Institute of Technology Tralee

Institiúid Teicneolaíochta Trá Lí

Database Concepts

Lab 05 – Inserting, Updating and Deleting Data

In this lab you will :

- Create an Oracle script to populate a database
- INSERT a row in a database table
- UPDATE a row(s) in a database table
- DELETE a row(s) from a database table

Before You Start:

During this lab, you will use the DVDSYS database which you created last week.
The tables in the database are empty, as we did not add any data to the database in last week's session.

INSERT a Record

Open the table ***Genres***.

Insert a row in the table ***Rates*** – use the following data:

- Rate_Code : 'NR'
- Description : 'New Release'
- Rate: 5.00

Since you are inserting a value for each column in the table, you must use the following syntax:

INSERT INTO <table name>
VALUES (val1, val2,…..);

Refresh the data view to see the inserted record.

Insert another row in the table ***Rates*** – use the following data:

- Rate_Code : 'OR'
- Description : 'Old Release'

Refresh the data view to see the inserted record.

Insert a row in the table ***Members*** – use the following data:

- Member_Id: 0001
- Surname: 'SMITH'
- Forename: 'JOHN'

Since you are NOT inserting a value for every column in the table, you must use the following syntax:

INSERT INTO <table name> (col1, col2,…)
VALUES (val1, val2,…..);

Insert a row in the table ***DVDs***– use the following data:

- DVD_Id: 1
- Title : 'SNOW WHITE'
- Rate_Code: 'XX'
- Genre_Code: 'TH'
- Age_Rating: '12'

What is the problem that arises?

Notice that the DBMS does not allow this row to be inserted since referential integrity would be violated.

Insert a row in the table ***DVDs***– use the following data:

- DVD_Id: 1
- Title : 'SNOW WHITE'
- Rate_Code: 'NR'
- Genre_Code: 'TH'
- Age_Rating: '12'

UPDATE a Row(s)

Remember the syntax for the SQL UPDATE statement:

UPDATE <table name>
SET col1 = val1 [,col2 = val2 (….)]
[WHERE <condition> = true];

The rental rate for the Old Release is now known to be €3.00. Update the record in the table rates for this Rate_Code.

UPDATE Rates
SET Rate = 3.00
WHERE Rate_Code = 'OR';

You now have ***Address*** details for Member_Id 1.
Update this row for Member_Id 1 – use the following data:

- Street: '12 MAIN STREET'
- Town: 'KILLARNEY'
- County: 'CO.KERRY'

UPDATE Members
SET Street = '12 MAIN STREET', Town = 'KILLARNEY', County = 'CO.KERRY'
WHERE Member_Id = 1;

DELETE a Row(s)

Remember the syntax for the SQL DELETE statement:

DELETE
FROM <table name>
[<condition> = true];

Delete the 'New Release' from the table ***Rates***.

```
DELETE
FROM  Rates
WHERE Rate_Code = 'NR';
```

What is the problem that arises?

Delete the row for DVD_Id 1 from the table DVDs.

```
DELETE
FROM  DVDs
WHERE DVD_Id = 1;
```

SQL COMMIT and ROLLBACK

A database transaction may require several DML SQL statements (SELECT/INSERT/UPDATE/DELETE) to be executed. A transaction can therefore consist of one or more SQL statements.

If all SQL statements in a transaction execute successfully, then the transaction is said to have completed successfully and any updates to the database are made permanent by executing an SQL COMMIT statement. Other transactions may **only then** see the updates.

If any SQL statements in a transaction fail, then any previous updates in the same transaction must be undone. To do this, a ROLLBACK statement must be executed.

Until an application/end user COMMITs a transaction:

- ***The application/end user*** can see any changes made during the transaction by querying the modified tables, but ***other users*** cannot see the changes. After you commit the transaction, the changes are visible to other users' statements that execute after the commit.

- You can roll back (undo) any changes made during the transaction with the ROLLBACK statement

NOTE:

- The Oracle Database issues an implicit COMMIT before and after any data definition language (DDL) statement.
- A ROLLBACK will only rollback to the last COMMIT point.
- INSERT, UPDATE and DELETE statements require an explicit COMMIT before other users can see the changes made by a transaction.

Exercise

Create an Oracle Script ***DVDData.sql*** which:

- Adds three rate categories to the table Rates (New Release, Old Release, Children)
- Adds five genres to the table Genres (Thriller, Romance, Comedy, Horror, Drama)
- Adds four age ratings to the table Age_Ratings (PG, U, 18, 12)
- Adds six dvds to the table DVDs (DVD_Id 1–6, data of your choosing)
- Adds five members to the table Members (Member_Id 1-5, data of your choosing)
- Adds ten rentals to the table Rentals (Rent_Id 1-3, data of your choosing)

Executing this script multiple times will result in primary key violations.
How might the script be modified to ensure primary key violations do not occur?