Spatial Data Management

Combining Your Data and Existing Data

Last Week

- We looked at data
  - Where to find it
  - How to create it
  - And how to link different bits of data together

This Week

- So far the examples of data linking we've seen have been very simple – but what about if there are more than 2 datasets?
  - A sketched diagram called an ER diagram can help to find common information
- We'll also start working with databases – which provide a central store that makes linking (and managing) all this data easier

This Lesson

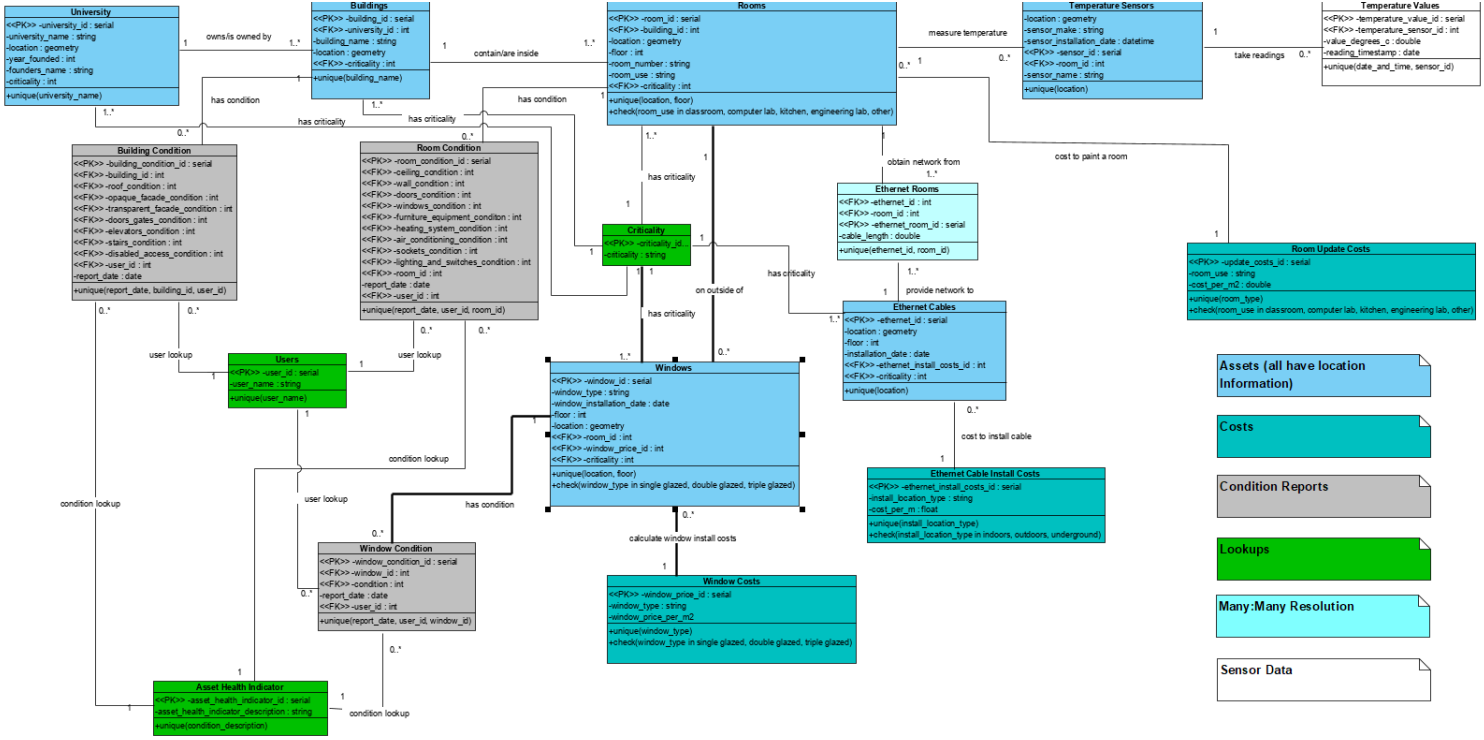
- We'll look at a type of sketch called and Entity Relationship diagram that can be useful to understand multiple datasets and how they link together

- In particular we'll look at relationships – how the datasets are linked/joined

How to Combine Data?

- So far, we've seen how to join two different sources of information – the *asset* and *condition report*
- But in reality we probably have lots more information to think about –so how do we do this?
- Understand the data first!
  - Maybe sketch an ER diagram

Example


University
<<PK>> -university_id : serial
-university_name : string
-location : geometry
-year_founded : int
-founders_name : string
-criticality : int
+unique(university_name)
owns/is owned by
Buildings
<<PK>> -building_id : serial
<<FK>> -university_id : int
-building_name : string
-location : geometry
<<FK>> -criticality : int
+unique(building_name)
contain/are inside
Rooms
<<PK>> -room_id : serial
<<FK>> -building_id : int
-location : geometry
-floor : int
-room_number : string
-room_use : string
<<FK>> -criticality : int
+unique(location, floor)
+check(room_use in classroom, computer lab, kitchen, engineering lab, other)
measure temperature
Temperature Sensors
-location : geometry
-sensor_make : string
-sensor_installation_date : datetime
<<PK>> -sensor_id : serial
<<FK>> -room_id : int
-sensor_name : string
+unique(location)
take readings
Temperature Values
<<PK>> -temperature_value_id : serial
<<FK>> -temperature_sensor_id : int
-value_degrees_c : double
-reading_timestamp : date
+unique(date_and_time, sensor_id)
has condition
has criticality
Building Condition
<<PK>> -building_condition_id : serial
<<FK>> -building_id : int
<<FK>> -roof_condition : int
<<FK>> -opaque_facade_condition : int
<<FK>> -transparent_facade_condition : int
<<FK>> -doors_gates_condition : int
<<FK>> -elevators_condition : int
<<FK>> -stairs_condition : int
<<FK>> -disabled_access_condition : int
<<FK>> -user_id : int
-report_date : date
+unique(report_date, building_id, user_id)
Room Condition
<<PK>> -room_condition_id : serial
<<FK>> -ceiling_condition : int
<<FK>> -wall_condition : int
<<FK>> -doors_condition : int
<<FK>> -windows_condition : int
<<FK>> -furniture_equipment_condition : int
<<FK>> -heating_system_condition : int
<<FK>> -air_conditioning_condition : int
<<FK>> -sockets_condition : int
<<FK>> -lighting_and_switches_condition : int
<<FK>> -room_id : int
-report_date : date
<<FK>> -user_id : int
+unique(report_date, user_id, room_id)
Criticality
<<PK>> -criticality_id
-criticality : string
obtain network from
Ethernet Rooms
<<FK>> -ethernet_id : int
<<FK>> -room_id : int
<<PK>> -ethernet_room_id : serial
-cable_length : double
+unique(ethernet_id, room_id)
provide network to
Ethernet Cables
<<PK>> -ethernet_id : serial
-location : geometry
-floor : int
-installation_date : date
<<FK>> -ethernet_install_costs_id : int
<<FK>> -criticality : int
+unique(location)
cost to install cable
Ethernet Cable Install Costs
<<PK>> -ethernet_install_costs_id : serial
-install_location_type : string
-cost_per_m : float
+unique(install_location_type)
+check(install_location_type in indoors, outdoors, underground)
cost to paint a room
Room Update Costs
<<PK>> -update_costs_id : serial
-room_use : string
-cost_per_m2 : double
+unique(room_type)
+check(room_use in classroom, computer lab, kitchen, engineering lab, other)
on outside of
Windows
<<PK>> -window_id : serial
-window_type : string
-window_installation_date : date
-floor : int
-location : geometry
<<FK>> -room_id : int
<<FK>> -window_price_id : int
<<FK>> -criticality : int
+unique(location, floor)
+check(window_type in single glazed, double glazed, triple glazed)
calculate window install costs
Window Costs
<<PK>> -window_price_id : serial
-window_type : string
-window_price_per_m2
+unique(window_type)
+check(window_type in single glazed, double glazed, triple glazed)
Window Condition
<<PK>> -window_condition_id : serial
<<FK>> -window_id : int
<<FK>> -condition : int
-report_date : date
<<FK>> -user_id : int
+unique(report_date, user_id, window_id)
has condition
Users
<<PK>> -user_id : serial
-user_name : string
+unique(user_name)
user lookup
condition lookup
Asset Health Indicator
<<PK>> -asset_health_indicator_id : serial
-asset_health_indicator_description : string
+unique(condition_description)
Assets (all have location Information)
Costs
Condition Reports
Lookups
Many:Many Resolution
Sensor Data

Understanding Data

- The Entity Relationship (ER) Diagram
  - Presents data requirements of a system in a manner that is easily understood by management
  - Very useful as a sketch tool to better understand your data
  - (I used Visual Paradigm to create the ER diagram for this module – but a paper sketch also works if you're not submitting an official document)

Understanding Data

- The Entity Relationship (ER) Diagram
  - In theory ..
    - An ER diagram would be used to create a database from scratch
    - Stages include
      - Conceptual
      - Logical
      - Normalisation
      - Physical
  - In practice ..
    - You rarely (never?!) get to create an entirely new database
    - We are using a simplified version for sketch purposes

E-R Diagrams - Notation

- UML
  - Was developed for programming, to model classes, properties and methods
  - Diagram Elements include:
    - Class diagrams – attributes, methods and relationships – WE WILL USE THIS
    - Use Case diagram – actors in a system and their goals – not used in this module

E-R Diagrams - Notation

- Learning UML notation is also useful because:
  - For pre-existing databases, you can also reverse engineer the E-R Diagram using tools provided by the DBMS – usually the result is presented as a logical UML diagram
    - This is useful when you are given a database and are not familiar with the tables and data it contains

E-R Diagrams

- Key Components (Constructs) of an E-R Diagram
  - Entities
  - Attributes – including data types and cardinality
  - Identifiers (eg. ID)
  - Joining datasets
    - Sketching Relationships
    - Sketching Cardinality of Relationships

Entities

具有共同属性且独立存在

- Entity
  - Represents classes of objects that have properties in common and an autonomous existence
  - Each entity must have a unique name
  - Graphically represented in the diagram by means of a UML CLASS

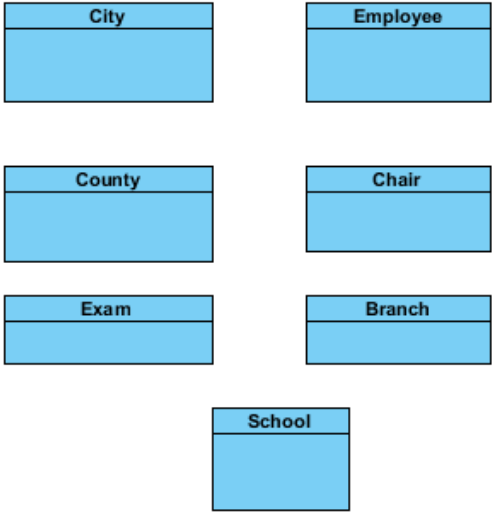
Entities

- An Entity:
  - Examples include: person, product, project assignment
  - Not: technological objects – files, PCs, screens, windows, "Project history"
  - In your sketch each entity will represent a different data source (e.g. a spreadsheet or the output of a questionnaire)

3. Barker, Richard. 1990. *CASE*Method: Entity Relationship Modeling*. (Wokingham, England: Addison-Wesley). (with thanks to David Hay)

E-R Diagrams

- Some Examples of Entities



notation of uml

E-R Diagrams

- Attributes
  - Describe the elementary properties of entities or relationships
  - Can be grouped for simplicity into composites
  - Graphically represented by means of a list within the CLASS

  - *These are the columns in your datasets*

E-R Diagrams

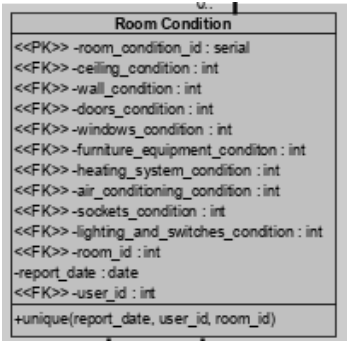
- Attributes
  - *An Attribute is a characteristic of an entity type.*
  - *It "serves to qualify, identify, classify, quantify, or express the state of an entity"*

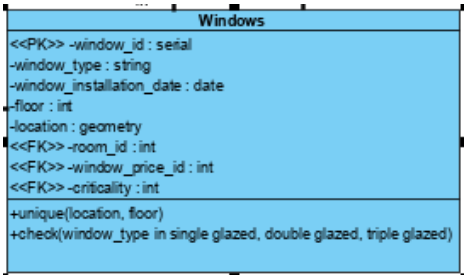
E-R Diagrams

- Some Examples of Attributes

They usually have the data type

- String / text
- Number
- Date
- Geometry





E-R Diagram Constructs

- Identifiers
  - If we're giving John Smith a pay rise, we need to be sure that we have the right John Smith
  - Identifiers allow unambiguous identification of each row of data
    - Formed from one or more attributes of the entity itself
    - E.g. for a person the combination of – name, surname, date of birth, address makes a person unique

E-R Diagram Constructs

- Identifiers

need unique constraint

  - Those of you who have worked with databases before may be familiar with the concept of a number column as a primary key in a table
  - On the diagram it is useful to have these ID values marked as PK (primary key)
  - But it is also helpful to have the real identifier as a *unique* constraint – a rule that will check that the data is correct
    - More about ID values and constraints in later lessons

primary key

is a key in a relational database that is unique for each record.

Cannot contain NULL value

(A relational database must have only one primary key)

in the table, primary key can consist of single or multiple columns

foreign key

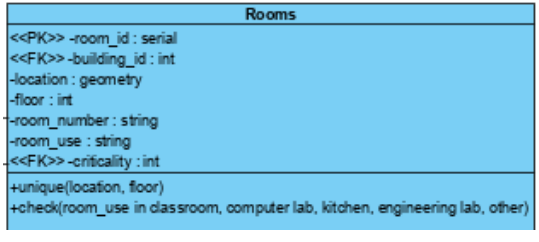
are columns of a table that points to the primary key of another table.

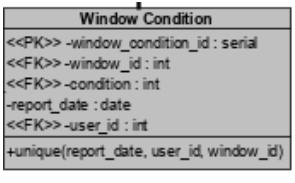
They act as a cross-reference between tables.

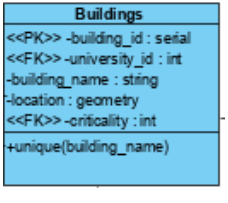
(两个 table 的交叉点)

E-R Diagram Constructs

- Examples of Identifiers
  - The PK and the *unique* rule are both required

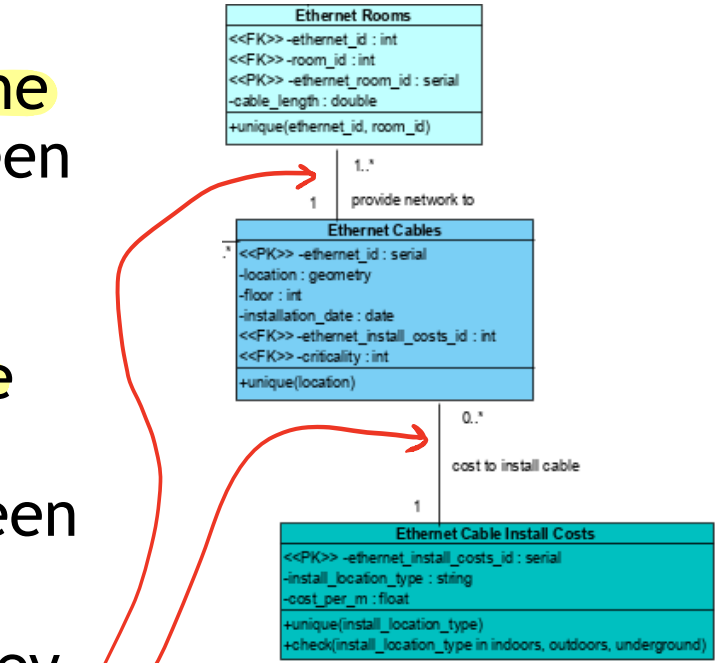






E-R Diagrams

- Relationships
  - These represent the links (joins) between two or more datasets
  - There can be more than one relationship between two datasets
  - On the diagram they are shown as lines



E-R Diagram Constructs

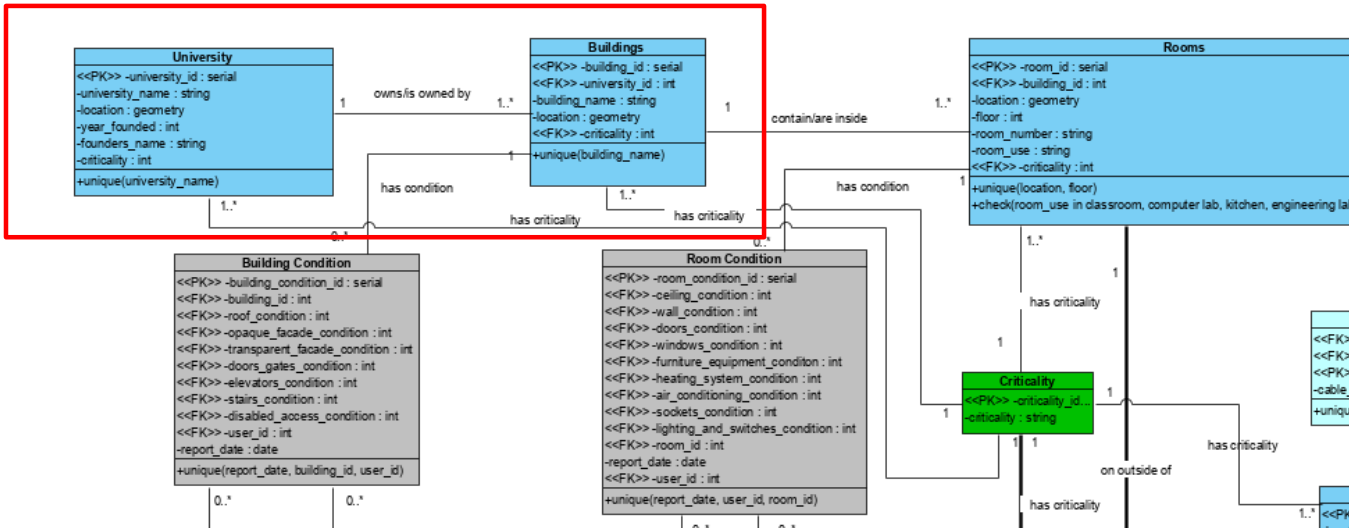
基数 定义了一个 entity 与另一个 entity 的关系里，某方可能出现的次数

- Cardinality of Relationships
  - These describe the minimum and maximum number of relationship occurrences in which an entity can participate
  - Can be (type)
    - One-one
    - One-many
    - Many-many

    } 1 – 1, 1 – n, n – n
  - Also
    - Mandatory (minimum value of 1)
    - Optional (minimum value of 0)

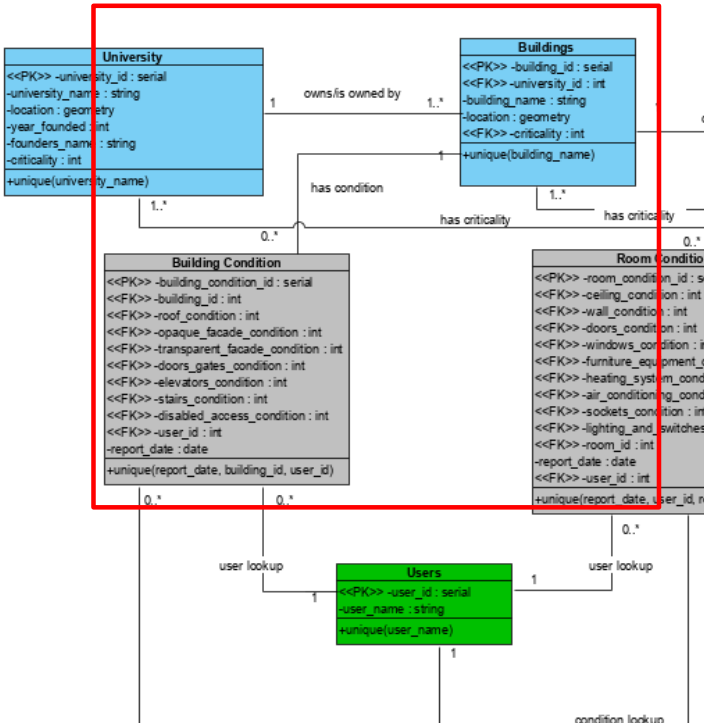
Cardinality of Relationships

- 1:many mandatory (1 1..*)
  - 1 university must have at least one building
  - building must be in one university



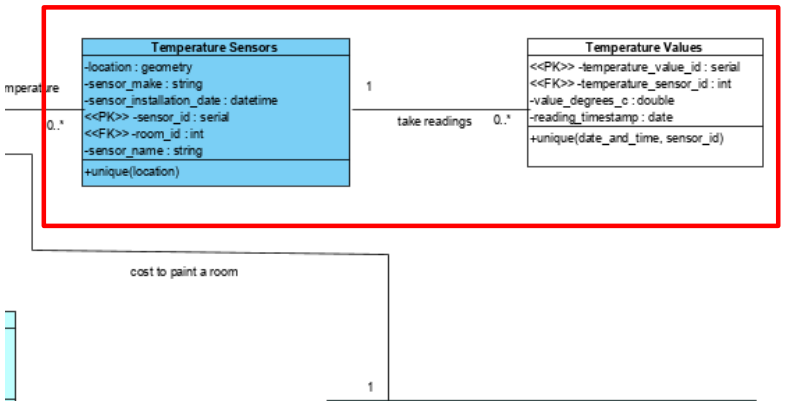
Cardinality of Relationships

- 1:many optional (1 0..*)
  - 1 building may have 0 or more condition reports
  - A building condition report must be associated with a building



Cardinality of Relationships

- 1:many optional (1 0.. *)
  - 1 temperature sensor may have 0 or more readings
  - A temperature reading must be associated with a temperature sensor

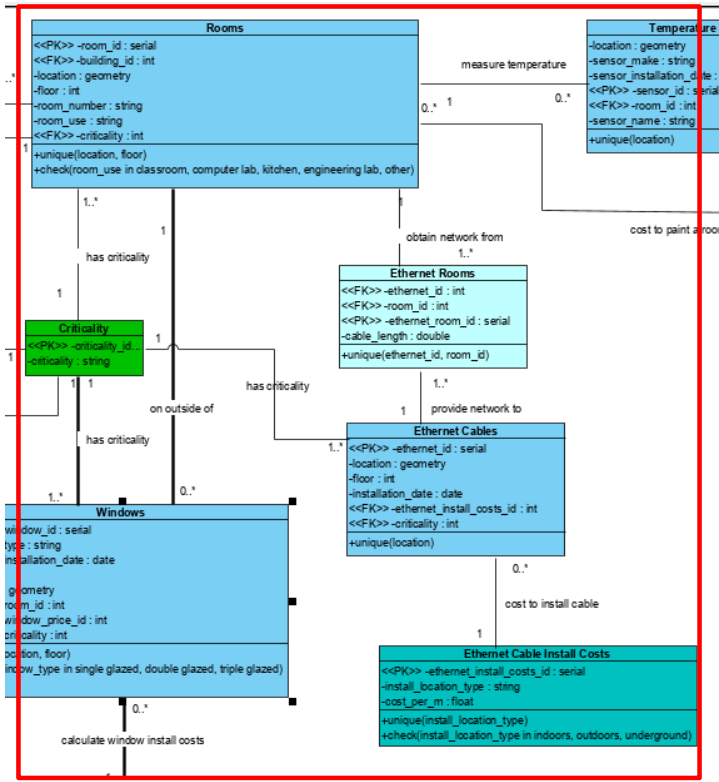


Many – Many Relationships

- In real life these occur quite frequently
  - One car goes to many petrol stations
  - One petrol station serves many cars
- However as we'll see a database can't work with many-many datasets
  - So we usually create an interim table to help out

Many – Many Relationships

- Many- Many:
  - One Ethernet cable passes through many rooms
  - One room contains more than one (many) Ethernet cables
- So if we were to need a link between Ethernet cables and specific rooms this becomes
  - 1 room links to many ethernet_rooms
  - 1 Ethernet_cable links to many ethernet_rooms
- (In the case study we calculate the relationship on-the-fly – i.e. only when needed - using spatial data so this table isn't shown)



This Lesson

- We looked at a type of sketch called and Entity Relationship diagram that can be useful to understand multiple datasets and how they link together

ERD

- In particular we looked at relationships – how the datasets are linked/joined

Next Lesson

- Now that we have some understanding of how information works, we can start to look at how to store and manage it
- One very good way to do this is a database – the next lesson will explain why

Spatial Data Management

Database Concepts

This Lesson

- Now that we have some understanding of how information works, we can start to look at how to store and manage it

- One very good way to do this is a database – this lesson will explain what a database is and why it is a great way to store data

How is data stored?

- On paper
- In a spreadsheet
- In a database management system (see next slides)
- In a PDF
- In a word document
- And many other formats ..

Database Fundamentals

- What is a database?
  - "A database is a collection of data used to represent information of interest to an information system" (Atzeni et al.)
  - Does not have to be computer-based

Database Fundamentals

- What is a DBMS?
  - DataBase Management System
  - A **software** system to manage collections of data
  - Tables, Primary Keys, Foreign Keys all stored in this software system

- NB: the terms database and DBMS are often swapped in practice – so when we say database we often mean DBMS

Database Fundamentals

- Characteristics of a DBMS (1)
  - Large
    - Millions of records
  - Shared
    - Millions of users
  - Persistent
    - Information not lost if computer is switched off
  - Reliable
    - Always gives same result for same question

Database Fundamentals

- Characteristics of a DBMS (2)
  - Efficient
    - No long wait for an answer
  - Secure
    - Multi-user access with varying privileges
    - Disaster recovery
    - Backup mechanism inbuilt
  - Minimises redundancy
    - Information only held once

Database Fundamentals

- Characteristics of a DBMS (3)
  - Self describing
    - You can find out what tables are in the database, what columns are in these tables, what data types the columns are using, the primary and foreign keys
  - Multiple views of the same data
    - Many programs can use the same data for different purposes
  - Has query language inbuilt
    - SQL (Structured Query Language)

Database Fundamentals

- Data Independence
  - Allows users to interact with the database without knowing how the database is physically structured
  - Equally, the data on disk can be moved to another location without the end user being aware

Should it be called a DATAbase?

Data	Numbers, text, points on a map that could mean something or nothing	67
Information	Meaning or relationship is applied to the raw data	67 – number of air conditioners that have broken down this year
Knowledge	Combine information from different sources	67 air conditioners broke down, and 55 of them were of type X, with only 12 of them of type y
Insight	the ability to synthesise knowledge in order obtain a deep understanding of a problem	Air conditioners of type x are far less reliable than those of type y
Wisdom	the ability to use insight to facilitate informed decision making.	Decision: Although air conditioner type Y is expensive, we should replace the 55 air conditioners of type x in the next 2 months

Advantages of a DBMS

- Co-operative approach to database design
  - Involvement of end users essential in designing database - therefore need an accessible framework
- Relational model very easy to understand
  - approachable and unimposing
  - simply a collection of tables - everybody understands tables
- The Structured Query Language (SQL - sequel) a language that (almost) anyone can understand
  - 'English like'
  - Non-procedural - specify what data you want rather than how to retrieve it
- Standard, consistent model of real world application

Advantages of a DBMS

- Data common resource, available to authorised members
- Central control = economy of scale
- Data independence favours flexible applications
- Data sharing implies less redundancy and improved quality
- Supports multiple users concurrently. Each user can benefit immediately from others' changes
- Different types of data in one central location which can be queried through one interface

Disadvantages of a DBMS

- Expensive and complex, training required, resource hungry
- May include extra services that are not required
- Not suitable for simple, single-user applications
- Operating costs may be high

DBMS versus Files

- Database ..
  - Central backup and recovery
  - Secure access, local and remote
  - File storage in controlled area, no files on users disks
  - Hides complexity of file system from users
  - Applies central rules base – constraints
  - Inbuilt query and reporting tools

  - Difficult to scale a single file (millions of rows?)
  - Data of interest to one piece of software has to be copied (perhaps in a different file format) for another application.
  - Difficulties in maintaining copies consistent, current etc.

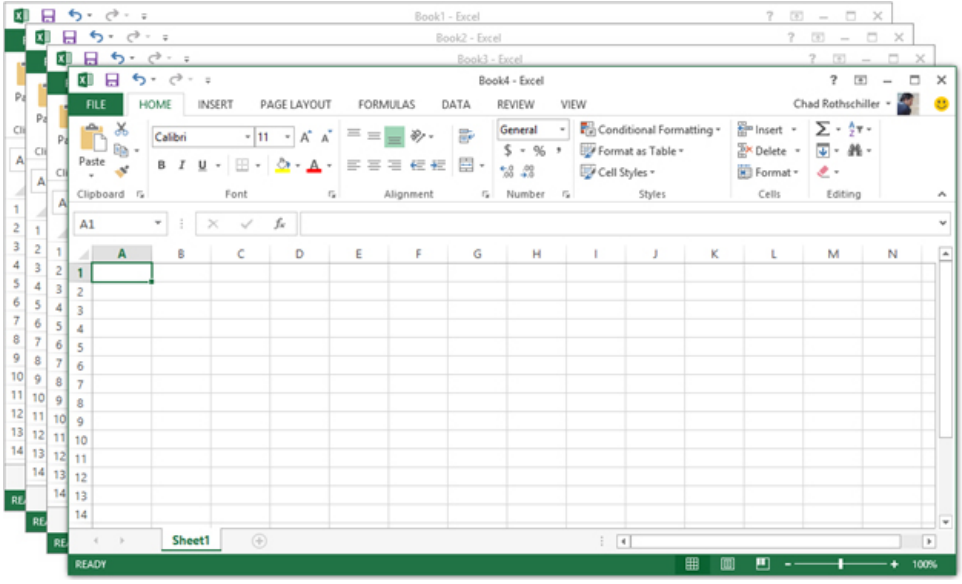
Database versus Spreadsheet

- In Assignment 1 you submitted
  - A spreadsheet
  - Data into a database

So I've ended up with 200+ separate spreadsheets which I can link by merging

Not easy as you all used different column names and orders

No one can see anyone else's



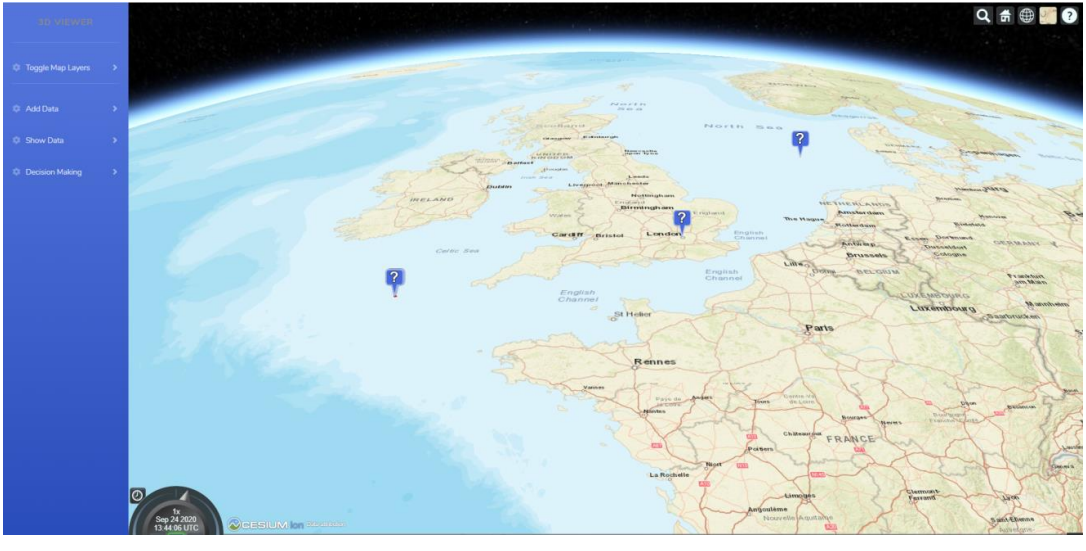
Database versus Spreadsheet

- I've also ended up with ONE table in my database with all the data and with the same column names!

You could also immediately see everyone's data on our 3D map

- But you don't have rights to change some else's data

I can easily check who entered which rows of data



Database versus Spreadsheet

- The spreadsheets are all located on your hard drives ..
  - Easy to lose them?!
- The database is a backed up central server

Oracle Exadata Database

- two 24-core x86 processors
- 384 GB of memory (expandable up to 1.5 TB)

This Lesson

- We looked at how to store and manage data in a centralised way to enable easier sharing and backup of data and to ensure that only people with permission can see data

Next Lesson

- Now that we've seen the benefits of a database, the next step is to understand how it works a little bit more in depth..
  - i.e. what are the components that make up a database

Spatial Data Management

Components of a Database

This Lesson

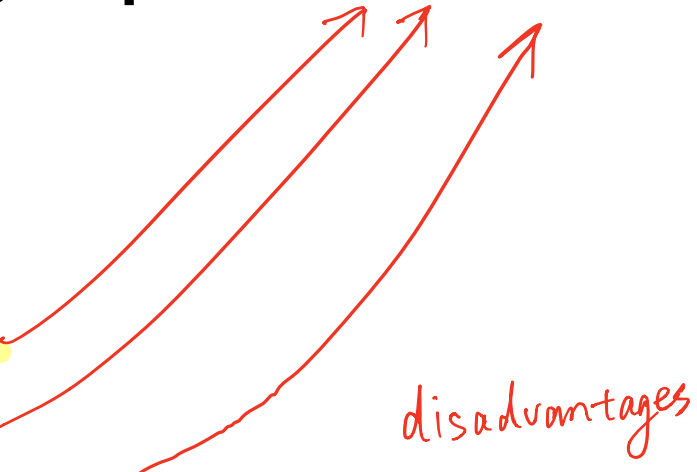
- Now that we've seen the benefits of a database, the next step is to understand how it works a little bit more in depth..
  - i.e. what are the components that make up a database

In the beginning- spreadsheets

- Data in different columns and rows
- Can store up to **1,048,576** rows by **16,384 columns** in excel
- Each column max 255 characters long
- Relatively easy to use

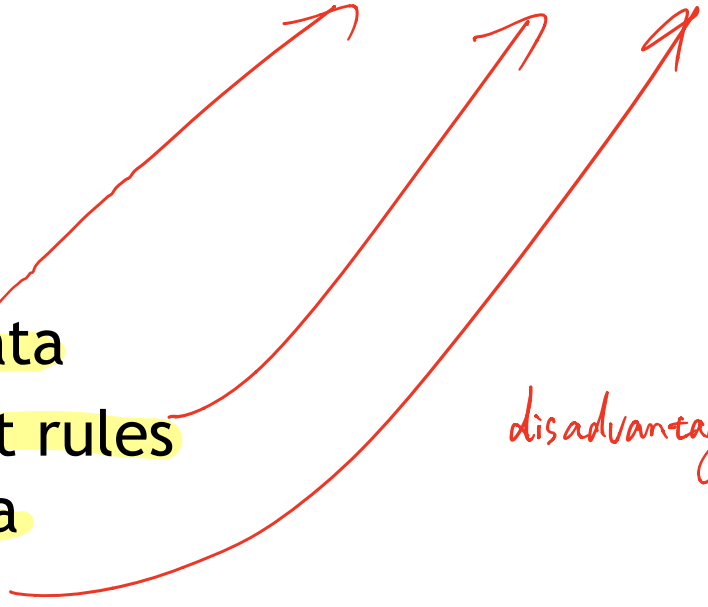
In the beginning- spreadsheets

- But …
  - Lots of duplicate data
  - Not so easy to share data
  - Difficult to edit as can have many items of data in a cell



In the beginning- spreadsheets

- But …
  - Easy to lose data
  - Lack of in-built rules gives poor data quality



disadvantages

Database Rules ..

- One fact, one place
  - Duplicate data costs money –e.g. storage
  - You also don't know which is the 'correct' version of the data if it is changed in one place and not another
- One item of data in one cell
  - So that you can change just that item without having to change something else

Modelling the World – DB Style

- To avoid duplicate data, a database splits the world as a lot of smaller interconnected objects about which we hold some data
- We can then join the information back together as needed

Database Components

only hold information for one entity

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

columns are fixed type

can have information about many things

spreadsheet

cells can interact with each others (it is alive)

consist of the figures in spreadsheet

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

Relation of database = table in that database

Database Components

- Terminology – Domains/Data Types

Domain Type	Postgres Name
String	character varying
Date	date
Number	integer
Spatial data (location)	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 Components

- 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 Components

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

    https://www.youtube.com/watch?v=JwZPaV2qqqg&list=PLCiOXwirraUDUYF_qDYcZV8Hce8dsE_Ho&index=166

Database Components

- 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)

Database Components

- Constraints – the rules that make your data better quality
  - Primary and foreign keys
  - Unique constraints
  - Check constraints
  - Tuple constraints —— programming level

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

This Lesson

- We've looked at the components of a database
  - Schemas, tables, and constraints
  - In particular primary and foreign keys used to link datasets together

Next Lesson

- In the next lesson, we'll start to look at how a database works in practice
- In particular we'll look at SQL – Structured Query Language, which helps us
  - Put data in the database
  - Change (edit, update, delete) the data
  - Use it for decision making

Spatial Data Management

What is SQL?

This Lesson

- In this lesson, we'll start to look at how a database works in practice

- In particular we'll look at SQL – Structured Query Language, which helps us
  - Put data in the database
  - Change (edit, update, delete) the data
  - Use it for decision making

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
  1. Data Definition Language (DDL) which allows users to create and modify the tables in the databases
  2. Data Manipulation Language (DML) which allows users to add, edit or delete data from the tables that have been created
  3. 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

most commonly used command in SQL

- 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.

This Lesson

- In this lesson, we've started to look at how a database works in practice
- In particular we've seen SQL which helps us
  - Put data in the database
  - Change (edit, update, delete) the data
  - Use it for decision making

Next Lesson

- The next thing to do is to create the structure of the database ourselves
  - i.e. decide what tables and columns we want
- We will do this in multiple stages
  - DDL - create the tables (next lesson)
  - DDL - create the constraints (later lesson)

Spatial Data Management

Data Definition Language

This Lesson

- In this lesson we'll seen how to create the tables in a database
  - From scratch using Data Definition Language SQL

Before you start ..

- You need to know the structure of the tables you want to create
  - What are the tables called
  - What columns do they have
  - What type of data is in these columns
- This information comes from
  - Looking at existing data (1)
  - Your own design (2) e.g. a questionnaire

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

first command :

```
DROP SCHEMA IF EXISTS ucfscde cascade;
```

(NB – the cascade is important – if the schema has data it won't be deleted without cascade)

DDL – Creating a Schema

- Using DROP SCHEMA deletes all the data !

- So if you are unsure, rename it instead:

if dont want to delete, then use :

```
alter schema ucfscde rename to
ucfscde_backup_23Nov2018;
```

DDL – Creating a Schema

- Now create the new schema:

create command;

```
CREATE SCHEMA ucfscde;
```

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, didn't show this but you could include 'cardinality of the attributes' as an additional piece of information
    - (same concept as cardinality of relationships – should we have a value for an attribute or can it be left blank or 'null')
  - 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!

Example for NOT NULL

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

If the table might already exist you can use DROP TABLE to delete it (see later on)

Field Types

- Some PostGIS Specific Terminology

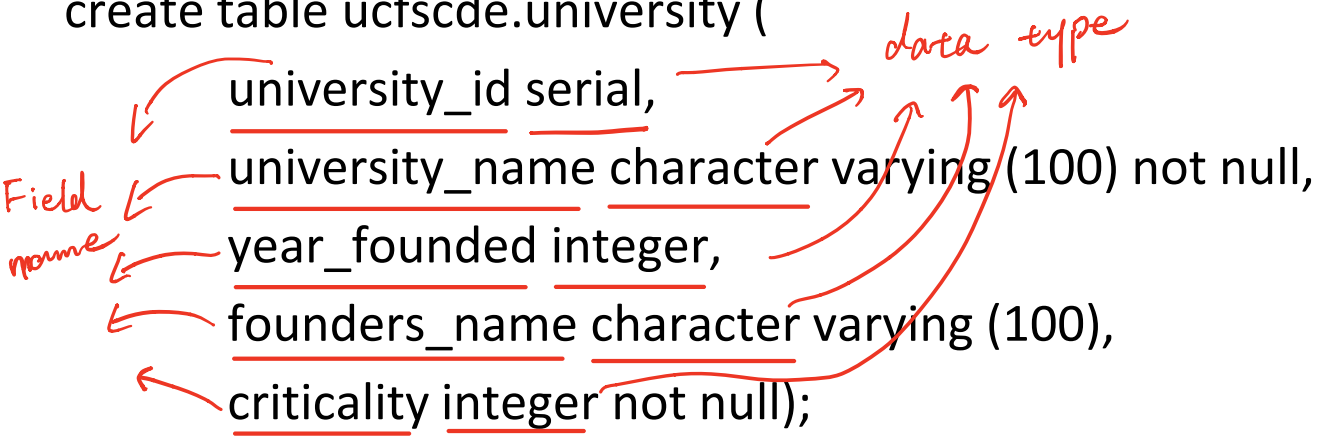
General Domain Type	PostGIS Terminology
String	character varying (length) varchar (length)
Date	date
Number	integer numeric (precision, scale)
Spatial Data	geometry
(automatically increasing number used for ID values)	serial

DDL

- Creating a table - example

```
drop table if exists ucfscde.university;

create table ucfscde.university (
    university_id serial,
    university_name character varying (100) not null,
    year_founded integer,
    founders_name character varying (100),
    criticality integer not null);
```



DDL

- Creating a Table - Example

```
drop table if exists ucfscde.buildings;

create table ucfscde.buildings (
   building_id serial,
   building_name character varying (200) not null,
   university_id integer not null,
   criticality integer not null
);
```

DDL

- Creating a Table - Example

```
drop table if exists ucfscde.rooms;
create table ucfscde.rooms (
   room_id serial,
   floor integer not null,
   building_id integer,
   room_use character varying(50) not null,
   room_number character varying (50) not null,
   criticality integer not null
);
```

DDL

```
drop table if exists ucfscde.window_costs;
create table ucfscde.window_costs (
        window_cost_id serial not null,
        window_type character varying(100) not null,
        window_cost_per_m2 numeric(7,2) precision not null
);
```

Note: numeric(7,2) = 7 digits in total, 2 of which after the decimal point, so 99999.99 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!

DDL

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

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

DDL

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

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

This Lesson

- In this lesson we've seen how to create the tables in a database
  - From scratch using Data Definition Language SQL
  - Creating the table in a specific schema
  - Using different domain (column) types

Next Lesson

- In the next lesson we'll see how to create the rules in a database that ensure that data has good quality
- We'll do this from scratch again, using Data Definition Language SQL

Spatial Data Management

Constraints

This Lesson

- In this lesson we'll see how to create the rules in a database that ensure that data has good quality
  - No missing information
  - No duplicate information
  - Information is fit for what you need
- We'll do this from scratch using Data Definition Language SQL

Before you start ..

- You need to know the rules that you want to apply
  - Which column/columns make each row in each table unique?
  - How are the different tables linked (*values, location)*
  - What other rules should be enforced to guarantee better data?
- This information comes from
  - Looking at existing data

Before you start ..

- The rules come from
  - Looking at existing data
  - Your own design e.g. a questionnaire
  - Your understanding of what you're trying to store information about
    - e.g. you might need a rule that says a condition survey is invalid after 6 months
    - Usually on larger systems a team will decide the rules

DDL - Integrity Constraints

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

integrity

DDL - Integrity Constraints

- Constraints - Primary Keys

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

  alter table ucfscde.buildings add constraint buildings_pk primary key (building_id);

  alter table ucfscde.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 ucfscde.buildings
        add constraint building_university_fk
        foreign key(university_id) references
        ucfscde.university(university_id);
```

DDL – Integrity Constraints

- Creation order is important
  - You must create the primary key first then reference it from the foreign key
  - DO ALL THE PRIMARY KEYS FIRST

```
alter table ucfscde.temperature_values
   add constraint temperature_values_temperature_fk
   foreign key(temperature_sensor_id) references
   ucfscde.temperature_sensors(sensor_id);
```

DDL - Integrity Constraints

- Constraints - Domain Constraints

  ALTER TABLE <TABLENAME>

  ADD CONSTRAINT <CONSTRAINTNAME_CHK>

  CHECK (<TEST>)

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

Check Constraints versus Lookup Tables

- Check constraints – you can't change the values once the constraint is created
  - Use only when you are sure you have all the values
- Lookup tables
  - Have a primary/foreign key relationship with the main table
  - New values can be added if necessary

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

```
alter table ucfscde.temperature_values
    add constraint temperature_values_unique
    unique(temperature_sensor_id,
reading_timestamp);
```

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

DDL - Integrity Constraints

```
CREATE OR REPLACE FUNCTION ucfscde.update_sensor_room_id() RETURNS trigger AS
$BODY$
    DECLARE
    originalroomid integer;
    newroomid integer;
   BEGIN
    select room_id into originalroomid from ucfscde.temperature_sensor where
    sensor_id=NEW.sensor_id;
    select room_id into newroomid from ucfscde.rooms a where st_contains(a.location,
    NEW.location);
    raise 'Original ID: %', originalroomid;
    raise 'New Length: %', newroomid;
    IF NEW.room_id <> originallength THEN
          UPDATE ucfscde.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 update_sensor_room
    AFTER INSERT OR UPDATE OF location ON ucfscde.temperature_sensor
    FOR EACH ROW
    EXECUTE PROCEDURE ucfscde.update_sensor_room_id ();
```

- The trigger then runs when data is inserted/updated in the table
  - NB- you don't specify the schema name, the schema name is taken from the schema of the table the trigger runs on

This Lesson

- In this lesson we've seen how to create the rules in a database that ensure that data has good quality
    - No missing information
    - No duplicate information
    - Information is fit for what you need
- We've done this from scratch using Data Definition Language SQL

Next Lesson

- So far, we've mentioned primary and foreign keys as the way in which the different tables in a database are linked
- In the next lesson, we'll look specifically at why we need a UNIQUE constraint on every table, to avoid data duplication

Spatial Data Management

Why an ID is not a unique primary key

This Lesson

- So far, we've mentioned primary and foreign keys as the way in which the different tables in a database are linked

- In this lesson, we'll look specifically at why we need a UNIQUE constraint on every table, to avoid data duplication

First Rule of Data Management

- One fact, one place

- Don't have copies of data because:
  - They takes extra space on your machine
  - You can end up changing the data in one copy and not in another
    - Users don't know which one is the right version!

First Rule of Data Management

- One fact, one place

- If your data has duplicate information, split it into smaller tables and link them using primary/foreign keys

Primary Keys

- The PRIMARY KEY constraint uniquely identifies each record in a table
- Primary keys must contain UNIQUE values, and cannot contain NULL values
- A table can have only ONE primary key; and in the table, this primary key can consist of single or multiple columns (fields)
- (from: https://www.w3schools.com/sql/sql_primarykey.ASP)

Primary and Foreign Keys – why we need unique constraints

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?

unique constraints make PK/FK unique

Primary and Foreign Keys – why we need unique constraints

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 – why we need unique constraints

ID	Name	Surname	Salary	Date of Birth	Address
100102	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

Why not IDs?

- Lets take another example
  - We have 7 windows in our UCL campus dataset, each having a different ID

	window_id integer	window_cost_id integer	window_type character varying (100)	window_installation_date date	room_id integer	floor integer	criticality integer	location geometry
1	1	2	double glazed	2014-05-23	1	0	1	010F0000A0346...
2	2	2	double glazed	2014-05-23	2	0	4	010F0000A0346...
3	3	2	double glazed	2017-05-23	4	0	1	010F0000A0346...
4	4	1	single glazed	2014-05-23	7	0	1	010F0000A0346...
5	5	1	single glazed	2014-05-23	6	1	1	010F0000A0346...
6	6	1	single glazed	1974-09-23	10	1	1	010F0000A0346...
7	7	1	single glazed	1944-02-02	11	1	4	010F0000A0346...

Why Not IDs

- When you're collecting condition information about the windows you have 3 options
  - Find some sort of ID tag (QR number?) on the window to tell you which one it is
    - but UCL is old and we don't have these!
    - also the window_id is arbitrary and could change if someone deletes and then re-inserts the data for the window
  - You could use a combination of building, room and floor but some rooms might have more than one window

Why Not IDs

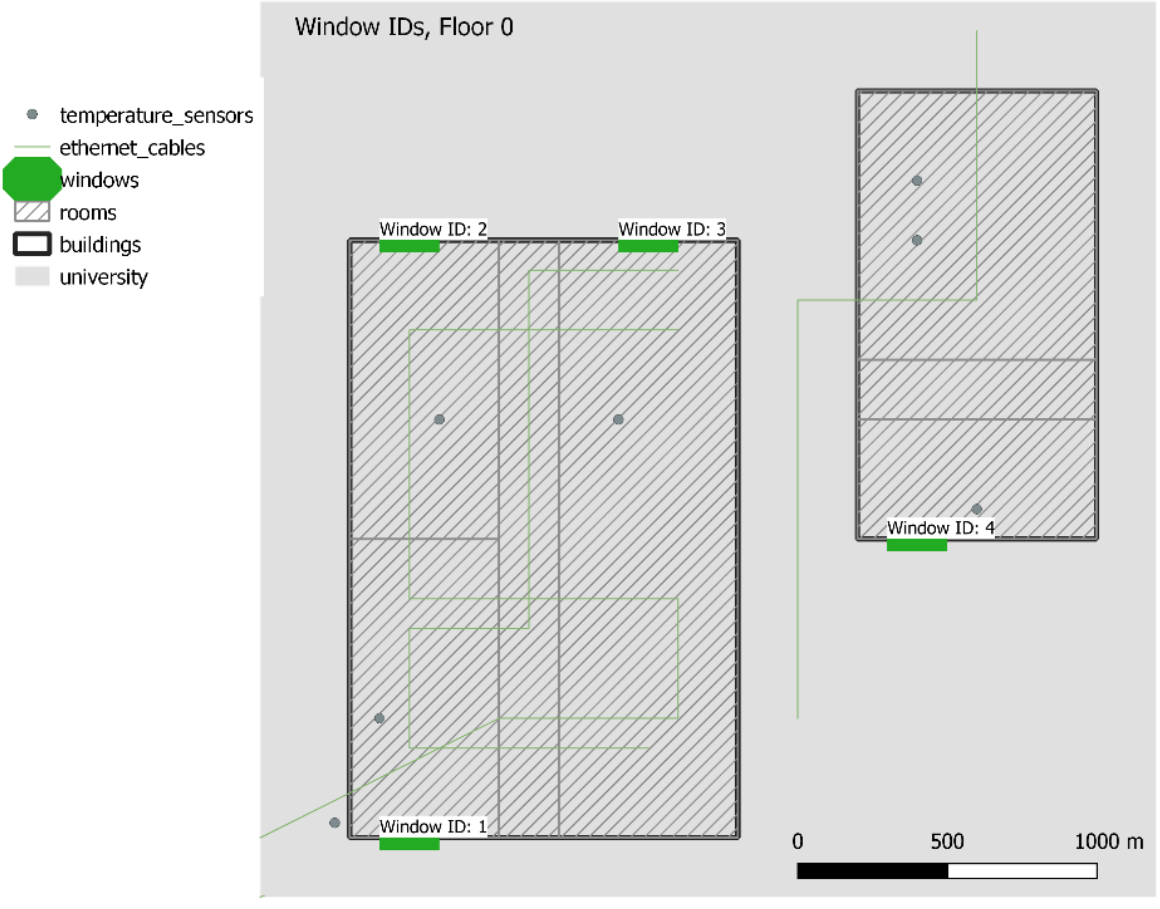
- When you're collecting condition information about the windows you have 3 options

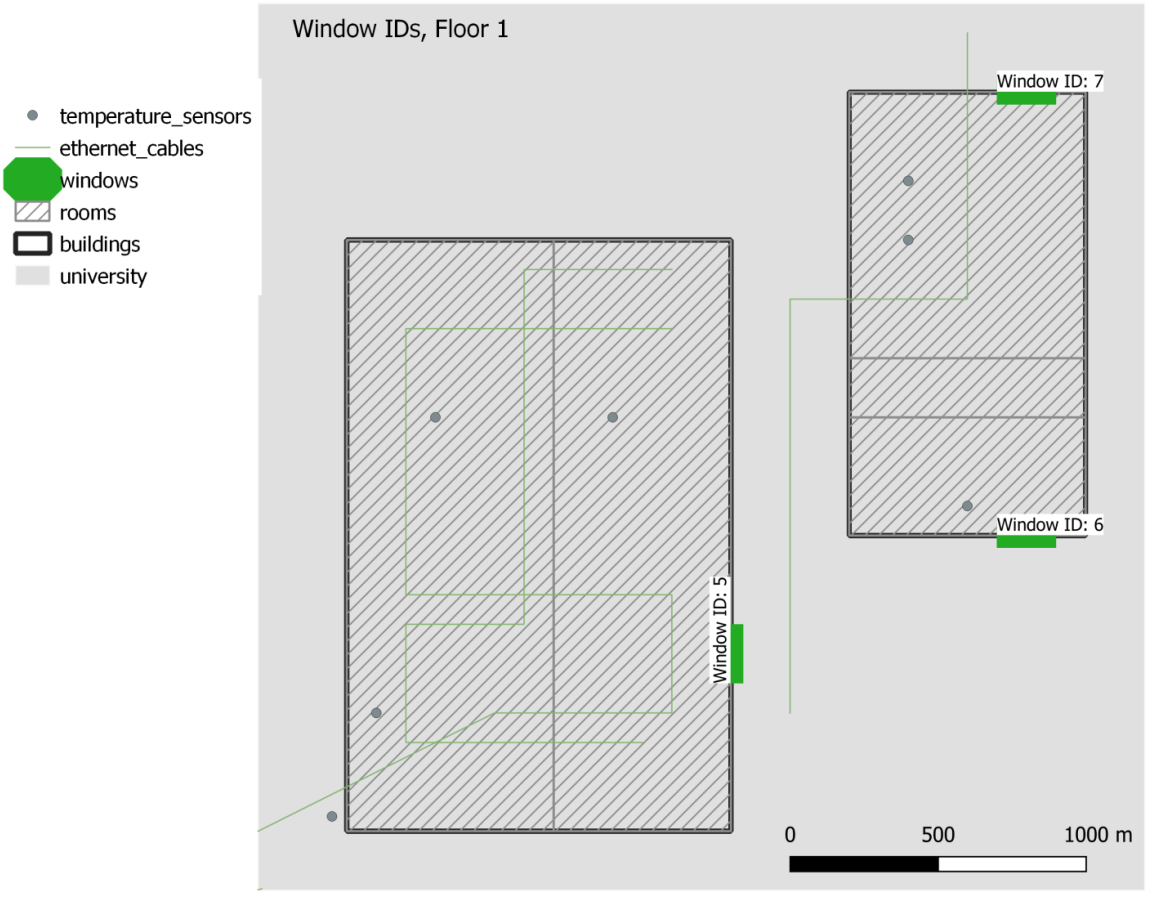
- Best to use the actual unique property of the window – it's location – and tag your condition report with a GPS reading
    - Assumes that you have a device that can give you your location with enough accuracy to link the condition report to only one window

Why Not IDs

- For this module we'll use a map with the Window IDs on it .. This allows us to link the real primary key/unique identifier – the location – to the ID short cut that we have in our database

- In real life maybe use a tablet with a 2D or 3D map – click on the 'asset' on the map and then add its condition.


Window IDs, Floor 0
temperature_sensors
ethernet_cables
windows
rooms
buildings
university
Window ID: 2
Window ID: 3
Window ID: 4
Window ID: 1
0
500
1000 m


Window IDs, Floor 1
temperature_sensors
ethernet_cables
windows
rooms
buildings
university
Window ID: 7
Window ID: 6
Window ID: 5
0
500
1000 m

Why Not IDs

- So to make sure we select the correct row in a table, we need to know which columns of data make that row unique (different from all the others)
  - This is the real primary key for the table
- So why do we bother with ID values then?

Primary and Foreign Keys – why we need unique constraints

- Why bother with IDs then?

Name	Surname	Date of Birth	Address	Start Date	Predicted Graduation Date

Name	Surname	Date of Birth	Address	Exam Name	Exam Date	Grade

Exam Name	Exam Date	Pass Mark

What happens if a student changes their surname and address?

Primary and Foreign Keys – why we need unique constraints

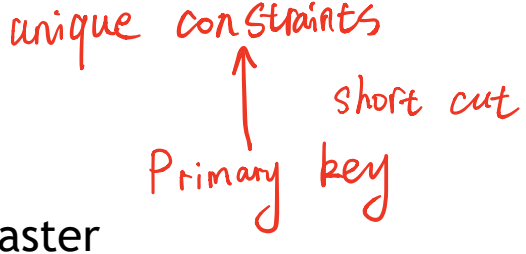
- Why bother with IDs then?

Student ID	Name	Surname	Date of Birth	Address	Start Date	Predicted Graduation Date

Student ID	Exam ID	Grade

Exam ID	Exam Name	Exam Date	Pass Mark

- Only ID values duplicated
  Less risk of inconsistent data
  Less storage space required
  Join queries (see following weeks) are faster



Primary and Foreign Keys – why we need unique constraints

- The process
  1. Create an ID column for every table
     - This makes joining data faster as it is quicker to compare 2 numbers in different ables
  2. Create a UNIQUE constraint for every table that makes sure the real primary key is valid

Primary and Foreign Keys – why we need unique constraints

- Every single table in your database should have
  - A primary key (1)
  - At least one UNIQUE constraint (2)
- This will help to ensure that 'one fact one place' is achieved

This Lesson

- Primary and foreign keys are the way in which the different tables in a database are linked
- They also help to ensure that data is consistent
  - Always use the same values in a lookup table
  - Avoid duplicate data

This Week

- We looked at how to sketch an E-R diagram to make joining more than 2 dataset easier
  - A sketched diagram called an ER diagram can help to find common information
- We started working with databases - what they are and how to create tables and constraints using SQL

Next Week

- So far we've looked at how to create tables that include text (varchar/character varying), number or date information

- Next week we'll look at how to add columns for location information
- And we'll also start to look at how to add data to the database

Spatial Databases

Modelling Spatial Data

Last Week

- We looked at how to sketch an E-R diagram to make joining more than 2 dataset easier
  - A sketched diagram can help to find common information
- We started working with databases - what they are and how to create tables and constraints using SQL

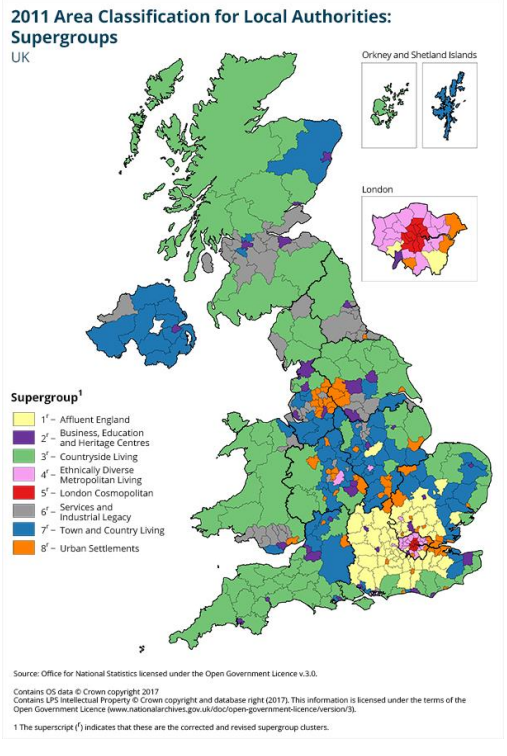
This Week

- So far we've looked at how to create tables that include text (varchar/character varying), number or date information

- This week we'll look at how to add columns for location information
- And we'll also start to look at how to add data to the database

This Lesson

- Spatial data is more complicated than simple text or numbers – e.g. the coastline of Great Britain is made up of millions of x/y locations joined in to form a polygon
- To understand how to model this data in a database, we'll first look at how we represent the world using spatial data



https://www.ons.gov.uk/methodology/geography/geographicalproducts/areaclassifications/2011areaclassifications/maps

Spatial Data

- Spatial data includes ***anything*** that can be modelled using some form of *location* information!
  - This is a particular type of data that is more complex than the numbers, text and date domains that are typically found in a database

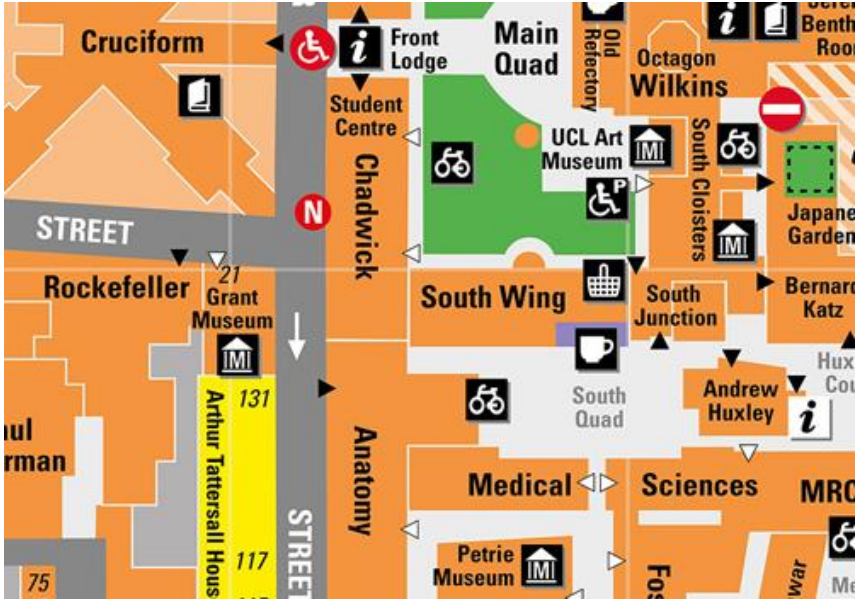
Spatial Data

- In spatial data, objects (entities, assets) can be represented as
  - Points (2D or 3D)
  - Lines (2D or 3D)
  - Polygons (2D or 3D)
  - Polyhedra (2D or 3D)

Think about what something would look like if you were looking from above

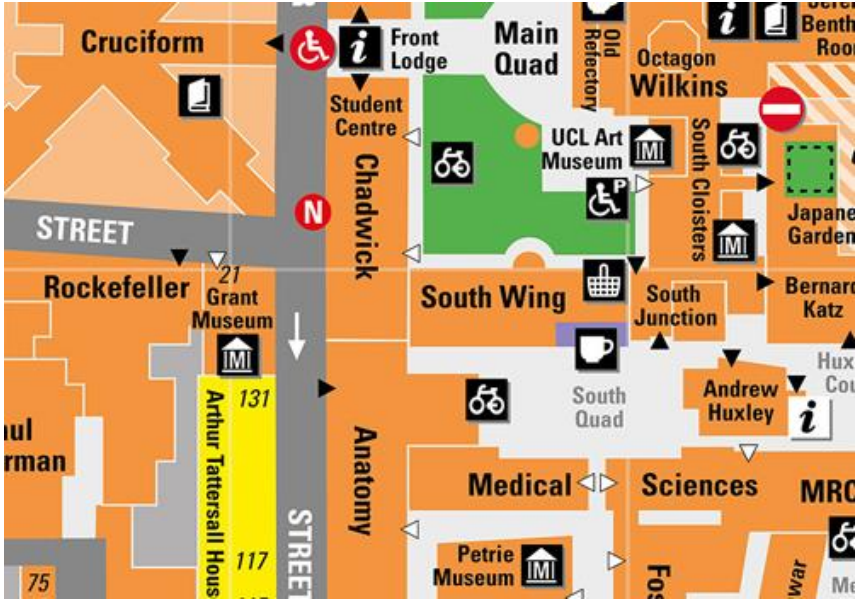
Spatial Data

- You can have multiples too:
  - School with multiple buildings (multiple polygons)



Spatial Data

- You can have multiples too:
  - Building with multiple entrances (multiple points, black arrows)



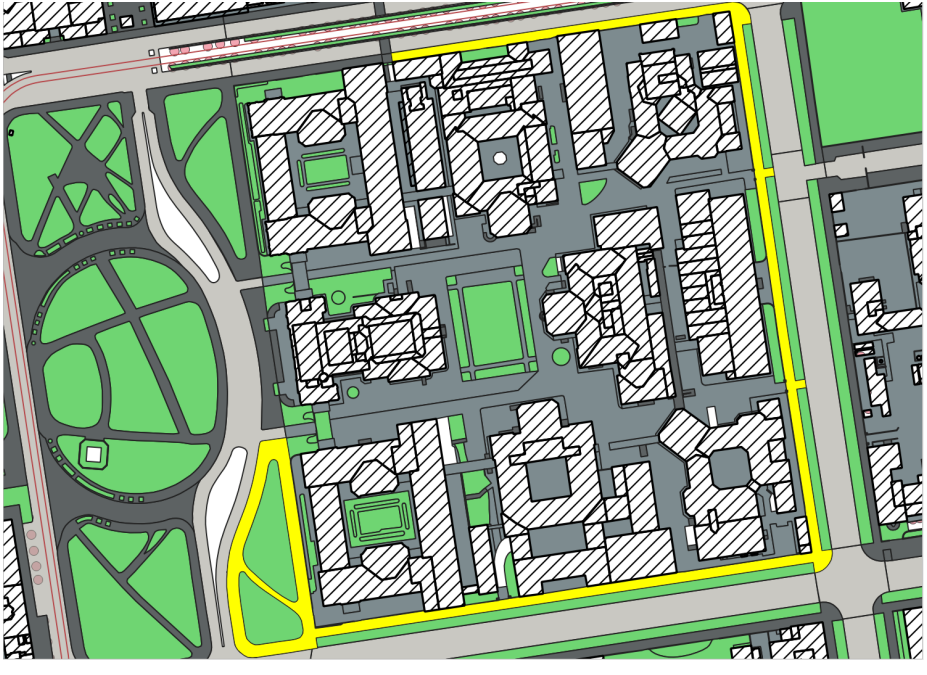
Spatial Data

- And your data can have holes
  - Building with internal courtyard

(data from Politecnico di Milano)

Spatial Data

- And your data can have holes
  - Pavement around green spaces

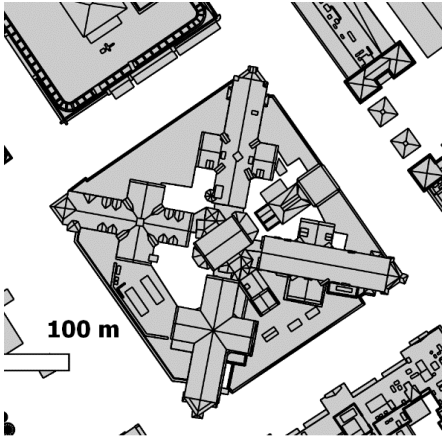


(data from Politecnico di Milano)

Spatial Data

- Can have complex shapes
  - UCL's Cruciform building

openstreetmap.org/search?query=wc1e%206bt#map=19/51.52404/-0.13504

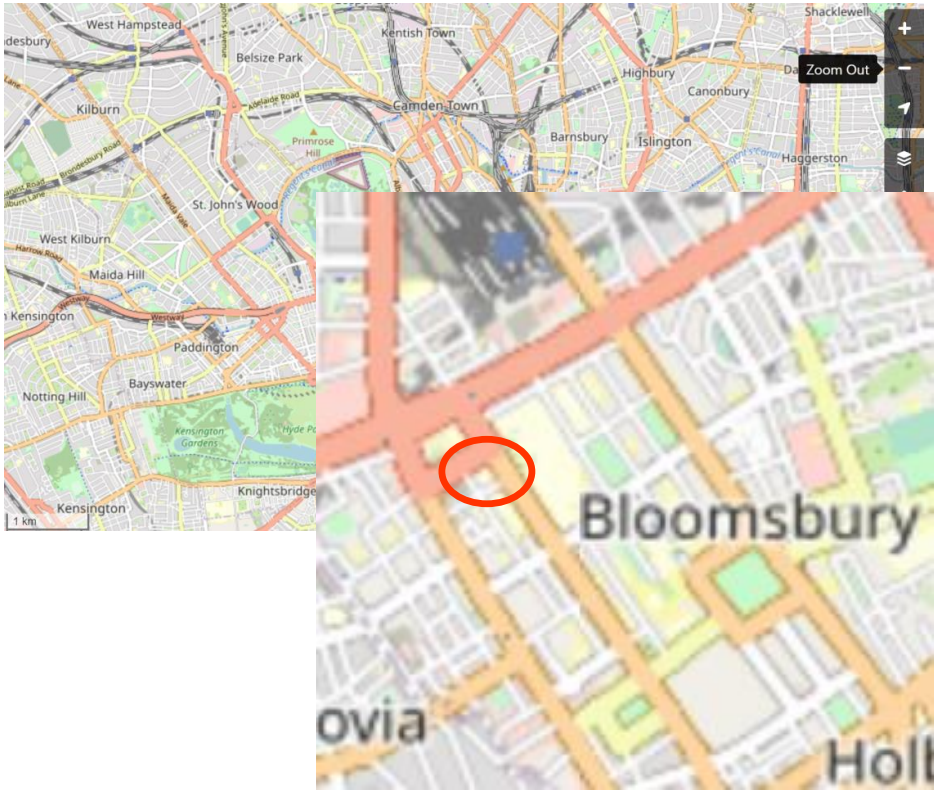


Bluesky data

Bluesky data. 3D view

Spatial Data

- Can change shape depending on how far out you zoom (i.e. when you change map scale)



openstreetmap.org/search?query=wc1e%206bt#map=19/51.52404/-0.13504

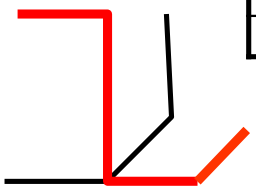
Spatial Data – Basic Representation

- Points represented as two coordinates – x,y

ID	X	Y	Attribute
1	100	**100**	**Well1**
2	400	**400**	**Well2**

- Lines are represented as a list of coordinate pairs

ID	X1	Y1	X2	Y2	X3	Y3	x4	Y4	X5	Y5	Attribute
1	125	150	325	150	450	375	450	500			**Road 1**
2	135	500	335	500	335	150	500	150	600	250	**Road 2**



How do you model lines with more than 5 nodes?

Spatial Data – Basic Representation

- Polygons are also represented as a series of x, y points – but the first and last point must be the same to close the loop

ID	X1	Y1	X2	Y2	X3	Y3	x4	Y4	X5	Y5	Attribute
1	125	150	300	150	300	450	125	450	125	150	**House 1**
2	225	50	450	50	450	200	225	200	225	50	**House 2**

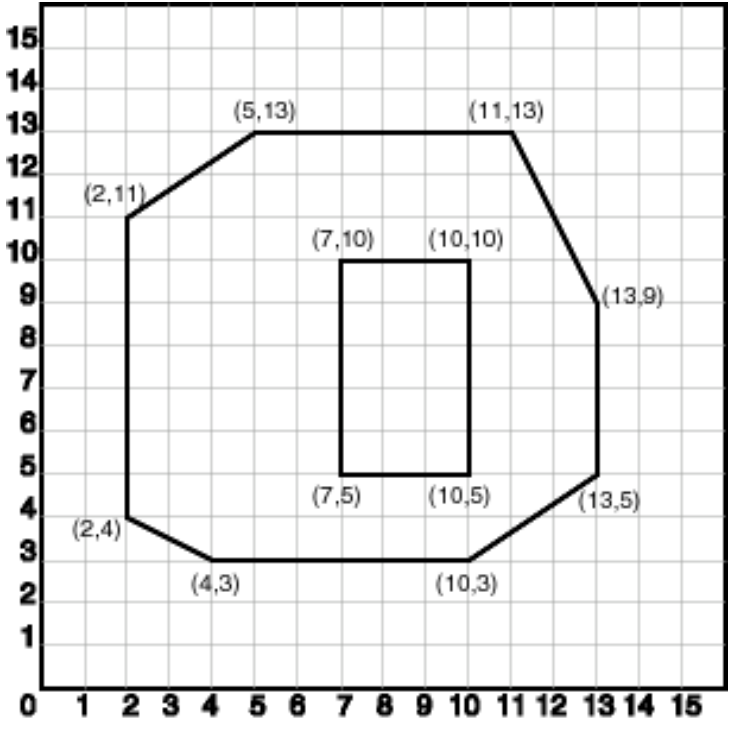
Databases – Storing Spatial Data

- Knowing the x,y values is not enough!
- What do the following numbers represent? Draw as many possible representations of these numbers (single points, lines, polygons, combinations of these etc).

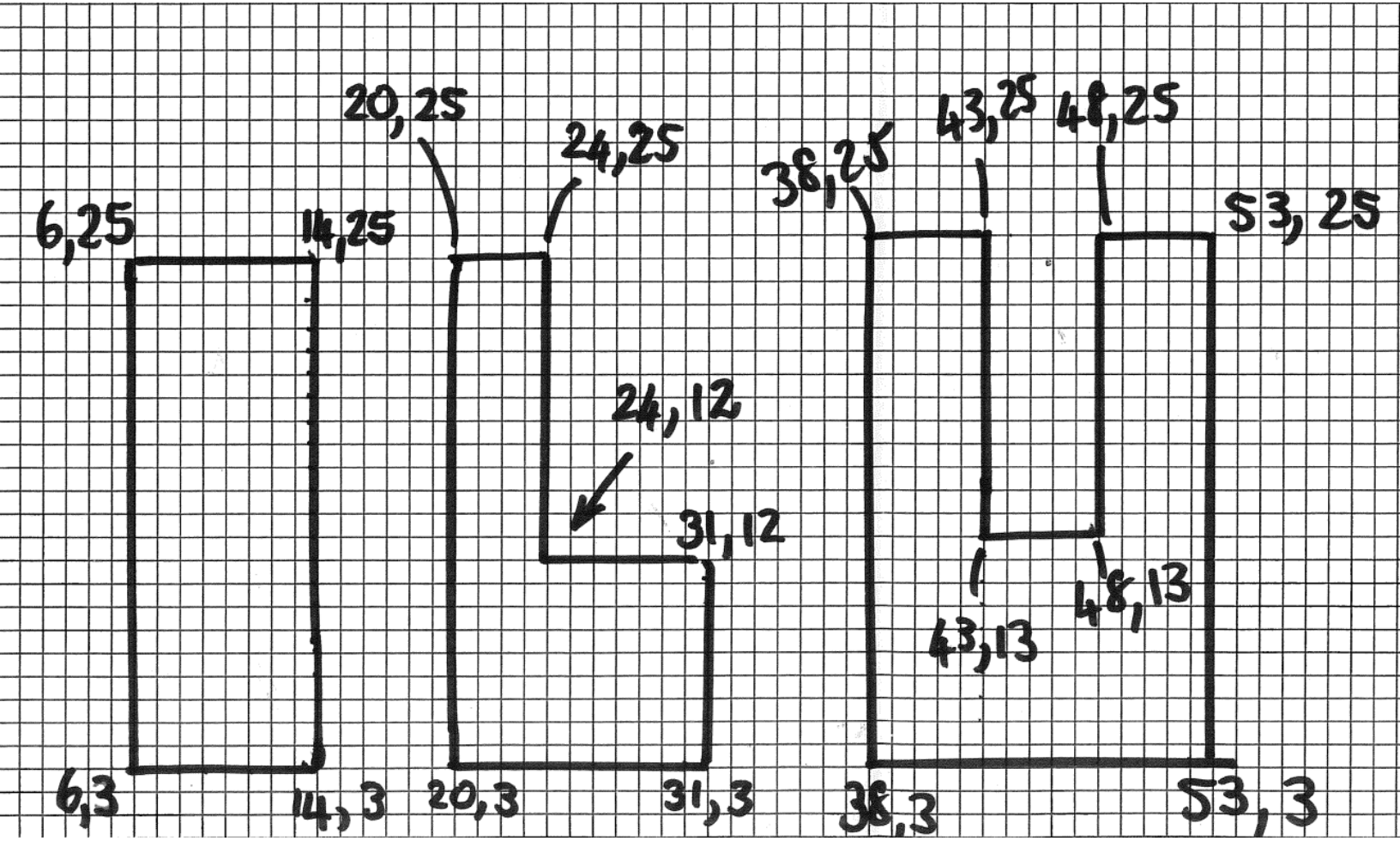
2,4, 4,3, 10,3, 13,5, 13,9, 11,13, 5,13, 2,11, 2,4, 7,5, 7,10, 10,10, 10,5, 7,5

Polygon with Hole

Taken from:
http://download.oracle.com/docs/cd/B19306_01/appdev.102/b14255/sdo_objrelschema.htm#i1004087



Modelling more complex objects



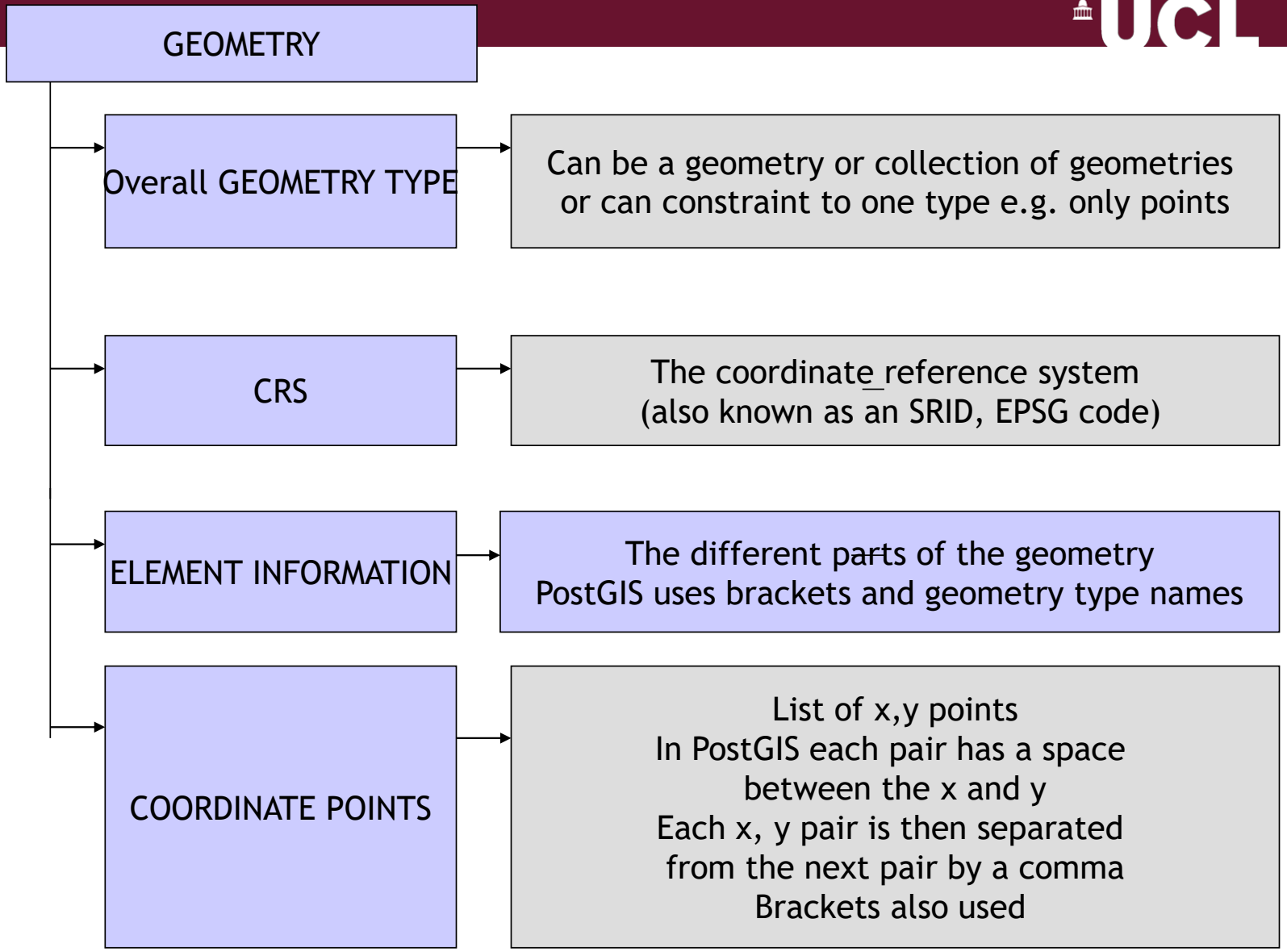
Modelling more complex objects

- Option 1
  - Keep adding columns
    - But you could have 1000s of coordinates for the GB boundary!
    - Also could have lots of empty space if you have another row with e.g. the Isle of Wight

ID	X1	Y1	X2	Y2	X3	Y3	x4	Y4	X5	Y5	X6	Y6	X7	Y7	X8	Y8	Attribute
1	125	150	300	150	300	450	125	450	125	150							House 1
2	225	50	450	50	450	200	225	200	225	50							House 2

Modelling more complex objects

- Option 2
  - Pack them into a list (an ‘array’)
    - Can hold 33.5 million vertices (x/y pairs) in practice
      - http://www.danbaston.com/posts/2016/11/28/what-is-the-maximum-size-of-a-postgis-geometry.html
  - Also store the other information
    - Is it a point, line, polygon, polyhedron
    - Any multiple parts or holes (‘elements’)
    - Where is it located on the globe – i.e. what coordinate reference system CRS (SRID) is used (see separate lesson)


GEOMETRY
Overall GEOMETRY TYPE
Can be a geometry or collection of geometries
or can constraint to one type e.g. only points
CRS
The coordinate reference system
(also known as an SRID, EPSG code)
ELEMENT INFORMATION
The different parts of the geometry
PostGIS uses brackets and geometry type names
COORDINATE POINTS
List of x,y points
In PostGIS each pair has a space
between the x and y
Each x, y pair is then separated
from the next pair by a comma
Brackets also used

Modelling the Geometry Type

- Option 1 we could have columns for the information we need

Building Name	Building Number	Date Built	Geometry Type	CRS/ SRID	Element Information	Coordinate Points
Character varying (string)	Integer	Date	Character varying (‘point’, ‘line’, ‘polygon’, ’multi-point’, ‘multi-line’ etc)	Integer	Array	Array

Modelling more complex objects

- Option 2
  - Use an ‘object relational’ approach
  - use a special data type (domain) to store all the required information
  - this is called a ‘geometry’ data type

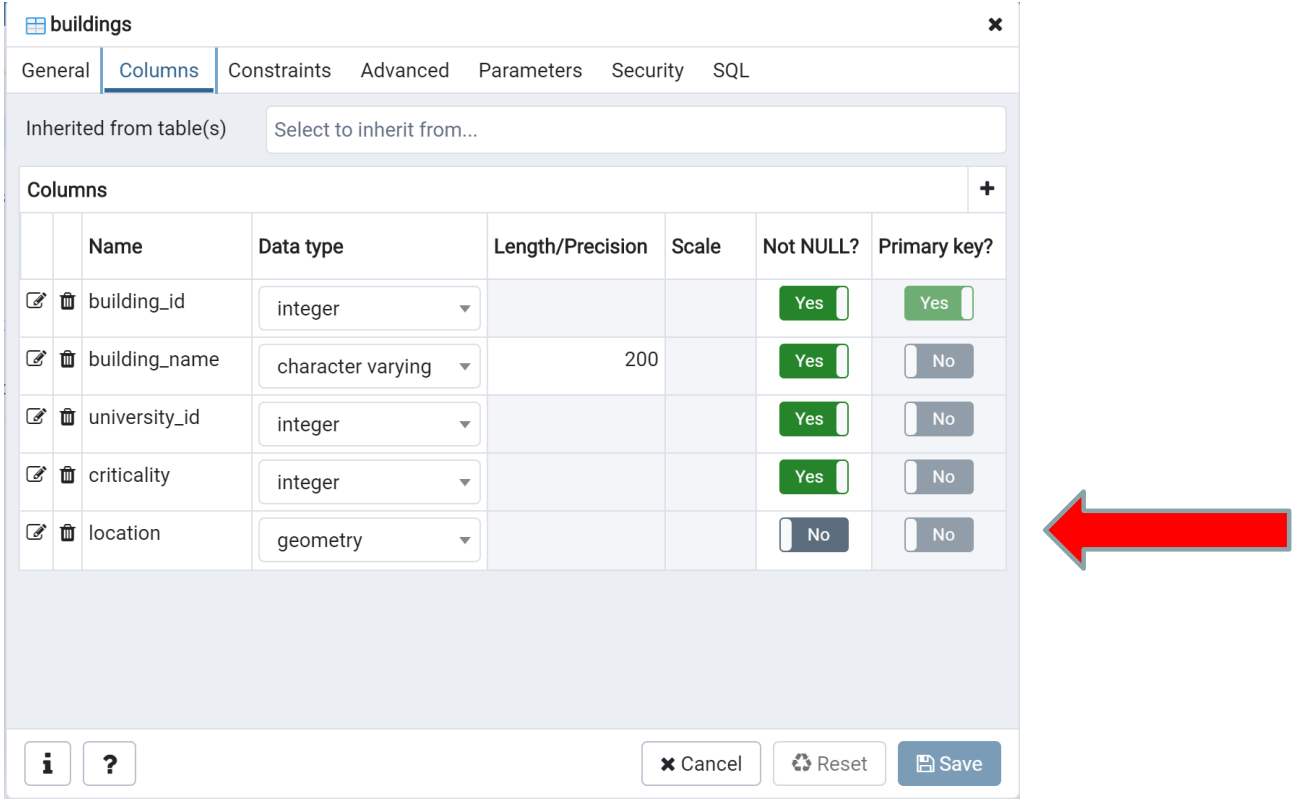
Modelling Geometry

- Object-Relational Approach
  - Geometry is just another data type

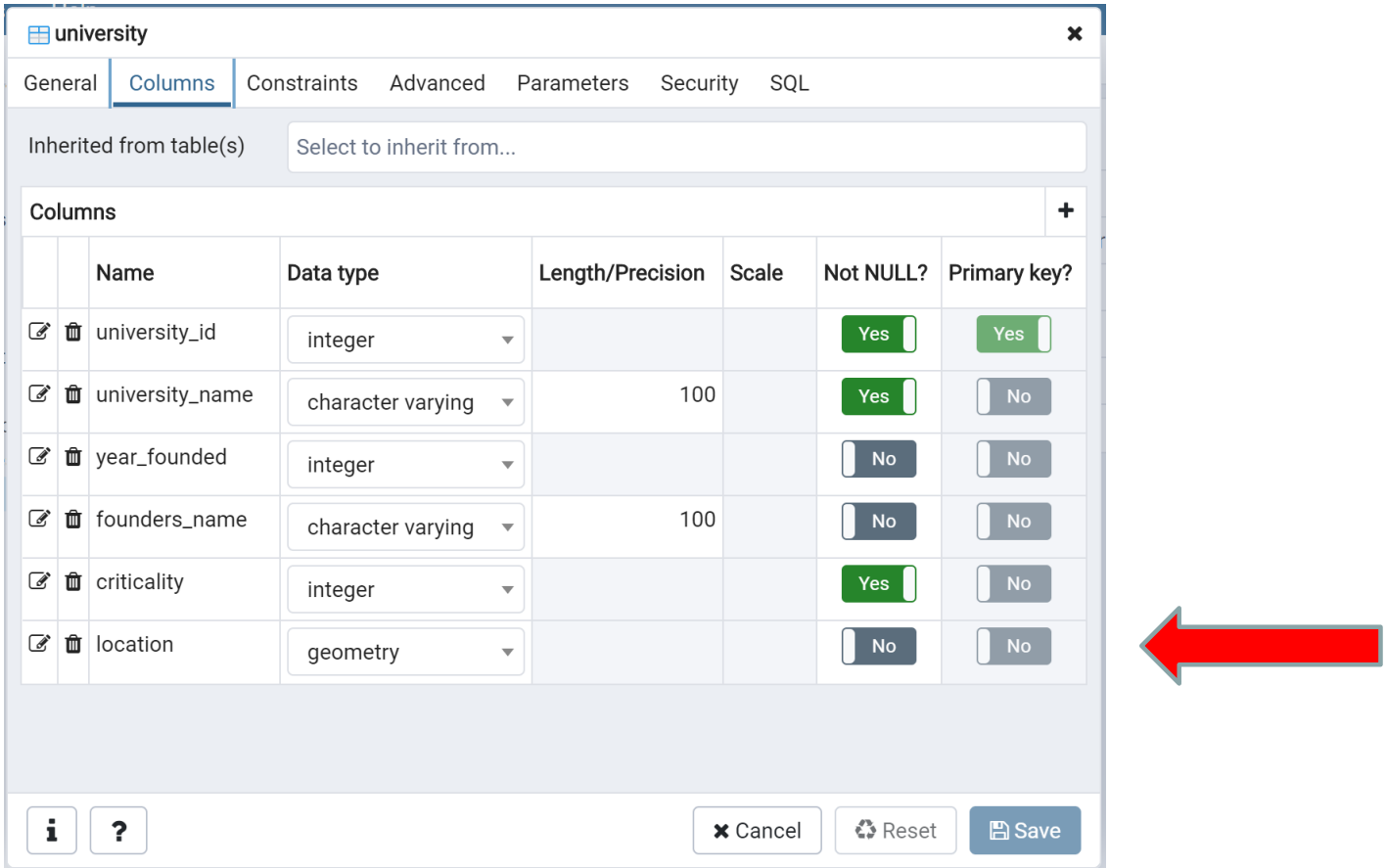
Building Name	Building Number	Date Built	Geometry
Character varying (string)	Integer	Date	<< all the geometry information stored in one column as a single object >>

- (Some additional information usually stored separately)

Geometry is a data type



Geometry is a data type



This Lesson

- We looked at how spatial data is modelled – we need
  - The list of coordinates
  - What ‘shape’ these coordinates represent
    - Point
    - Line
    - Polygon
    - Polyhedron
  - Holes?
  - 2D or 3D?
- In our database, spatial data is another data type

Next Lesson

- In the next lesson, we'll look at georeferencing – even if you know the shape, how do you know where it is on the globe?

Spatial Databases

Georeferencing

This Lesson

- We will look at georeferencing and the different types of coordinate systems you can have
  - Local
  - National
  - World
- We will also look at EPSG codes, which are used to tie the coordinates to specific countries

Spatial Data

- Spatial data includes ***anything*** that can be modelled using some form of *location* information!
  - i.e. where something is, referenced to a shared framework (could be a coordinate system, a map of London Boroughs, countries of the world, UK counties and many more)
- This referencing is called *geo-referencing*

Georeferencing – a reminder

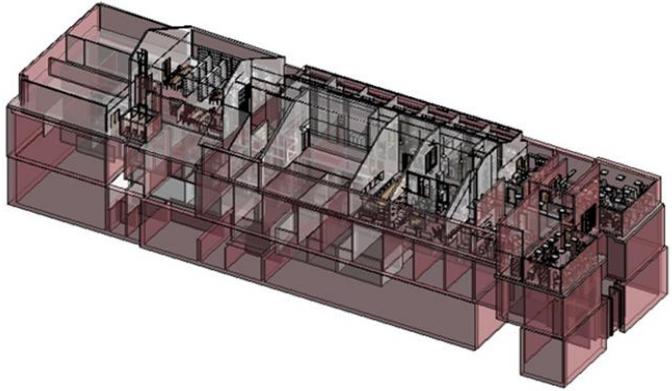
- Can be direct:
  - E.g. a map that shows a building or another object, x/y coordinates, GPS coordinates
- Or indirect
  - For example, a Post Code or a Street Address is an indirect geo-reference that can be used to link non-spatial data to a position on the map. A PDF file containing the specification of a water pipe can be linked to the location of that pipe.

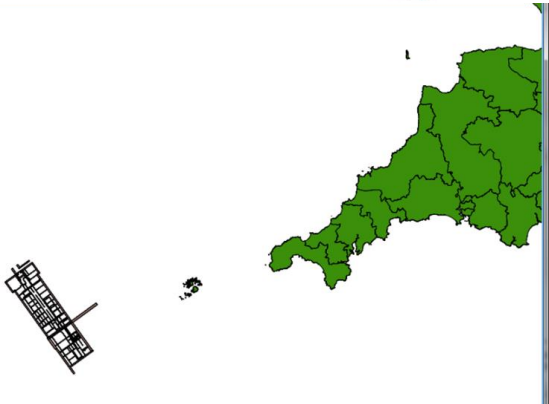
Direct - Coordinate Systems

- Direct referencing works by mapping objects using their real coordinates (e.g. the coordinates that a GPS captures)
- Depending on where you are in the world, and what system you are using these coordinates may be referenced to different ‘origin’ points ..

Direct – Local Coordinate Systems

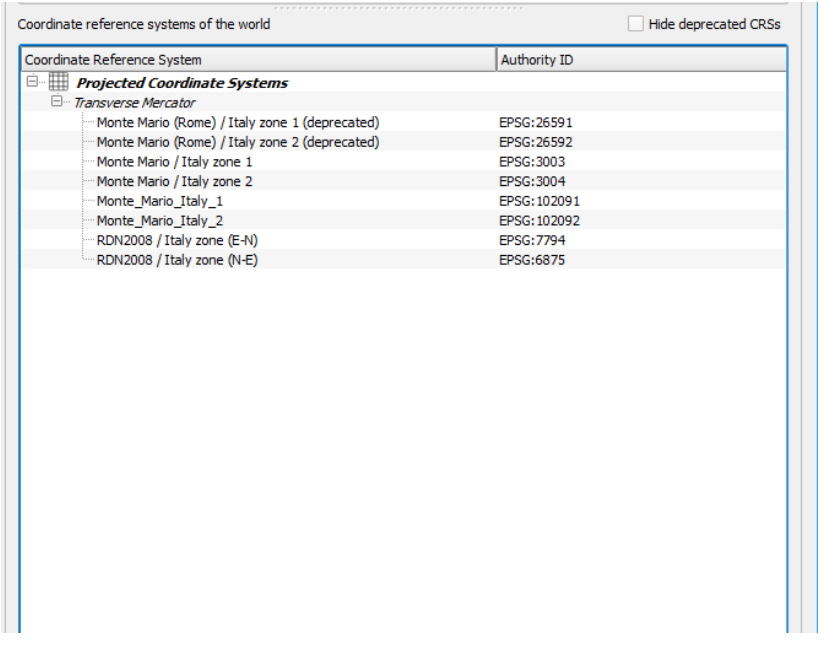
- Used in CAD/BIM
- Have a local reference point as the 0,0 point
  - Usually the edge of a construction site
- All distances and angles measured from this local reference point
- Also Cartesian (flat surface)





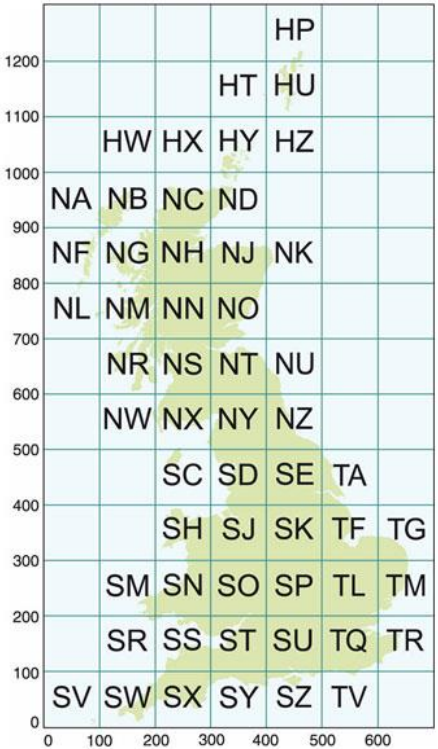
Direct – National Coordinate Systems (country level)

- National coordinate systems are usually created by an organisation such as a National Mapping Agency
- Some countries have more than one



Direct – National Coordinate Systems

- In Great Britain, our mapping system uses “British National Grid” which has its 0,0 of the south west coast



- Cartesian system

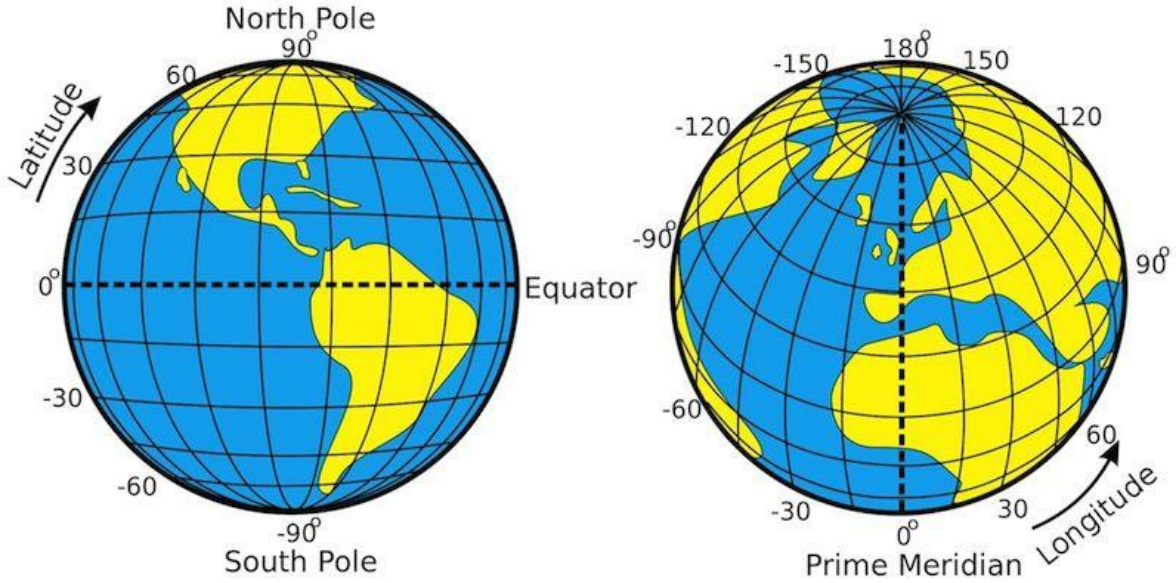
https://timetrail.warwickshire.gov.uk/assets/exhibition/Image/Toolkit/gb-map.jpg

Direct – Global Coordinate Systems

- As satellite systems such as GPS don't only map Great Britain, they use a reference system that covers the world
  - Coordinates are latitude/longitude
    - Longitude ranges from 0 at the Prime Meridian passing through Greenwich, England, to +180 toward the east and 0 to -180 toward the west.
    - Latitude ranges from 0 at the equator to +90 at the North Pole and 0 to -90 at the South Pole. For example, Denver's position shown in the figure is -104.9 degrees longitude (west) and +39.8 degrees latitude (north).

http://www.innovativegis.com/basis/pfprimer/topic7/topic7.html

Direct – Global Coordinate Systems



https://thumbs-prod.si-cdn.com/0rQSHAWkucV1O0dmyJVN8Ml0sS4=/800x600/filters:no_upscale()/https://public-media.smithsonianmag.com/filer/5c/ea/5cea567c-050b-432a-834f-fc94dcb1b49e/coordinates.jpg

Coordinate Reference Systems – Standard Codes

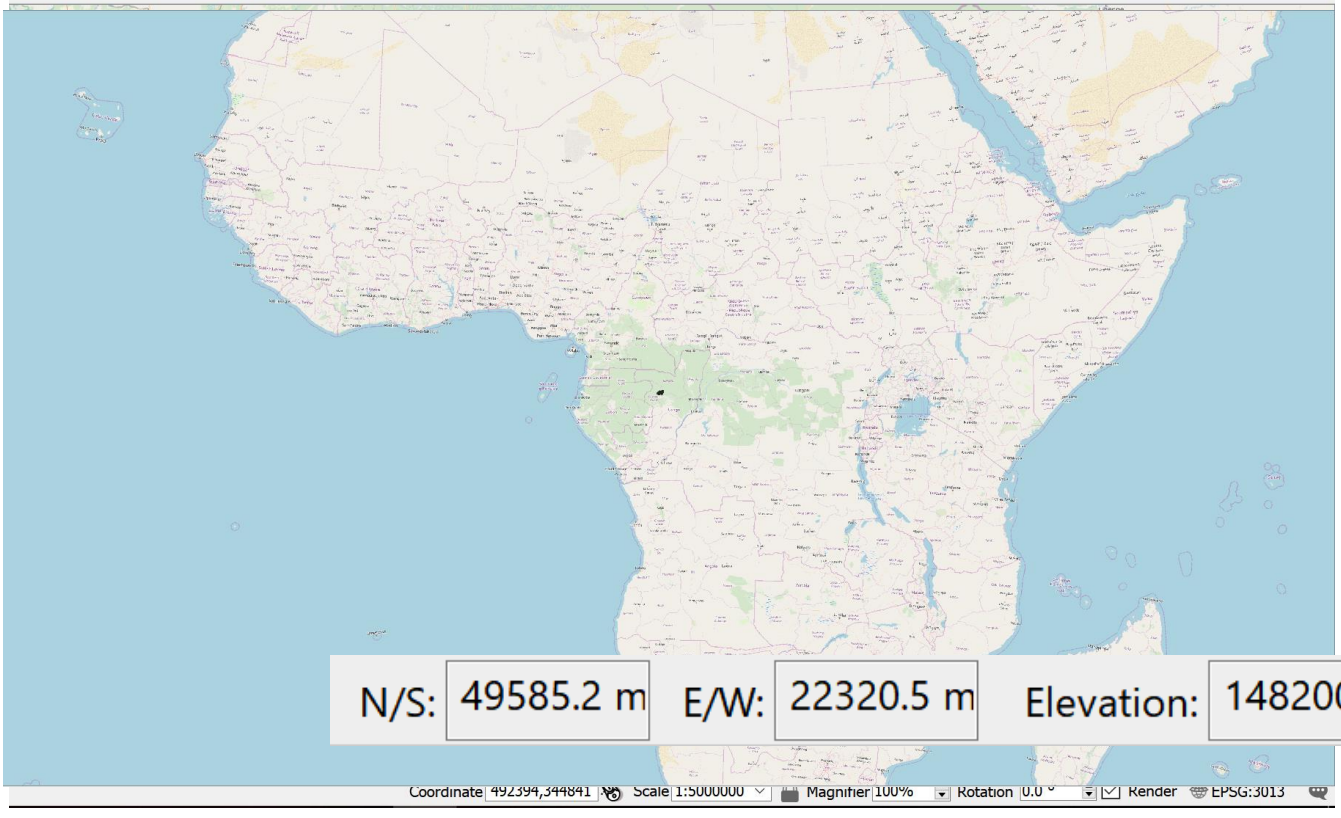
- Local coordinate reference systems are not set by any authority but are just defined by whoever is working on a project
- However, national and international systems are public and are assigned a code by the European Petroleum Standards Group
  - This is called an EPSG code
- In the UK
  - EPSG 27700 – British National Grid
  - EPSG 4326 – the WGS84 system used by GPS

Coordinate Reference Systems – Standard Codes

Coordinate Reference Systems – Linking Local and National Data

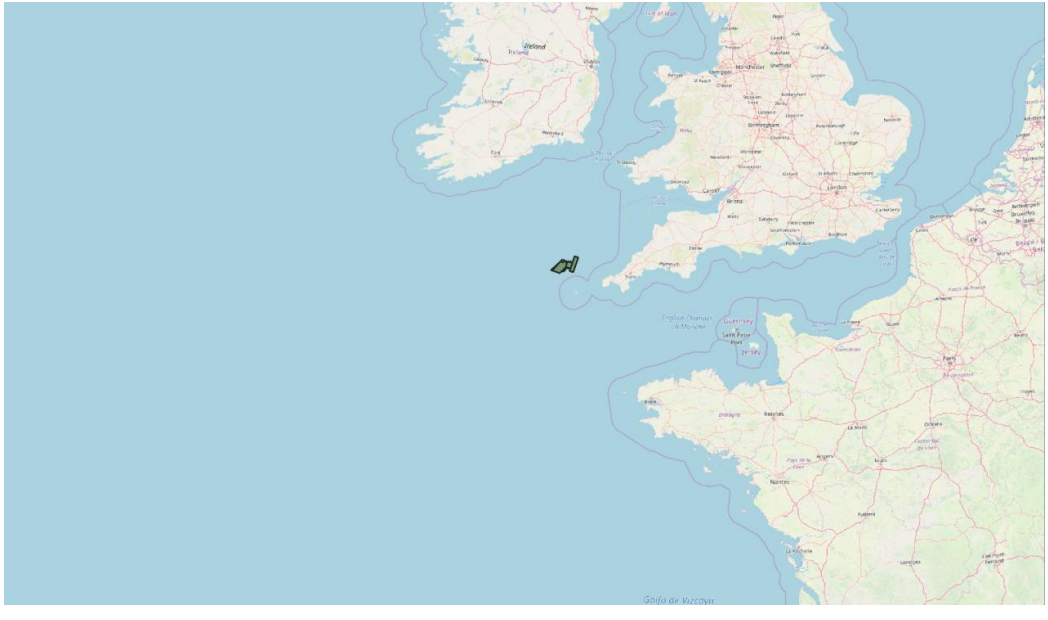
- If you have a locally referenced dataset, you can transform the data to a national reference system
- At a very basic level, if you know the real world x/y of one point (e.g. a corner of the building) in national units, then you can use this to shift all the coordinates
  - Tools such as Revit (for BIM) allow you to do this
  - Might also need to change the units from mm to m
- More sophisticated methods also exist (the geospatial students might learn some of these over the coming year)

Without Georeferencing – data defaults to Africa



Coordinates are not enough – we need the EPSG code too

- Geo-referenced BIM Dataset - in British National Grid



Where it should be – Sweden EPSG 3013

This Lesson

- We looked at georeferencing and the different types of coordinates you can have
  - Local
  - National
  - World
- We also looked at EPSG codes, which are used to tie the coordinates to specific countries

Next Lesson

- In the next lesson, we'll see how to add columns to store 2D and 3D spatial information to our tables

- This turns the tables into mappable data

Storing Spatial Data

This Lesson

- In this lesson, we'll see how to add columns to store 2D and 3D spatial information to our tables

- This turns the tables into mappable data

Databases – Storing Spatial Data

- Data Types
  - So far, we have seen the following data types in our databases:
    - VARCHAR (or *CHARACTER VARYING*)
    - Numeric (p, s) (or *integer, decimal, double precision)*
    - DATE
  - Most modern Databases also have a special data type for storing vector spatial data
  - In PostGIS this is a GeometryColumn

Adding a Geometry (Spatial) Column

- Generic column can take any geometry type

SRID = Spatial Reference identifier

```
select
AddGeometryColumn(<<schema>>,<<tablename>>,<<column
name>>,SRID, <<type of geometry>>,<<number of dimensions>>);

alter table ucfscde.buildings drop column if exists location;

elect AddGeometryColumn('ucfscde','buildings','location',27700,
'geometry',3);
```

Adding a Geometry (Spatial) Column

- Column for a specific geometry type
  - This is a form of constraint

```
alter table ucfscde.rooms drop column location;

select AddGeometryColumn('ucfscde','rooms','location',0, 'polygon',3);
--          (3 dimensions, polygons)

alter table ucfscde.buildings drop column if exists location;

select AddGeometryColumn('ucfscde',‘temperature_sensors','location',0,
'point',3);
            (3 dimensions, points)
```

Adding a Geometry (Spatial) Column

- Coordinate reference systems and constraints – local reference system

```
alter table ucfscde.buildings drop column if exists location;

select AddGeometryColumn('ucfscde','buildings','location',0,
'polyhedralsurface',3);
```

defined by urself

Adding a Geometry (Spatial) Column

- Coordinate reference systems and constraints – British National Grid

```
alter table ucfscde.London_counties drop column if exists location;


select AddGeometryColumn
(‘ucfscde',‘london_counties','location',27700, ‘polygon',2); -- British National Grid
```

Adding a Geometry (Spatial) Column

- Coordinate reference systems and constraints – WGS84 (world wide)

```
alter table ucfscde.london_highway drop column if exists location;
select AddGeometryColumn ('ucfscde','london_highway','location',4326, 'linestring',2);

alter table ucfscde.london_highway drop column if exists location;
Select addGeometryColumn ('ucfscde','london_poi','location',4326, 'point',2);
```

Adding a Geometry (Spatial) Column

(In Global system, coordinates are in degree) ⇒

use national system in this module

- Coordinate reference systems and constraints – WGS84 (world wide)
  - NB – some spatial operations – e.g. distance measurement – don't give useful results using this coordinate reference system (you can't express a distance in degrees)
    - Avoid if possible
    - If you have GPS data try and convert to a national system

Adding a 3D Geometry (Spatial) Column

- This is identical to 2D DDL
  - Use AddGeometryColumn
  - Note the number of dimensions is 3!

```
alter table ucfscde.buildings drop column if exists location;

select AddGeometryColumn('ucfscde','buildings','location',0,
'geometry',3);
```

3D - DDL

- As above, you can constrain the data type
  - This will prevent any invalid surfaces being inserted

```
alter table ucfscde.buildings drop column if exists location;

select AddGeometryColumn('ucfscde','buildings','location',27700,
'polyhedralsurface',3);
```

Adding a 3D Geometry (Spatial) Column

- DO NOT DO THIS: create a table with a geometry column type
  - Not a good idea as no constraints at all on SRID or dimension
  - Also defaults to 2D so 3D won't work

~~create table ucfscde.OSBuildings~~

~~(id serial,~~

~~location geometry);~~

This Lesson

- In the next lesson, we'll see how to add columns to store 2D and 3D spatial information to our tables

- This turns the tables into mappable data

Next Lesson

- Now that we've seen how to create the structure of our tables and the rules to make the data better, we can start looking at how to put data into the database

- We'll start with non-spatial data

Spatial Data Management

Inserting Data

This Lesson

- Now that we've seen how to create the structure of our tables and the rules to make the data better, we can start looking at how to put data into the database

- We'll start with non-spatial data

SQL - 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

} DML

SQL - DML

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

SQL - 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

SQL - DML

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


```
insert into ucfscde.window_costs
(window_type, window_cost_per_m2)
values
('single glazed',1050.2);
```

Insert Data

- NB: there must be a match between the first and second parts of the *insert* statement
  - Same number of columns!

```
insert into ucfscde.room_update_costs
(room_use, cost_per_m2)
values
('computer lab', 10000);
```

Insert Data

- Be careful about quotes
  - Any text needs to have single quotes around it
  - Dates also need quotes

```
insert into ucfscde.temperature_sensors (sensor_make,
sensor_name, sensor_installation_date)
values ('Siemens','Sensor 1', '24-May-2015');
```

SQL – DML – How SERIAL works

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

```
select * from ucfscde.window_costs;
```

```
-- insert some data (with an error)
insert into ucfscde.users (user_name)
values
('usera1');
```

- (in this case the user name is incorrect)

DML – how SERIAL works

- Delete the error and create the correct data

```
delete from ucfscde.users where user_name = 'usera1';
```

```
insert into ucfscde.users (user_name)
values
('user1');
```

```
select * from ucfscde.users;
```

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 – Working with Foreign Keys

- In a database you cannot guarantee that just because the row is the fifth one you insert into the table, it will have ID = 5
- This is because:
  1. Other people may also be inserting data
  2. If you delete a row it doesn't reset the SERIAL counter
- This causes problems with foreign keys as you don't know the ID to reference

DML – Working with Foreign Keys

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

```
select criticality_id from ucfscde.criticality where criticality like
    '%Critical%';

insert into
    ucfscde.university(university_name,year_founded,founders_name,
    criticality)
values ('Centennial University','1826','Claire Ellul', select criticality_id from
    ucfscde.criticality where criticality like  '%Critical%'));
```

Inserting Data – Pro Tip #1

- Insert data into parent tables first – the ones that are the 1: side of the relationship (not the many side)
  - You can find which ones these are from the ER Diagram
- That way you will be able to find the ID required for the foreign key

Inserting Data – Pro Tip #2

- There is a balance between making the constraints very strict and reality
  - e.g. if you force all the windows to have a price, you can't insert a window where you don't know the price
  - BUT if you don't force a price, you can't be sure you have all the price information you need

In reality …

- You can use SQL to insert data
  - This is the best way to really understand what is going on in your database and to identify problems with your data
  - And to understand what is going on under the hood of a database and how the design impacts the information you'll get for decision making

In reality …

- Or you can use other tools that then use SQL to insert data
- For spatial data there are two useful tools

Also great for non-spatial data e.g. CSV, spreadsheets!

This Lesson

- We've seen how to insert data in the database
  - And how to make sure that the 'rules' are followed – i.e. the primary and foreign keys

Next Lesson

- So far we don't know how to change data that is in the database
  - Suppose you discover a mistake?
  - You can use an UPDATE statement to do this – we'll see how in the next lesson

Spatial Data Management

Updating Data

This Lesson

- So far we don't know how to change data that is in the database
  - Suppose you discover a mistake?
    - You could delete and re-insert – but lots of work, and you might have primary/foreign key dependencies
  - You can use an UPDATE statement to do this – we'll see how here

DML

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

```
update ucfscde.buildings set building_name = 'Building 2'
where building_name = 'Building 22';
```

DML

- Updating multiple columns:

```
update ucfscde.temperature_values set value_degrees_c = '43.5',
reading_timestamp='2017-02-02 10:05:00'
where value_degrees_c = '16.3' ;
```

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 ucfscde.rooms set floor = 2 where building_id =
(select building_id from ucfscde.buildings where building_name = 'Building 2');
```

example about updating value of unknown id (FK)

  - Will change the data for all the Building 2 rooms (setting floor to 2)

DML

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

Example

```
update ucfscde.rooms
set last_repainted = '14-Oct-2019'
where (last_repainted <'14-Feb-2005'
AND floor = 2 AND building_id = (select building_id from ucfscde.buildings where
building_name = 'Building 2')) OR (building_id = (select building_id from ucfscde.buildings
where building_name = 'Building 1'));
```

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 ucfscde.university where
  university_name = 'Campus 2';
```

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

```
delete from ucfscde.university where
  university_name = 'Centennial University';
```

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".

You need to delete the ‘child’ row first!

This Lesson

- We've seen how to change / edit data in the database if there are changes, using the UPDATE statement
  - And also how the primary foreign key constraint helps with data quality - e.g. you can't delete a building if it has rooms, as otherwise you'd be left with rooms without a building

This Week

- This week we looked at how to add columns for location information to our tables
  - Including some theory about why location is more complicated that simple numbers
- We also looked at how to add data to the database and edit / update it

Next Week

- So far we haven't inserted any map/location information into our tables
- Next week we'll see how to do this to create 2D and 3D data and show this on a map
- We'll also start to look at how all this data can be used in decision making
  - By looking at basic analysis you can perform on data