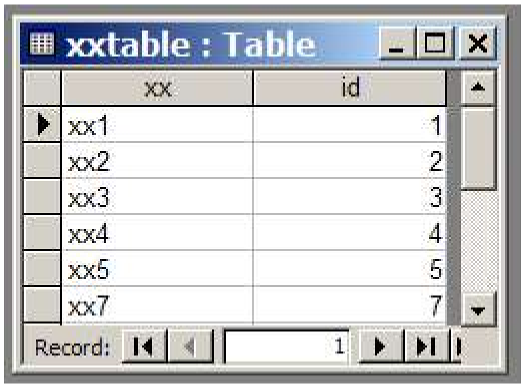
JOINS in Action

Introduction

The purpose of this handout is to give you mini-example of the different types of joins we covered in class using two tables. We will illustrate cross join, inner join, left outer join, right outer join, and the full outer join. For this example, I used MS Access (the screen shots look much nicer than MySQL). But the same SQL code can be executed in MySQL to achieve the same results.

The Tables

In this example, let's assume we have two tables. Table 1 has two fields called "xx", a text field, and "id", the primary key for the table.



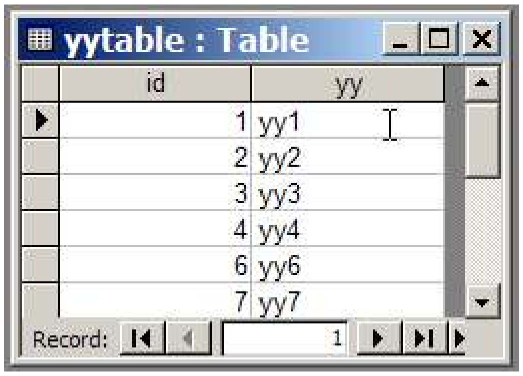
xxtable : Table

xx	id
xx1	1
xx2	2
xx3	3
xx4	4
xx5	5
xx7	7

Record: 1

Note that Table 1 is missing id=6.

Table 2 has two fields called "yy", a text field, and "id", the primary key for the table.



yytable : Table

id	yy
1	yy1
2	yy2
3	yy3
4	yy4
6	yy6
7	yy7

Record: 1

Note that Table 2 is missing id=5. We will now link the two tables using the different joins we covered in class.

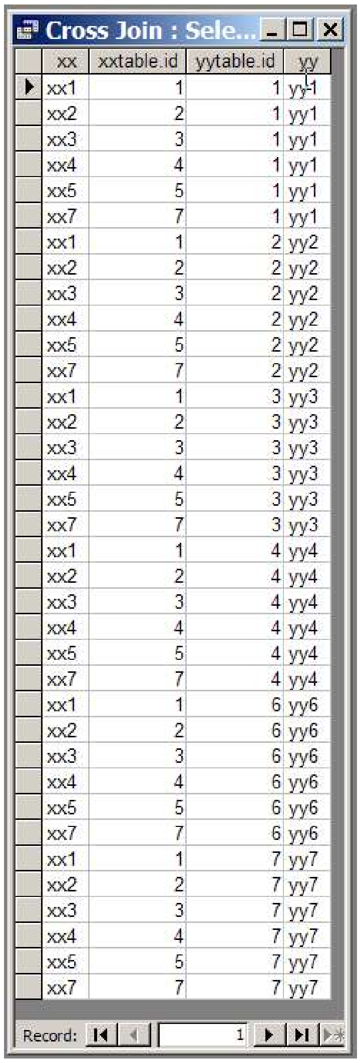
CROSS JOIN

We can combine each row from table 1 with every row from table 2 using a cross join. In order to do this, we do not need to specify a WHERE clause or an ON clause since we will not combine the two tables based on a particular field (like id). We know that we want to take each row from table 1 and combine it with every row in table 2 and give us the result.

In order to achieve this, we would type the following SQL query (I want to display both id fields from the two tables so that you can see which rows from table 1 are being paired with which rows from table 2.:

```
SELECT xx, xxtable.id, yytable.id, yy
FROM xxtable, yytable;
```

This SQL query will give us the following result:


Cross Join : Sele...

xx	xxtable.id	yytable.id	yy
xx1	1	1	yy1
xx2	2	1	yy1
xx3	3	1	yy1
xx4	4	1	yy1
xx5	5	1	yy1
xx7	7	1	yy1
xx1	1	2	yy2
xx2	2	2	yy2
xx3	3	2	yy2
xx4	4	2	yy2
xx5	5	2	yy2
xx7	7	2	yy2
xx1	1	3	yy3
xx2	2	3	yy3
xx3	3	3	yy3
xx4	4	3	yy3
xx5	5	3	yy3
xx7	7	3	yy3
xx1	1	4	yy4
xx2	2	4	yy4
xx3	3	4	yy4
xx4	4	4	yy4
xx5	5	4	yy4
xx7	7	4	yy4
xx1	1	6	yy6
xx2	2	6	yy6
xx3	3	6	yy6
xx4	4	6	yy6
xx5	5	6	yy6
xx7	7	6	yy6
xx1	1	7	yy7
xx2	2	7	yy7
xx3	3	7	yy7
xx4	4	7	yy7
xx5	5	7	yy7
xx7	7	7	yy7

Record: 1

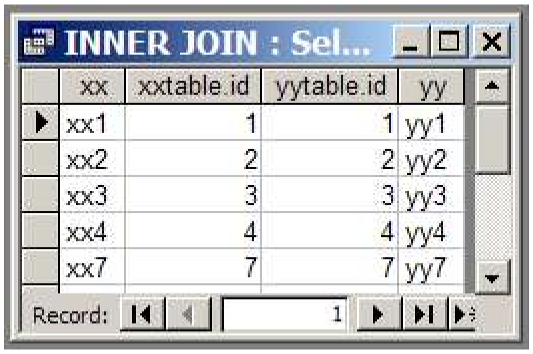
Note that every row from table 2 is matched with every row from table 1. However, now we may want to join the two tables based on rows that match on the two id fields rather than creating all possible pairs between the rows of the two tables.

INNER JOIN

If we wanted to pull only the records where the id fields were equal to one another (and drop the row where id=5 from table 1 and drop the row where id=6 from table 2), we would issue the following SQL query:

```
SELECT xx, xxtable.id, yytable.id, yy
FROM xxtable
INNER JOIN yytable
    ON xxtable.id=yytable.id;
```

The SQL query would give the following result:


INNER JOIN : Sel...

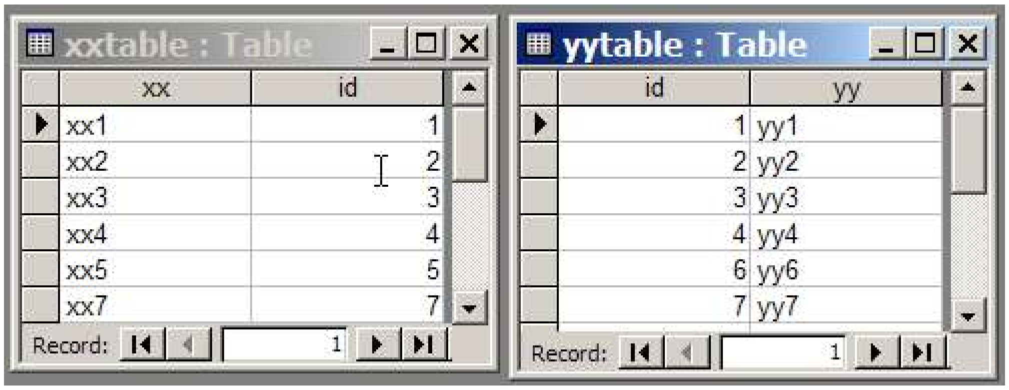
	xx	xxtable.id	yytable.id	yy
▶	xx1	1	1	yy1
	xx2	2	2	yy2
	xx3	3	3	yy3
	xx4	4	4	yy4
	xx7	7	7	yy7

Record: 1

This result only contains the records where id of table 1 equals id of table2. The result of this INNER JOIN excludes the record from table 1 where xxtable.id=5 and excludes the record from table 2 where yytable.id=6 since they do not match any records in the other table.

LEFT OUTER JOIN

In order to better illustrate the outer joins, let's take our original tables:


xxtable : Table

	xx	id
▶	xx1	1
	xx2	2
	xx3	3
	xx4	4
	xx5	5
	xx7	7

Record: 1

yytable : Table

	id	yy
▶	1	yy1
	2	yy2
	3	yy3
	4	yy4
	6	yy6
	7	yy7

Record: 1

and align the two tables according to values in the id fields like this:

xx	*xxtable.id*	*yytable.id*	*yy*
xx1	1	1	yy1
xx2	2	2	yy2
xx3	3	3	yy3
xx4	4	4	yy4
xx5	5	NULL	

xx	*xxtable.id*	*yytable.id*	*yy*
NULL		6	yy6
xx7	7	7	yy7

If we wanted all of the rows in the table 1 (xx table, the "left" table) in our result, regardless of whether they matched with rows in table 2 (the yy table, the "right" table), we could perform a LEFT OUTER JOIN between tables xx and yy.

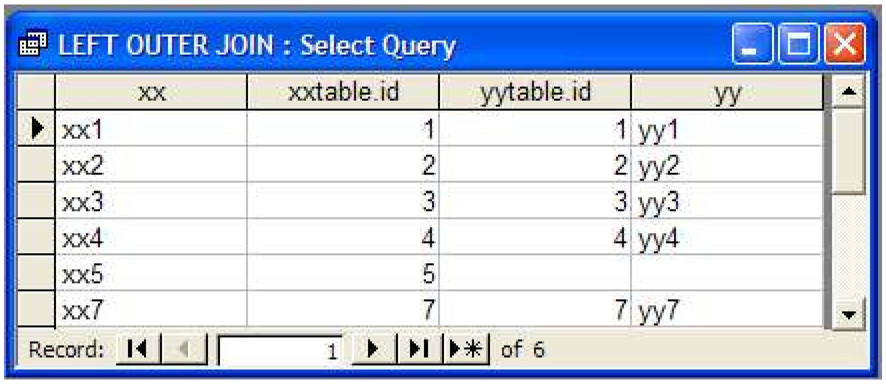
The following SQL syntax:

```
SELECT xx, xxtable.id, yytable.id, yy
FROM xxtable

LEFT OUTER JOIN yytable

    ON xxtable.id=yytable.id;
```

would give the following result:

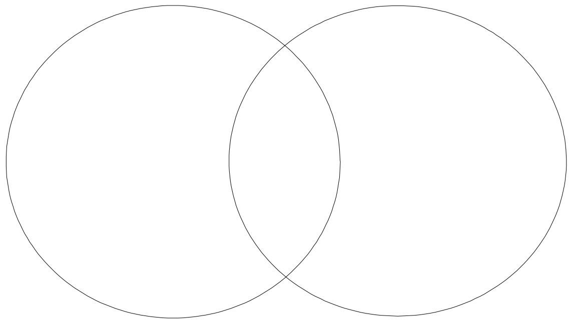

LEFT OUTER JOIN : Select Query

xx	xxtable.id	yytable.id	yy
xx1	1	1	yy1
xx2	2	2	yy2
xx3	3	3	yy3
xx4	4	4	yy4
xx5	5		
xx7	7	7	yy7

Record: 1 of 6

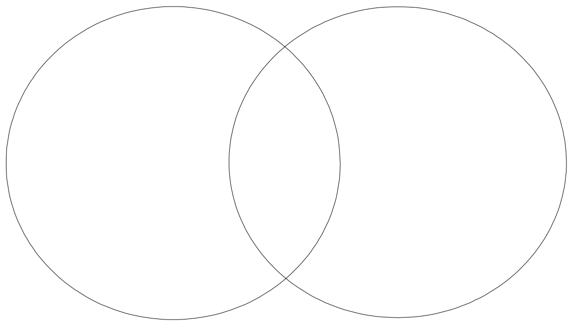
Since the right table, yytable, does not have a record with yytable.id=5, there is a NULL value for the fields yytable.id and yy where xxtable.id=5. Note that all of the records of the left table in a LEFT OUTER JOIN get displayed, regardless of whether or not they match rows in the right table.

You can think of the join between two tables as a Venn diagram.



You can think of the left circle as all of the records that are in the left table (the xx table) and right circle as all of the records that are in the right table (the yy table). The middle section where the two circles overlap represents all of the records that match; that is, that are in both the left table and the right table.

So when we did the LEFT OUTER JOIN, we took all rows that were in the left table and displayed them, irrespective of whether they matched rows in the right table.



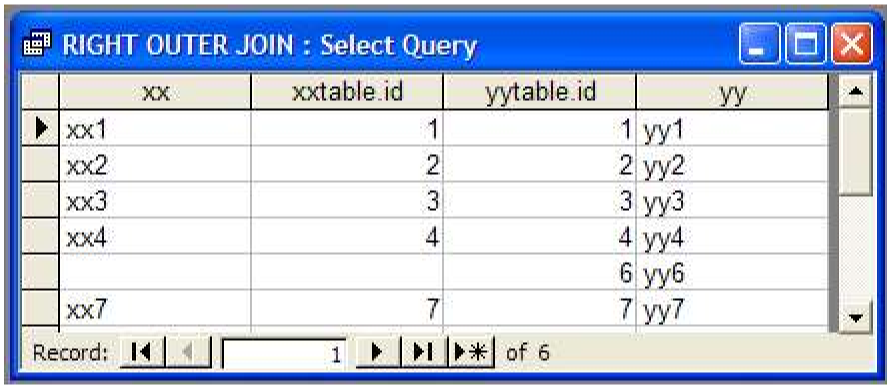
RIGHT OUTER JOIN

Now, let's say we wanted all of the rows in the table 2 (yy table, the "right" table) in our result, regardless of whether they matched with rows in table 1 (the xx table, the "left" table), we could perform a RIGHT OUTER JOIN between tables xx and yy.

The following SQL syntax:

```
SELECT xx, xxtable.id, yytable.id, yy
FROM xxtable
RIGHT OUTER JOIN yytable
    ON xxtable.id=yytable.id;
```

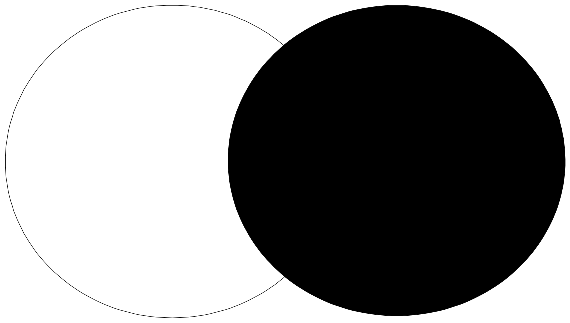
would give the following result:


RIGHT OUTER JOIN : Select Query

xx	xxtable.id	yytable.id	yy
xx1	1	1	yy1
xx2	2	2	yy2
xx3	3	3	yy3
xx4	4	4	yy4
		6	yy6
xx7	7	7	yy7

In this example, there is a NULL value for the fields xxtable.id and xx where yytable.id=6 since xxtable does not have a record with xxtable.id=6. Note that all of the records of the right table in a RIGHT OUTER JOIN get displayed, regardless of whether or not they match rows in the left table.

Using the Venn diagram, we now grabbed all rows of the right table, irrespective of whether they matched rows of the left table.



FULL OUTER JOIN

Now let's say we want all records from both tables, regardless of whether they are records that match both tables. In order to do this, we would perform a FULL OUTER JOIN. If the relational database management system that you are using supports FULL OUTER JOINS, then the following SQL syntax:

```
SELECT xx, xxtable.id, yytable.id, yy
FROM xxtable

FULL OUTER JOIN yytable

    ON xxtable.id=yytable.id;
```

would produce the results you want. However, if your relational database management system does not support FULL OUTER JOINS (like MS Access and MySQL), then you need to revise your SQL statement by doing two queries and take the UNION of them.

We've already seen what the two SQL queries, the LEFT OUTER JOIN:

```
SELECT xx, xxtable.id, yytable.id, yy
FROM xxtable
LEFT OUTER JOIN yytable
    ON xxtable.id=yytable.id;
```

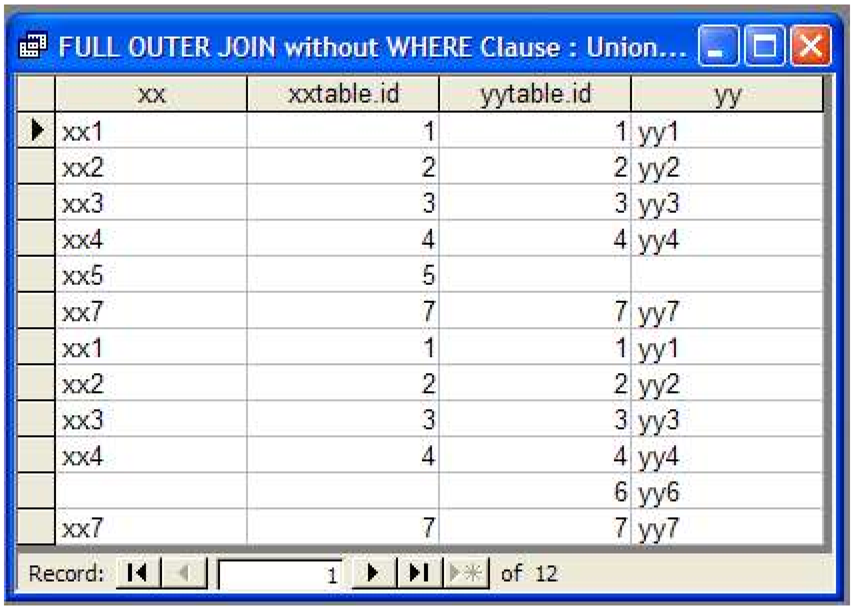
and the RIGHT OUTER JOIN:

```
SELECT xx, xxtable.id, yytable.id, yy
FROM xxtable
RIGHT OUTER JOIN yytable
    ON xxtable.id=yytable.id;
```

do. So if we were to take the UNION ALL of the two:

```
SELECT xx, xxtable.id, yytable.id, yy
FROM xxtable
LEFT OUTER JOIN yytable
    ON xxtable.id=yytable.id
UNION ALL
SELECT xx, xxtable.id, yytable.id, yy
FROM xxtable
RIGHT OUTER JOIN yytable
    ON xxtable.id=yytable.id;
```

we would get the following result:



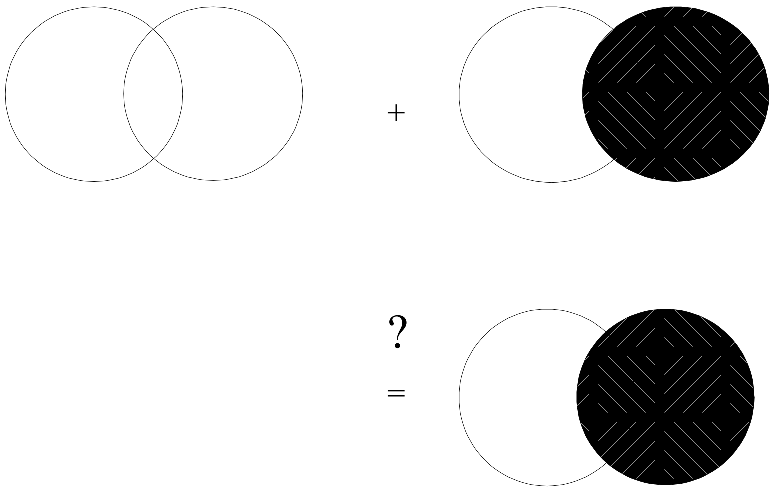
FULL OUTER JOIN without WHERE Clause : Union...

xx	xxtable.id	yytable.id	yy
xx1	1	1	yy1
xx2	2	2	yy2
xx3	3	3	yy3
xx4	4	4	yy4
xx5	5		
xx7	7	7	yy7
xx1	1	1	yy1
xx2	2	2	yy2
xx3	3	3	yy3
xx4	4	4	yy4
		6	yy6
xx7	7	7	yy7

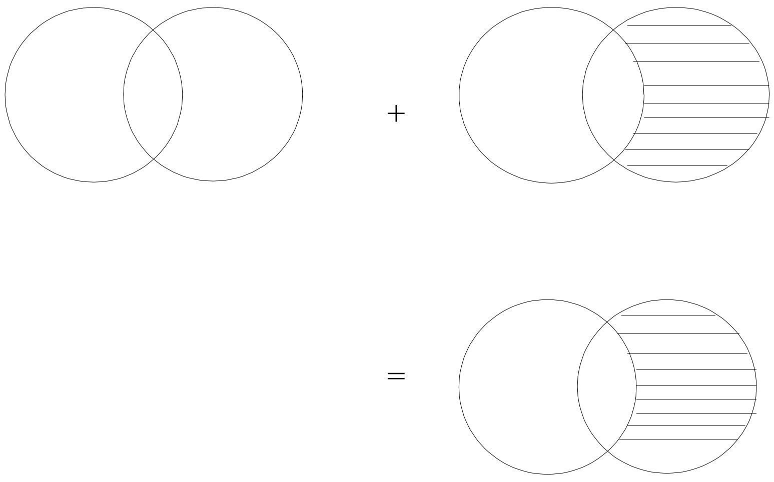
Record: 1 of 12

Note that we have duplicate records with xxtable.id= 1, 2, 3, and 4 in our result. Why is this?

In this query, we've combined the results of two queries by using the UNION clause. Using our Venn diagram, we added the result of the two queries. As performed, is this really what we want? Does the union of the two queries as written really give us what we want?



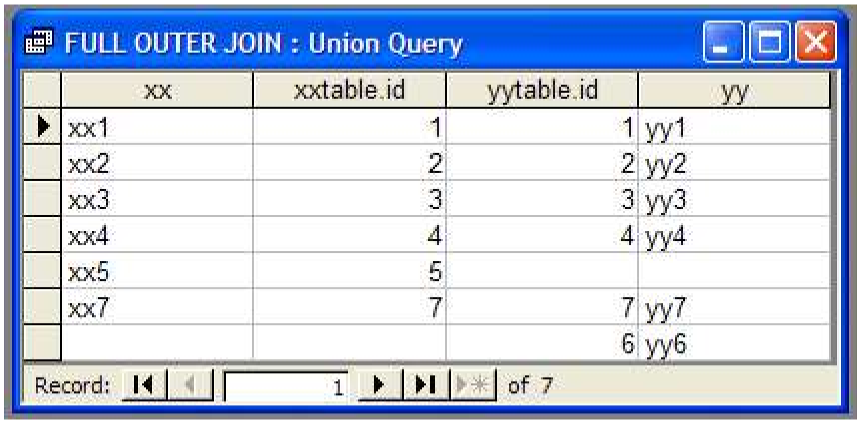
No! Actually, we added the intersecting piece twice (hence, the duplication of the records that match). Therefore, we want this:



In words, we want all of the result from the LEFT OUTER JOIN and combine that with only the records that did not match from the RIGHT OUTER JOIN (ie, where xx IS NULL). So we can revise our query to achieve the correct union:

```
SELECT xx, xxtable.id, yytable.id, yy
FROM xxtable
LEFT JOIN yytable
    ON xxtable.id=yytable.id
UNION ALL
SELECT xx, xxtable.id, yytable.id, yy
FROM xxtable
RIGHT JOIN yytable
    ON xxtable.id=yytable.id
WHERE xx IS NULL;
```

This revised query gives the following result:


FULL OUTER JOIN : Union Query

xx	xxtable.id	yytable.id	yy
xx1	1	1	yy1
xx2	2	2	yy2
xx3	3	3	yy3
xx4	4	4	yy4
xx5	5		
xx7	7	7	yy7
		6	yy6

Record: 1 of 7

Alternatively, a UNION statement without the "WHERE xx IS NULL" would also work and give you what you want.

```
SELECT xx, xxtable.id, yytable.id, yy
FROM xxtable
LEFT JOIN yytable
    ON xxtable.id=yytable.id
```

```
UNION
SELECT xx, xxtable.id, yytable.id, yy
FROM xxtable
RIGHT JOIN yytable
    ON xxtable.id=yytable.id;
```