CSE 302
Database Management Systems Sessional

Lab - 4

Topics

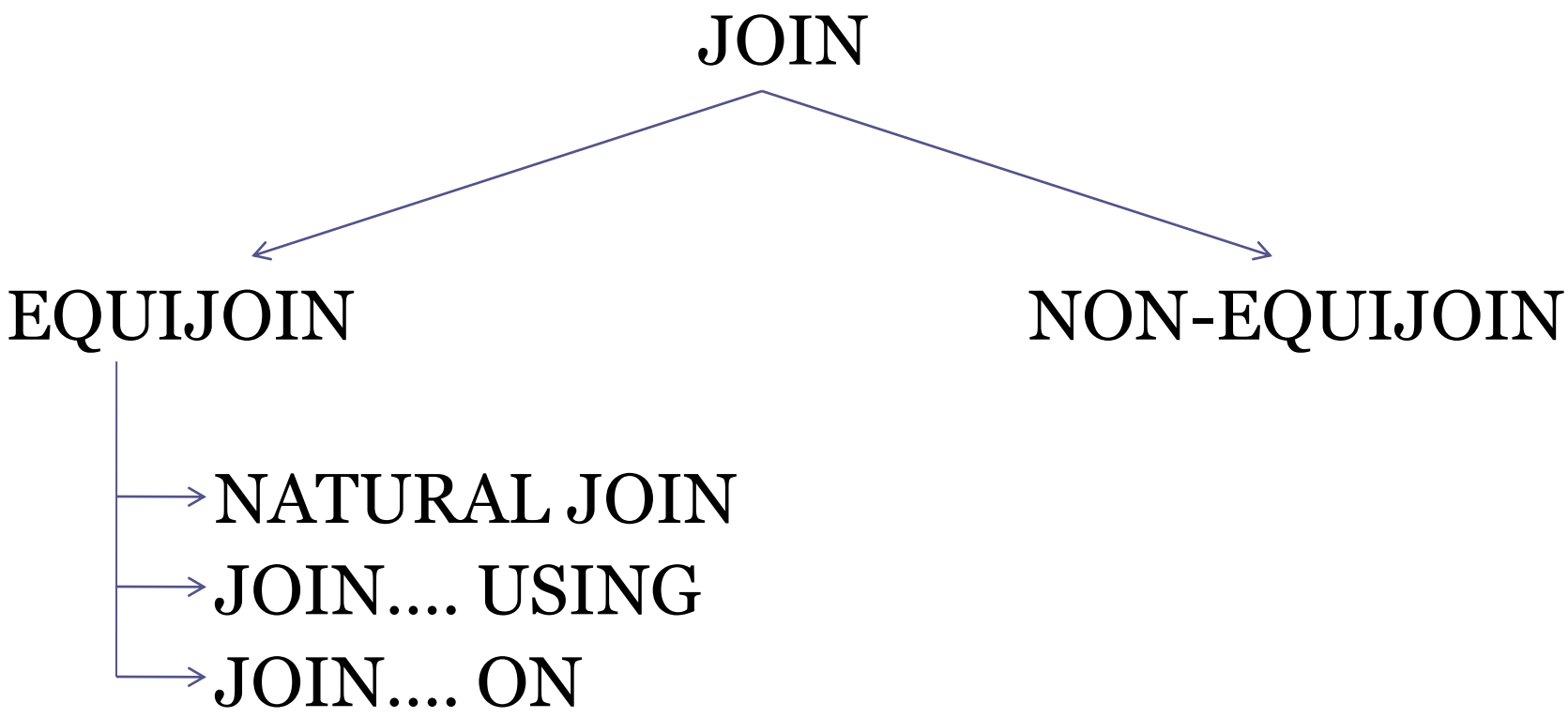
- JOIN
- GROUP FUNCTIONS

JOINS

Objective

- Displaying Data from Multiple Tables

Types of JOIN



Some Additional JOIN Methods

- Outer Joins
- Self Joins

EQUALITY JOINS

EQUALITY JOINS

- Two (or more) tables having equivalent data stored in a "**common column“.**

SELECT *Column Names*
FROM *Multiple Table Names*
WHERE *Condition / Access Path*

EQUALITY JOINS - Example

Customer (Cust_id, Cust_name, Cust_dob, Cust_street, Cust_city)
Account (Account_id, Balance, Type)
Depositor (Cust_id, Account_id)

Display a list showing the Customer Names and Account Numbers.

```
SELECT Cust_name, Account_id
FROM CUSTOMER, DEPOSITOR
WHERE CUSTOMER.Cust_id = DEPOSITOR.Cust_id;
```

EQUALITY JOINS - Example

Include the Cust_id field in the displayed rows.

ERROR!!!

```
SELECT Cust_name, Cust_id, Account_id
FROM CUSTOMER, DEPOSITOR
WHERE CUSTOMER.Cust_id = DEPOSITOR.Cust_id;
```

The Cust_id in the SELECT clause is ambiguous because it appears in both the Customer table and the Depositor table.

EQUALITY JOINS - Example

Include the Cust_id field in the displayed rows.

```
SELECT Cust_name, CUSTOMER .Cust_id, Account_id
FROM CUSTOMER, DEPOSITOR
WHERE CUSTOMER.Cust_id = DEPOSITOR.Cust_id;
```

EQUALITY JOINS - Example

Use Table Aliases.

```
SELECT CUST_NAME, C.CUST_ID, ACCOUNT_ID
FROM CUSTOMER C, DEPOSITOR D
WHERE C.CUST_ID = D.CUST_ID;
```

Practice Problem

- Display a list showing the customer's name and the customer's loan id for all of the records in the Customer table.

Additional Search Conditions in JOIN

- *Display a list showing the Customer Names and Account Numbers for the customers who live in Harrison city.*

```
SELECT CUST_NAME, ACCOUNT_ID
FROM CUSTOMER C, DEPOSITOR D
WHERE C.CUST_ID = D.CUST_ID AND
C.CUST_CITY='Harrison';
```

JOINING MORE THAN TWO TABLES

- *Display the Customer Name, Account ID and Balance.*

- You need to join the
  - CUSTOMER,
  - DEPOSITOR &
  - ACCOUNT.

JOINING MORE THAN TWO TABLES

- *Display the Customer Name, Account ID and Balance.*

```
SELECT C.CUST_NAME, D.CUST_ID, A.ACCOUNT_ID, BALANCE
FROM CUSTOMER C, DEPOSITOR D, ACCOUNT A
WHERE C.CUST_ID=D.CUST_ID AND
            D.ACCOUNT_ID=A. ACCOUNT_ID;
```

Practice Problem

- *Display a list showing the customer's name, the customer's loan id & the loan amount.*

NATURAL JOIN

- Automatically joins the two tables that have a commonly named and defined field.

- SELECT …
  FROM Table 1 NATURAL JOIN Table 2;

For more than 2 tables,

- SELECT ….
  FROM T1 NATURAL JOIN T2 NATURAL JOIN T3;

NATURAL JOIN

- *Display a list showing the Customer Names and Account Numbers.*

```
SELECT CUST_NAME, CUST_ID, ACCOUNT_ID
FROM CUSTOMER NATURAL JOIN DEPOSITOR;
```

- *Display the Customer Name, Account ID and Balance.*

```
SELECT CUST_NAME, CUST_ID, ACCOUNT_ID,BALANCE
FROM CUSTOMER NATURAL JOIN DEPOSITOR NATURAL
JOIN ACCOUNT;
```

JOIN….USING

- Joins based on a column that has the same name and definition in both tables can be created with the USING clause.

```
SELECT ….
FROM T1 JOIN T2
USING (Common Column Name);
```

- USING clause can only contain the name of the common column (enclosed in parentheses).

JOIN….USING

- *Display a list showing the Customer Names and Account Numbers.*

```
SELECT CUST_NAME, CUST_ID, ACCOUNT_ID
FROM CUSTOMER JOIN DEPOSITOR
USING(CUST_ID);
```

JOIN….ON

- When tables have related (common) columns with different names, ON clause is useful.

```
SELECT ….
FROM T1 A JOIN T2 B
ON A.C1=B.C2;
```

- Add the table alias before the column names

JOIN….ON

- *Display a list showing the Customer Names and Account Numbers.*

```
SELECT CUST_NAME, ACCOUNT_ID
FROM CUSTOMER C JOIN DEPOSITOR D
ON C.CUST_ID=D.CUST_ID;
```

JOIN….ON

- *Display a list showing the Customer Names and Account Numbers for the customers who live in Harrison city.*

```
SELECT CUST_NAME, C.CUST_ID, ACCOUNT_ID
FROM CUSTOMER C JOIN DEPOSITOR D
ON C.CUST_ID=D.CUST_ID
WHERE C.CUST_CITY= 'Harrison';
```

NON-EQUALITY JOINS

NON-EQUALITY JOINS

- With an equality join, the two tables must have exactly the same value in their common columns.
- But that's not always possible.

- A non-equality join is used when the related columns cannot be joined through the use of an equal sign.

Example

- Employee (Employee_id, Employee_name, Employee_dob, Employee_street, Employee_city, Employee_startdate, Salary, Manager_id);
- SALGRADE (GRADE,LOSAL,HISAL);

- The relationship between the EMPLOYEE and the SALGRADE table is a non-equijoin.

- No column in the EMPLOYEE table corresponds directly to a column in the SALGRADE TABLE.

Example

- *Display a list showing the Employee Names, their Salaries and Grades.*

```
SELECT E.EMPLOYEE_NAME, E.SALARY,S.GRADE
FROM EMPLOYEE E, SALGRADE S
WHERE E.SALARY BETWEEN S.LOSAL AND S.HISAL;
```

Practice Problem

- *Display a list showing the Employee Names, their Salaries and Grades. (Use JOIN.... ON)*

Some Additional JOINS

OUTER JOINS

- The Outer Join operator is a plus sign enclosed in parenthesis(+) and it is placed on the side of the join that is deficient in information.
- This operator has the effect of creating one or more null rows, to which one or more rows from the non deficient table can be joined.
- The outer join can appear on only one side of the expression-the side that has information missing.
- A condition involving an outer join can’t use the IN operator or be linked to another condition by the OR operator.

OUTER JOINS

- *Display a list showing the Customer Names and Account Numbers. Show all of the customers, regardless of which customers have ACCOUNT.*

```
SELECT CUST_NAME, C.CUST_ID, ACCOUNT_ID
FROM CUSTOMER C, DEPOSITOR D
WHERE C.CUST_ID= D.CUST_ID(+)
ORDER BY C.CUST_ID;
```

SELF-JOINS

- Data in a table references other data in the same table.
- Sometimes you need to join a table to itself.

SELF-JOINS

- *Find the name of each employee's manager.*

```
SELECT  WORKER.EMPLOYEE_NAME ||
        ' WORKS FOR ' || MANAGER.EMPLOYEE_NAME
        "WORKER AND MANAGER"
FROM EMPLOYEE WORKER, EMPLOYEE MANAGER
WHERE WORKER.Manager_id=MANAGER.EMPLOYEE_ID;
```

Practice Problems for JOINs

- **Find all the customers who have an account as well as a loan .**
- **Find all the customers who have an account but not a loan.**
- **Display the Employee name and Employee Id along with their manager's name and manager ID.**
- **Display the list of Customer name, Customer DOB and Account Type.**

GROUP FUNCTIONS

Group Function

- Also known as "Multiple-Row Functions".
- They operates on set of rows to give one result per group.
- These set may be the whole table or the table split into groups.
- These are similar to the "aggregate functions" or "Group By" functions in Access

Group Functions

- SUM
- AVG
- COUNT
- MIN
- MAX

Group Functions

- **GROUP BY clause**
  - To identify groups of records to be processed
- **ORDER BY clause**
  - To sort the records
- **HAVING clause**
  - To restrict the groups displayed

```
SELECT * | column1, column2, ...
FROM tableName
WHERE Condition
GROUP BY column1, column2, ...
HAVING group condition
```

Group Functions

SUM function

- Calculates the total amount in a numeric field for a group of records.
  - SUM(n) - where n is a numeric column
  - SUM(ALL n) - the same as above
  - SUM(DISTINCT n) - returns only the unique numeric values

SUM function

- *Display total salary of all employees.*

```
SELECT SUM(Salary) "Total Salary"
FROM Employee;
```

- *Display total salary of the employees of e_city_001.*

```
SELECT SUM(Salary) "Total Salary of E_CITY_001"
FROM Employee
WHERE Employee_city='e_city_001';
```

AVG function

- AVG(column containing numeric data)
- AVG(DISTINCT [column containing numeric data])
  - DISTINCT keyword returns only unique values

AVG function

- *Display average salary of all employees.*

```
SELECT AVG(Salary) " Average Salary"
FROM Employee;
```

- *Display average salary of the employees of e_city_001.*

```
SELECT AVG (Salary) " Average Salary of E_CITY_001"
FROM Employee
WHERE Employee_city='e_city_001';
```

MAX and MIN function

- Returns the largest and smallest values in a specified column.

- MAX(ALL c) or MIN(ALL c)
  - where c is any numeric, character, or date field

- MAX(c) or MIN(c)
  - the same result as above

- MAX(DISTINCT c) or MIN(DISTINCT c)
  - returns the highest or lowest distinct value

MAX and MIN function

- *Display the maximum salary of employees.*

```
SELECT MAX(Salary) "Highest Salary"
FROM Employee;
```

- *Display the minimum DOB of employees.*

```
SELECT MIN(EMPLOYEE_DOB)
FROM EMPLOYEE;
```

COUNT function

- Counts the records that have non-NULL values
- Counts the total records meeting a specific condition

COUNT function

- *Display the count of cities.*

```
SELECT COUNT(EMPLOYEE_CITY)
FROM EMPLOYEE;
```

  - This counts all categories (including duplicates)

- *Display the count of unique cities only.*

```
SELECT COUNT(DISTINCT  EMPLOYEE_CITY)
FROM EMPLOYEE;
```

  - This counts unique (or distinct) categories

Group functions and NULL values

- All Group functions except COUNT(*) ignore null values in the column.

COUNT Function – NULL Values

- Including the NULL values
  - COUNT(*) counts all the records, even NULLS
  - Whenever NULL values may affect the COUNT the function, use an * as the argument, rather than a column name.

```
SELECT COUNT(*)
FROM EMPLOYEE;
```

GROUP BY Clause

GROUP BY Clause

- Divides the table of information into smaller groups.

```
SELECT .....
FROM .....
GROUP BY column1, column2,... ;
```

GROUP BY Clause

- *Divide the Employee table into groups by City. Then calculate the average salary for each group.*

```
SELECT Employee_city, Avg(Salary)
FROM Employee
GROUP BY Employee_city;
```

- The query execution goes like this:
  - The records in the Employee table are grouped by City
  - The average Salary for each City is calculated.

GROUP BY Clause

- *Display the Sum of All Loan of the Same City.*

```
SELECT Cust_city, SUM(Balance), Type
FROM Customer NATURAL JOIN Depositor
                    NATURAL JOIN Account
GROUP BY Cust_city, Type;
```

- The GROUP BY first groups the results by cust_city
- Then groups the Account TYPE within each customer City group.
- Then the SUM function calculates the Balance total.

ORDER BY Clause

ORDER BY Clause

- *Divide the Employee table into groups by City. Then calculate the average salary for each group and order the result by average salary.*

```
SELECT Employee_city, Avg(Salary)
FROM Employee
GROUP BY Employee_city
ORDER BY Avg(Salary);
```

- *Order by Descending order-*

```
SELECT Employee_city, Avg(Salary)
FROM Employee
GROUP BY Employee_city
ORDER BY Avg(Salary) DESC;
```

HAVING Clause

HAVING Clause

- To further restrict groups returned by a query (Specifies which groups will be returned)
- Use a HAVING clause instead of a WHERE clause when group functions are involved.

HAVING(condition)

HAVING Clause

- *Display the cust_city, total balance and account type of customers with balance>1000.*

```
SELECT Cust_city, SUM(Balance), Type
FROM Customer NATURAL JOIN Depositor
                    NATURAL JOIN Account
GROUP BY Cust_city, Type
HAVING SUM(Balance)>1000;
```

WHERE and HAVING

- Both can be used in the same query.

```
SELECT Cust_city, SUM(Balance), Type
FROM Customer NATURAL JOIN Depositor NATURAL JOIN
                                                Account
WHERE Cust_dob > '01-JAN-80'
GROUP BY Cust_city, Type
HAVING SUM(Balance)>1000;
```

```
SELECT Employee_city, Avg(Salary)
FROM Employee
WHERE Employee_startdate>'01-JAN-80'
GROUP BY Employee_city
HAVING AVG(Salary)>1000;
```

Nesting Group Functions

- Group Functions can be nested to a **depth of two.**

```
SELECT Max(Avg(Salary))
FROM Employee
GROUP BY Employee_city;
```

Some general rules

- For using a mixture of individual items(Employee_city) and group functions (AVG) in the same SELECT statement, you must include a GROUP BY Clause that specifies the individual items.

- You can’t use WHERE Clause to restrict groups.

- You have to use the HAVING Clause to restrict groups.

Practice Problems for Functions

- **Write a query to display the number of customer with the same city.**

- **Display the Manager Number and the Salary of the lowest paid employee for that manager.**

- **Display the Manager Number and the difference between the highest and the lowest Salary of the employee for that manager.**

- **Display the minimum, maximum , sum and average salary for each group of employee having the same city.**

THANK YOU