**DB Project 1 Steps**

1. Database Design:

- Review the COMPANY schema specified in Figures 3.7 and 4.1 of your textbook (6th edition).

- Modify the EMPLOYEE table to include the 'OverTimeCount' attribute as described.

2. Create Tables:

- Write SQL `CREATE TABLE` statements for EMPLOYEE, DEPARTMENT, PROJECT, WORKS\_ON, DEPT\_LOCATIONS, and DEPENDENT tables.

- Include primary keys, foreign keys, and other appropriate constraints.

- Save these SQL statements in a text file.

3. Execute SQL Statements:

- Open SQLPLUS.

- Log in to your Oracle or MySQL database.

- Execute the SQL statements from the text file using the `@` command to create the tables.

- Use the `SPOOL` command to capture the execution in a spool file.

4. Load Data:

- Write database programs in a language you are familiar with (e.g., Java with JDBC, PHP, Python) to load the provided records into each of the tables.

- Handle circular references by inserting records with NULL foreign keys and then updating them after the referenced records are inserted.

5. Execute Update Commands:

- Write SQL commands for the updates provided on the course website.

- Execute the updates one by one and observe which ones violate integrity constraints.

- Use the `SPOOL` command to capture the commands and their results in spool files.

- Document any integrity constraints violations.

6. Execute Retrieval Queries:

- Write SQL queries for the English queries listed on the course website.

- Execute each query and display its results.

- Capture the queries and results in spool files.

7. Insert Records Violating Integrity Constraints:

- Write three additional `INSERT` commands that intentionally violate different types of integrity constraints (e.g., foreign key, unique constraint).

- Capture the commands and any error messages in spool files.

- Document the nature of the constraint violations.

8. Delete Records Violating Referential Integrity:

- Write a `DELETE` command to remove a record that violates a referential integrity constraint.

- Capture the command in a spool file.

- Document why the record violated the constraint.

9. Insert Records Without Violating Constraints:

- Insert three new records that do not violate any integrity constraints.

- Capture the commands and successful execution in spool files.

10. Create Trigger:

- Write a SQL command to create a trigger that updates the `EMPLOYEE.OverTimeCount` each time an entry in the `WORKS\_ON` table records more than 40 hours.

- Capture the command in a spool file.

11. Documentation:

- Edit the spool files to add explanations for each step, especially for constraint violations.

12. Submission:

- Upload the spool files for SQLPLUS commands and query results to Canvas.

- Upload the source code for your data loading programs.

Remember to test each step carefully and document your work thoroughly to ensure a successful submission. If you encounter any issues or errors, be sure to troubleshoot and address them. Additionally, follow any specific instructions or guidelines provided by your course instructor or on the course website.

What are spool files?

Spool files are text files that capture the output of SQL commands and queries executed in command-line database tools like SQL\*Plus. They are used for logging and auditing purposes and can also be used to document your work or to save query results for further analysis or reporting. Spooling is a common feature in database tools to save the output to a file.

To create spool files and get the output from your SQL commands or queries, you can follow these steps:

1. In SQL\*Plus (Oracle) or MySQL Command-Line Client:

For SQL\*Plus (Oracle):

- Open SQL\*Plus.

- Start spooling by using the `SPOOL` command followed by the file path where you want to save the spool file. For example:

>SPOOL /path/to/spool\_file.txt

For MySQL Command-Line Client:

- Start spooling by using the `tee` command followed by the file path where you want to save the spool file. For example:

>tee /path/to/spool\_file.txt

2. Execute SQL Commands or Queries:

Now, execute your SQL commands or queries as usual in the database tool. All the output, including query results, error messages, and command execution messages, will be saved in the spool file.

3. Stop Spooling:

To stop spooling in SQL\*Plus, use the `SPOOL OFF` command:

>SPOOL OFF

To stop spooling in MySQL Command-Line Client, use the `notee` command:

>notee

4. Accessing Spool Files:

Once you've stopped spooling, you can access the spool file using a text editor, file explorer, or any other tool that allows you to read text files. You will find all the output and messages from your SQL session recorded in the spool file.

Spool files are useful for keeping records of your work, sharing query results, and troubleshooting issues. They are particularly handy when you need to document the steps you've taken in your database interactions or when you need to provide evidence of your work for auditing purposes.

**Create Table Commands:**

CREATE TABLE EMPLOYEE (

Fname VARCHAR(15),

Minit CHAR,

Lname VARCHAR(15),

Ssn CHAR(9) NOT NULL PRIMARY KEY,

Bdate DATE,

Address VARCHAR(30),

Sex CHAR,

Salary DECIMAL(10, 2),

Super\_ssn CHAR(9),

Dno INT,

OverTimeCount INT DEFAULT 0, -- New attribute for tracking overtime

FOREIGN KEY (Dno) REFERENCES DEPARTMENT(Dnumber),

FOREIGN KEY (Super\_ssn) REFERENCES EMPLOYEE(Ssn)

);

CREATE TABLE DEPARTMENT (

Dname VARCHAR(15),

Dnumber INT NOT NULL PRIMARY KEY,

Mgr\_ssn CHAR(9) NOT NULL UNIQUE,

Mgr\_start\_date DATE,

FOREIGN KEY (Mgr\_ssn) REFERENCES EMPLOYEE(Ssn)

);

CREATE TABLE DEPT\_LOCATIONS (

Dnumber INT NOT NULL,

Dlocation VARCHAR(15) NOT NULL,

PRIMARY KEY (Dnumber, Dlocation),

FOREIGN KEY (Dnumber) REFERENCES DEPARTMENT(Dnumber)

);

CREATE TABLE PROJECT (

Pname VARCHAR(15) NOT NULL,

Pnumber INT NOT NULL PRIMARY KEY,

Plocation VARCHAR(15),

Dnum INT NOT NULL,

FOREIGN KEY (Dnum) REFERENCES DEPARTMENT(Dnumber)

);

CREATE TABLE WORKS\_ON (

Essn CHAR(9),

Pno INT,

Hours DECIMAL(3, 1),

PRIMARY KEY (Essn, Pno),

FOREIGN KEY (Essn) REFERENCES EMPLOYEE(Ssn),

FOREIGN KEY (Pno) REFERENCES PROJECT(Pnumber)

);

CREATE TABLE DEPENDENT (

Essn CHAR(9),

Dependent\_name VARCHAR(15) NOT NULL,

Sex CHAR,

Bdate DATE,

Relationship VARCHAR(8),

PRIMARY KEY (Essn, Dependent\_name),

FOREIGN KEY (Essn) REFERENCES EMPLOYEE(Ssn)

);

**Trigger to update OverTimeCount:**

DELIMITER //

CREATE TRIGGER calculate\_overtime

AFTER INSERT ON WORKS\_ON

FOR EACH ROW

BEGIN

DECLARE total\_hours DECIMAL(10, 2);

DECLARE weekly\_threshold DECIMAL(10, 2) DEFAULT 40.0;

-- Calculate the total hours worked by the employee on all projects in the current week

SELECT SUM(Hours) INTO total\_hours

FROM WORKS\_ON

WHERE Essn = NEW.Essn;

-- Check if the total hours exceed the weekly threshold

IF total\_hours > weekly\_threshold THEN

-- Update the OverTimeCount for the employee

UPDATE EMPLOYEE

SET OverTimeCount = OverTimeCount + 1

WHERE Ssn = NEW.Essn;

END IF;

END;

//

DELIMITER ;

**Queries:**

#1 Enter a department name, and retrieve all the names and salaries of all employees who work in that department.

SELECT Fname, Minit, Lname, Salary

FROM EMPLOYEES AS E

WHERE E.Dept\_num IN

(SELECT Dept\_number

FROM DEPARTMENT

WHERE Name = 'HR');

#2.Enter an employee last name and first name and retrieve a list of projects names/hours per week that the employee works on.

SELECT DISTINCT P.Name, W.Hours

FROM PROJECT AS P, WORKS\_ON AS W

WHERE (P.project\_number, W.Hours) IN

(SELECT W.project\_number, W.Hours

FROM WORKS\_ON AS W

WHERE W.Ssn IN

(SELECT E.Ssn

FROM EMPLOYEES AS E

WHERE Fname = 'Roy' AND Lname = 'Lewallen'));

#3 Enter a department name and retrieve the total of all employee salaries who work in the department.

SELECT Name AS Department\_Name, SUM(Salary) AS Total\_salary

FROM EMPLOYEES, DEPARTMENT

WHERE (Dept\_num, Name) IN

(SELECT Dept\_number, Name

FROM DEPARTMENT

WHERE Name = 'Sales');

#4 For each department, retrieve the department name and the number (count) of employees who work in that department.

# Order the result by number of employees in descending order.

SELECT Name, COUNT(\*) AS 'Total Employees'

FROM DEPARTMENT

INNER JOIN EMPLOYEES

ON Dept\_number = Dept\_num

GROUP BY Dept\_number, Name

ORDER BY COUNT(\*) DESC;

#5 For each employee who is a supervisor, retrieve the employee first and last name and

# the number (count) of employees that are supervised. Order the result in descending order.

SELECT S.Fname, S.Lname, COUNT(\*) AS 'Employees Supervised'

FROM EMPLOYEES AS E, EMPLOYEES AS S

WHERE E.SuperSSN = S.Ssn

GROUP BY S.Ssn

ORDER BY COUNT(\*) DESC;