

## Lecture 8

- Inner join
- Left/Right join
- Outer join and UNION
- ORDER BY

## Grabbing data from different tables

- Remember this?

```
SELECT GivenName, Surname, Subject
FROM User, Message
WHERE idUser = idOwner;
```

- This is also called an *inner join*.
- The result set contains only data from records where the **idUser** in the **User** table has a corresponding **idOwner** in the **Message** table. Other users are not shown.

## Left join

- What if we want to see a recordset with all users and their messages, and still want to see users who have posted no messages?
- In this case, we need a *left (outer) join*

```
SELECT GivenName, Surname, Subject
FROM User
LEFT JOIN Message
ON idUser = idOwner;
```

## Right join

- A *right (outer) join* is in principle just a left join with the order of the tables switched.
- The following right join returns exactly the same data set as our left join:

```
SELECT GivenName, Surname, Subject
FROM Message
RIGHT JOIN User
ON idUser = idOwner;
```

## Outer join

- A (full) outer join also retrieves records from both tables in a join that do not have a counterpart in the respective other table
- The official syntax of a full outer join is:

```
SELECT <field1>, <field2>, ...
  FROM <table1>
 FULL OUTER JOIN <table2>
   ON <table1>.<table1_idfield> =
    <table2>.<table2_idfield>;
```

- However, this is not available in some RDMBS, including MySQL

## Outer join example

AnimalTable	
Animal ID	Animal
1	Elephant
3	Snake
4	Tiger
6	Snake
7	Emu

KeeperTable	
Animal ID	Keeper
1	Fred
1	Zoe
4	Jenny
6	Steph
9	Martin

```
SELECT Animal, Keeper
  FROM AnimalTable
 FULL OUTER JOIN KeeperTable
   ON AnimalTable.`Animal ID` =
    KeeperTable.`Animal ID`;
```

## Outer join example – the result

Animal	Keeper
Elephant	Fred
Elephant	Zoe
Snake	NULL
Tiger	Jenny
Snake	Steph
Emu	NULL
NULL	Martin

## Outer join workaround

- A way of doing a full outer join in MySQL is to combine a left and a right join with a UNION:

```
SELECT Animal, Keeper
  FROM AnimalTable
 LEFT JOIN KeeperTable
   ON AnimalTable.`Animal ID` =
    KeeperTable.`Animal ID`
 UNION
 SELECT Animal, Keeper
  FROM AnimalTable
 RIGHT JOIN KeeperTable
   ON AnimalTable.`Animal ID` =
    KeeperTable.`Animal ID`;
```

## What does a UNION do?

- In MySQL (and various other SQL dialects), a UNION combines the result sets from two SELECT queries
- By default, records that are found in both result sets are included only once (DISTINCT)
- Generally, there is also a UNION ALL option to include all duplicate records

## UNION example

- The following queries are the same:

```
SELECT * FROM Table1
    WHERE Field1 = 'value1'
    OR Field2 = 'value2';
```

and

```
SELECT * FROM Table1
    WHERE Field1 = 'value1'
UNION
SELECT * FROM Table1
    WHERE Field2 = 'value2';
```

## ORDER BY

- Sometimes, we would like to have our result set sorted:

```
SELECT GivenName, Surname,
    DateOfBirth
FROM User
ORDER BY DateOfBirth ASC;
```

- This sorts our users from oldest to youngest (ascending)
- To sort from youngest to oldest (descending), use **DESC** instead of **ASC**

## ORDER BY ... more than one field

- Consider this query:

```
SELECT GivenName, Surname,
    City
FROM User ORDER BY City ASC;
```

- This sorts our users by the name of the city they live in – but within each city, the names are not in alphabetical order
- Sorting by more than one field fixes this (note **ASC** is the default):

```
SELECT GivenName, Surname,
    City
FROM User
ORDER BY City, Surname, GivenName;
```

## Today's lab sheet

- ...is on the web.
- Today: inner, left, right and outer joins, unions, order by, and – as an extra – a sneak preview of the GROUP BY clause and of aggregate functions