

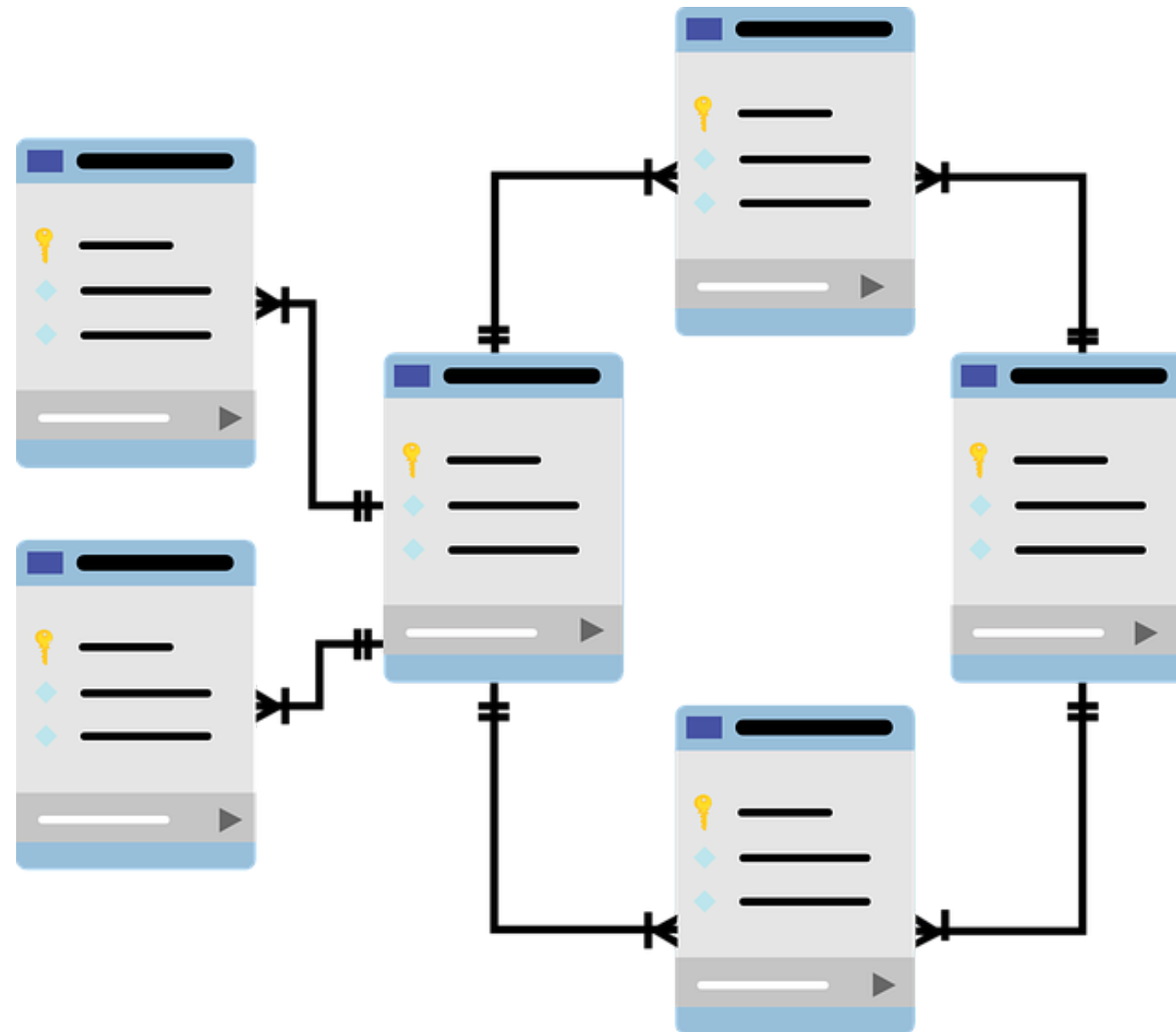
# Joining tables

INTRODUCTION TO SQL SERVER



John MacKintosh  
Instructor

# Relational Databases



# Primary Keys

- Primary keys: Uniquely identify each row in a table

```
+-----+-----+
| artist_id | name          |
+-----+-----+
| 1         | AC/DC         |
| 2         | Accept        |
| 3         | Aerosmith     |
| 4         | Alanis Morissette |
| 5         | Alice In Chains |
+-----+-----+
```

- Primary key: `artist_id`

```

+-----+-----+-----+
| album_id | title                | artist_id |
+-----+-----+-----+
| 1         | For Those About To Rock | 1         |
| 2         | Balls to the Wall       | 2         |
| 3         | Restless and Wild       | 2         |
| 4         | Let There Be Rock       | 1         |
| 5         | Big Ones                | 3         |
+-----+-----+-----+

```

- Primary key: `album_id`
- What about `artist_id` ?

# Foreign keys

- `artist` table

```
+-----+-----+
| artist_id | name          |
+-----+-----+
| 1          | AC/DC         |
| 2          | Accept        |
| 3          | Aerosmith     |
| 4          | Alanis Morissette |
| 5          | Alice In Chains |
+-----+-----+
```

- `album` table

```
+-----+-----+-----+
| album_id | title          | artist_id |
+-----+-----+-----+
| 1         | For Those About To Rock | 1         |
| 2         | Balls to the Wall      | 2         |
| 3         | Restless and Wild      | 2         |
| 4         | Let There Be Rock      | 1         |
| 5         | Big Ones             | 3         |
+-----+-----+-----+
```

- `artist_id` : Foreign key to `artist`

# Joining album and artist

- `artist` table

```
+-----+-----+
| artist_id | name           |
+-----+-----+
| 1         | AC/DC          |
| 2         | Accept         |
| 3         | Aerosmith      |
| 4         | Alanis Morissette |
| 5         | Alice In Chains |
+-----+-----+
```

- AC/DC has `artist_id` = 1

- `album` table

```
+-----+-----+-----+
| album_id | title           | artist_id |
+-----+-----+-----+
| 1         | For Those About To Rock | 1         |
| 2         | Balls to the Wall      | 2         |
| 3         | Restless and Wild      | 2         |
| 4         | Let There Be Rock      | 1         |
| 5         | Big Ones             | 3         |
+-----+-----+-----+
```

- Rows 1 and 4 have `artist_id` = 1

# Joining album and artist

```
+-----+-----+-----+-----+
| album_id | title                | artist_id | artist_name |
+-----+-----+-----+-----+
| 1        | For Those About To Rock | 1         | AC/DC       |
| 4        | Let There Be Rock       | 1         | AC/DC       |
+-----+-----+-----+-----+
```

- Return album details from `album` table
- Return corresponding artist details from `artist` table
- Joined using `artist_id` column

# INNER JOIN

```
SELECT
  album_id,
  title,
  album.artist_id,
  name AS artist_name
FROM album
INNER JOIN artist ON artist.artist_id = album.artist_id
WHERE album.artist_id = 1;
```

```
+-----+-----+-----+-----+
| album_id | title                | artist_id | artist_name |
+-----+-----+-----+-----+
| 1        | For Those About To Rock | 1        | AC/DC      |
| 4        | Let There Be Rock      | 1        | AC/DC      |
+-----+-----+-----+-----+
```



# INNER JOIN syntax

**SELECT**

```
table_A.columnX,  
table_A.columnY,  
table_B.columnZ
```

**FROM** table\_A

**INNER JOIN** table\_B **ON** table\_B.foreign\_key = table\_A.primary\_key;

```
SELECT
  album_id,
  title,
  album.artist_id,
  name AS artist_name
FROM album
INNER JOIN artist on artist.artist_id = album.artist_id;
```

```
+-----+-----+-----+-----+
| album_id | title                | artist_id | artist_name |
+-----+-----+-----+-----+
| 1        | For Those About To Rock | 1         | AC/DC       |
| 4        | Let There Be Rock      | 1         | AC/DC       |
| 2        | Balls To The Wall      | 2         | Accept      |
| 3        | Restless and Wild      | 2         | Accept      |
+-----+-----+-----+-----+
```

- Returns **all** combinations of **all** matches between `album` and `artist`

# Multiple INNER JOINS

```
SELECT
```

```
    table_A.columnX,  
    table_A.columnY,  
    table_B.columnZ table_C columnW
```

```
FROM table_A
```

```
INNER JOIN table_B ON table_B.foreign_key = table_A.primary_key
```

```
INNER JOIN table_C ON table_C.foreign_key = table_B.primary_key;
```

# Let's join some tables!

INTRODUCTION TO SQL SERVER

# Mix n match - LEFT & RIGHT joins

INTRODUCTION TO SQL SERVER



John MacKintosh  
Instructor

# The rationale for LEFT and RIGHT joins

- Why do we need LEFT and RIGHT joins?
- One table may not have an exact match in another:
  - Customer order history for marketing campaign
  - Product list and returns history
  - Patients admitted but not yet discharged

# The rationale for LEFT and RIGHT joins

- Why do we need LEFT and RIGHT joins?
- One table may not have an exact match in another:
  - Customer order history for marketing campaign
  - Product list and returns history
  - **Patients admitted but not yet discharged**

## Admissions table

```
+-----+-----+
| Patient_ID | Admitted |
+-----+-----+
| 1          | 1        |
| 2          | 1        |
| 3          | 1        |
| 4          | 1        |
| 5          | 1        |
+-----+-----+
```

## Discharges table

```
+-----+-----+
| Patient_ID | Admitted |
+-----+-----+
| 1          | 1        |
| 3          | 1        |
| 4          | 1        |
+-----+-----+
```

## INNER JOIN:

```
+-----+-----+-----|
| Patient_ID | Admitted | Discharged |
+-----+-----+-----|
| 1          | 1        | 1          |
| 3          | 1        | 1          |
| 4          | 1        | 1          |
+-----+-----+-----+
```

## LEFT JOIN:

```
+-----+-----+-----|
| Patient_ID | Admitted | Discharged |
+-----+-----+-----|
| 1          | 1        | 1          |
| 2          | 1        | NULL       |
| 3          | 1        | 1          |
| 4          | 1        | 1          |
| 5          | 1        | NULL       |
+-----+-----+-----+
```



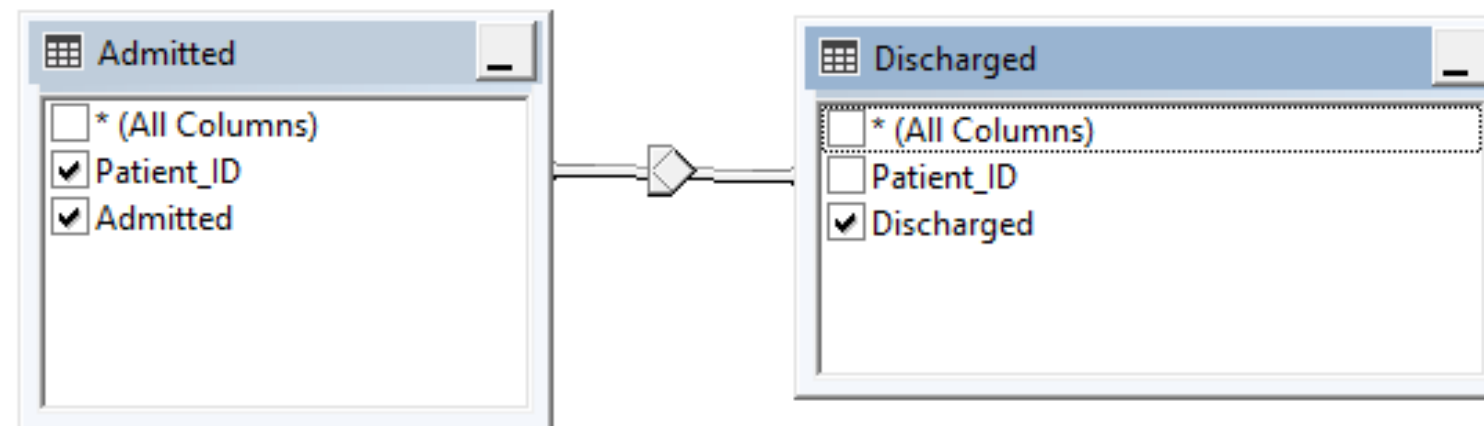
# LEFT JOIN SYNTAX

## SELECT

```
Admitted.Patient_ID,  
Admitted,  
Discharged
```

**FROM** Admitted

**LEFT JOIN** Discharged **ON** Discharged.Patient\_ID = Admitted.Patient\_ID;



**SELECT**

Admitted.Patient\_ID,  
Admitted,  
Discharged

**FROM** Admitted

**LEFT JOIN** Discharged **ON** Discharged.Patient\_ID = Admitted.Patient\_ID;

Patient_ID	Admitted	Discharged
1	1	1
2	1	NULL
3	1	1
4	1	1
5	1	NULL

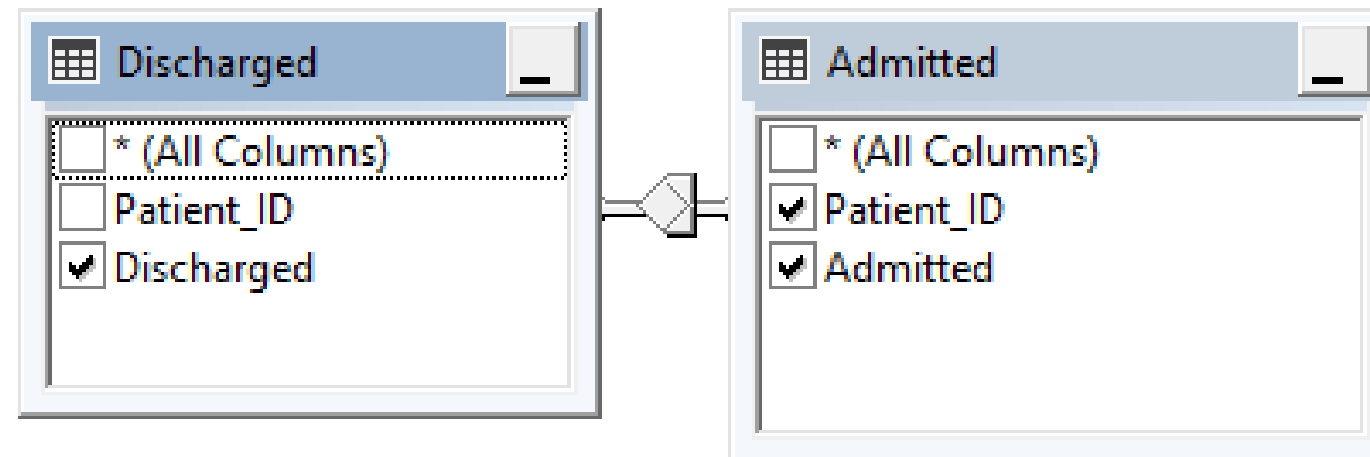
# RIGHT JOIN

**SELECT**

Admitted.Patient\_ID,  
Admitted,  
Discharged

**FROM** Discharged

**RIGHT JOIN** Admitted **ON** Admitted.Patient\_ID = Discharged.Patient\_ID;



# RIGHT JOIN results

```
SELECT
  Admitted.Patient_ID,
  Admitted,
  Discharged
FROM Discharged
RIGHT JOIN Admitted ON Admitted.Patient_ID = Discharged.Patient_ID;
```

Patient_ID	Admitted	Discharged
1	1	1
2	1	NULL
3	1	1
4	1	1
5	1	NULL

# Summary

- `INNER JOIN` : Only returns matching rows
- `LEFT JOIN` (or `RIGHT JOIN` ): All rows from the main table plus matches from the joining table
- `NULL` : Displayed if no match is found
- `LEFT JOIN` and `RIGHT JOIN` can be interchangeable

## INNER JOIN

LEFT TABLE	RIGHT TABLE
MATCHES RETURNED, NON MATCHES DISCARDED	MATCHES RETURNED, NON MATCHES DISCARDED

## LEFT JOIN

LEFT - MAIN TABLE	RIGHT - JOINING TABLE
ALL ROWS RETURNED	MATCHES RETURNED, NON MATCHES RETURN NULL

## RIGHT JOIN

LEFT - JOINING TABLE	RIGHT - MAIN TABLE
MATCHES RETURNED, NON MATCHES RETURN NULL	ALL ROWS RETURNED

# Let's Practice!

INTRODUCTION TO SQL SERVER

# UNION & UNION ALL

INTRODUCTION TO SQL SERVER



John MacKintosh  
Instructor



```
SELECT
  album_id,
  title,
  artist_id
FROM album
WHERE artist_id IN (1, 3)
```

```
+-----+-----+-----+
| album_id | title                | artist_id |
+-----+-----+-----+
| 1        | For Those About To Rock | 1        |
| 4        | Let There Be Rock      | 1        |
| 5        | Big Ones               | 3        |
+-----+-----+-----+
```

```
SELECT
  album_id,
  title,
  artist_id
FROM album
WHERE artist_id IN (1, 4, 5)
```

```
+-----+-----+-----+
| album_id | title                | artist_id |
+-----+-----+-----+
| 1        | For Those About To Rock | 1        |
| 4        | Let There Be Rock      | 1        |
| 6        | Jagged Little Pill     | 4        |
| 7        | Facelift               | 5        |
+-----+-----+-----+
```

# Combining results

```
SELECT
  album_id,
  title,
  artist_id
FROM album
WHERE artist_id IN (1, 3)

UNION

SELECT
  album_id,
  title,
  artist_id
FROM album
WHERE artist_id IN (1, 4, 5);
```

album_id	title	artist_id
1	For Those About To Rock	1
4	Let There Be Rock	1
5	Big Ones	3
6	Jagged Little Pill	4
7	Facelift	5

- Duplicate rows are excluded

# UNION ALL

```
SELECT
  album_id,
  title,
  artist_id
FROM album
WHERE artist_id IN (1, 3)
UNION ALL
SELECT
  album_id,
  title,
  artist_id
FROM album
WHERE artist_id IN (1, 4, 5);
```

album_id	title	artist_id
1	For Those About To Rock	1
4	Let There Be Rock	1
5	Big Ones	3
1	For Those About To Rock	1
4	Let There Be Rock	1
6	Jagged Little Pill	4
7	Facelift	5

- Includes duplicate rows

# Creating new column names for final results

```
SELECT
  album_id AS ALBUM_ID,
  title AS ALBUM_TITLE,
  artist_id AS ARTIST_ID
FROM album
WHERE artist_id IN(1, 3)
UNION ALL
SELECT
  album_id AS ALBUM_ID,
  title AS ALBUM_TITLE,
  artist_id AS ARTIST_ID
FROM album
WHERE artist_id IN(1, 4, 5)
```

```
+-----+-----+-----+
| ALBUM_ID | ALBUM_TITLE           | ARTIST_ID |
+-----+-----+-----+
| 1        | For Those About To Rock | 1         |
| 4        | Let There Be Rock      | 1         |
| 5        | Big Ones               | 3         |
| 1        | For Those About To Rock | 1         |
| 4        | Let There Be Rock      | 1         |
| 6        | Jagged Little Pill     | 4         |
| 7        | Facelift               | 5         |
+-----+-----+-----+
```

# Summary

- `UNION` or `UNION ALL` : Combines queries from the same table or different tables

If combining data from different tables:

- Select the same number of columns in the same order
- Columns should have the same data types

If source tables have different column names

- Alias the column names

`UNION` : Discards duplicates (slower to run)

`UNION ALL` : Includes duplicates (faster to run)

# Let's practice!

INTRODUCTION TO SQL SERVER