

SQL: Outer Join

by Ryan Price



RyanPriceMedia.com



Wednesday, July 30, 14

1

You may follow this presentation using the `outline.md` text file or the presenter notes, [here](#).

Joins: Inner vs. Outer Joins

=====

Welcome to Part 2 of the lesson on database joins with SQL. This lesson builds on the previous lesson about joins, introducing the Outer Join concept. The database uses a "Battle School" as the example.

Table of Contents

➔ 01 What is an Outer Join for?

02 Live Demo

03 Syntax Review

04 Code Challenge

SQL: Outer Join



1. What is an Outer Join for?
2. Live demo
3. Syntax review
4. Code challenge

What is an Outer Join for?

SQL: Outer Join



Wednesday, July 30, 14

3

##What is an Outer Join for?

If one of your tables references another and allows NULL keys, you may need an Outer Join to get all the records you selected.

What is an Outer Join for?

If you have NULL values, you want to think Outer Join

SQL: Outer Join



What is an Outer Join for?

SQL: Outer Join



Wednesday, July 30, 14

4

##What is an Outer Join for?

Let's talk again about the Principle of *Normalization*. In our last example, we say how a Join can bring our data back together after we split it into tables using Normalization. One reason for Normalization is so we don't have bunches of NULL columns in rows where that information is not relevant. However, when we join tables together, there is more than one way to do that.

What is an Outer Join for?

SQL: Outer Join



Wednesday, July 30, 14

4

##What is an Outer Join for?

Let's talk again about the Principle of *Normalization*. In our last example, we say how a Join can bring our data back together after we split it into tables using Normalization. One reason for Normalization is so we don't have bunches of NULL columns in rows where that information is not relevant. However, when we join tables together, there is more than one way to do that.

What is an Outer Join for?

Normalization

Multiple rows storing similar data; reduce redundancy.

Use NULL IDs if data is not present.

SQL: Outer Join



Let's talk again about the Principle of *Normalization*. In our last example, we say how a Join can bring our data back together after we split it into tables using Normalization. One reason for Normalization is so we don't have bunches of NULL columns in rows where that information is not relevant. However, when we join tables together, there is more than one way to do that.

What is an Outer Join for?

Normalization

Multiple rows storing similar data; reduce redundancy.

Use NULL IDs if data is not present.

Cadet Table with Awards

SQL: Outer Join



Let's talk again about the Principle of *Normalization*. In our last example, we say how a Join can bring our data back together after we split it into tables using Normalization. One reason for Normalization is so we don't have bunches of NULL columns in rows where that information is not relevant. However, when we join tables together, there is more than one way to do that.

What is an Outer Join for?

Normalization

Multiple rows storing similar data; reduce redundancy.

Use NULL IDs if data is not present.

Cadet Table with Awards

I	“Andrew”	“MVP”
---	----------	-------

SQL: Outer Join



Let's talk again about the Principle of *Normalization*. In our last example, we say how a Join can bring our data back together after we split it into tables using Normalization. One reason for Normalization is so we don't have bunches of NULL columns in rows where that information is not relevant. However, when we join tables together, there is more than one way to do that.

What is an Outer Join for?

Normalization

Multiple rows storing similar data; reduce redundancy.

Use NULL IDs if data is not present.

Cadet Table with Awards

1	"Andrew"	"MVP"
2	"Julian"	"MVP"

SQL: Outer Join



Let's talk again about the Principle of *Normalization*. In our last example, we say how a Join can bring our data back together after we split it into tables using Normalization. One reason for Normalization is so we don't have bunches of NULL columns in rows where that information is not relevant. However, when we join tables together, there is more than one way to do that.

What is an Outer Join for?

Normalization

Multiple rows storing similar data; reduce redundancy.

Use NULL IDs if data is not present.

Cadet Table with Awards

1	"Andrew"	"MVP"
2	"Julian"	"MVP"
3	"Petra"	"MVP"

SQL: Outer Join



Let's talk again about the Principle of *Normalization*. In our last example, we say how a Join can bring our data back together after we split it into tables using Normalization. One reason for Normalization is so we don't have bunches of NULL columns in rows where that information is not relevant. However, when we join tables together, there is more than one way to do that.

What is an Outer Join for?

Normalization

Multiple rows storing similar data; reduce redundancy.

Use NULL IDs if data is not present.

Cadet Table with Awards

1	"Andrew"	"MVP"
2	"Julian"	"MVP"
3	"Petra"	"MVP"
4	"Alai"	NULL

SQL: Outer Join



Let's talk again about the Principle of *Normalization*. In our last example, we say how a Join can bring our data back together after we split it into tables using Normalization. One reason for Normalization is so we don't have bunches of NULL columns in rows where that information is not relevant. However, when we join tables together, there is more than one way to do that.

What is an Outer Join for?

Normalization

Multiple rows storing similar data; reduce redundancy.

Use NULL IDs if data is not present.

Cadet Table with Awards

1	"Andrew"	"MVP"
2	"Julian"	"MVP"
3	"Petra"	"MVP"
4	"Alai"	NULL
5	"Andrew"	"Tactics"

SQL: Outer Join



Let's talk again about the Principle of *Normalization*. In our last example, we say how a Join can bring our data back together after we split it into tables using Normalization. One reason for Normalization is so we don't have bunches of NULL columns in rows where that information is not relevant. However, when we join tables together, there is more than one way to do that.

What is an Outer Join for?

Normalization

Multiple rows storing similar data; reduce redundancy.

Use NULL IDs if data is not present.

Cadet Table with Awards

1	"Andrew"	"MVP"
2	"Julian"	"MVP"
3	"Petra"	"MVP"
4	"Alai"	NULL
5	"Andrew"	"Tactics"



SQL: Outer Join



Let's talk again about the Principle of *Normalization*. In our last example, we say how a Join can bring our data back together after we split it into tables using Normalization. One reason for Normalization is so we don't have bunches of NULL columns in rows where that information is not relevant. However, when we join tables together, there is more than one way to do that.

What is an Outer Join for?

Normalization

Multiple rows storing similar data; reduce redundancy.

Use NULL IDs if data is not present.

Cadet Table with Awards

1	"Andrew"	"MVP"
2	"Julian"	"MVP"
3	"Petra"	"MVP"
4	"Alai"	NULL
5	"Andrew"	"Tactics"



Cadet Table

1	"Andrew"
2	"Julian"
3	"Petra"
4	"Alai"

SQL: Outer Join



Let's talk again about the Principle of *Normalization*. In our last example, we say how a Join can bring our data back together after we split it into tables using Normalization. One reason for Normalization is so we don't have bunches of NULL columns in rows where that information is not relevant. However, when we join tables together, there is more than one way to do that.

What is an Outer Join for?

Normalization

Multiple rows storing similar data; reduce redundancy.

Use NULL IDs if data is not present.

Cadet Table with Awards

1	"Andrew"	"MVP"
2	"Julian"	"MVP"
3	"Petra"	"MVP"
4	"Alai"	NULL
5	"Andrew"	"Tactics"



Cadet Table

1	"Andrew"
2	"Julian"
3	"Petra"
4	"Alai"

Award Table

1	"MVP"
1	"Tactics"
2	"MVP"
3	"MVP"

SQL: Outer Join



Let's talk again about the Principle of *Normalization*. In our last example, we say how a Join can bring our data back together after we split it into tables using Normalization. One reason for Normalization is so we don't have bunches of NULL columns in rows where that information is not relevant. However, when we join tables together, there is more than one way to do that.

What is an Outer Join for?

Normalization

Multiple rows storing similar data; reduce redundancy.

Use NULL IDs if data is not present.

Cadet Table with Awards

1	"Andrew"	"MVP"
2	"Julian"	"MVP"
3	"Petra"	"MVP"
4	"Alai"	NULL
5	"Andrew"	"Tactics"

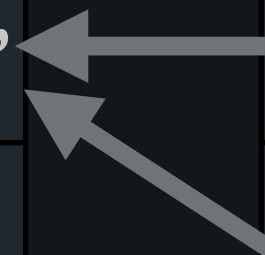


Cadet Table

1	"Andrew"
2	"Julian"
3	"Petra"
4	"Alai"

Award Table

1	"MVP"
1	"Tactics"
2	"MVP"
3	"MVP"



SQL: Outer Join



Let's talk again about the Principle of *Normalization*. In our last example, we say how a Join can bring our data back together after we split it into tables using Normalization. One reason for Normalization is so we don't have bunches of NULL columns in rows where that information is not relevant. However, when we join tables together, there is more than one way to do that.

What is an Outer Join for?

Normalization

Multiple rows storing similar data; reduce redundancy.

Use NULL IDs if data is not present.

Cadet Table with Awards

1	"Andrew"	"MVP"
2	"Julian"	"MVP"
3	"Petra"	"MVP"
4	"Alai"	NULL
5	"Andrew"	"Tactics"



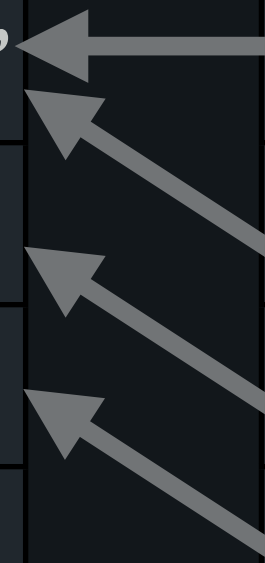
SQL: Outer Join

Cadet Table

1	"Andrew"
2	"Julian"
3	"Petra"
4	"Alai"

Award Table

1	"MVP"
1	"Tactics"
2	"MVP"
3	"MVP"



Let's talk again about the Principle of *Normalization*. In our last example, we say how a Join can bring our data back together after we split it into tables using Normalization. One reason for Normalization is so we don't have bunches of NULL columns in rows where that information is not relevant. However, when we join tables together, there is more than one way to do that.

What is an Outer Join for?

Normalization

Multiple rows storing similar data; reduce redundancy.

Use NULL IDs if data is not present.

Cadet Table with Awards

1	"Andrew"	"MVP"
2	"Julian"	"MVP"
3	"Petra"	"MVP"
4	"Alai"	NULL
5	"Andrew"	"Tactics"



SQL: Outer Join

Cadet Table

1	"Andrew"
2	"Julian"
3	"Petra"
4	"Alai"

Award Table

1	"MVP"
1	"Tactics"
2	"MVP"
3	"MVP"



Let's talk again about the Principle of *Normalization*. In our last example, we say how a Join can bring our data back together after we split it into tables using Normalization. One reason for Normalization is so we don't have bunches of NULL columns in rows where that information is not relevant. However, when we join tables together, there is more than one way to do that.

Inner Join vs. Left Outer Join

SQL: Outer Join



Wednesday, July 30, 14

5

If you look at a Venn Diagram of two tables A and B using the default Join, or **Inner Join**, you see that the default join only shows data where the two circles overlap.

What this means is that the table on the left may have lots of rows that match our WHERE clause, but the conditions of the Inner Join are preventing them from being returned. That's where the **Outer Join** comes in. The Outer Join allows us to include matching rows from the table on the left, without a corresponding row on the right. That's why an Outer join can also be referred to as a **Left Join**.

Inner Join vs. Left Outer Join

“Inner Join” includes rows that match in both tables vs. all of the “Left” table.

SQL: Outer Join



Wednesday, July 30, 14

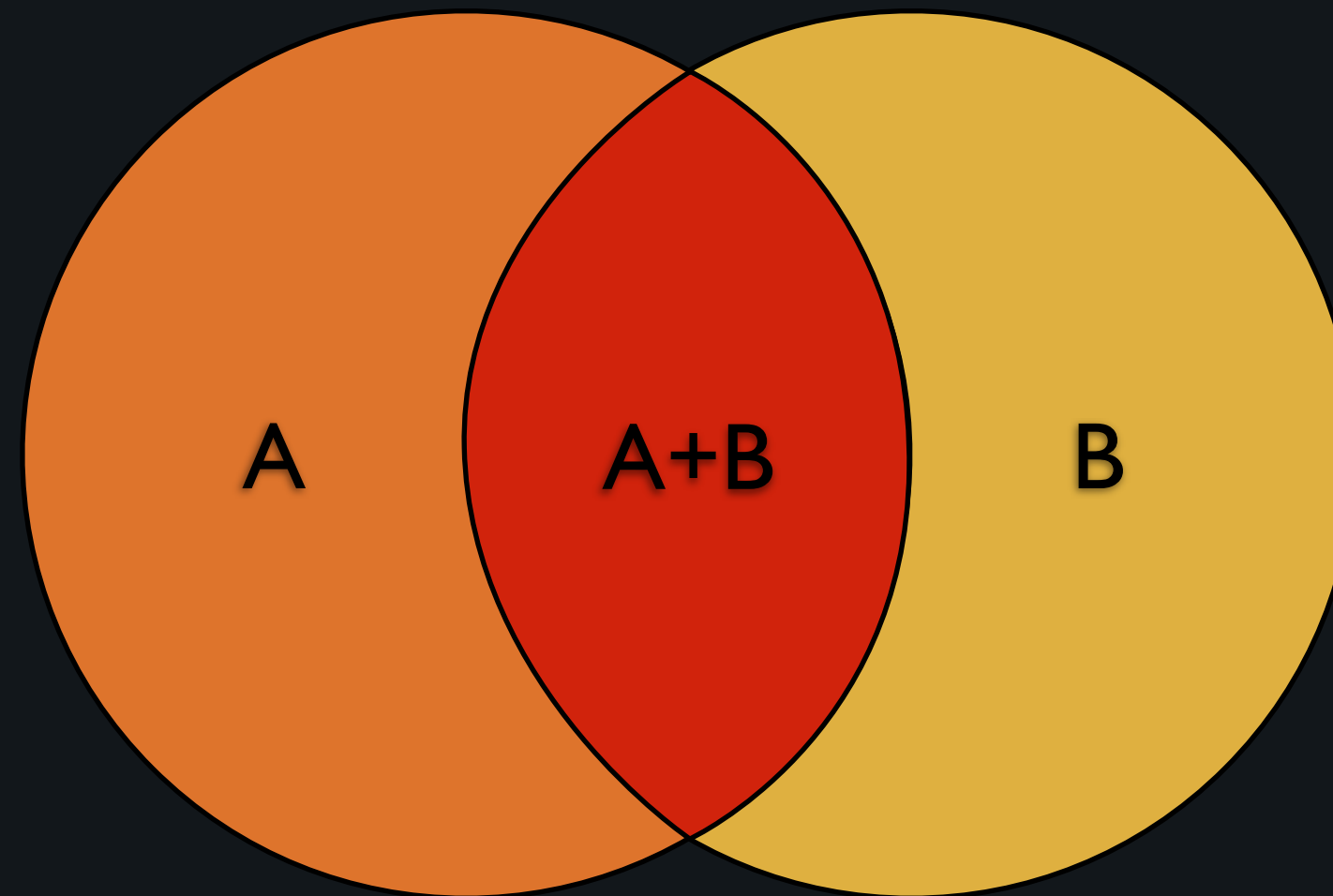
5

If you look at a Venn Diagram of two tables A and B using the default Join, or `*Inner Join*`, you see that the default join only shows data where the two circles overlap.

What this means is that the table on the left may have lots of rows that match our WHERE clause, but the conditions of the Inner Join are preventing them from being returned. That's where the `*Outer Join*` comes in. The Outer Join allows us to include matching rows from the table on the left, without a corresponding row on the right. That's why an Outer join can also be referred to as a `*Left Join*`.

Inner Join vs. Left Outer Join

“Inner Join” includes rows that match in both tables vs. all of the “Left” table.



SQL: Outer Join



Wednesday, July 30, 14

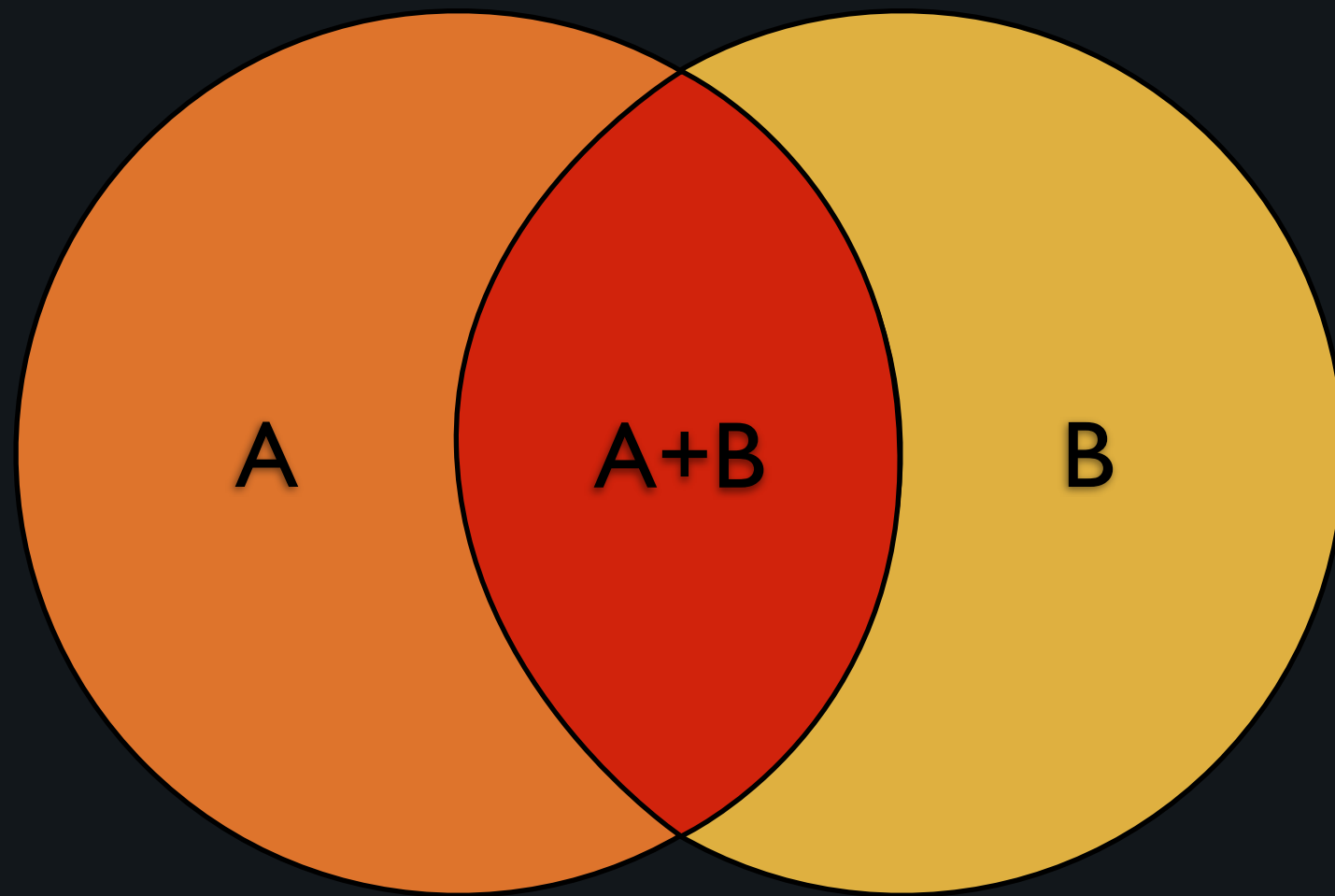
5

If you look at a Venn Diagram of two tables A and B using the default Join, or *Inner Join*, you see that the default join only shows data where the two circles overlap.

What this means is that the table on the left may have lots of rows that match our WHERE clause, but the conditions of the Inner Join are preventing them from being returned. That's where the *Outer Join* comes in. The Outer Join allows us to include matching rows from the table on the left, without a corresponding row on the right. That's why an Outer join can also be referred to as a *Left Join*.

Inner Join vs. Left Outer Join

“Inner Join” includes rows that match in both tables vs. all of the “Left” table.



SQL: Outer Join

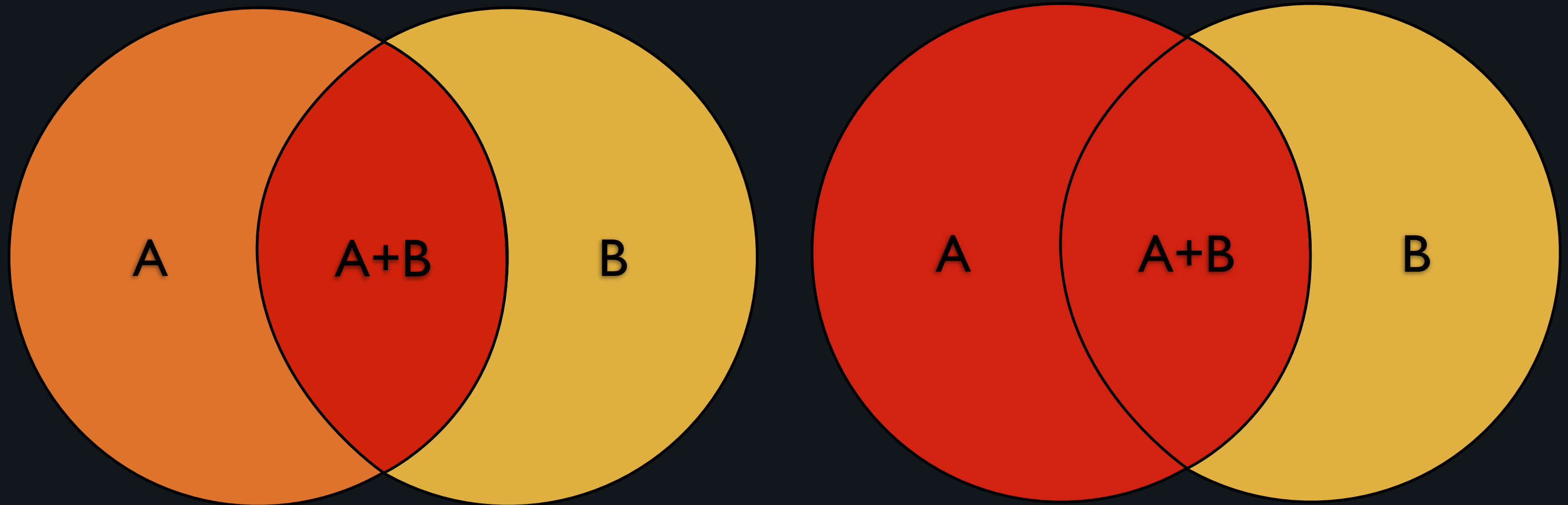


If you look at a Venn Diagram of two tables A and B using the default Join, or *Inner Join*, you see that the default join only shows data where the two circles overlap.

What this means is that the table on the left may have lots of rows that match our WHERE clause, but the conditions of the Inner Join are preventing them from being returned. That's where the *Outer Join* comes in. The Outer Join allows us to include matching rows from the table on the left, without a corresponding row on the right. That's why an Outer join can also be referred to as a *Left Join*.

Inner Join vs. Left Outer Join

“Inner Join” includes rows that match in both tables vs. all of the “Left” table.



SQL: Outer Join



If you look at a Venn Diagram of two tables A and B using the default Join, or *Inner Join*, you see that the default join only shows data where the two circles overlap.

What this means is that the table on the left may have lots of rows that match our WHERE clause, but the conditions of the Inner Join are preventing them from being returned. That's where the *Outer Join* comes in. The Outer Join allows us to include matching rows from the table on the left, without a corresponding row on the right. That's why an Outer join can also be referred to as a *Left Join*.

Inner Join vs. Left Outer Join

SQL: Outer Join



Wednesday, July 30, 14

6

- * Show all of A, and include B if it exists, matching on a condition, given with the ON statement.
 - *e.g. "Show a list of all cadets and awards they have won, if any."*
- * Inner Joins include just those rows that match. Even if some rows from table A meet the criteria, an Inner Join would not include them unless there is a corresponding row in table B.
 - *e.g. an Inner Join would ONLY show cadets who have won awards*

Inner Join vs. Left Outer Join

SQL: Outer Join



Wednesday, July 30, 14

6

- * Show all of A, and include B if it exists, matching on a condition, given with the ON statement.
 - *e.g. "Show a list of all cadets and awards they have won, if any."*
- * Inner Joins include just those rows that match. Even if some rows from table A meet the criteria, an Inner Join would not include them unless there is a corresponding row in table B.
 - *e.g. an Inner Join would ONLY show cadets who have won awards*

Inner Join vs. Left Outer Join

Inner Join

```
SELECT c.cadet_id, c.name, a.name  
FROM cadet AS c JOIN award AS a  
WHERE c.cadet_id = a.cadet_id ;
```

SQL: Outer Join



- * Show all of A, and include B if it exists, matching on a condition, given with the ON statement.
 - *e.g. "Show a list of all cadets and awards they have won, if any."*
- * Inner Joins include just those rows that match. Even if some rows from table A meet the criteria, an Inner Join would not include them unless there is a corresponding row in table B.
 - *e.g. an Inner Join would ONLY show cadets who have won awards*

Inner Join vs. Left Outer Join

Inner Join

```
SELECT c.cadet_id, c.name, a.name  
FROM cadet AS c JOIN award AS a  
WHERE c.cadet_id = a.cadet_id ;
```

Cadet Table Inner Join with Awards

1	"Andrew"	1	"MVP"
1	"Andrew"	2	"Tactics"
2	"Julian"	3	"MVP"
3	"Petra"	4	"MVP"

SQL: Outer Join



- * Show all of A, and include B if it exists, matching on a condition, given with the ON statement.
 - *e.g. "Show a list of all cadets and awards they have won, if any."*
- * Inner Joins include just those rows that match. Even if some rows from table A meet the criteria, an Inner Join would not include them unless there is a corresponding row in table B.
 - *e.g. an Inner Join would ONLY show cadets who have won awards*

Inner Join vs. Left Outer Join

Left Outer Join -- `query1.sql`

```
SELECT c.cadet_id, c.name, a.name
FROM cadet AS c LEFT OUTER JOIN award AS a ON c.cadet_id = a.cadet_id ;
```

Cadet Table Outer Join with Awards

1	"Andrew"	1	"MVP"
1	"Andrew"	2	"Tactics"
2	"Julian"	3	"MVP"
3	"Petra"	4	"MVP"
4	"Alai"	-	NULL

SQL: Outer Join



- * Show all of A, and include B if it exists, matching on a condition, given with the ON statement.
 - *e.g. "Show a list of all cadets and awards they have won, if any."*
- * Outer Joins include all matching rows from the table on the Left; a match on the Right is not needed.
 - *i.e. All cadets will be shown, their awards only appear if they exist.*
- * If there are no NULL values on the key to the join, there will be no difference between Inner and Outer Join.
- * The most common type of Outer Join is a Left Join. SQLite only implements a Left Outer Join.
- * You may use `LEFT JOIN` and `LEFT OUTER JOIN` interchangeably in SQLite.

Inner Join vs. Left Outer Join

Left Outer Join -- `query1.sql`

```
SELECT c.cadet_id, c.name, a.name  
FROM cadet AS c LEFT OUTER JOIN award AS a ON c.cadet_id = a.cadet_id ;
```



Cadet Table Outer Join with Awards

1	"Andrew"	1	"MVP"
1	"Andrew"	2	"Tactics"
2	"Julian"	3	"MVP"
3	"Petra"	4	"MVP"
4	"Alai"	-	NULL

SQL: Outer Join




- * Show all of A, and include B if it exists, matching on a condition, given with the ON statement.
 - *e.g. "Show a list of all cadets and awards they have won, if any."*
- * Outer Joins include all matching rows from the table on the Left; a match on the Right is not needed.
 - *i.e. All cadets will be shown, their awards only appear if they exist.*
- * If there are no NULL values on the key to the join, there will be no difference between Inner and Outer Join.
- * The most common type of Outer Join is a Left Join. SQLite only implements a Left Outer Join.
- * You may use `LEFT JOIN` and `LEFT OUTER JOIN` interchangeably in SQLite.

Inner Join vs. Left Outer Join

Left Outer Join -- `query1.sql`

```
SELECT c.cadet_id, c.name, a.name  
FROM cadet AS c LEFT OUTER JOIN award AS a ON c.cadet_id = a.cadet_id ;
```



Outer Join needs ON

Cadet Table Outer Join with Awards

1	"Andrew"	1	"MVP"
1	"Andrew"	2	"Tactics"
2	"Julian"	3	"MVP"
3	"Petra"	4	"MVP"
4	"Alai"	-	NULL

SQL: Outer Join



Wednesday, July 30, 14

7

- * Show all of A, and include B if it exists, matching on a condition, given with the ON statement.
 - *e.g. "Show a list of all cadets and awards they have won, if any."*
- * Outer Joins include all matching rows from the table on the Left; a match on the Right is not needed.
 - *i.e. All cadets will be shown, their awards only appear if they exist.*
- * If there are no NULL values on the key to the join, there will be no difference between Inner and Outer Join.
- * The most common type of Outer Join is a Left Join. SQLite only implements a Left Outer Join.
- * You may use `LEFT JOIN` and `LEFT OUTER JOIN` interchangeably in SQLite.

Inner Join vs. Left Outer Join

Gotcha I

```
SELECT c.cadet_id, c.name, a.name  
FROM award AS a LEFT OUTER JOIN cadet AS c ON c.cadet_id = a.cadet_id ;
```

Award Table Outer Join with Cadet

1	"Andrew"	1	"MVP"
1	"Andrew"	2	"Tactics"
2	"Julian"	3	"MVP"
3	"Petra"	4	"MVP"

SQL: Outer Join



* The Left and Right tables in Inner Joins can be reordered without affecting the results, but for Outer Joins the ordering of the tables matters.

i.e. Placing awards on the Left would mean all awards were included, not all Cadets.

Inner Join vs. Left Outer Join

Gotcha 1

```
SELECT c.cadet_id, c.name, a.name  
FROM award AS a LEFT OUTER JOIN cadet AS c ON c.cadet_id = a.cadet_id ;
```



Award Table Outer Join with Cadet

1	"Andrew"	1	"MVP"
1	"Andrew"	2	"Tactics"
2	"Julian"	3	"MVP"
3	"Petra"	4	"MVP"

SQL: Outer Join



* The Left and Right tables in Inner Joins can be reordered without affecting the results, but for Outer Joins the ordering of the tables matters.

i.e. Placing awards on the Left would mean all awards were included, not all Cadets.

Inner Join vs. Left Outer Join

Gotcha 1

```
SELECT c.cadet_id, c.name, a.name  
FROM award AS a LEFT OUTER JOIN cadet AS c ON c.cadet_id = a.cadet_id ;
```



Reordering tables changes results!

Award Table Outer Join with Cadet

1	"Andrew"	1	"MVP"
1	"Andrew"	2	"Tactics"
2	"Julian"	3	"MVP"
3	"Petra"	4	"MVP"

SQL: Outer Join



* The Left and Right tables in Inner Joins can be reordered without affecting the results, but for Outer Joins the ordering of the tables matters.

i.e. Placing awards on the Left would mean all awards were included, not all Cadets.

Inner Join vs. Left Outer Join

Gotcha 2

```
SELECT c.cadet_id, c.name, a.name  
FROM cadet AS c LEFT OUTER JOIN award AS a ON c.cadet_id = a.cadet_id  
WHERE a.name = "MVP" ;
```

Cadet Left Join with Award name = "MVP"

1	"Andrew"	1	"MVP"
2	"Julian"	3	"MVP"
3	"Petra"	4	"MVP"

SQL: Outer Join



* In a Left join, you want to use the ON or USING syntax. If you use WHERE, you may omit rows.

i.e. If a row in the `award` table is NULL, its `cadet_id` cannot be equal to an `id` in the `cadet` table.

Inner Join vs. Left Outer Join

Gotcha 2

```
SELECT c.cadet_id, c.name, a.name  
FROM cadet AS c LEFT OUTER JOIN award AS a ON c.cadet_id = a.cadet_id  
WHERE a.name = "MVP" ;
```



Cadet Left Join with Award name = "MVP"

1	"Andrew"	1	"MVP"
2	"Julian"	3	"MVP"
3	"Petra"	4	"MVP"

SQL: Outer Join




* In a Left join, you want to use the ON or USING syntax. If you use WHERE, you may omit rows.

i.e. If a row in the `award` table is NULL, its `cadet_id` cannot be equal to an `id` in the `cadet` table.

Inner Join vs. Left Outer Join

Gotcha 2

```
SELECT c.cadet_id, c.name, a.name  
FROM cadet AS c LEFT OUTER JOIN award AS a ON c.cadet_id = a.cadet_id  
WHERE a.name = "MVP" ;
```



WHERE can negate the Outer Join!

Cadet Left Join with Award name = "MVP"

1	"Andrew"	1	"MVP"
2	"Julian"	3	"MVP"
3	"Petra"	4	"MVP"

SQL: Outer Join



* In a Left join, you want to use the ON or USING syntax. If you use WHERE, you may omit rows.

i.e. If a row in the `award` table is NULL, its `cadet_id` cannot be equal to an `id` in the `cadet` table.

Procedure: SQL SELECT

SQL: Outer Join



Procedure: SQL SELECT

1. Which tables to select **FROM**?

SQL: Outer Join



Procedure: SQL SELECT

1. Which tables to select **FROM**?
2. Which to **LEFT JOIN**? Table with possible **NULL** rows on the Right.

SQL: Outer Join



Procedure: SQL SELECT

1. Which tables to select **FROM**?
2. Which to **LEFT JOIN**? Table with possible **NULL** rows on the Right.
3. Instead of WHERE, use **ON** or **USING**.

SQL: Outer Join



Procedure: SQL SELECT

1. Which tables to select **FROM**?
2. Which to **LEFT JOIN**? Table with possible **NULL** rows on the Right.
3. Instead of WHERE, use **ON** or **USING**.
4. Specify the **WHERE** clause.

SQL: Outer Join



Procedure: SQL SELECT

1. Which tables to select **FROM**?
2. Which to **LEFT JOIN**? Table with possible **NULL** rows on the Right.
3. Instead of WHERE, use **ON** or **USING**.
4. Specify the **WHERE** clause.
5. Decide which fields to **SELECT**.

SQL: Outer Join



Procedure: SQL SELECT

1. Which tables to select **FROM**?
2. Which to **LEFT JOIN**? Table with possible **NULL** rows on the Right.
3. Instead of WHERE, use **ON** or **USING**.
4. Specify the **WHERE** clause.
5. Decide which fields to **SELECT**.
6. Decide which fields to **ORDER BY**.

SQL: Outer Join



Procedure: SQL SELECT

1. Which tables to select **FROM**?
2. Which to **LEFT JOIN**? Table with possible **NULL** rows on the Right.
3. Instead of WHERE, use **ON** or **USING**.
4. Specify the **WHERE** clause.
5. Decide which fields to **SELECT**.
6. Decide which fields to **ORDER BY**.
7. Add a **LIMIT**.

SQL: Outer Join



Table of Contents

01 What is an Outer Join for?

➔ **02 Live Demo**

03 Syntax Review

04 Code Challenge

SQL: Outer Join



In this section, we will re-visit our systematic approach to writing queries, adding a rule for Outer Joins.

Live Demo

SQL: Outer Join



Wednesday, July 30, 14

12

##Live demo

1. Open the `join2.db` from the folder `join2-outer-inner`.
`\$ cd join2-outer-inner`
`\$ sqlite3 join2.db`
2. List the available tables, and get a schema description for the `cadet` and `award` table.
`sqlite3> .tables`
`sqlite3> .schema cadet`

Live Demo

1. Show a list of cadets and awards they have won, if any.

SQL: Outer Join



Wednesday, July 30, 14

12

##Live demo

1. Open the `join2.db` from the folder `join2-outer-inner`.
`\$ cd join2-outer-inner`
`\$ sqlite3 join2.db`
2. List the available tables, and get a schema description for the `cadet` and `award` table.
`sqlite3> .tables`
`sqlite3> .schema cadet`

Live Demo

1. Show a list of cadets and awards they have won, if any.

query1.sql

SQL: Outer Join



1. Open the `join2.db` from the folder `join2-outer-inner`.
`\$ cd join2-outer-inner`
`\$ sqlite3 join2.db`
2. List the available tables, and get a schema description for the `cadet` and `award` table.
`sqlite3> .tables`
`sqlite3> .schema cadet`

Live Demo

1. Show a list of cadets and awards they have won, if any.

`query1.sql`

2. Show a list of cadets and the names of cadets who froze them in battle.

SQL: Outer Join



1. Open the ``join2.db`` from the folder ``join2-outer-inner``.
``$ cd join2-outer-inner``
``$ sqlite3 join2.db``
2. List the available tables, and get a schema description for the ``cadet`` and ``award`` table.
``sqlite3> .tables``
``sqlite3> .schema cadet``

Live Demo

1. Show a list of cadets and awards they have won, if any.

`query1.sql`

2. Show a list of cadets and the names of cadets who froze them in battle.

`query2.sql`

SQL: Outer Join



1. Open the ``join2.db`` from the folder ``join2-outer-inner``.
``$ cd join2-outer-inner``
``$ sqlite3 join2.db``
2. List the available tables, and get a schema description for the ``cadet`` and ``award`` table.
``sqlite3> .tables``
``sqlite3> .schema cadet``

Live Demo

cadet
cadet_id
name
army_id
...

army
army_id
name
commander
...

cadet_battle
cadet_battle_id
cadet_id
battle_id
frozen_by_id
...

award
award_id
cadet_id
battle_id
...

battle
battle_id
fought
army1_id
army2_id
...

SQL: Outer Join



1. ****Show a list of cadets and awards they have won, if any.****
 1. Decide which table to select FROM, and which table to LEFT JOIN.
`cadet` and `award`
 2. Make sure you place the table with potential NULL columns on the Right.
`award` may not have rows for every `cadet`, or may have multiple matches for one `cadet`.
 3. Remember to use the ON or USING keywords to specify the conditions for the LEFT JOIN.
`FROM cadet, LEFT OUTER JOIN award ON award.cadet_id = cadet.cadet_id`
 4. Specify the WHERE clause(s).

Live Demo

cadet
cadet_id
name
army_id
...

army
army_id
name
commander
...

cadet_battle
cadet_battle_id
cadet_id
battle_id
frozen_by_id
...

award
award_id
cadet_id
battle_id
...

battle
battle_id
fought
army1_id
army2_id
...

SQL: Outer Join



1. ****Show a list of cadets and awards they have won, if any.****
 1. Decide which table to select FROM, and which table to LEFT JOIN.
`cadet` and `award`
 2. Make sure you place the table with potential NULL columns on the Right.
`award` may not have rows for every `cadet`, or may have multiple matches for one `cadet`.
 3. Remember to use the ON or USING keywords to specify the conditions for the LEFT JOIN.
`FROM cadet, LEFT OUTER JOIN award ON award.cadet_id = cadet.cadet_id`
 4. Specify the WHERE clause(s).

Live Demo

cadet
cadet_id
name
army_id
...

army
army_id
name
commander
...

cadet_battle
cadet_battle_id
cadet_id
battle_id
frozen_by_id
...

award
award_id
cadet_id
battle_id
...

battle
battle_id
fought
army1_id
army2_id
...

SQL: Outer Join



1. ****Show a list of cadets and awards they have won, if any.****
 1. Decide which table to select FROM, and which table to LEFT JOIN.
`cadet` and `award`
 2. Make sure you place the table with potential NULL columns on the Right.
`award` may not have rows for every `cadet`, or may have multiple matches for one `cadet`.
 3. Remember to use the ON or USING keywords to specify the conditions for the LEFT JOIN.
`FROM cadet, LEFT OUTER JOIN award ON award.cadet_id = cadet.cadet_id`
 4. Specify the WHERE clause(s).

Live Demo

cadet
cadet_id
name
army_id
...

army
army_id
name
commander
...

cadet_battle
cadet_battle_id
cadet_id
battle_id
frozen_by_id
...

award
award_id
cadet_id
battle_id
...

battle
battle_id
fought
army1_id
army2_id
...

0..∞
Outer
Join

SQL: Outer Join



1. ****Show a list of cadets and awards they have won, if any.****
 1. Decide which table to select FROM, and which table to LEFT JOIN.
`cadet` and `award`
 2. Make sure you place the table with potential NULL columns on the Right.
`award` may not have rows for every `cadet`, or may have multiple matches for one `cadet`.
 3. Remember to use the ON or USING keywords to specify the conditions for the LEFT JOIN.
`FROM cadet, LEFT OUTER JOIN award ON award.cadet_id = cadet.cadet_id`
 4. Specify the WHERE clause(s).

Live Demo

cadet
cadet_id
name
army_id
...

army
army_id
name
commander
...

cadet_battle
cadet_battle_id
cadet_id
battle_id
frozen_by_id
...

award
award_id
cadet_id
battle_id
...

battle
battle_id
fought
army1_id
army2_id
...

SQL: Outer Join



2. ****Show a list of cadets and the names of cadets who froze them in battle.****
 1. Decide which table to select FROM, and which table to LEFT JOIN.
`cadet`, `cadet_battle`, and `cadet`
We will need aliases to write this query
 2. Make sure you place the table with potential NULL columns on the Right.
If there is a NULL in `frozen_by_id` there will be no match in `cadet` when we join, we must make sure this table uses LEFT OUTER JOIN.
 3. Remember to use the ON or USING keywords to specify the conditions for the LEFT JOIN.

Live Demo

cadet
cadet_id
name
army_id
...

army
army_id
name
commander
...

cadet_battle
cadet_battle_id
cadet_id
battle_id
frozen_by_id
...

award
award_id
cadet_id
battle_id
...

battle
battle_id
fought
army1_id
army2_id
...

SQL: Outer Join



Wednesday, July 30, 14

14

2. ****Show a list of cadets and the names of cadets who froze them in battle.****
 1. Decide which table to select FROM, and which table to LEFT JOIN.
`cadet`, `cadet_battle`, and `cadet`
We will need aliases to write this query
 2. Make sure you place the table with potential NULL columns on the Right.
If there is a NULL in `frozen_by_id` there will be no match in `cadet` when we join, we must make sure this table uses LEFT OUTER JOIN.
 3. Remember to use the ON or USING keywords to specify the conditions for the LEFT JOIN.

Live Demo

cadet
cadet_id
name
army_id
...

army
army_id
name
commander
...

cadet_battle
cadet_battle_id
cadet_id
battle_id
frozen_by_id
...

award
award_id
cadet_id
battle_id
...

battle
battle_id
fought
army1_id
army2_id
...

SQL: Outer Join



2. ****Show a list of cadets and the names of cadets who froze them in battle.****
 1. Decide which table to select FROM, and which table to LEFT JOIN.
`cadet`, `cadet_battle`, and `cadet`
We will need aliases to write this query
 2. Make sure you place the table with potential NULL columns on the Right.
If there is a NULL in `frozen_by_id` there will be no match in `cadet` when we join, we must make sure this table uses LEFT OUTER JOIN.
 3. Remember to use the ON or USING keywords to specify the conditions for the LEFT JOIN.

Live Demo

cadet
cadet_id
name
army_id
...

Inner
Join
0..∞

cadet_battle
cadet_battle_id
cadet_id
battle_id
frozen_by_id
...

battle
battle_id
fought
army1_id
army2_id
...

army
army_id
name
commander
...

award
award_id
cadet_id
battle_id
...

SQL: Outer Join



2. ****Show a list of cadets and the names of cadets who froze them in battle.****
 1. Decide which table to select FROM, and which table to LEFT JOIN.
`cadet`, `cadet_battle`, and `cadet`
We will need aliases to write this query
 2. Make sure you place the table with potential NULL columns on the Right.
If there is a NULL in `frozen_by_id` there will be no match in `cadet` when we join, we must make sure this table uses LEFT OUTER JOIN.
 3. Remember to use the ON or USING keywords to specify the conditions for the LEFT JOIN.

Live Demo

cadet
cadet_id
name
army_id
...

Inner
Join
0..∞

0..1
Outer
Join

cadet_battle
cadet_battle_id
cadet_id
battle_id
frozen_by_id
...

battle
battle_id
fought
army1_id
army2_id
...

army
army_id
name
commander
...

award
award_id
cadet_id
battle_id
...

SQL: Outer Join



Wednesday, July 30, 14

14

2. ****Show a list of cadets and the names of cadets who froze them in battle.****
 1. Decide which table to select FROM, and which table to LEFT JOIN.
`cadet`, `cadet_battle`, and `cadet`
We will need aliases to write this query
 2. Make sure you place the table with potential NULL columns on the Right.
If there is a NULL in `frozen_by_id` there will be no match in `cadet` when we join, we must make sure this table uses LEFT OUTER JOIN.
 3. Remember to use the ON or USING keywords to specify the conditions for the LEFT JOIN.

Table of Contents

01 What is an Outer Join for?

02 Live Demo

➔ **03 Syntax Review**

04 Code Challenge

SQL: Outer Join



Syntax Review

```
SELECT * FROM {table1} LEFT OUTER JOIN {table2} ON {condition} WHERE ...;
```

SQL: Outer Join



Syntax Review

```
SELECT * FROM {table1} LEFT OUTER JOIN {table2} ON {condition} WHERE ...;
```

- Join these tables based on the condition given after **ON**.

SQL: Outer Join



Syntax Review

```
SELECT * FROM {table1} LEFT OUTER JOIN {table2} ON {condition} WHERE ...;
```

- Join these tables based on the condition given after **ON**.
- A Left Join must use the **ON** or **USING** syntax.

SQL: Outer Join



Syntax Review

```
SELECT * FROM {table1} LEFT OUTER JOIN {table2} ON {condition} WHERE ...;
```

- Join these tables based on the condition given after **ON**.
- A Left Join must use the **ON** or **USING** syntax.
- The “Left” table includes rows that may not exist in the “Right” table.

SQL: Outer Join



Syntax Review

```
SELECT * FROM {table1} LEFT OUTER JOIN {table2} ON {condition} WHERE ...;
```

- Join these tables based on the condition given after **ON**.
- A Left Join must use the **ON** or **USING** syntax.
- The “Left” table includes rows that may not exist in the “Right” table.
- Order of tables matters.

SQL: Outer Join



Syntax Review

```
SELECT * FROM {table1} LEFT OUTER JOIN {table2} ON {condition} WHERE ...;
```

SQL: Outer Join



Syntax Review

```
SELECT * FROM {table1} LEFT OUTER JOIN {table2} ON {condition} WHERE ...;
```

- **LEFT JOIN** and **LEFT OUTER JOIN** are synonymous in SQLite.

SQL: Outer Join



Syntax Review

```
SELECT * FROM {table1} LEFT OUTER JOIN {table2} ON {condition} WHERE ...;
```

- **LEFT JOIN** and **LEFT OUTER JOIN** are synonymous in SQLite.
- The most common type of Outer Join is a Left Outer Join.

SQL: Outer Join



Syntax Review

```
SELECT * FROM {table1} LEFT OUTER JOIN {table2} ON {condition} WHERE ...;
```

SQL: Outer Join



Syntax Review

```
SELECT * FROM {table1} LEFT OUTER JOIN {table2} ON {condition} WHERE ...;
```

- **Outer Join** - all rows that match A, and any that correspond in B.

SQL: Outer Join



Syntax Review

```
SELECT * FROM {table1} LEFT OUTER JOIN {table2} ON {condition} WHERE ...;
```

- **Outer Join** - all rows that match A, and any that correspond in B.
- **Inner Join** - only rows that match A and B.

SQL: Outer Join



Syntax Review

```
SELECT * FROM {table1} LEFT OUTER JOIN {table2} ON {condition} WHERE ...;
```

- **Outer Join** - all rows that match A, and any that correspond in B.
- **Inner Join** - only rows that match A and B.
- If there are no NULL keys in A, the same records are returned from INNER and OUTER join.

SQL: Outer Join



Syntax Review

SQL: Outer Join



Syntax Review

1. Which tables to select **FROM**?

SQL: Outer Join



Syntax Review

1. Which tables to select **FROM**?
2. Which to **LEFT JOIN**? Table with possible **NULL** rows on the Right.

SQL: Outer Join



Syntax Review

1. Which tables to select **FROM**?
2. Which to **LEFT JOIN**? Table with possible **NULL** rows on the Right.
3. Instead of WHERE, use **ON** or **USING**.

SQL: Outer Join



Syntax Review

1. Which tables to select **FROM**?
2. Which to **LEFT JOIN**? Table with possible **NULL** rows on the Right.
3. Instead of WHERE, use **ON** or **USING**.
4. Specify the **WHERE** clause.

SQL: Outer Join



Syntax Review

1. Which tables to select **FROM**?
2. Which to **LEFT JOIN**? Table with possible **NULL** rows on the Right.
3. Instead of WHERE, use **ON** or **USING**.
4. Specify the **WHERE** clause.
5. Decide which fields to **SELECT**.

SQL: Outer Join



Syntax Review

1. Which tables to select **FROM**?
2. Which to **LEFT JOIN**? Table with possible **NULL** rows on the Right.
3. Instead of WHERE, use **ON** or **USING**.
4. Specify the **WHERE** clause.
5. Decide which fields to **SELECT**.
6. Decide which fields to **ORDER BY**.

SQL: Outer Join



Syntax Review

1. Which tables to select **FROM**?
2. Which to **LEFT JOIN**? Table with possible **NULL** rows on the Right.
3. Instead of WHERE, use **ON** or **USING**.
4. Specify the **WHERE** clause.
5. Decide which fields to **SELECT**.
6. Decide which fields to **ORDER BY**.
7. Add a **LIMIT**.

SQL: Outer Join



Table of Contents

01 What is an Outer Join for?

02 Live Demo

03 Syntax Review

➔ **04 Code Challenge**

SQL: Outer Join



Code Challenge

SQL: Outer Join



Wednesday, July 30, 14

21

Edit ``query3.sql`` and ``query4.sql`` files, and write a query to find the following:

1. Show a list of cadets and their awards, and show the name and date of the battle it was awarded for, if any.
2. Show a list of cadets who fought in battles from September 4th to 11th, 2032 and the cadet who froze them, if any.

Open the `join2.db` and load ``query3.sql`` then review the results.

```
$ sqlite3 join2.db
```

Code Challenge

1. Show cadets and their awards, and show the name and date of the battle it was awarded for, if any.

SQL: Outer Join



Wednesday, July 30, 14

21

Edit ``query3.sql`` and ``query4.sql`` files, and write a query to find the following:

1. Show a list of cadets and their awards, and show the name and date of the battle it was awarded for, if any.
2. Show a list of cadets who fought in battles from September 4th to 11th, 2032 and the cadet who froze them, if any.

Open the `join2.db` and load ``query3.sql`` then review the results.

```
$ sqlite3 join2.db
```

Code Challenge

1. Show cadets and their awards, and show the name and date of the battle it was awarded for, if any.

query3.sql

SQL: Outer Join



Wednesday, July 30, 14

21

Edit `query3.sql` and `query4.sql` files, and write a query to find the following:

1. Show a list of cadets and their awards, and show the name and date of the battle it was awarded for, if any.
2. Show a list of cadets who fought in battles from September 4th to 11th, 2032 and the cadet who froze them, if any.

Open the join2.db and load `query3.sql` then review the results.

\$ sqlite3 join2.db

Code Challenge

1. Show cadets and their awards, and show the name and date of the battle it was awarded for, if any.

`query3.sql`

2. Show cadets who fought in battles from September 4th to 11th, 2032 and the cadet who froze them, if any.

SQL: Outer Join



Wednesday, July 30, 14

21

Edit ``query3.sql`` and ``query4.sql`` files, and write a query to find the following:

1. Show a list of cadets and their awards, and show the name and date of the battle it was awarded for, if any.
2. Show a list of cadets who fought in battles from September 4th to 11th, 2032 and the cadet who froze them, if any.

Open the `join2.db` and load ``query3.sql`` then review the results.

```
$ sqlite3 join2.db
```

Code Challenge

1. Show cadets and their awards, and show the name and date of the battle it was awarded for, if any.

`query3.sql`

2. Show cadets who fought in battles from September 4th to 11th, 2032 and the cadet who froze them, if any.

`query4.sql`

SQL: Outer Join



Wednesday, July 30, 14

21

Edit ``query3.sql`` and ``query4.sql`` files, and write a query to find the following:

1. Show a list of cadets and their awards, and show the name and date of the battle it was awarded for, if any.
2. Show a list of cadets who fought in battles from September 4th to 11th, 2032 and the cadet who froze them, if any.

Open the `join2.db` and load ``query3.sql`` then review the results.

`$ sqlite3 join2.db`

Tables

cadet
cadet_id
name
army_id
...

army
army_id
name
commander
...

cadet_battle
cadet_battle_id
cadet_id
battle_id
frozen_by_id
...

award
award_id
cadet_id
battle_id
...

battle
battle_id
fought
army1_id
army2_id
...

SQL: Outer Join



Here is a visual diagram of the database tables and some of their fields, seen in `join2.db`.

For a fully detailed version, see `cadet.sql` in the `create` folder inside this example.

SQL: Outer Join

by Ryan Price



RyanPriceMedia.com



Wednesday, July 30, 14

23

##Coming Up

In our next tutorial, we will cover the concept of GROUP BY and functions like COUNT() MIN() SUM() and AVG()