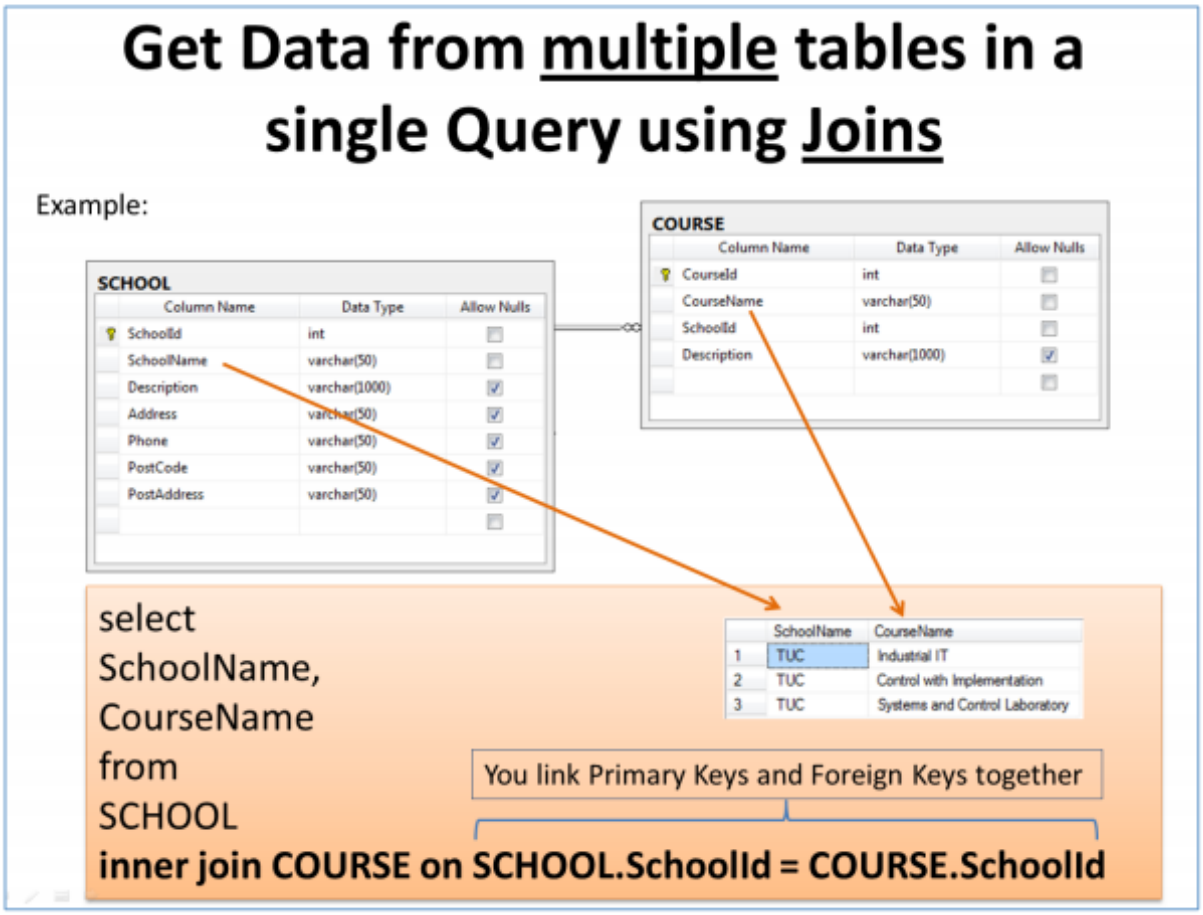
Database Lab

CSE 3104

Lab-05

5 Joins

SQL joins are used to query data from two or more tables, based on a relationship between certain columns in these tables.



We want to get the following information using a query:

SchoolName	ClassName
…	…
…	…

In order to get information from more than one table we need to use the JOIN. The JOIN is used to join the primary key in one table with the foreign key in another table.

```
select
SCHOOL.SchoolName,
CLASS.ClassName
from
SCHOOL
INNER JOIN CLASS ON SCHOOL.SchoolId = CLASS.SchoolId
```

5.1 Different SQL Joins

Before we continue with examples, we will list the types of JOIN you can use, and the differences between them.

- JOIN: Return rows when there is at least one match in both tables
- LEFT JOIN: Return all rows from the left table, even if there are no matches in the right table
- RIGHT JOIN: Return all rows from the right table, even if there are no matches in the left table
- FULL JOIN: Return rows when there is a match in one of the tables

5.2 Join Operations

Consider the CUSTOMER and ORDERS table

Now, let us join these two tables in our SELECT statement as follows:

```
SELECT CUSTOMER.CustomerId , Name, Age, Amount
FROM CUSTOMER, ORDERS
WHERE CUSTOMER.CustomerId = ORDERS.CustomerId
```

Here, it is noticeable that the join is performed in the WHERE clause. Several operators can be used to join tables, such as =, <, >, <>, <=, >=,! =, BETWEEN, LIKE, and NOT; they can all be used to join tables. However, the most common operator is the equal symbol.

5.2.1 Inner Join

The most frequently used and important of the joins is the INNER JOIN. They are also referred to as an EQUIJOIN.

The INNER JOIN creates a new result table by combining column values of two tables (table1 and table2) based upon the join-predicate. The query compares each row of table1 with each row of table2 to find all pairs of rows which satisfy the join-predicate. When the join-predicate is satisfied, column values for each matched pair of rows of A and B are combined into a result row.

The basic syntax of INNER JOIN is as follows:

```
SELECT table1.column1, table2.column2...
FROM table1
INNER JOIN table2
ON table1.common_filed = table2.common_field
```

```
SELECT CUSTOMER.CustomerId , Name, Age,
Amount,Date
FROM CUSTOMER
INNER JOIN ORDERS
ON CUSTOMER.CustomerId = ORDERS.CustomerId
```

5.2.2 Left Join

The SQL LEFT JOIN returns all rows from the left table, even if there are no matches in the right table. This means that if the ON clause matches 0 (zero) records in right table, the join will still return a row in the result, but with NULL in each column from right table.

This means that a left join returns all the values from the left table, plus matched values from the right table or NULL in case of no matching join predicate.

The basic syntax of LEFT JOIN is as follows:

```
SELECT table1.column1, table2.column2...
FROM table1
LEFT JOIN table2
ON table1.common_filed = table2.common_field
```

```
SELECT CUSTOMER.CustomerId , Name, Age,
Amount,Date
FROM CUSTOMER
LEFT JOIN ORDERS
ON CUSTOMER.CustomerId = ORDERS.CustomerId
```

5.2.3 Right Join

The SQL RIGHT JOIN returns all rows from the right table, even if there are no matches in the left table. This means that if the ON clause matches 0 (zero) records in left table, the join will still return a row in the result, but with NULL in each column from left table.

This means that a right join returns all the values from the right table, plus matched values from the left table or NULL in case of no matching join predicate.

The basic syntax of RIGHT JOIN is as follows:

```
SELECT table1.column1, table2.column2...
FROM table1
RIGHT JOIN table2
ON table1.common_filed = table2.common_field
```

```
SELECT CUSTOMER.CustomerId , Name, Age,
Amount,Date
FROM CUSTOMER
RIGHT JOIN ORDERS
ON CUSTOMER.CustomerId = ORDERS.CustomerId
```

5.2.3 Full Join

The SQL FULL JOIN combines the results of both left and right outer joins.

The joined table will contain all records from both tables, and fill in NULLs for missing matches on either side.

```
SELECT table1.column1, table2.column2...
FROM table1
FULL JOIN table2
ON table1.common_filed = table2.common_field
```

6 SQL Sub Queries

A Subquery or Inner query or Nested query is a query within another SQL query and embedded within the WHERE clause.

A subquery is used to return data that will be used in the main query as a condition to further restrict the data to be retrieved.

Subqueries can be used with the SELECT, INSERT, UPDATE, and DELETE statements along with the operators like =, <, >, >=, <=, IN, BETWEEN etc.

There are a few rules that subqueries must follow:

→ Subqueries must be enclosed within parentheses.

→ A subquery can have only one column in the SELECT clause, unless multiple columns are in the main query for the subquery to compare its selected columns.

→An ORDER BY cannot be used in a subquery, although the main query can use an ORDER BY. The GROUP BY can be used to perform the same function as the ORDER BY in a subquery.

→Subqueries that return more than one row can only be used with multiple value operators, such as the IN operator.

→A subquery cannot be immediately enclosed in a set function.

→The BETWEEN operator cannot be used with a subquery; however, the BETWEEN operator can be used within the subquery.

Subqueries with the SELECT Statement:

At first try with the following:

```
SELECT *
FROM CUSTOMER
WHERE CustomerId = (SELECT CustomerId
FROM CUSTOMER
WHERE Salary > 4500)
```

This will show an error message like:

```
Msg 512, Level 16, State 1, Line 1
Subquery returned more than 1 value. This is not permitted when the subquery
follows =, !=, <, <= , >, >= or when the subquery is used as an expression.
```

Correct form is:

```
SELECT *
FROM CUSTOMER
WHERE CustomerId IN (SELECT CustomerId
FROM CUSTOMER
WHERE Salary> 4500)
```

Note: Sub queries also can be performed on multiple tables.

Subqueries with the INSERT Statement:

Subqueries also can be used with INSERT statements. The INSERT statement uses the data returned from the subquery to insert into another table. The selected data in the subquery can be modified with any of the character, date or number functions.

Consider a table CUSTOMER_BKP with similar structure as CUSTOMER table. Now to copy complete CUSTOMER table into CUSTOMERS_BKP, following is the query:

```
INSERT INTO CUSTOMER_BKP
SELECT Name,Age,Address,Salary FROM CUSTOMER
WHERE CustomerId IN (SELECT CustomerId
FROM CUSTOMER)
```

Subqueries with the UPDATE Statement:

The subquery can be used in conjunction with the UPDATE statement. Either single or multiple columns in a table can be updated when using a subquery with the UPDATE statement.

Following example updates SALARY by 0.25 times in CUSTOMER table for all the customers whose AGE is greater than or equal to 27:

```
UPDATE CUSTOMER
SET SALARY = SALARY * 0.25
WHERE AGE IN (SELECT AGE FROM CUSTOMER_BKP
WHERE AGE >= 27 )
```

Subqueries with the DELETE Statement:

The subquery can be used in conjunction with the DELETE statement like with any other statements mentioned above.

```
UPDATE CUSTOMER
SET SALARY = SALARY * 0.25
WHERE AGE IN (SELECT AGE FROM CUSTOMER_BKP
WHERE AGE >= 27 )
```

Practice Session:

1. **List Product ID , Product Name , Product Type Name of all products .**
2. **Show all product information along with Product Type Name that has a price greater than $29.95(use alias)**
3. **List all the last name of the customers, Order IDs and Product Name in customer name order.(all of the products a given customer has ordered)**
4. **Show those orders which have product ID = 3.**