

Course outline first half – week 2

- Lecture 4 (8 March): more complex where clauses involving multiple fields, table aliases, select distinct
- Lecture 5 (9 March): working with multiple tables, master-detail relationships (one-to-many), one-to-one, implementing many-to-many relationships
- Lecture 6 (11 March): mapping tables and relationships in UML

Lecture 4

- More complex where clauses involving multiple fields
- Table aliasing
- SELECT DISTINCT

Warm-up exercise: Bestsellers!

- This assumes the stock table with the extra `Items sold` field:

```
select description, price
  from stock
 where `Items sold` > `Items in stock`;
```

- This gives us all goods of which we have sold more than we have in stock (so they must be selling well!)
- NB: Exercise: To make this example truly work, some records need non-zero values in this field. How can you put these in?

Bundling our stock for sale

- Task: get a dataset such that each record shows a combination of two different items from our **stock** table
- Problem: How can we select the same field in the table twice?
- Answer: by using a *table alias*

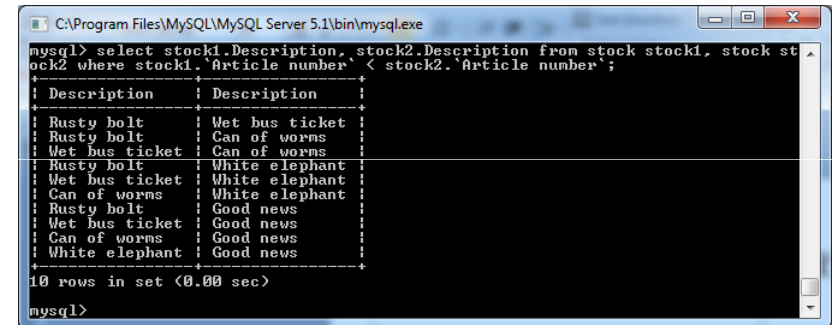
Table aliases

- The following query does the trick:

```
SELECT
    stock1.`Description`,
    stock2.`Description`
FROM
    stock AS stock1,
    stock AS stock2
WHERE
    stock1.`Article number` <
    stock2.`Article number`;
```

- stock1 and stock2 are table aliases

Table aliases



```
C:\Program Files\MySQL\MySQL Server 5.1\bin\mysql.exe
mysql> select stock1.Description, stock2.Description from stock stock1, stock stock2
where stock1.`Article number` < stock2.`Article number`;
+-----+-----+
| Description | Description |
+-----+-----+
| Rusty bolt  | Wet bus ticket |
| Rusty bolt  | Can of worms  |
| Wet bus ticket | Can of worms  |
| Rusty bolt  | White elephant |
| Wet bus ticket | White elephant |
| Can of worms | White elephant |
| Rusty bolt  | Good news     |
| Wet bus ticket | Good news     |
| Can of worms | Good news     |
| White elephant | Good news     |
+-----+-----+
10 rows in set (0.00 sec)

mysql>
```

Points to ponder

- What is the where clause for? What happens when we leave it away?
- The "AS" is optional and can be left out:

```
SELECT
    stock1.`Description`,
    stock2.`Description`
FROM
    stock stock1, stock stock2
WHERE
    stock1.`Article number` <
    stock2.`Article number`;
```

What have people bought?

- Consider the **purchases** table, which is defined as:

```
CREATE TABLE purchases (
    purchaseId int unsigned auto_increment,
    customerId int unsigned,
    `Article number` varchar(8),
    PRIMARY KEY (purchaseId)
);
```

On the **purchases** table

- The purchases table is of a kind that we will meet a lot. It establishes a *relation* between customers and the article numbers of the items that the customers bought.
- Note that we don't store any details except a customerId. This serves as a *key* into another table with customer data, but we'll ignore that for now.
- The purchaseId is the primary key on the table

Which items did customers buy?

- If we want a list of the items that we have sold, we could of course do this:

```
SELECT `Article number`  
FROM stock  
WHERE `Items sold` > 0;
```

Which items did customers buy?

- Alternatively, we can also query the **purchases** table:

```
SELECT `Article number`  
FROM purchases;
```

- There's just one problem here: We get vastly more records if customers bought several items of the same article

Which items did customers buy?

- We can fix the problem of multiple identical records in a dataset by using SELECT DISTINCT:

```
SELECT DISTINCT `Article number`  
FROM purchases;
```

- NB: Exercise: To make this example truly work, you will need to add some records to the **purchases** table!

Today's lab sheet

- ...is on the web
- Today: table aliasing and SELECT DISTINCT