UCL

Spatial Data Management

- Dr Claire Ellul
- c.ellul@ucl.ac.uk

Assignment Progress ..

- By now you should have written your system specification and created the corresponding conceptual and logical ER Diagrams
  - (At least in draft format)

ER Diagrams

- An ERD is a model of the world, containing information of interest to the particular system that you are trying to build
- As it is a model, it is usual to make some assumptions when constructing the diagram

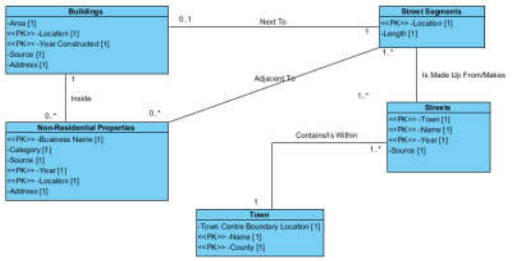
ER Diagrams

- This means that there isn't a 'perfect' answer, and that it is possible that your answers to the UCL Facilities Management will be SLIGHTLY (only slightly!) different to mine if you have made some different assumptions

ER Diagrams

- In practice, you can use the ERD to discuss and refine your understanding of the system with the client – especially as it is easy to understand by a non-technical person

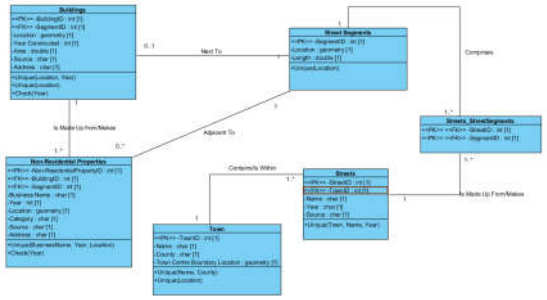
Conceptual to Logical



Conceptual to Logical

- Main Conversion Steps
  - Remove any many:many relationships (more about that today)
  - Replace identifiers with ID column
  - Add unique constraint
  - Add domain types to the attributes (string, date etc)
  - Add any foreign key fields (see today's lecture)
  - Add any other constraints (see today's lecture)

Conceptual to Logical



Overview

- Database vocabulary, primary and foreign keys
- SQL – an introduction
- SQL – DDL
- SQL - DML

Database Vocabulary

- Terminology
  - Tables/Relations
  - Rows/Records
  - Columns/Fields

S#	STUDENT_SURNAME	STUDENT_NAME
100	Smith	Joe
200	Jones	Robert
300	Francis	Alex
400	Morley	Jeremy

Database Vocabulary

- Terminology – Domains/Data Types

Domain Type	Postgres Name
String	character varying
Date	date
Number	integer
Spatial data	geometry

  - Each column stores information using one data type
  - In reality there are a whole range of data types for strings and numbers and many other data types – see here: https://www.postgresql.org/docs/9.5/static/datatype.html

Database Fundamentals

- Schemas and Instances
  - Schema does not change with time
    - For example, the structure of a table containing employee information will always have columns:
      - Name:Surname:DateOfBirth:DateOfEmployment
  - Instance changes with time
    - But the contents of that table will change as more employees are employed or leave

Database Fundamentals

- Transactions
  - A Transaction is an atomic unit of interaction between user and Database
    - Insertion
    - Modification/update
    - Deletion
    - Retrieval

Database Fundamentals

- Transactions
  - Many databases are optimised for typical transaction processing applications, and handle:
    - Many simultaneous users
    - Generally short transaction times
  - Support for concurrent access to database
    - File, table, row, field level locking
  - Commit & Rollback operations

Database Vocabulary

- Terminology
  - Key fields
    - *Primary Key*
      - Unique identifier for each row in a table
      - Often composite (made up of multiple fields)
    - *Foreign Key*
      - Attribute (possibly composite) of a table whose values are required to match those of the primary key of some other table
    - Primary Key in one table (parent) becomes the Foreign Key in the other table (child)

Primary and Foreign Keys

STUDENTS

STUDENT_SURNAME	STUDENT_NAME
Smith	Joe
Jones	Robert
Francis	Alex
Morley	Jeremy

COURSES

COURSE_NAME
Spatial Decision Support
Topographic & Base mapping
Remote Sensing and GIS
Cadastral & Land Information Systems

Primary Key in **Students** Becomes foreign key in **Exam Grades**

Primary Key in **Courses** Becomes foreign key in **Exam Grades**

EXAM GRADES

STUDENT_SURNAME	STUDENT_NAME	COURSE_NAME	GRADE
Smith	Joe	Spatial Decision Support	75
Smith	Joe	Topographic and Base Mapping	85
Francis	Alex	Spatial Decision Support	87
Smith	Joe	Remote Sensing and GIS	51

Primary and Foreign Keys

Name	Surname	Salary	Address
John	Smith		33 Acacia Avenue
John	Smith		33 Acacia Avenue

•Which John Smith earns £33,023 and which one earns £100,929?

Primary and Foreign Keys

ID	Name	Surname	Salary	Address
1001	John	Smith		33 Acacia Avenue
1002	John	Smith		33 Acacia Avenue

•NB: It is not enough just to add an ID field. You still don't know which John Smith earns £33,023 and which one earns £100,929!

Primary and Foreign Keys

ID	Name	Surname	Salary	Date of Birth	Address
1001	John	Smith		21/01/1946	33 Acacia Avenue
1002	John	Smith		30/01/1970	33 Acacia Avenue

•Sometimes, you need to add another field – in this case the date of birth.

NB: The ID column is a substitute short cut for the REAL primary key

Primary and Foreign Keys

- Using Numerical ID Fields
  - In many DBMS (for example Microsoft Access) it is possible to create a primary key that is simply an ID value that is automatically incremented when a new row is inserted
  - However, this does NOT uniquely define a row, as the link between the number and the rest of the record is arbitrary
  - So you need to have a correct primary key first, and you can then use the ID as an alternate key
  - Use NOT NULL constraints (see below) to make sure that all required values for the true primary key are filled in

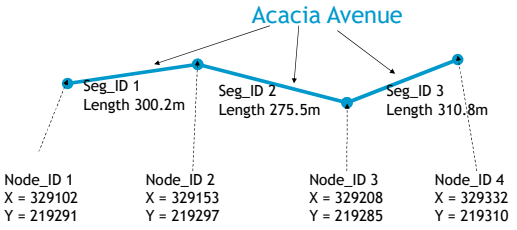
Primary and Foreign Keys

- In a relational database
  - a primary key can have multiple foreign keys that reference it
  - a foreign key only ever references ONE primary key
  - this a 1:many relationship
  - You can have many:many relationships in a conceptual diagram – this is how the real world works
  - But many:many relationships have to be modelled using an extra entity (in the logical diagram)

Primary and Foreign Keys

- A many:many Example - Acacia Avenue
  - Acacia Avenue is made of multiple segments
  - Each segment has a unique Seg_ID (which is the substitute key for the geometry) and a length
  - Each segment also has a start and end node, which contain the coordinate information
  - Information must be stored ONLY ONCE

Primary and Foreign Keys



Primary and Foreign Keys

STREETS

Street_ID	Street Name
1	Acacia Avenue

Primary Key in STREETS table becomes Foreign Key in STREET_SEGMENTS table

STREET_SEGMENTS

Seg_ID	Length	Street_ID
1	300.2	1
2	275.5	1
3	310.8	1

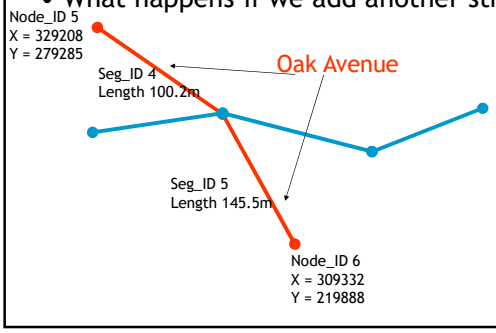
Primary Key in STREET_SEGMENTS table becomes Foreign Key NODES table

NODES

Node_ID	X	Y	Start Node for Segment ID	End Node for Segment ID
1	329102	219291	1	NULL
2	329153	219297	2	1
3	329208	219285	3	2
4	329332	219310	NULL	3

Primary and Foreign Keys

- What happens if we add another street?



Primary and Foreign Keys

STREET_SEGMENTS

Seg_ID	Length	Street_ID
1	300.2	1
2	275.5	1
3	310.8	1
4	100.2	2
5	145.5	2

NODES

Node_ID	X	Y	Start Node For Segment ID	End Node for Segment ID
1	329102	219291	1	NULL
2	329153	219297	2	1
3	329208	219285	3	2
4	329332	219310	NULL	3
5	329208	279285	NULL	4
6	309332	219888	5	NULL
2	329153	219297	5	4

Primary and Foreign Keys

- Issue with Many to Many relationships:
  - In the above data structure, coordinate data for Node ID 2 appears TWICE (as the Node is used by 4 linked segments)
  - If someone moves Node 2, it is possible that the coordinate data is only updated in one place, leading to an inconsistent database
  - In more general terms
    - 1 Node is linked to one or more segments
    - 1 segment is linked to 2 nodes
  - This is known as a Many:Many relationship
  - Resolve this by creating an additional entity in the logical diagram (which becomes an additional table)

STREET_SEGMENTS

Seg_ID	Length	Street_ID
1	300.2	1
2	275.5	1
3	310.8	1
4	100.2	2
5	145.5	2

STREETS

Street_ID	Street Name
1	Acacia Avenue
2	Oak Avenue

NODES

Node_ID	X	Y
1	329102	219291
2	329153	219297
3	329208	219285
4	329332	219310
5	329208	279285
6	309332	219888

NODE_SEGMENTS

Seg_ID	Node_ID	Start Node?
1	1	TRUE
1	2	FALSE
2	2	TRUE
2	3	FALSE
3	3	TRUE
3	4	FALSE
4	5	TRUE
4	2	FALSE
5	2	TRUE
5	6	FALSE

Exercise: many:many

- Sketch out an entity for Underground Railway Tracks, with attributes
  - Installation date
  - Last maintained
  - Next maintenance due
  - Location
- Assume that one TRACK represents the ENTIRE line, and is made up of many track segments

http://s0.geograph.org.uk/geophotos/05/03/42/5034239_00c2f915.jpg

Exercise: many:many

- Sketch out an entity for Underground Stations, with attributes
  - Name
  - Location
  - Opening Hours
  - Maintenance Hours

https://upload.wikimedia.org/wikipedia/commons/thumb/e/e6/Baker5tEntrance..JPG/1200px-Baker5tEntrance..JPG

Exercise: many:many

- Draw the relationship between tracks and stations:
  - At conceptual level
  - At logical level

Overview

- Database vocabulary, primary and foreign keys
- **SQL - an introduction**
- SQL – DDL
- SQL - DML

Spatial Data Management

- SQL – Structured Query Language
  - Standard language for accessing and manipulating databases
  - Allows users to create tables, add constraints and insert, update or delete data in the tables
  - Also allows users to interrogate or query the data – i.e. answers questions
  - International Standard – supported by International Organisation for Standardisation (ISO) and American National Standards Institute (ANSI)

Spatial Data Management

- The SQL Language
  - Scripting language (not compiled)
  - 4th Generation Language
  - Not a programming language although can be accessed through many programming languages

Spatial Data Management

- SQL – A History
  - Originally developed for an IBM *System R* database in the 1970s
  - Has since been adopted by all mainstream relational databases
  - Standardised, but many 'additions to' and 'flavours of' the standard by individual vendors
  - We will be learning about the PostGreSQL flavour

Spatial Data Management

- SQL is composed of three parts
  - Data Definition Language (DDL) which allows users to create and modify the tables in the databases
  - Data Manipulation Language (DML) which allows users to add, edit or delete data from the tables that have been created
  - Query Language which allows users to interrogate the data

Data Definition Language

- Creates the STRUCTURE of the database
  - What tables exist, the columns in the tables, the type of data that you can put in each column
  - Data definition language (DDL) is used to create and destroy databases and database objects such as tables and constraints.

Data Manipulation Language

- Used to create/edit/delete the DATA that goes into the tables created by DDL
- Data manipulation language (DML) is used to insert, update and delete data from the database once the database and associated objects have been created using the Data Definition Language (DDL).

Query – The Select Statement

- This is how you USE the data that has been created with DML
- The select statement is the third and final part of SQL. It is key to extracting data from the database and using the database to answer questions. It is the most commonly used command in SQL.

Overview

- Database vocabulary, primary and foreign keys
- SQL – an introduction
- **SQL - DDL**
- SQL - DML

DDL

- Data Definition Language - DDL
  - Creating a schema
  - Creating a table
  - Removing a table
  - Modifying a table
  - In some database systems, each statement must be followed by a ; or /

DDL – Creating a Schema

- A schema is a place that holds a set of tables needed for a particular system
- First remove the schema if it already exists

```
DROP SCHEMA IF EXISTS assetsclass;
```

DDL – Creating a Schema

- Using DROP SCHEMA deletes all the data !
- So if you are unsure, rename it instead:

alter schema assetsclass rename to assetsclass_backup_23Nov2018;

DDL – Creating a Schema

- Now create the new schema:

CREATE SCHEMA assetsclass;

DDL

- Creating a Table
  - Use the CREATE TABLE statement
  - Domains used to identify the types of the attributes
  - Each attribute definition except the last one should be followed by a ,

CREATE TABLE <TABLENAME>
(FIELD_NAME_1 <FIELD_TYPE>,
FIELD_NAME_2 <FIELD_TYPE>,
FIELD_NAME_3 <FIELD_TYPE>);

DDL

- Creating a Table – Nulls
  - In the ER diagram, we defined the cardinality of the attributes
  - If we want to FORCE an attribute to have a value, we should use NOT NULL

DDL

- Only use NOT NULL when you are really sure that there will always be a value in the column!

CREATE TABLE <TABLENAME>
(FIELD_NAME_1 <FIELD_TYPE> NOT NULL,
FIELD_NAME_2 <FIELD_TYPE>,
FIELD_NAME_3 <FIELD_TYPE>);

Database Vocabulary

- Some PostGIS Specific Terminology

General Domain Type	PostGIS Terminology
String	character varying (length)
Date	date
Number	integer numeric (precision, scale)
Spatial Data	geometry
(automatically increasing number used for ID values)	serial

DDL

- Creating a table – example

```
drop table if exists assetsclass.university;
create table assetsclass.university (
        university_id serial,
        university_name character varying (100),
        year_founded integer,
        founders_name character varying (100));
```

DDL

- Creating a Table - Example

```
create table assetsclass.buildings (
   building_id serial,
   building_name character varying (200) NOT NULL,
   university_id integer NOT NULL
);
```

DDL

- Creating a Table - Example

```
create table assetsclass.rooms (
   room_id serial,
   floor integer NOT NULL,
   last_repainted date NOT NULL,
   building_id integer NOT NULL,
   room_use character varying(50) NOT NULL,
   room_number character varying (50) NOT NULL
);
```

DDL

- Creating a Table - Example

```
drop table if exists assetsclass.windows;
create table assetsclass.windows (
        window_id serial,
        building_id integer,
        window_type character varying(100),
        window_installation_date date,
        room_id integer,
        floor integer
);
```


DDL

- Creating a table – Example

```
drop table if exists assetsclass.cleaner;
create table assetsclass.cleaner (
        cleaner_id serial,
        cleaner_name character varying (100),
        cleaner_surname character varying (100),
        date_of_birth date,
        contact_number character varying (100)
        );
```

DDL

```
DROP TABLE IF EXISTS assetsclass.noise;
CREATE TABLE assetsclass.noise (
noise_in_dB numeric(5,2),
date_and_time date);
```

Note: numeric(5,2) = 5 digits in total, 2 of which after the decimal point, so 999.99 is the maximum value

DDL

- Deleting a Table
  - Drop Table <tablename>
  - This will delete the table AND all the data in the table (if the table exists)
  - Also useful to run this before a CREATE TABLE statement – but NB all data is lost!

```
DROP TABLE IF EXISTS assetsclass.buildings;
```

DDL

- Modifying a Table – Adding a Column
  - Alter table <tablename> add (<column description>)

```
ALTER TABLE assetsclass.buildings add
  number_of_inhabitants integer;
```

DDL

- Modifying a Table – Removing a Column
  - Alter table <tablename> drop column <column name>

```
ALTER TABLE assetsclass.buildings drop
column number_of_inhabitants;
```

DDL

- Worksheet – Table Creation

DDL - Integrity Constraints

- Constraint Types
  - Primary Key - Keys
  - Foreign Key - Referential Constraints
  - Not Nulls -
  - Domain Restrictions
  - Tuple Restrictions

DDL - Integrity Constraints

- Constraints - Primary Keys

```
ALTER TABLE <TABLENAME>
ADD CONSTRAINT <CONSTRAINTNAME_PK>
PRIMARY KEY(<FIELDNAME>)
```

```
alter table assetsclass.buildings add CONSTRAINT
  buildings_pk PRIMARY KEY (building_id);
```

```
alter table assetsclass.university add CONSTRAINT
  university_pk
PRIMARY KEY (university_id);
```

DDL - Integrity Constraints

- Constraints - Foreign Keys
  - ALTER TABLE <TABLENAME>
  - ADD CONSTRAINT <CONSTRAINTNAME_FK> FOREIGN KEY(<FIELDNAME>)
  - REFERENCES <TABLENAME>(<FIELDNAME>)

```
alter table assetsclass.buildings
add constraint buildings_university_fk
foreign key(university_id) references
assetsclass.university(university_id);
```

DDL – Integrity Constraints

```
alter table assetsclass.windows
add constraint windows_rooms_fk
foreign key(room_id) references
assetsclass.rooms(room_id);
```

DDL – Integrity Constraints

- You will get an error:

There is no unique constraint matching given keys for referenced table "rooms"

As the primary key is missing, so create that first

DDL – Integrity Constraints

```
-- add the required primary key first
alter table assetsclass.rooms add CONSTRAINT
rooms_pk PRIMARY KEY (room_id);

-- then rerun the foreign key constraint
alter table assetsclass.windows
add constraint windows_rooms_fk
foreign key(room_id) references
assetsclass.rooms(room_id);
```


DDL - Integrity Constraints

- Constraints - Domain Constraints
  - ALTER TABLE <TABLENAME>
  - ADD CONSTRAINT <CONSTRAINTNAME_CHK>
  - CHECK (<TEST>)

```
alter table assetsclass.windows
add constraint window_type_check
check (window_type in ('single glazed','double
glazed','triple glazed'));
```

DDL - Integrity Constraints

- Constraints – Unique Constraints
  - You MUST have these if you use an ID value as a primary key
  - These make sure your REAL primary key is kept unique
  - Every table you create should have a UNIQUE constraint (except if it was created to resolve a many:many relationship)

```
alter table assetsclass.cleaner
add constraint cleaners_unique
unique(cleaner_name, cleaner_surname,
date_of_birth);
```

DDL -Integrity Constraints

- Constraints - Tuple Constraints
  - These are created as triggers, which are procedures that are run when data is inserted into the database
  - Triggers are associated to the table on which they act

DDL - Integrity Constraints

- Constraints - Tuple Constraints

CREATE TRIGGER <schema>.<trigger_name>
BEFORE DELETE OR INSERT OR UPDATE ON
<schema>.<tablename>
<code>

DDL - Integrity Constraints

- Constraints - Tuple Constraints
  - Automatically update the room_id for the temperature sensor if the temperature sensor is moved
  - NB: These are **not required for your assignment** but could be something that gains you bonus points if you are good at programming

DDL - Integrity Constraints

```
CREATE OR REPLACE FUNCTION assetsclass.update_sensor_room_id() RETURNS trigger AS
$BODY$
    DECLARE
    originalroomid integer;
    newroomid integer;
    BEGIN
    select room_id into originalroomid from assetsclass.temperature_sensor where
    sensor_id=NEW.sensor_id;
    select room_id into newroomid from assetsclass.rooms a where st_contains(a.location,
    NEW.location);
    raise 'Original ID: %', originalroomid;
    raise 'New Length: %', newroomid;
    IF NEW.room_id <> originallength THEN
        UPDATE assetsclass.temperature_sensor set room_id = newroomid where id =
    NEW.id;
    end if;
      RETURN NULL;
    END;
$BODY$ LANGUAGE 'plpgsql';
```

DDL - Integrity Constraints

- To associate the trigger with the table:

```
CREATE TRIGGER assetsclass.update_sensor_room_id
    AFTER INSERT OR UPDATE OF location ON
     assetsclass.temperature_sensor
    FOR EACH ROW
    EXECUTE PROCEDURE update_sensor_room_id ();
```

- The trigger then runs when data is inserted/updated in the table
  - (NB: the trigger requires the temperature_sensor table and also uses spatial functionality so we can't run it yet)

Constraints

- Worksheet - Constraints

Overview

- Database vocabulary, primary and foreign keys
- SQL – an introduction
- SQL – DDL
- **SQL - DML**

Spatial Data Management - DML

- Data Manipulation Language - DML
  - Adding data to the database
  - Changing data in the database
  - Removing data from the database
- Three operations
  - Insert
  - Update
  - Delete

Spatial Data Management - DML

- DML
  - Each statement must be ended by a semi-colon ;

Spatial Data Management - DML

- A useful SELECT statement
  - Will allow you to see what has been added to the table

```
SELECT * FROM <TABLENAME>
```

- In SQL * means all rows

DML

- Insert
  - INSERT INTO <TABLENAME> (<FIELDLIST, SEPARATED BY COMMAS>) VALUES (<DATA LIST, SEPARATED BY COMMAS>)

```
insert into
  assetsclass.university(university_name,year_fou
  nded,founders_name)
values ('UCL','1826','Jeremy Bentham');
```

DML – how SERIAL works

- The ID values in a database are assigned automatically using the SERIAL data type

```
select * from assetsclass.university;


-- insert some data (with an error)
insert into
  assetsclass.university(university_name,year_fou
  nded,founders_name)
values ('UCL Stratford', '2017',Jeremy Benthom');
```

DML – how SERIAL works

- Delete the error and create the correct data

```
delete from assetsclass.university where
  founders_name = 'Jeremy Benthom';

insert into
  assetsclass.university(university_name,ye
  ar_founded,founders_name)
values ('UCL Stratford', '2017', 'Jeremy
  Bentham');

select * from assetsclass.university;
```

DML – how SERIAL works

- If a row is deleted the ID is not reused
- You can also have a situation with multiple people inserting data at the same time
  - So even though this might be the third row YOU inserted, it may not have ID = 3;

DML

- Insert – referencing another value
  - So, you can't guarantee the ID – you need to find it out every time
  - This is done using an SQL statement as follows:

```
select university_id from assetsclass.university where
  university_name = 'UCL';

insert into assetsclass.buildings (building_name,
  university_id)
values ('Chadwick', (select university_id from
  assetsclass.university where university_name =
  'UCL'));
insert into assetsclass.buildings (building_name,
  university_id)
values ('Parson', (select university_id from
  assetsclass.university where university_name =
  'UCL'));
```

DML

- Multiple Insert Statements

```
insert into assetsclass.windows(building_id, window_type, window_installation_date,
floor, room_id=1) values
((select building_id from assetsclass.buildings where building_name = 'Chadwick'), 'triple
glazed','23-May-2014',null,1),
((select building_id from assetsclass.buildings where building_name = 'Chadwick'), 'triple
glazed','23-May-2014',null,1),
((select building_id from assetsclass.buildings where building_name = 'Chadwick'), 'triple
glazed','23-May-2017',null,1),
((select building_id from assetsclass.buildings where building_name = 'Pearson'), 'triple
glazed','23-May-2014',null,1),
((select building_id from assetsclass.buildings where building_name = 'Pearson'), 'single
glazed','22-May-2014',null,1),
((select building_id from assetsclass.buildings where building_name = 'Pearson'), 'single
glazed','23-May-2014',null,1);
```

DML

- Update
  - UPDATE TABLE <TABLENAME> SET <FIELDNAME> = <VALUE> WHERE <FIELDNAME> = <VALUE>

```
update assetsclass.buildings set building_name = 'Pearson'
where building_name = 'Parson';
```

DML

- Updating multiple columns:

```
update assetsclass.windows set
window_installation_date = '15-Jun-2012',
floor = 2, room_id=1 where building_id = (select
building_id from assetsclass.buildings);
```

DML

- The where clause
  - Allows you to select a subset of the rows in a database, so that you don't modify all the data

```
update assetsclass.windows set
window_installation_date = '15-Jun-2012', floor =
2, room_id=1 where building_id = (select
building_id from assetsclass.buildings where
building_name = 'Pearson');
```

  - Will change the data for all the PEARSON windows

DML

- The WHERE clause:
  - Also allows you to combine more than one criterion using BOOLEAN logic

```
update assetsclass.windows set
window_installation_date =
'15-Jun-2012', floor = 2, room_id=1 where
building_id =
(select building_id from assetsclass.buildings
where building_name = 'Pearson')
 and window_installation_date = '23-May-2014' and
window_type = 'single glazed';
```


DML

- Boolean Logic

Logical Operators	Description
OR	For the row to be selected at least one of the conditions must be true.
AND	For a row to be selected all the specified conditions must be true.
NOT	For a row to be selected the specified condition must be false.

Source: http://beginner-sql-tutorial.com/sql-logical-operators.htm

DML

- Delete
  - Will remove ALL the rows from a table

DELETE FROM <TABLENAME> WHERE FIELDNAME = <VALUE>

```
delete from assetsclass.university where
university_name = 'UCL Stratford';
```

What happens if there is a primary/foreign key reference?

```
delete from assetsclass.university where
university_name = 'UCL';
```

DML

ERROR: update or delete on table "university" violates foreign key constraint "buildings_university_fk" on table "buildings"
DETAIL: Key (university_id)=(5) is still referenced from table "buildings".

DML

- Worksheet – Data Manipulation Language