POWERING POTENTIAL

MySQL Part 2

Module 19

MYSQL

Contents

DCL: Data Control Language

Section Overview

Data Control Language (DCL)

- SQL Privileges
- Creating users
- Granting privileges
- Revoking privileges

MySQL Privileges

So far we have logged in to MySQL as `root`

- → This account has far more privilege than our web application really needs
- → No damage limitation in the event of a SQL injection attack, for example

Best practice is to create additional accounts that have only the privileges required for the task at hand

- → Some pages of our application require read-only access
- → Some pages require read-write access

MySQL allows assignment of user privileges at four levels

- → Global
- → Database
- → Table
- → Column

User Privileges

User privileges determine whether a user is allowed to run specific SQL commands. Privileges include:

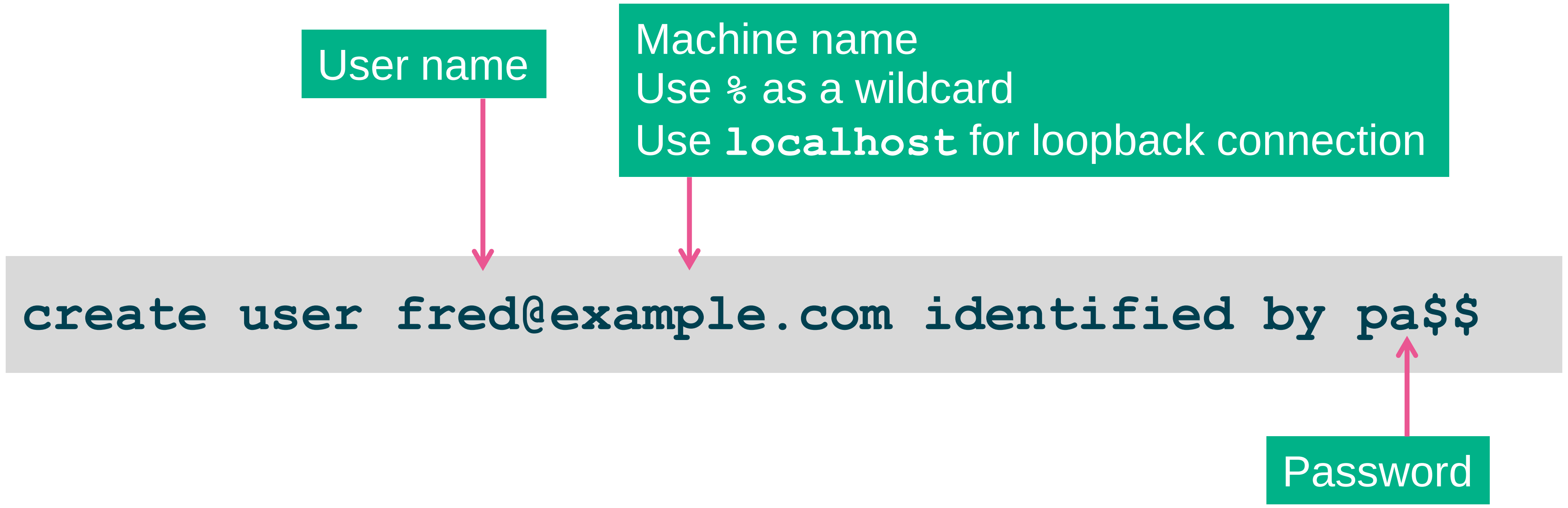
Privilege	Applies to	Allows users to:
SELECT	Tables, columns	Select rows from a table
INSERT	Tables, columns	Insert new rows into tables
UPDATE	Tables, columns	Update values in existing rows
DELETE	Tables	Delete existing table rows
CREATE	Databases, tables	Create new databases or tables
DROP	Databases, tables	Delete databases or tables
USAGE	Global	Do nothing
ALL	Global	Do everything. Usually applied only to the "root" account

The owner / creator of a table automatically has all the privileges

Creating Users

The `create user` command creates user accounts

- User names and passwords need not be related to Linux accounts
- Passwords are hashed before storage in the database



```
create user fred@example.com identified by pa$$
```

Granting Privileges

The `grant` command assigns MySQL privileges

The list of privileges to be assigned
Example: select, insert, update

The database and table to apply the privilege to. '`*`' means all tables in the database

```
grant select on store.* to someUser
identified by 'pa$$'
```

The MySQL account to assign the privilege to

The password to assign to the account (may be omitted if the account already exists)

Revoking Privileges and Deleting Accounts

The REVOKE command takes away privileges

General form:

```
revoke privileges on item from user
```

Example:

```
revoke update, delete on cart.users from badUser
```

The drop user command deletes an account

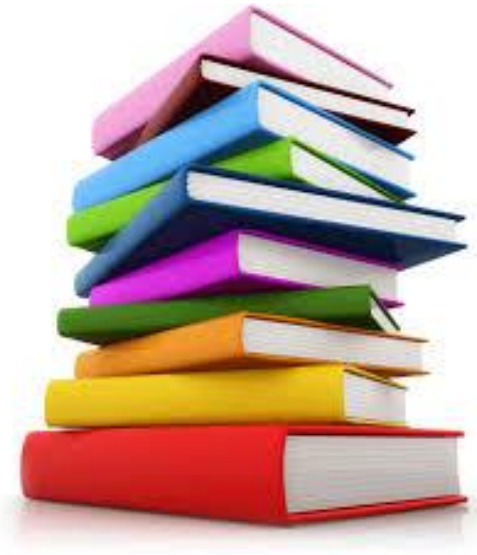
```
drop user badUser
```

Fine grained control

A **Library** application has five pages that access the database. For each one, identify what type of access is required for each table

- select, insert, update, delete (or some combination)

Page	Access to book table	Access to borrower table
Addbook	?	?
Addborrower	?	?
Booksearch	?	?
Checkin	?	?
Checkout	?	?



DML: Data Manipulation Language

Section Overview

Data Manipulation Language (DML)

- Referential Integrity
- Drop table / database
- Selecting data
- Select… from
- Selecting columns
- Column aliases
- Distinct entries
- Filtering using where statements
- And / Or logic

Updating Records

The **update** statement will change values stored in the database.

You can update everything in a single table

Or target it more specifically based on a where clause

```
update [table] set [column] = [value]
      -- will update all the records in the table
      -- with this value!

update [table] set [column] = [value]
where [some condition is true];
      -- will update just the records that are
      -- referenced by this condition

UPDATE users SET username = 'sherlock'
      WHERE username = 'sholmes';
```

Deleting Records

The delete command will delete all rows that match the criteria you specify

If no criteria is specified all rows are deleted

```
delete from [table] where [condition is true];

delete from items;
      -- will delete everything in the table!

delete from items where id = 2;
      -- will only delete the item with id 2
```

Drop Table / Drop Database

The **drop** keyword deletes an entire table or database

The **delete** command removes records

```
drop table [table name];
drop table users;

drop database [databasename];
drop database store;
```

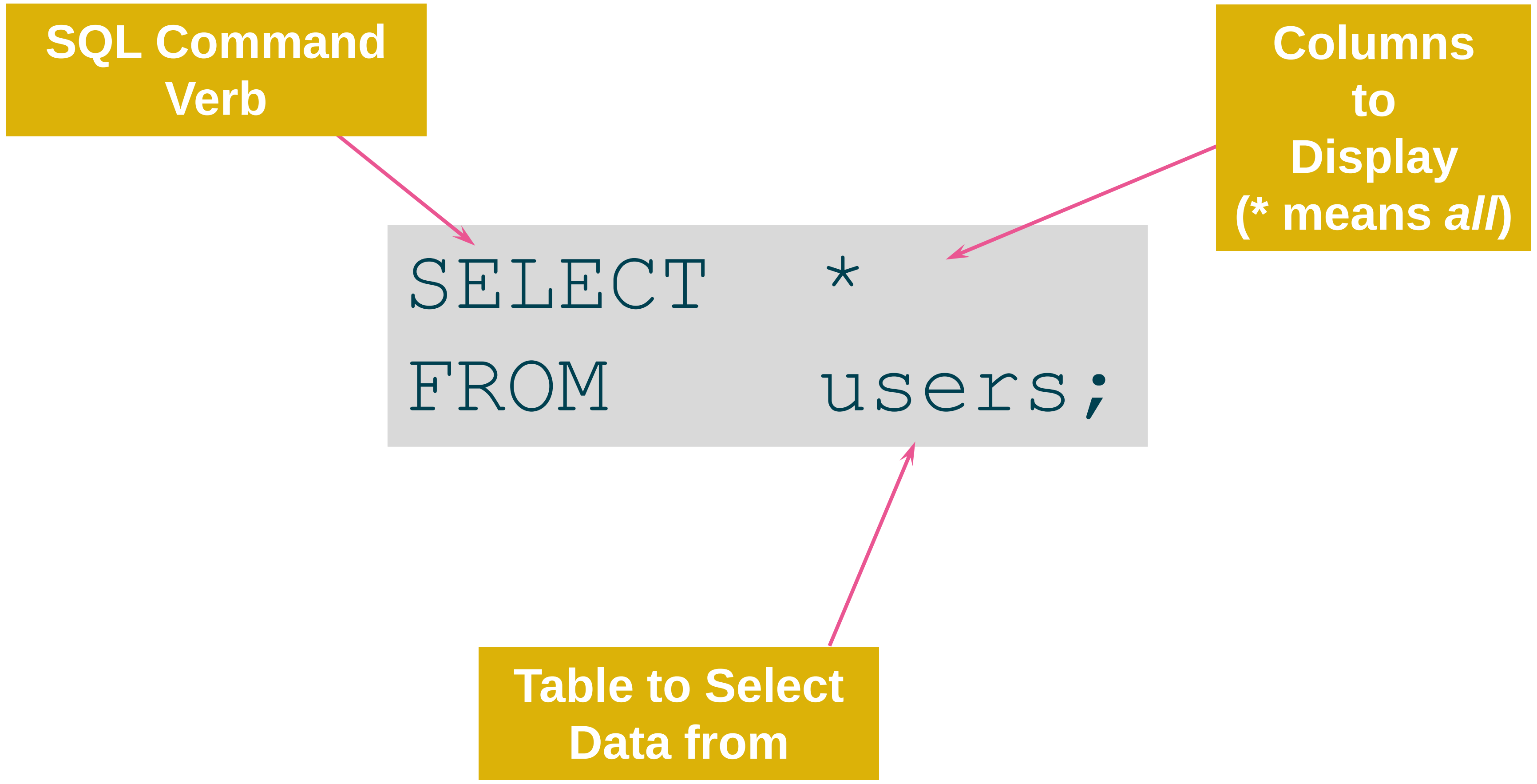
THERE IS NO PROMPT

THERE IS NO WARNING

THERE IS NO UNDO FUNCTION

Be careful with these commands!

Simple SELECT



```
SELECT   *
FROM     users;
```

Statement Format

SQL is a free format language

→ Use new lines, tab keys and indentation to make it readable

→ White space is ignored by the parser

Make use of comments, ignored by runtime engine

Comment

```
SELECT  *       -- all columns
FROM    items;
```

Specifying Columns

Columns to Display

```
SELECT    username, email
FROM       users;
```

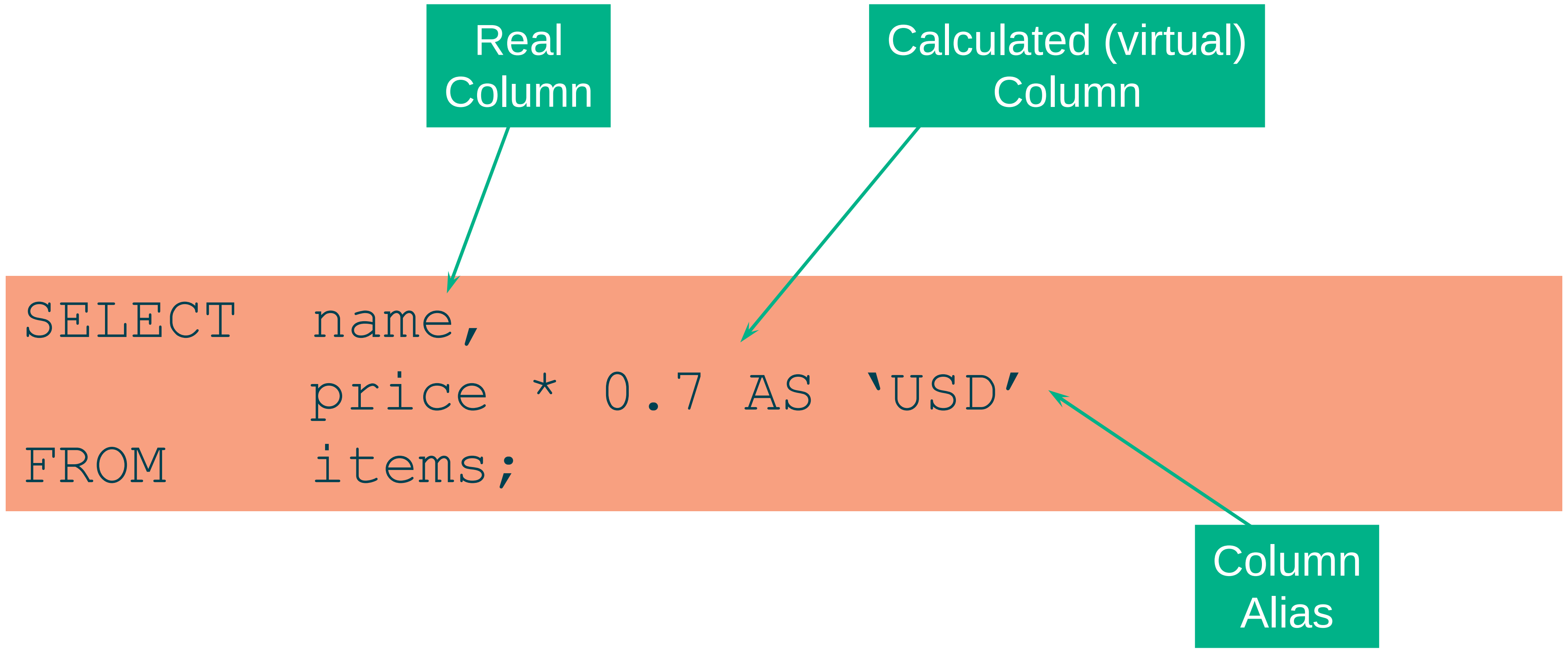
You have two choices

Use * or list the columns separated by commas

Columns may be listed in any order

Columns are displayed in the order you specify

Calculated (Virtual) Columns and Aliases



```
SELECT  name,
        price * 0.7 AS 'USD'
FROM    items;
```

‘AS’ keyword can be omitted

- Quotes can also be omitted if no spaces in alias name

Combining Strings - Concatenation

```
SELECT concat(firstname,' ', lastname)
FROM users;
```

Concat is a function in SQL

It takes in a number of strings (identified using the quotation marks ‘’)

The output is a single string with all the parts in order

We can run these as a standalone statements

```
mysql> select concat('hello', ' ', 'world');
+-------------------------------+
| concat('hello', ' ', 'world') |
+-------------------------------+
| hello world                   |
+-------------------------------+
1 row in set (0.00 sec)
```

Keyword - Distinct

Distinct returns only unique values

```
mysql> select distinct price from items;
+---------------+
| price         |
+---------------+
|           500 |
|          1700 |
|          2000 |
+---------------+
```

The distinct keywords will look for unique combinations of the fields given

Sorting the Results

```
SELECT      *
FROM        users
ORDER BY username, firstname
```

By Specific Column(s)

ORDER BY username ASC

ORDER BY username DESC

Boolean Operators

We can filter the results of any select statement by using the **where** clause
→ This requires a boolean statement (predicate)

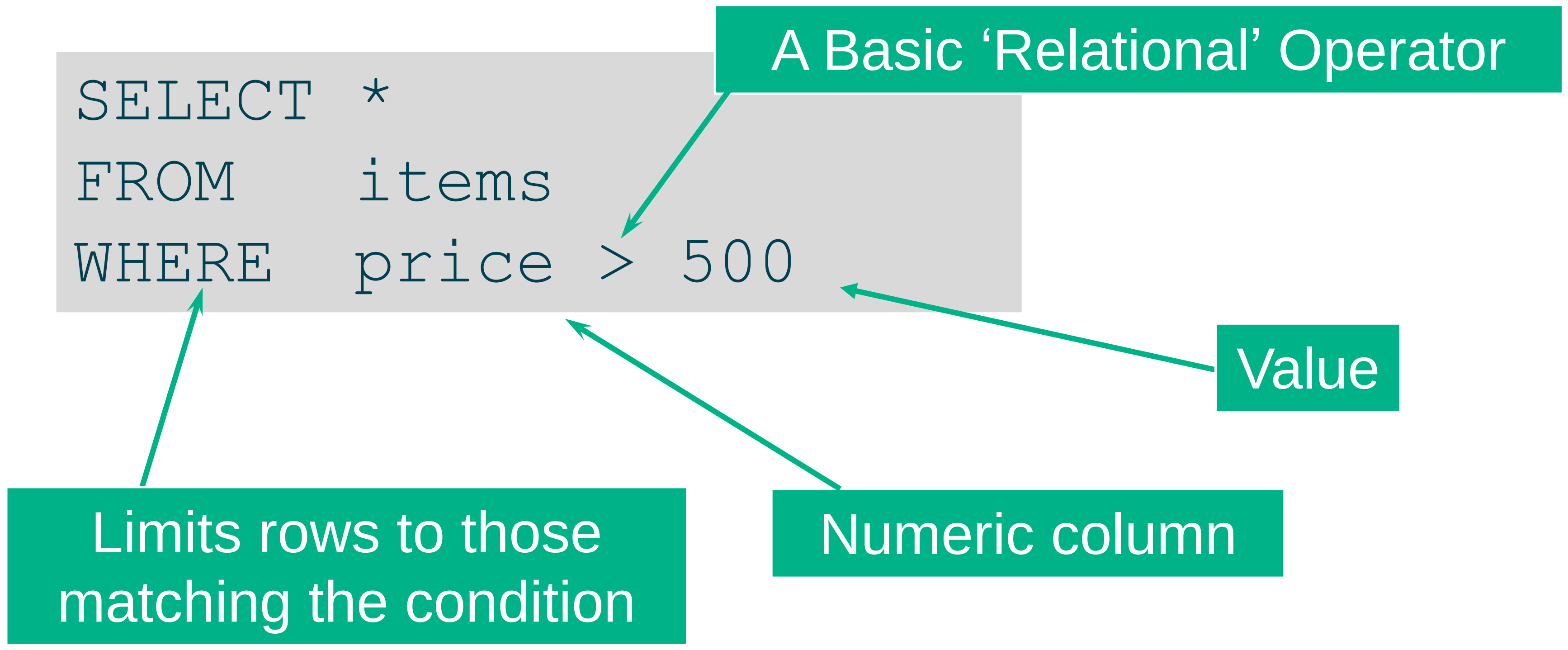
Less than, greater than, equal to, in a range

• 1 < 2	➔	true
• 1 > 2	➔	false
• ‘hello’ = ‘world’	➔	false
• ‘harry’ in (‘tom’,’harry’)	➔	true
• 55 between 10 and 100	➔	true

In SQL queries, use fields in your predicates

- where username = ‘sholmes’

Limiting Rows with Basic Operators



```
SELECT *
FROM   items
WHERE  price > 500
```

A Basic ‘Relational’ Operator
Value
Limits rows to those matching the condition
Numeric column

Think of a WHERE clause as an ‘IF’ statement

Include this row in result set ‘if’ the test returns true

Basic operators include >, >=, <, <=, =, <>

BETWEEN

```
SELECT *
FROM   items
WHERE  price BETWEEN 500 AND 1000
```

Starting Value

Stopping Value

A lot easier than typing

```
WHERE  price >= 500 AND
       price <= 900
```

Values are inclusive

IN

Peter
George
Tom
Mike
Sandy
Eleanor
Bill
Gary
Grace
Harry
Samantha
Dick

```
SELECT   firstname
FROM     users
WHERE    firstname IN ('Tom',
'Dick', 'Harry')

-- easier than coding
WHERE    firstname = 'Tom' OR
         firstname = 'Dick' OR
         firstname = 'Harry'
```

$\longrightarrow$

Tom
Dick
Harry

NOT

```
Peter
George
Tom
Mike
Sandy
Eleanor
Bill
Gary
Grace
Harry
Samantha
Dick
```

```sql
SELECT   firstname
FROM     users
WHERE    firstname NOT IN
                  ('Tom', 'Dick', 'Harry')
```

→

```
Peter
George
Mike
Sandy
Eleanor
Bill
Gary
Grace
Samantha
```

LIKE

When used with 'LIKE'

'_' Matches any single character

'%' Matches any number of characters (including none)

```
WHERE surname LIKE 'a%'

WHERE surname LIKE '%a'
```

starts with 'A'

ends with 'A'

```
WHERE surname LIKE '%a%'

WHERE surname LIKE '_t%n%r_'
```

contains an 'A'

has a 'T' in pos 2
has an 'R' 1 char from end
and an 'N' between them

LIKE is only used with character columns

e.g 'stationary'

Logical Operators AND / OR

Both conditions must be true

```
SELECT   *
FROM     users
WHERE    lastname = ‘Holmes’
         AND password = 'sherlock';
```

Every time you say AND you are likely to get less rows

Either (or both) conditions can be true

```
SELECT   *
FROM     users
WHERE    lastname = ‘Holmes’
         OR password = 'sherlock';
```

Every time you say OR you are likely to get more rows

Nulls

Basic premise of an RDBMS is the concept of optional columns

→ NULL means 'not applicable' or 'unknown', different from zero or blank

On INSERT of a row, must supply values for mandatory columns

→ Other columns may be left as NULL (assuming no 'DEFAULT' value)

NULL propagates through expressions: (5 + null) is null, not 5

→ Nothing is equal to null, not even null = null

WHERE clause expressions will evaluate to TRUE, FALSE or NULL

→ Need to think 3 way logic

→ Only rows whose expressions evaluate to TRUE are output

Can use IS NULL to retrieve rows with NULL entries:

```
select *
from users
where email is null
```

Current Ordering

The order of the statements is as follows:

```
select [columns, calculated columns] [as alias]
from [table name]
where [some condition is true]
order by [a field] [asc/desc]
```

SELECT , FROM, WHERE, ORDER BY

Joins

The syntax of a join:

```
select [column1] [as alias], [columnX]
from [tableA]
INNER JOIN [tableB]
ON tableA.ColumnA = tableB.ColumnA
where [some condition is true]
order by [a field] [asc/desc]
```

INNER, OUTER (Left / right / full)

Outer Joins

The syntax of an outer join:

```
select [column1] [as alias], [columnX]
from [tableA]
LEFT / RIGHT / FULL OUTER JOIN [tableB]
ON tableA.ColumnA = tableB.ColumnA
where [some condition is true]
order by [a field] [asc/desc]
```

Multiple Joins

The syntax of a 3 table join:

```
select [column1] [as alias], [columnX]
from [tableA]
INNER JOIN [tableB]
ON tableA.ColumnA = tableB.ColumnA
INNER JOIN [tableC]
ON tableB.ColumnX = tableC.ColumnX
where [some condition is true]
order by [a field] [asc/desc]
```

Stored Procedures

The syntax of a stored procedure:

```
DELIMITER //

create procedure searchProductByKeyword
(IN keyword varchar(10))
begin
select *
from product
where name like concat('%',keyword,'%');
end //
delimiter ;
```

Any questions?

Activity:
Exercise 19

RESOURCES

MySQL Documentation

https://dev.mysql.com/doc/

Select Syntax

https://dev.mysql.com/doc/refman/5.7/en/select.html