Session 4 – Manual

ERDs and Table Joins

Objective:

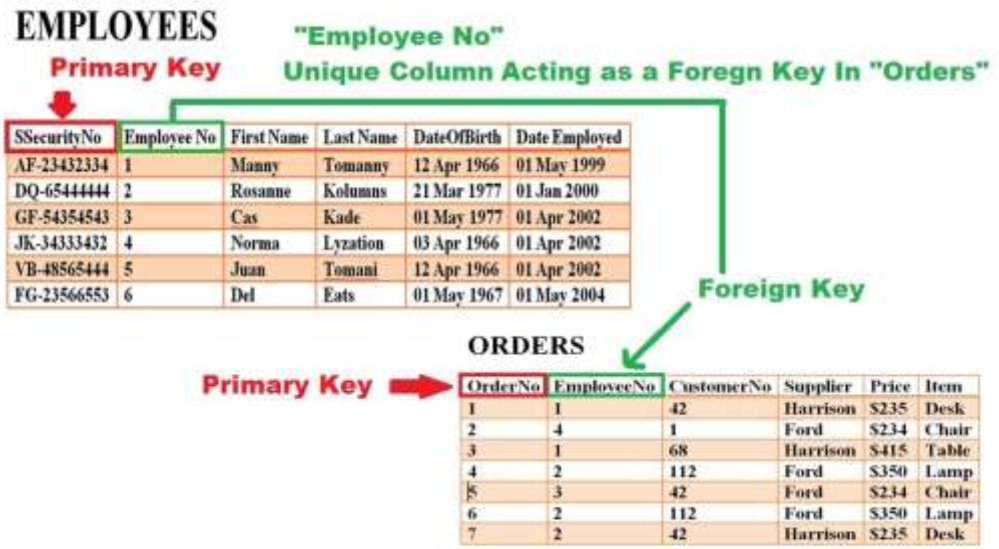
Session: Table Joins

- Comprehend the significance of primary and foreign keys in establishing relationships between tables.
- Identify and understand different types of relationships between tables (one-to-one, one-to-many, many-to-many).
- Learn how to read and interpret Entity-Relationship Diagrams (ERDs).
- Gain proficiency in using various types of joins (INNER, LEFT, RIGHT, FULL, CROSS, SELF) to combine data from multiple tables.

Understanding Relationships Between Tables

Key Points:

- **Primary Key**: A unique identifier for each record in a table.
- **Foreign Key**: A field in one table that uniquely identifies a row of another table.



EMPLOYEES

SSecurityNo	Employee No	First Name	Last Name	DateOfBirth	Date Employed
AF-23432334	1	Manny	Tomanny	12 Apr 1966	01 May 1999
DQ-65444444	2	Rosanne	Kolumns	21 Mar 1977	01 Jan 2000
GF-54354543	3	Cas	Kade	01 May 1977	01 Apr 2002
JK-34333432	4	Norma	Lyzation	03 Apr 1966	01 Apr 2002
VB-48565444	5	Juan	Tomani	12 Apr 1966	01 Apr 2002
FG-23566553	6	Del	Eats	01 May 1967	01 May 2004

ORDERS

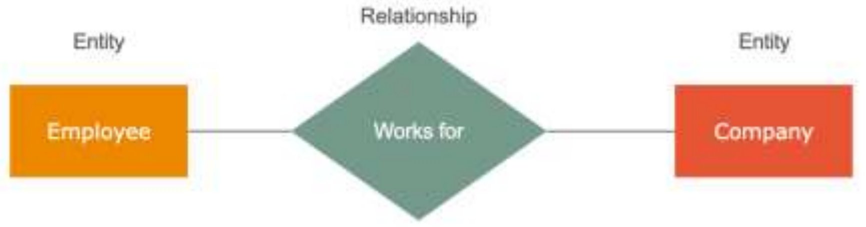
OrderNo	EmployeeNo	CustomerNo	Supplier	Price	Item
1	1	42	Harrison	$235	Desk
2	4	1	Ford	$234	Chair
3	1	68	Harrison	$415	Table
4	2	112	Ford	$350	Lamp
5	3	42	Ford	$234	Chair
6	2	112	Ford	$350	Lamp
7	2	42	Harrison	$235	Desk

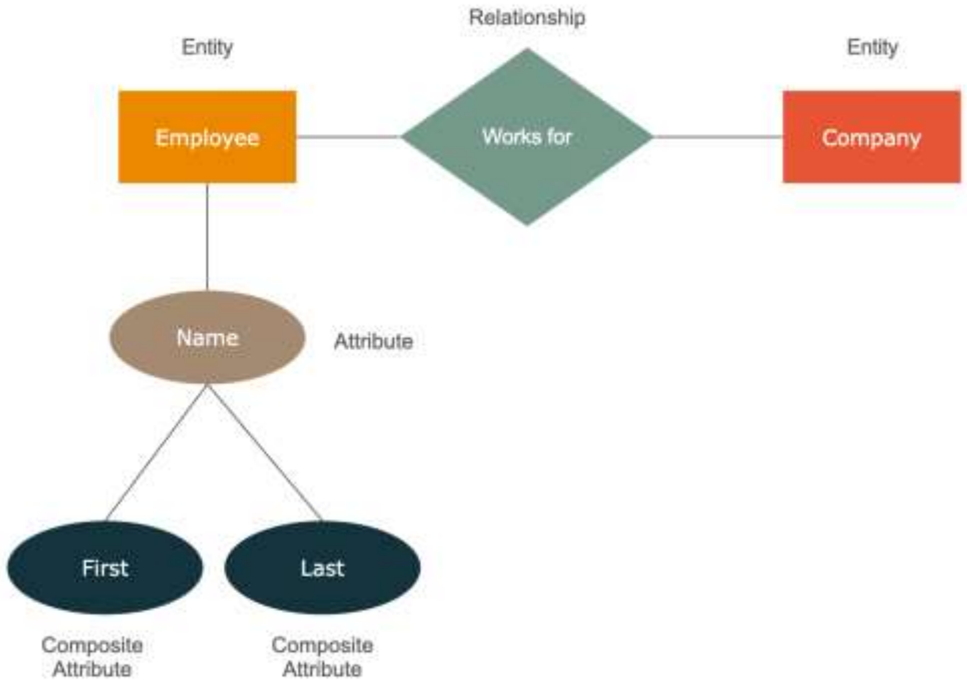
Types of Relationships:

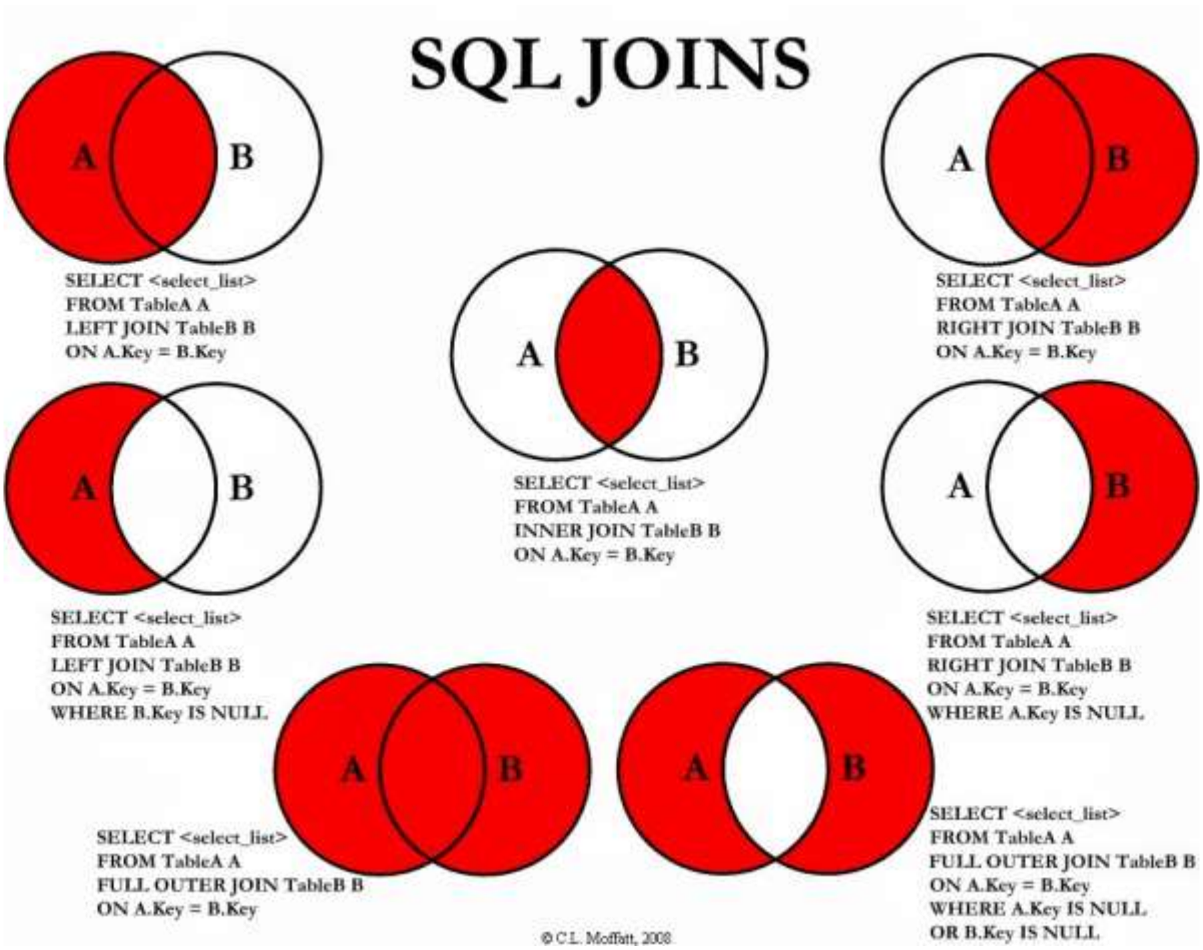
1. **One-to-One**: Each row in Table A is linked to one and only one row in Table B.
2. **One-to-Many**: Each row in Table A is linked to one or more rows in Table B.
3. **Many-to-Many**: Rows in Table A are linked to multiple rows in Table B and vice versa, typically implemented using a junction table.

Entity-Relationship Diagrams (ERDs):

- **Purpose**: To visually represent the relationships between tables.
- **Components**:
  - Entities (tables)
  - Attributes (columns)
  - Relationships (lines connecting entities)







```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```

```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
```

```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```

```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
```

```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
```

```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
```

© C.L. Moffatt, 2008

Using Joins to Combine Data from Multiple Tables

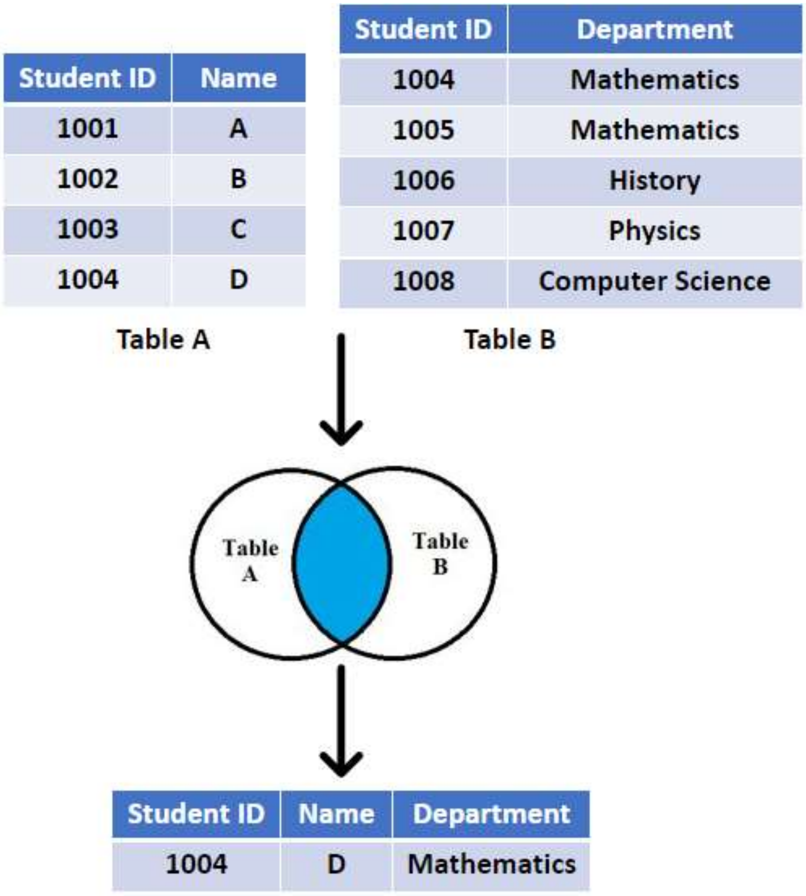
Key Points:

- **Purpose**: To retrieve related data from multiple tables.

Types of Joins:

1. **INNER JOIN**: Selects records with matching values in both tables.

```
SELECT
  a.column_name, b.column_name
FROM
  table1 a
INNER JOIN
  table2 b ON a.common_field = b.common_field;
```



Student ID	Name
1001	A
1002	B
1003	C
1004	D

Table A

Student ID	Department
1004	Mathematics
1005	Mathematics
1006	History
1007	Physics
1008	Computer Science

Table B

Student ID	Name	Department
1004	D	Mathematics

2. **LEFT JOIN (LEFT OUTER JOIN)**: Selects all records from the left table and matched records from the right table.

```
SELECT
  a.column_name, b.column_name
FROM
  table1 a
LEFT JOIN
  table2 b ON a.common_field = b.common_field;
```

3. **RIGHT JOIN (RIGHT OUTER JOIN)**: Selects all records from the right table and matched records from the left table.

```
SELECT
  a.column_name, b.column_name
FROM
  table1 a
RIGHT JOIN
  table2 b ON a.common_field = b.common_field;
```

4. **FULL JOIN (FULL OUTER JOIN)**: Selects all records when there is a match in either left or right table.

```
SELECT
  a.column_name, b.column_name
FROM
  table1 a
FULL JOIN
  table2 b ON a.common_field = b.common_field;
```

5. **Cross Join**: Combines all rows of Table A with all rows of Table B.

```
SELECT
  a.column_name, b.column_name
FROM
  table1 a
CROSS JOIN
  table2 b;
```

6. **Self Join**: Joins a table with itself.

```
SELECT
  a.column_name, b.column_name
FROM
  table1 a, table1 b
WHERE
  a.common_field = b.common_field;
```

Happy querying!