_value, … );
```

- Basic syntax for inserting into selected columns

```
INSERT into tablename (columnname1,
  columnname2, … );
VALUES (column1_value,
  column2_value, … );
```

Slide 66

CE6 There is no slide for A-head "Inserting Data into Tables" (pg 88) - okay?
CE, 7/29/2005

Using the INSERT Command (continued)

- Ensure all foreign keys that new row references have already been added to database

Format Models

- Also called format mask
- Used to specify different output format from default
- For NUMBER data types
  - 9 represents digit
- For DATE/TIMESTAMP data types
  - Choose formats for year day, date, etc.

Format Models

- Also called format mask
- Used to specify different output format from default
- For NUMBER data types
  - 9 represents digit
- For DATE/TIMESTAMP data types
  - Choose formats for year day, date, etc.

Inserting Date and Interval Values

- Inserting values into DATE columns
  - Use TO_DATE function to convert string to DATE
  - Syntax
    - `TO_DATE('date_string', 'date_format_model')`
- Inserting values into INTERVAL columns
  - Syntax
    - `TO_YMINTERVAL('years-months')`
    - `TO_DSINTERVAL('days HH:MI:SS.99')`

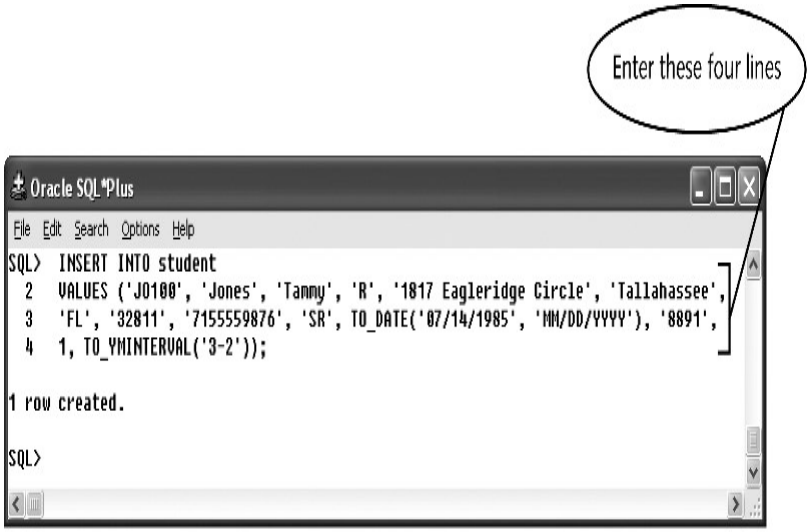


Figure 3-4 Action query to insert data values with DATE and INTERVAL data types

Inserting LOB Column Locators

- Oracle stores LOB data in separate physical location from other types of data
- LOB locator
  - Structure containing information that identifies LOB data type
  - Points to alternate memory location
- Create blob locator
  - `EMPTY_BLOB()`

```
INSERT INTO faculty
 ( F_ID, F_LAST, F_FIRST, F_IMAGE )

VALUES
(2, 'Zhulin', 'Mark', EMPTY_BLOB( ) );
```


CE7

Updating Table Rows

- UPDATE action query syntax

```
UPDATE tablename
SET column1 = new_value1, column2 =
  new_value2, …
WHERE search condition;
```

Slide 74

CE7 There is no slide for A-head "Updating and Deleting Existing Table Rows" (pg 102) - okay?
CE, 7/29/2005

Deleting Table Rows

- SQL DELETE action query
  - Remove specific rows
- Truncate table
  - Remove all rows
- DELETE query syntax

```
DELETE FROM tablename
WHERE search condition;
```

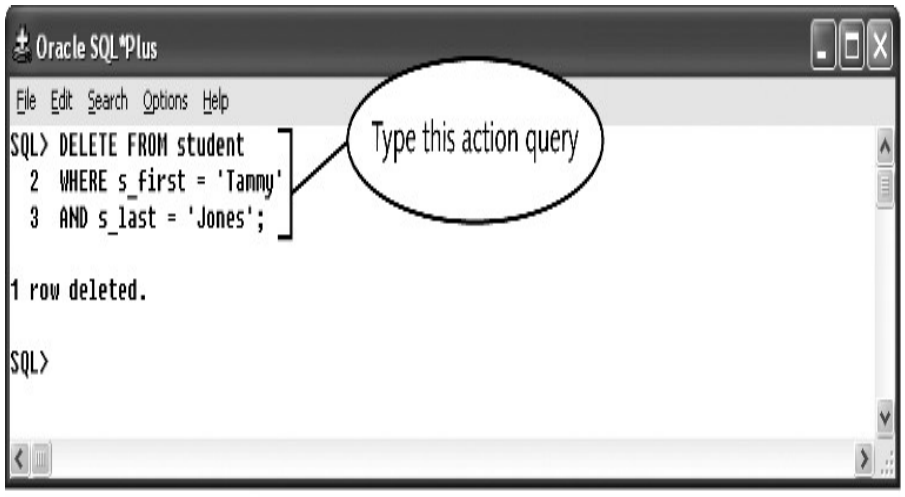



Figure 3-8 Creating a DELETE action query

Deleting Table Rows (continued)

- Child row
  - Row's value is foreign key
  - Cannot delete row if it has child row
    - Unless first delete row in which foreign key value exists
- TRUNCATE syntax
  - `TRUNCATE TABLE tablename;`
- Cannot truncate table with foreign key constraints
  - Must disable constraints first

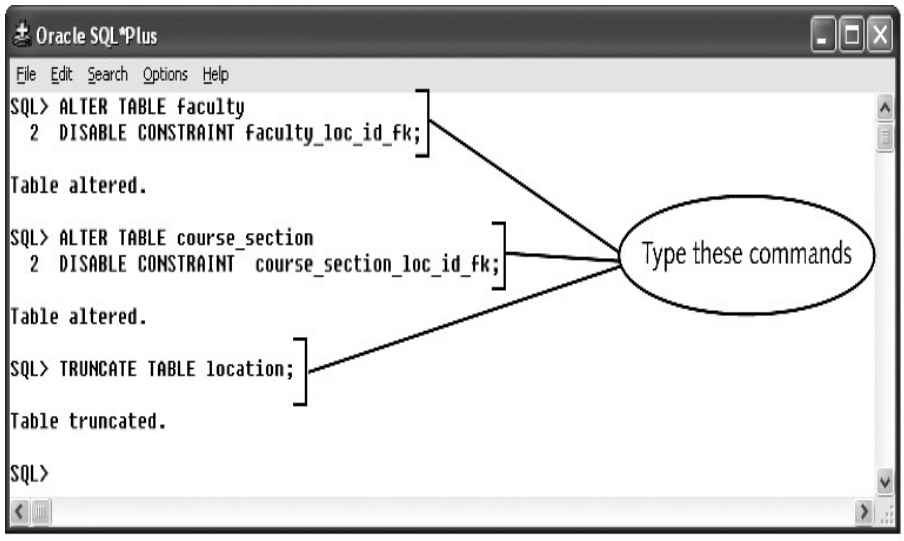


Figure 3-10 Commands that disable constraints and then truncate a table

CE8

Creating New Sequences

- CREATE SEQUENCE command
  - DDL command
  - No need to issue COMMIT command

CE9

General Syntax Used to Create a New Sequence

```
CREATE SEQUENCE sequence_name
   [INCREMENT BY number]
   [START WITH start_value]
   [MAXVALUE maximum_value] | [NOMAXVALUE]
   [MINVALUE minimum_value] | [NOMINVALUE]
   [CYCLE] | [NOCYCLE]
   [CACHE number_of_values] | [NOCACHE]
   [ORDER] | [NOORDER];
```

Figure 3-11 General syntax used to create a new sequence

Slide 79

CE8 There is no slide for A-head "Sequences" (pg 105) - okay?
CE, 7/29/2005

Slide 80

CE9 This slide's title is figure caption rather than A-head (like slide 23) - okay?
CE, 7/29/2005

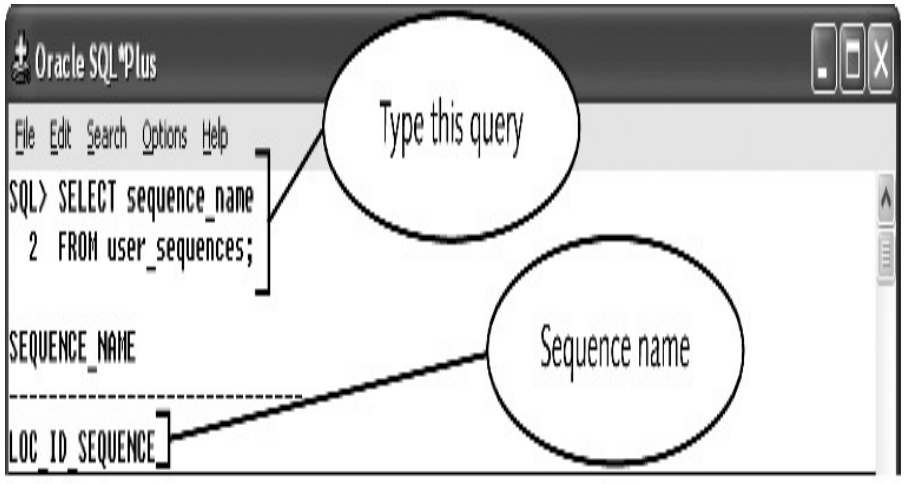


Figure 3-13 Viewing sequence names

Using Sequences

- Pseudocolumn
  - Acts like column in database table
  - Actually command that returns specific value
- CURRVAL
  - Returns most recent sequence value retrieved
- NEXTVAL
  - Next available sequence value
  - *sequence_name*.NEXTVAL

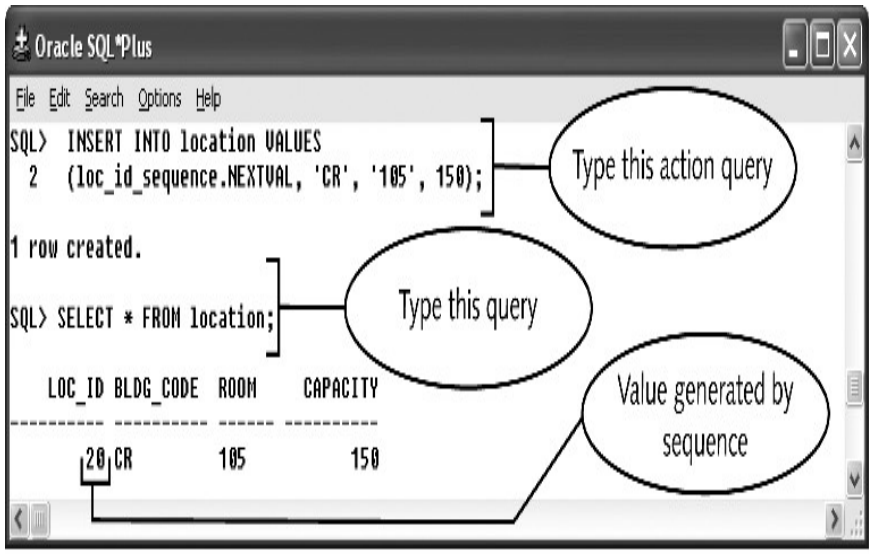


Figure 3-14 Inserting a new row using a sequence

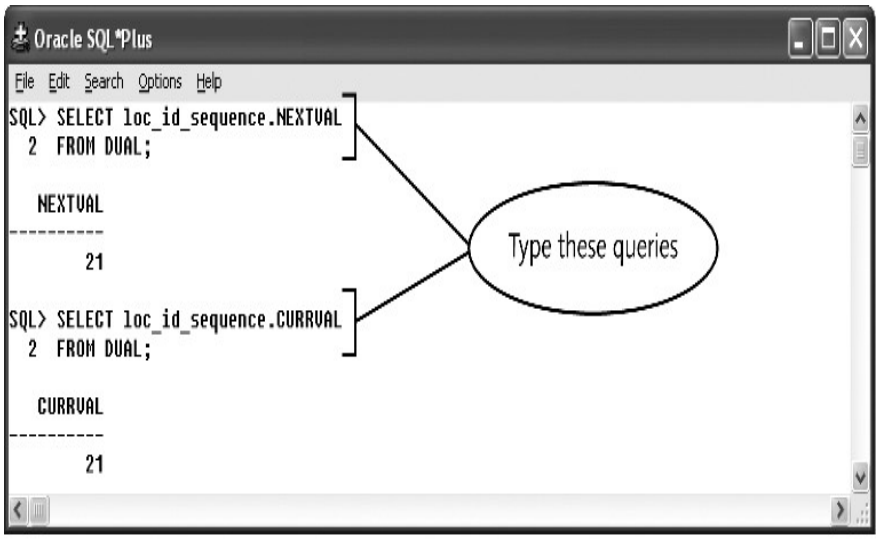


Figure 3-15 Using the NEXTVAL and CURRVAL pseudocolumns in SELECT queries

Granting Object Privileges

- SQL GRANT command
  - Syntax

```
GRANT privilege1, privilege2, …
ON object_name
TO user1, user2, …;
```


Database Object Privileges

Object Type(s)	Privilege	Description
Table, Sequence	ALTER	Allows the user to change the object's structure using the ALTER command
Table, Sequence	DROP	Allows the user to drop (delete) the object
Table, Sequence	SELECT	Allows the user to view table records or select sequence values
Table	INSERT, UPDATE, DELETE	Allows the user to insert, update, or delete table records
Any object	ALL	Allows the user to perform all possible operations on the object

Table 3-5 Commonly used Oracle10g object privileges

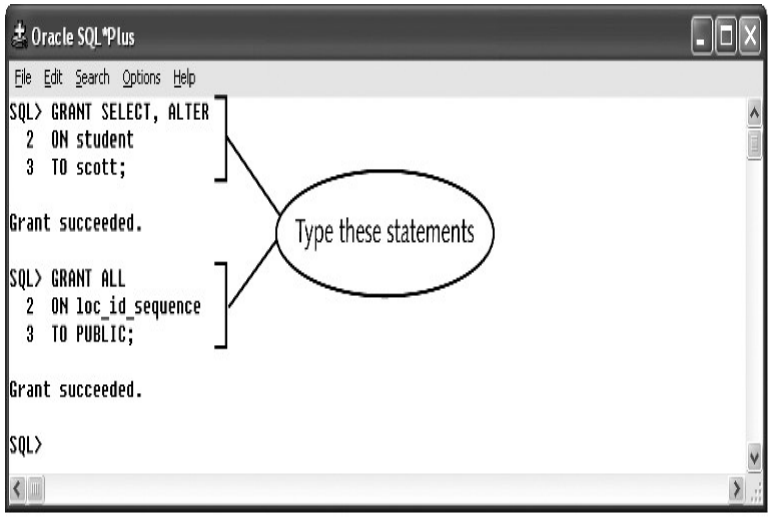


Figure 3-16 Granting object privileges

Revoking Table Privileges

- REVOKE command
  - Syntax

```
REVOKE privilege1, privilege2, …
ON object_name
FROM user1, user2, …;
```

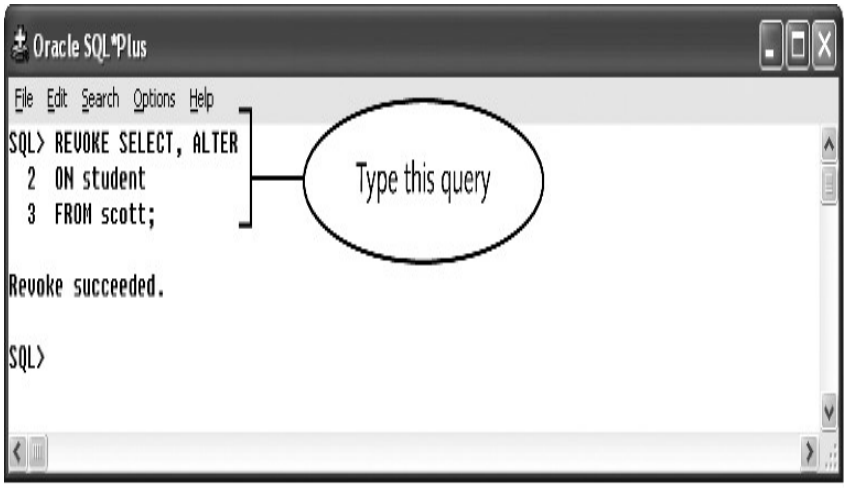


Figure 3-17 Revoking object privileges

Retrieving Data From a Single Database Table

- Syntax

```
SELECT columnname1, columnname2, …
FROM ownername.tablename
[WHERE search_condition];
```

- Retrieve all of columns
  - Use asterisk (*) as wildcard character in SELECT clause
  - `SELECT * from …`

Suppressing Duplicate Rows

- SQL DISTINCT qualifier
  - Examines query output before it appears on screen
  - Suppresses duplicate values
- Syntax
  - `SELECT DISTINCT columnname;`

Using Search Conditions in SELECT Queries

- Use search conditions to retrieve rows matching specific criteria
  - Exact search conditions
    - Use equality operator
  - Inexact search conditions
    - Use inequality operators
- Search for NULL or NOT NULL values
  - `WHERE columnname IS NULL`
  - `WHERE columnname IS NOT NULL`

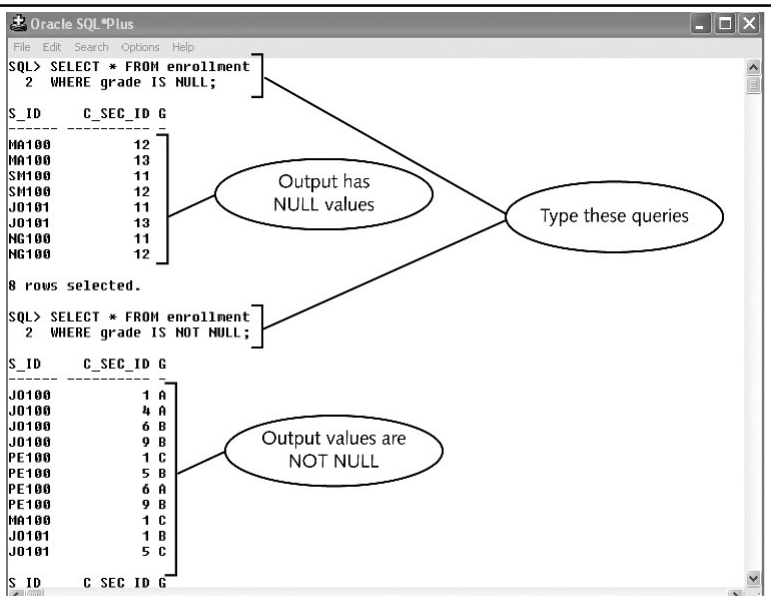


Figure 3-20 Queries using IS NULL and IS NOT NULL search conditions (partial output shown)

Using Search Conditions in SELECT Queries (continued)

- IN comparison operator
  - Match data values that are members of a set of search values
- LIKE operator
  - Use to match part of character string
  - Syntax
    - `WHERE columnname LIKE 'string'`
    - Character string should contain wildcard character %, or _, or both

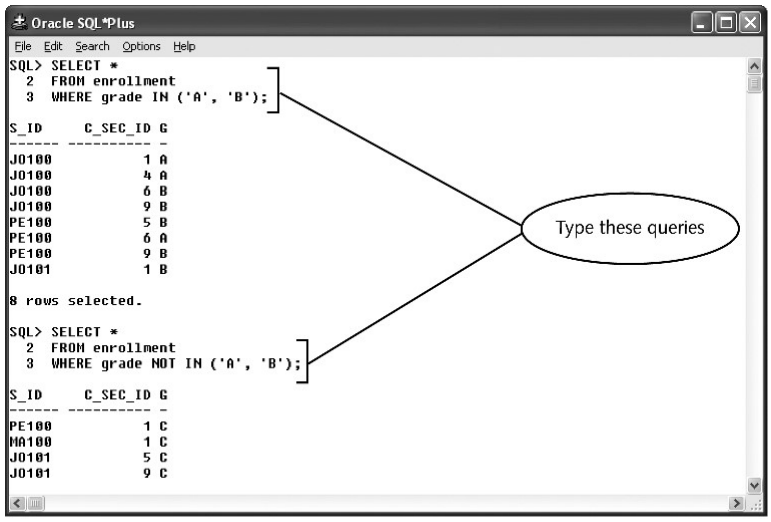


Figure 3-21 Using the IN and NOT IN comparison operators

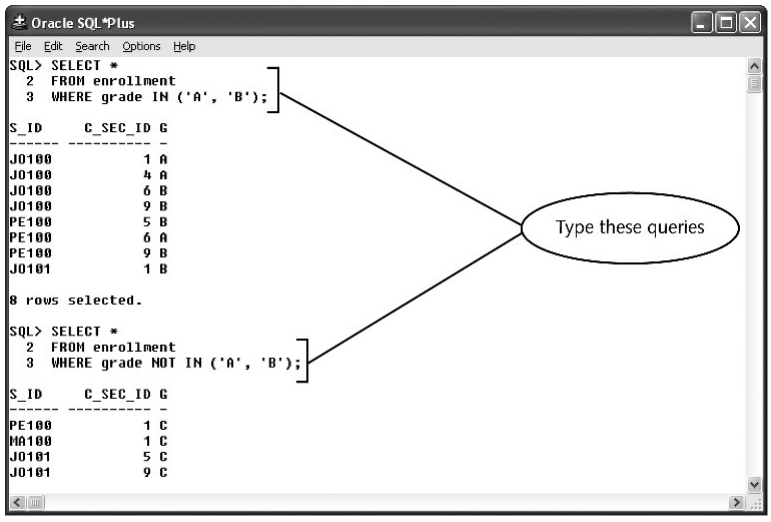


Figure 3-21 Using the IN and NOT IN comparison operators

Sorting Query Output

- ORDER BY clause
  - Sort query output
  - Syntax for select with ordered results

```
SELECT columnname1, columnname2, …
FROM ownername.tablename
WHERE search_condition
ORDER BY sort_key_column;
```

  - Sort can be ascending or descending
  - Can specify multiple sort keys

Performing Arithmetic Calculations

- Perform arithmetic calculations on columns that have data types
  - NUMBER
  - DATE
  - INTERVAL
- SYSDATE pseudocolumn
  - Retrieves current system date
- Use + and – to calculate differences between dates

Oracle 10*g* SQL Functions

- Built-in functions perform calculations and manipulate retrieved data values
- Called single-row functions
  - Return single result for each row of data retrieved
- To use:
  - List function name in SELECT clause followed by required parameter in parentheses

Oracle 10*g* SQL Group Functions

- Group function
  - Performs operation on group of queried rows
  - Returns single result such as column sum
- To use:
  - List function name followed by column name in parentheses

Using the COUNT Group Function

- COUNT group function
  - Returns integer representing number of rows that query returns
- COUNT(*) version
  - Calculates total number of rows in table that satisfy given search condition
  - Includes NULL values.
- The COUNT(*columnname*) version
  - Does not include NULL values

Using the GROUP BY Clause to Group Data

- GROUP BY clause
  - Group output by column with duplicate values
  - Apply group functions to grouped data
- Syntax
  - `GROUP BY group_columnname;`
  - Follows FROM clause
- All columns listed in SELECT clause must be included in GROUP BY clause

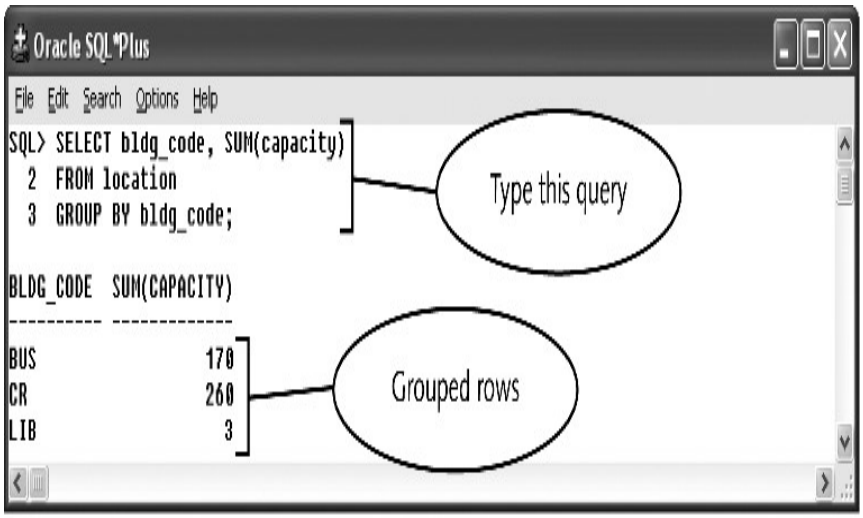


Figure 3-34 SQL query that uses the GROUP BY clause to group rows

Using the HAVING Clause to Filter Grouped Data

- HAVING clause
  - Place search condition on results of queries that display group function calculations
- Syntax
  - `HAVING group_function comparison_operator value`
- Example
  - `HAVING sum(capacity) >= 100`

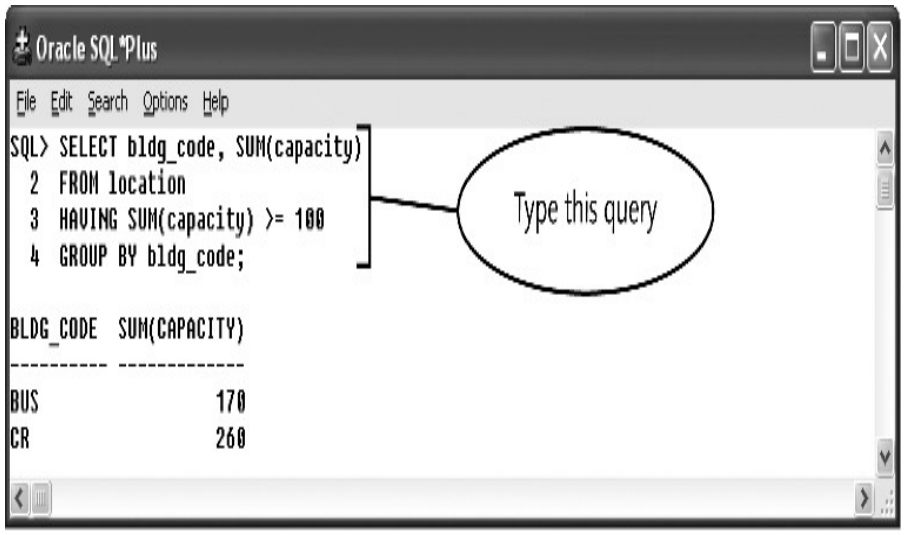


Figure 3-36 Using the HAVING clause with a group function

Joining Multiple Tables

- Join
  - Combine data from multiple database tables using foreign key references
- Syntax

```
SELECT column1, column2, …
FROM table1, table2
WHERE table1.joincolumn =
  table2.joincolumn
AND search_condition(s);
```

Joining Multiple Tables (continued)

- Must qualify column name in SELECT clause
  - Specify name of table that contains column followed by period then column name
- Join condition
  - Specifies table names to be joined and column names on which to join tables
- SQL supports multiple types of join queries

Inner Joins

- Simplest type of join
- VALUES in one table equal to values in other table
- Also called:
  - Equality join
  - Equijoin
  - Natural join
- Query design diagram

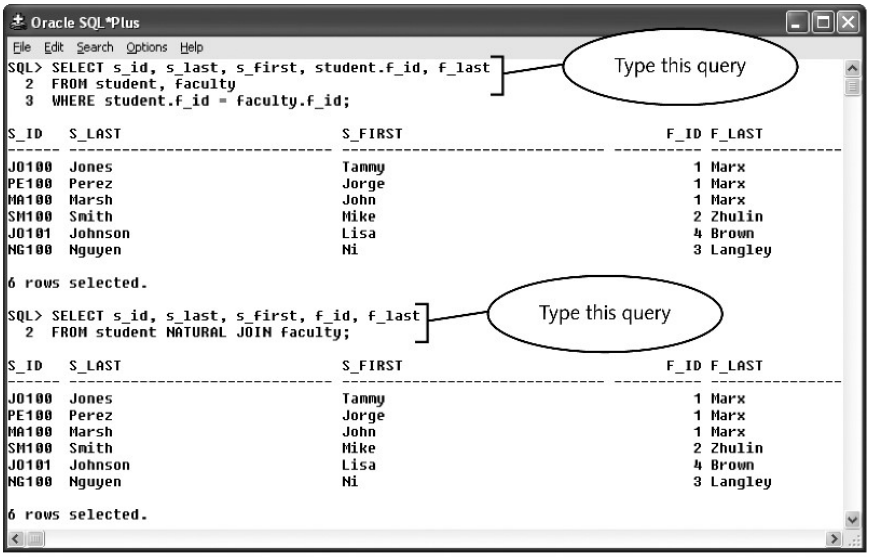


Figure 3-43 Queries joining two tables

Outer Joins

- Returns all rows from one table
  - Called inner table
- And matching rows from second table
  - Called outer table
- Syntax
  - *inner_table.join_column =*
    outer_table.join_column(+)

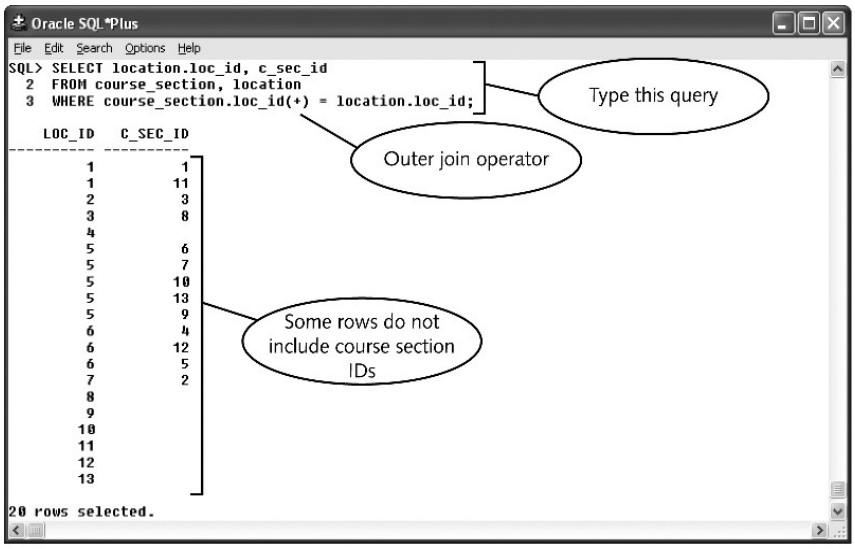


Figure 3-50 Query using an outer join

Self-joins

- Query that joins table to itself
- Must create table alias
  - Alternate name assigned to table in query's FROM clause
  - Syntax
    - FROM *table1 alias1*, …

FACULTY (Table name: FAC)

F_ID	F_LAST	F_FIRST	F_MI	LOC_ID	F_PHONE	F_RANK	F_SUPER
1	Marx	Teresa	J	9	4075921695	Associate	4
2	Zhulin	Mark	M	10	4073875682	Full	
3	Langley	Colin	A	12	4075928719	Assistant	4
4	Brown	Jonnel	D	11	4078101155	Full	
5	Sealy	James	L	13	4079817153	Associate	2

SUPERVISOR
(Table name: SUPER)

Figure 3-51 Tables referenced in self-join query

Guide to Oracle 10g 113

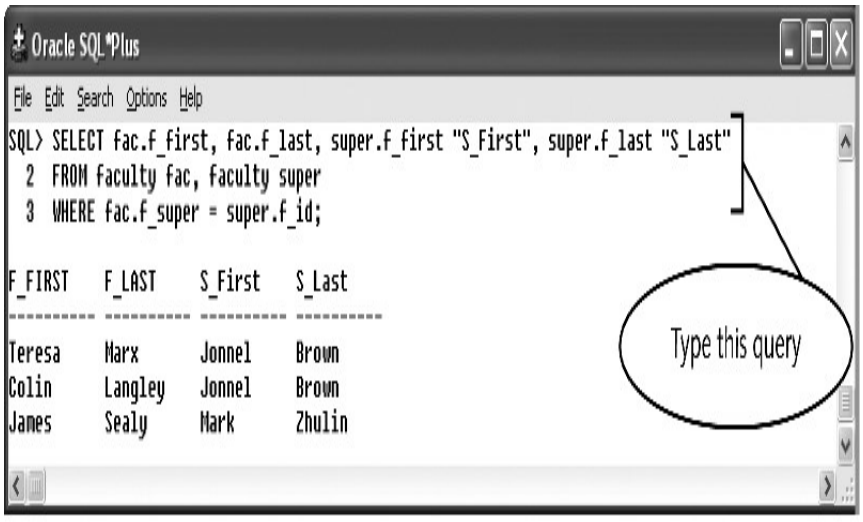


Figure 3-52 Self-join of the FACULTY table

Guide to Oracle 10g 114

Creating Nested Queries

- Nested query
  - Consists of main query and one or more subqueries
  - Main query
    - First query that appears in SELECT command
  - Subquery
    - Retrieves values that main query's search condition must match

Creating Nested Queries with Subqueries that Return a Single Value

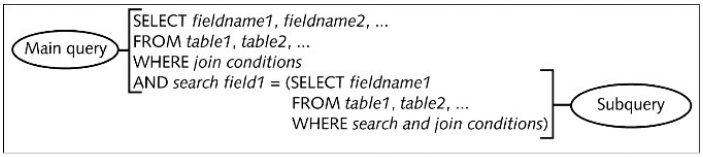


Figure 3-53 Syntax for nested query

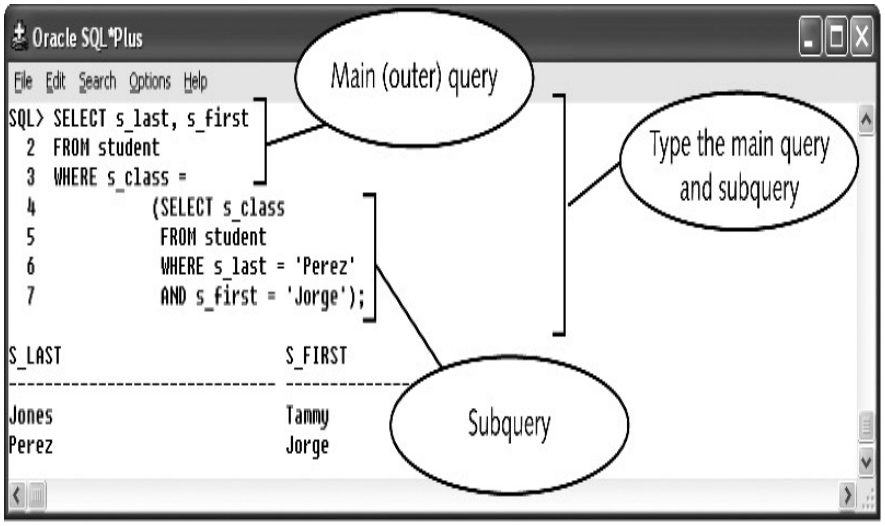


Figure 3-54 Creating a nested query that returns a single value

Using Multiple Subqueries Within a Nested Query

- Use AND and OR operators
  - To join search conditions associated with subqueries

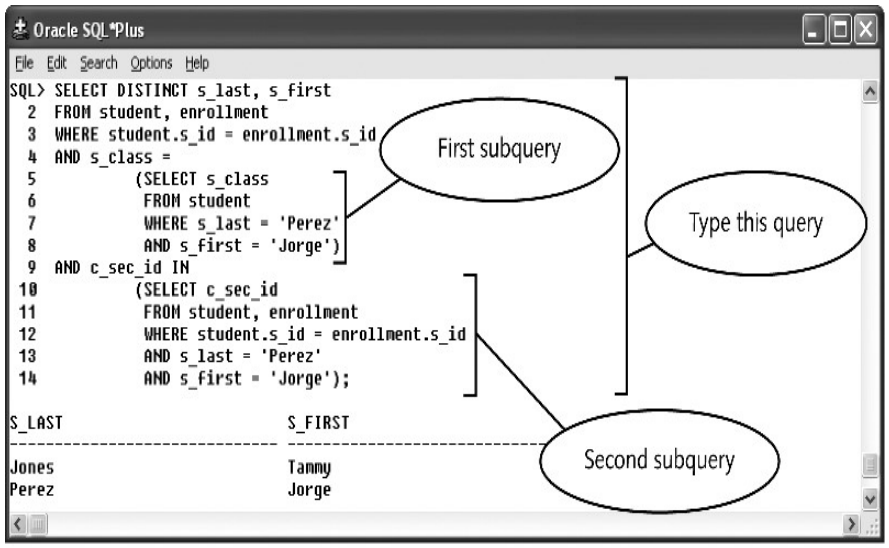


Figure 3-56 Nested query that has multiple subqueries

Creating Nested Subqueries

- Nested subquery
  - Subquery that contains second subquery that specifies its search expression

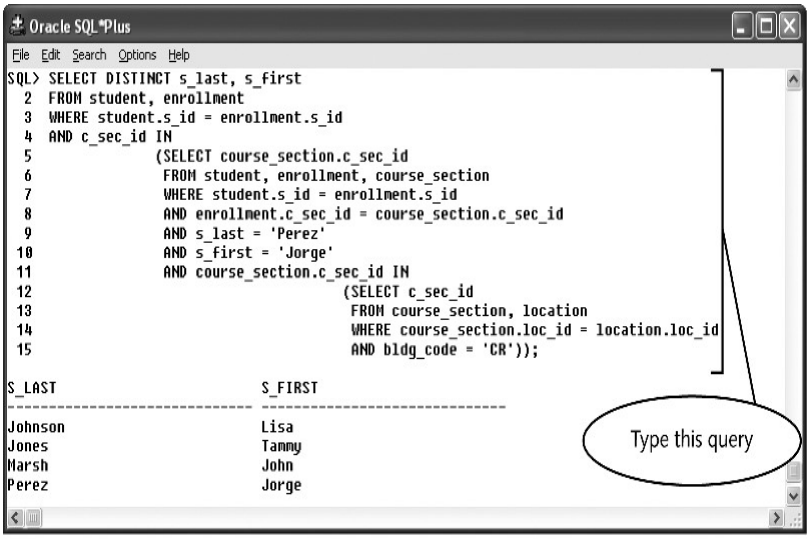


Figure 3-57 Creating a query with a nested subquery

CE11

UNION and UNION ALL

- UNION set operator
  - Joins output of two unrelated queries into single output result
  - Syntax
    - *query1* UNION *query2*;
- UNION ALL operator
  - Same as UNION but includes duplicate rows

Slide 122

CE11 There is no slide for A-head "Using Set Operators to Combine Query Results" (pg 175) - okay?
CE, 7/29/2005

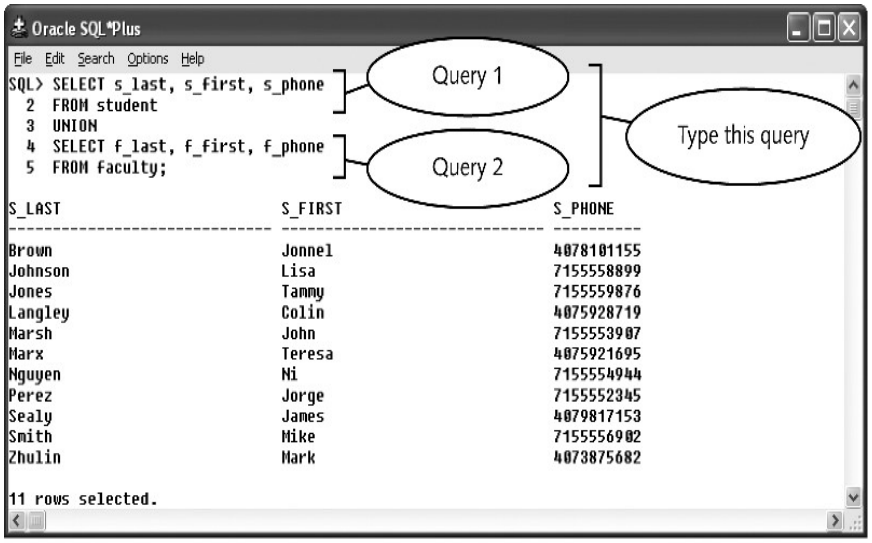


Figure 3-58 Query using the set UNION operator

INTERSECT

- Finds intersection in two queries
- Requires that both queries have same number of display columns in SELECT statement
- Automatically suppresses duplicate rows

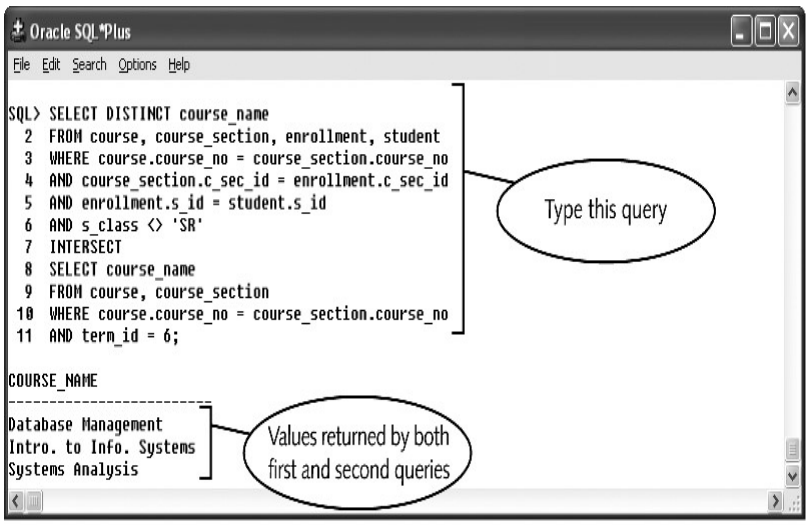


Figure 3-61 Query using the INTERSECT set operator

MINUS

- To find difference between two unrelated query result list

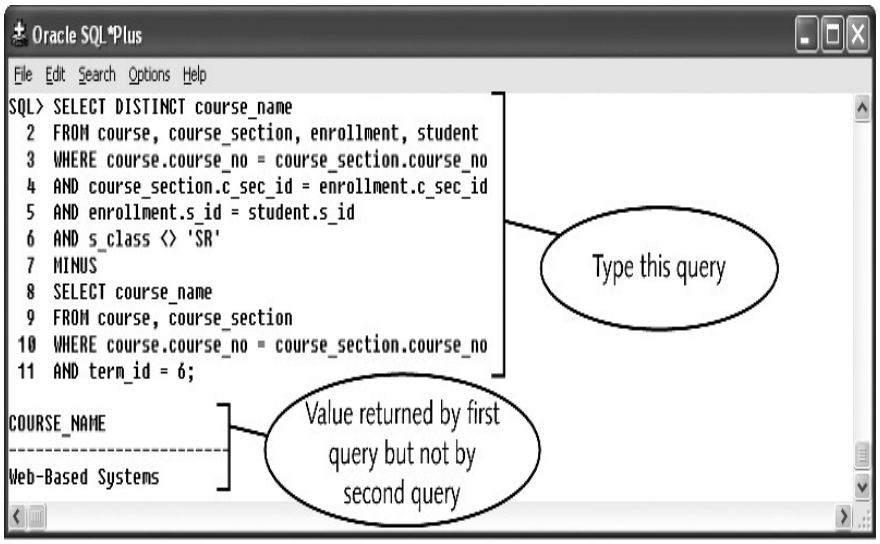


Figure 3-62 Query using the MINUS set operator

Creating Views

- Syntax

  CREATE VIEW *view_name*
  AS *source_query*;

  – Or

  CREATE OR REPLACE VIEW *view_name*
  AS *source_query*;

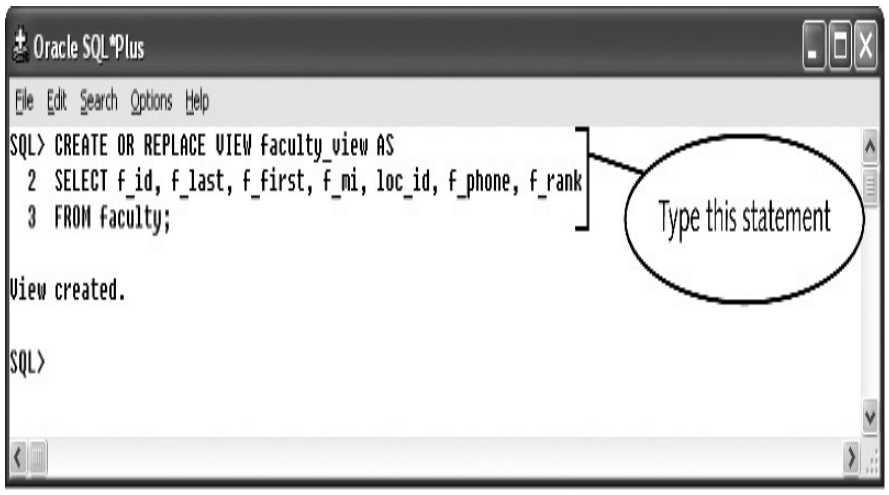


Figure 3-63 Creating an updateable view

Executing Action Queries Using Views

- Use view to execute action queries that:
  - Insert
  - Update
  - Delete data in underlying source tables
- Can also execute update action queries and delete action queries using view
  - Just as with database table

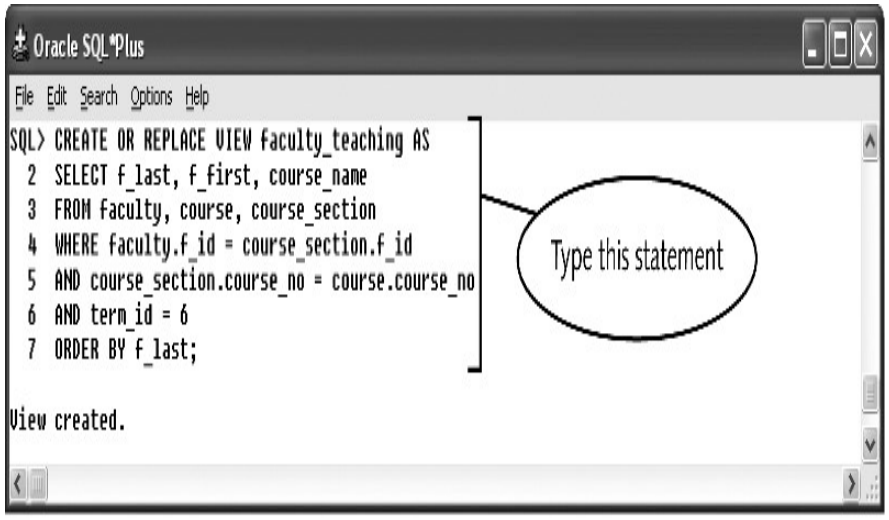


Figure 3-64 Creating a view that cannot be updated

Removing Views

- DROP VIEW command
  - Remove view from user schema
  - Syntax
    - `DROP VIEW view_name;`
    - **Example:**

DROP VIEW faculty_view;