

Lecture 3

- Deleting records from a table
- Updating records in a table
- Changing table definitions
- Deleting tables
- Design considerations (data types)

Deleting records from a table

- This is done with the DELETE statement:

```
DELETE FROM <table_name>;
```

- Note that this deletes ALL RECORDS from the table!



Deleting an individual record or group of records from a table

- This is done by adding a where clause to the delete statement. Example:

```
DELETE FROM stock
WHERE
  `Article number`='344-28299';
```

- Be careful to ensure that the where clause contains the proper data!

Spot the danger!

- This happened to a student of mine once, on a production system he'd updated with new code:

```
delete from participants
  where canDelete='?';
```

- The "?" marks a parameter supplied by his application.
- The parameter was meant to be something other than 0 for those participants that could be deleted from the table.
- Due to a typo in the application, the parameter was left empty. But an empty string is interpreted as a zero. So what was the result?

Updating the contents of a table

- This is done with the UPDATE statement:

```
UPDATE <table_name>
  SET
    <field1_name>=<value1>,
    <field2_name>=<value2>,
    ...;
```

- Again note that this updates **all** records in the table!

Updating individual records in a table

- This is also done with the UPDATE statement:

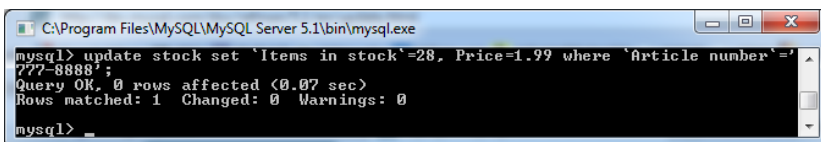
```
UPDATE <table_name>
  SET
    <field1_name>=<value1>,
    <field2_name>=<value2>,
    ...
  WHERE <some_condition>;
```

- The records in the table for which *<some_condition>* is true are updated.

Example: Updating a record

- If we wanted to update the price and number of items in stock:

```
UPDATE stock
  SET
    `Items in stock`=28,
    Price=1.99
  WHERE `Article number`='777-8888';
```



Changing a table definition

- Sometimes, we have to make changes to a table's definition
- We may need an extra field
- The requirements for a field may have changed, e.g., we may need a `bigint` instead of an `int`.
- We may need to add a key
- We may want to remove a field or a key that we don't need
- ... all these and many more use ALTER TABLE statements

Adding new fields to a table

- The general form of the command is this:

```
ALTER TABLE <table_name> ADD (  
    <field_name1> <field_definition1>,  
    <field_name2> <field_definition2>,  
    ...)  
AFTER <existing_field_name>;
```

- This adds the new fields to the existing table after the field *<field_name>*.

- If we only want to add one field, we omit the parentheses:

```
ALTER TABLE <table_name> ADD  
    <field_name> <field_definition>  
AFTER <existing_field_name>;
```

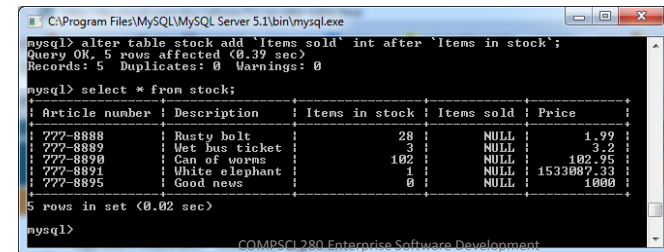
- If the field is to be added as the first field in the table, we use FIRST instead of the AFTER clause.

Adding a new field to a table

- If we wanted to add an additional field to our stock table, this would work:

```
ALTER TABLE stock  
    ADD `Items sold` int  
    AFTER `Items in stock`;
```

- Note that if we only want to add a single field, we must leave the parentheses away



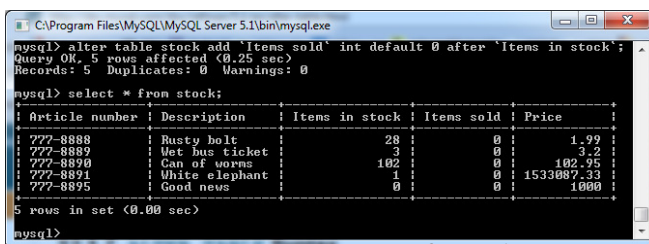
```
mysql> alter table stock add `Items sold` int after `Items in stock`;  
Query OK, 5 rows affected (0.39 sec)  
Records: 5 Duplicates: 0 Warnings: 0  
  
mysql> select * from stock;  
+-----+-----+-----+-----+-----+  
| Article number | Description | Items in stock | Items sold | Price |  
+-----+-----+-----+-----+-----+  
| 777-8888      | Rusty bolt | 28             | NULL       | 1.99  |  
| 777-8889      | Wet bus ticket | 3             | NULL       | 3.2   |  
| 777-8890      | Can of worms | 102            | NULL       | 102.95|  
| 777-8891      | White elephant | 1             | NULL       | 1533087.33|  
| 777-8895      | Good news | 0              | NULL       | 1000  |  
+-----+-----+-----+-----+-----+  
5 rows in set (0.02 sec)  
  
mysql>
```

NULL what?

- Notice how the change of the table put a NULL value into the field by default?

- We can specify a proper default value as part of a field declaration in CREATE TABLE or ALTER TABLE, like so:

```
ALTER TABLE stock  
    ADD `Items sold` int DEFAULT 0  
    AFTER `Items in stock`;
```

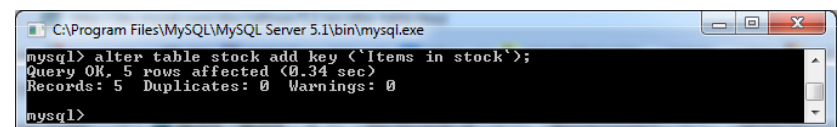


```
mysql> alter table stock add `Items sold` int default 0 after `Items in stock`;  
Query OK, 5 rows affected (0.25 sec)  
Records: 5 Duplicates: 0 Warnings: 0  
  
mysql> select * from stock;  
+-----+-----+-----+-----+-----+  
| Article number | Description | Items in stock | Items sold | Price |  
+-----+-----+-----+-----+-----+  
| 777-8888      | Rusty bolt | 28             | 0          | 1.99  |  
| 777-8889      | Wet bus ticket | 3             | 0          | 3.2   |  
| 777-8890      | Can of worms | 102            | 0          | 102.95|  
| 777-8891      | White elephant | 1             | 0          | 1533087.33|  
| 777-8895      | Good news | 0              | 0          | 1000  |  
+-----+-----+-----+-----+-----+  
5 rows in set (0.00 sec)  
  
mysql>
```

Adding a key to a table

- Let's say we'd like to be able to quickly find all items that are out of stock. Then having the number of items in stock as a key would help:

```
ALTER TABLE stock  
    ADD KEY (`Items in stock`);
```



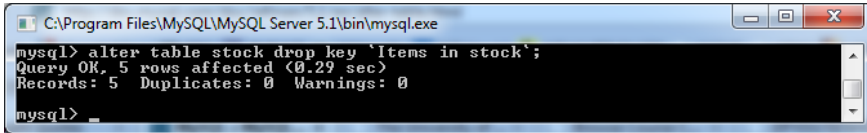
```
mysql> alter table stock add key (`Items in stock`);  
Query OK, 5 rows affected (0.34 sec)  
Records: 5 Duplicates: 0 Warnings: 0  
  
mysql>
```

- Note that we need the parentheses here even if we don't stretch the key across more than one field

Don't need the key after all? Drop it!

- Getting rid of an unnecessary key speeds the RDBMS up a little:

```
ALTER TABLE stock
  DROP KEY `Items in stock`;
```



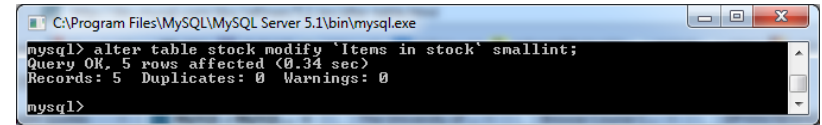
```
C:\Program Files\MySQL\MySQL Server 5.1\bin\mysql.exe
mysql> alter table stock drop key `Items in stock`;
Query OK, 5 rows affected (0.29 sec)
Records: 5 Duplicates: 0 Warnings: 0
mysql>
```

- As logic would have it, you don't need parentheses here (because this is the name of the key, not the field over which it has been defined)

Modifying a field definition in a table

- If we're not a big business, we can probably do with a small integer (smallint) for the items in stock. We can modify the field like this:

```
ALTER TABLE stock
  MODIFY `Items in stock` smallint;
```



```
C:\Program Files\MySQL\MySQL Server 5.1\bin\mysql.exe
mysql> alter table stock modify `Items in stock` smallint;
Query OK, 5 rows affected (0.34 sec)
Records: 5 Duplicates: 0 Warnings: 0
mysql>
```

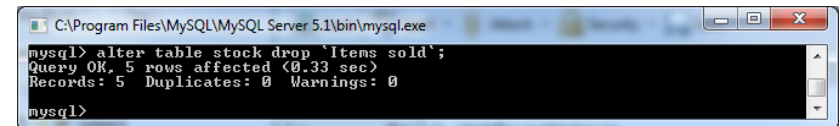
Note on modifying columns

- When modifying column definitions, the RDBMS will generally attempt to convert existing data
- When this is not possible (e.g., conversion from non-numeric values in a text type to a numeric type in the modified field), you may end up with empty field values, or the query may not work (e.g., when trying to convert a string value to a numeric field type that is declared as being NOT NULL).
- If the conversion is too complex, a temporary field with a value computed via a DB procedure may help (see later in the semester)

Deleting a field in a table

- This is also done with ALTER TABLE:

```
ALTER TABLE stock
  DROP `Items sold`;
```



```
C:\Program Files\MySQL\MySQL Server 5.1\bin\mysql.exe
mysql> alter table stock drop `Items sold`;
Query OK, 5 rows affected (0.33 sec)
Records: 5 Duplicates: 0 Warnings: 0
mysql>
```

- Note that this means that all data in the field will be lost!

Deleting a table

- In database speak, this is known as "dropping" a table (you also "drop" a database as you've already dropped keys and fields – easy really!):

```
DROP TABLE <table_name>;
```

- For example, dropping our **stock** table would be done like this:

```
DROP TABLE stock;
```

Deleting a table conditionally

- The following is quite useful in practice if you're having to overwrite tables with a fresh version (e.g., in an SQL script):

```
DROP TABLE IF EXISTS <table_name>;
```

- Typically, you would put this before a CREATE TABLE statement to ensure the old version of the table you want to create has been removed

What can go into a table field?

- Booleans: which can be true (0) or false (1)
- Bit masks
- Numbers: e.g., integers, floats, doubles
- Sets and enumerations
- Date and time values
- Characters and strings: char, varchar (limited size short string), text
- BLOBs (binary large objects): can be used to store binary data such as, e.g., images, sounds, videos etc.
- Exact range of data types supported depends on RDBMS

Which data type should you choose?

- Try to keep storage space small
- Floating point numbers: Don't overdo the precision. If your field is to store only values accurate to, say, 0.01% or less, then it makes no sense to store them as a double – float is more than enough!
- Integers: consider which range you will really need – will your client ever have several billion cars in stock? If it's never going to be smaller than 0, use unsigned
- Text: don't use a varchar for text longer than 255 characters, and don't use the text type if it's always shorter than that
- Autoincrement fields should be integers
- Choose commonly understood data types for portability – your RDBMS is bound to have some that other RDBMS don't understand, and if you have to port...

Today's lab sheet

- ...is on the web
- Today: Using PHPMyAdmin, updating records, backing up your database content with PHPMyAdmin, dropping tables
- Is not assessed, but its content is examinable in test and exam!