Web Mapping Back End: Spatial Databases

Dr Hai Nguyen

Department of Geography and Planning

UNIVERSITY OF LIVERPOOL

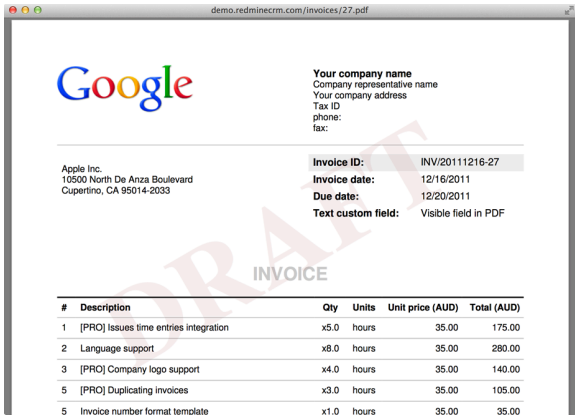
Overview and Structure

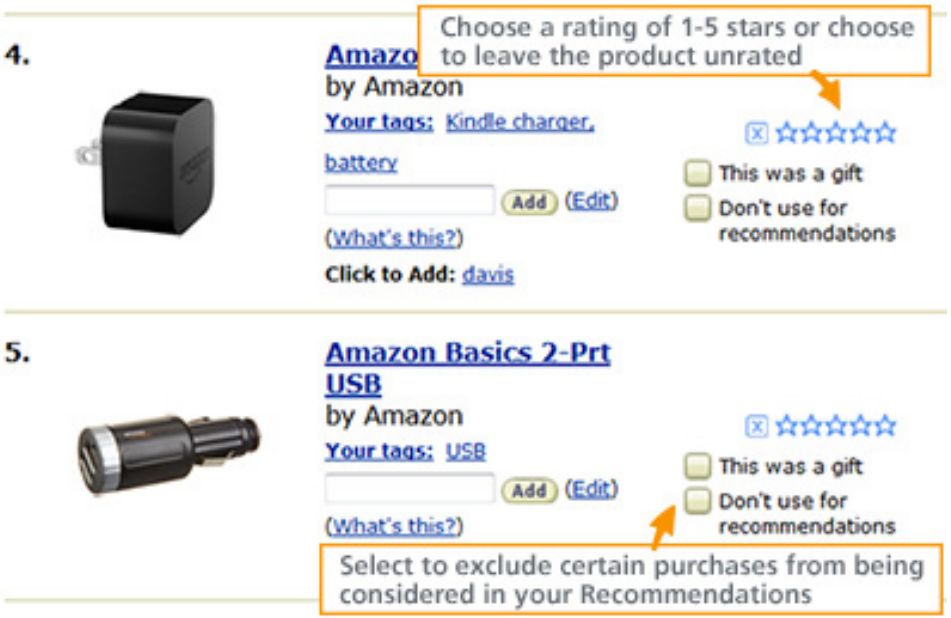
- Databases: an introduction
- Relational Database Systems
  - MYSQL
  - PostgreSQL
  - Example Queries
- Spatial Databases
  - PostGIS
  - Spatial query

Databases

- **Database**: *"a collection of data containing information relevant to an enterprise"*
  - "***Database System Concepts***" (Silberschatz, Korth and Sudarshan, 2010)
- Data in a database are interrelated
- Databases can also be connected (for security/maintenance, etc.)
- Examples:
  - ***A company***: Human Resources, Finance, Customers,Stocks/Servies etc.

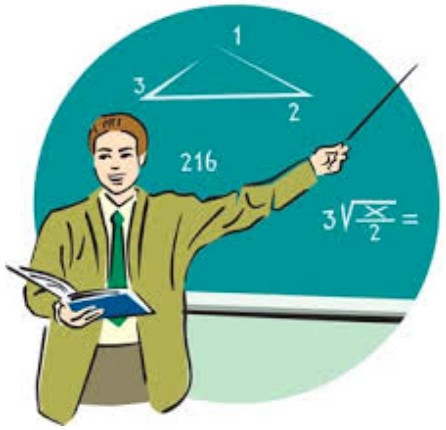
Example: E-commerce





UNIVERSITY OF LIVERPOOL

Example: University



» Programme and Module Catalogue

18001: German Language 1 - Advanced

10/11 Session, Semester 1

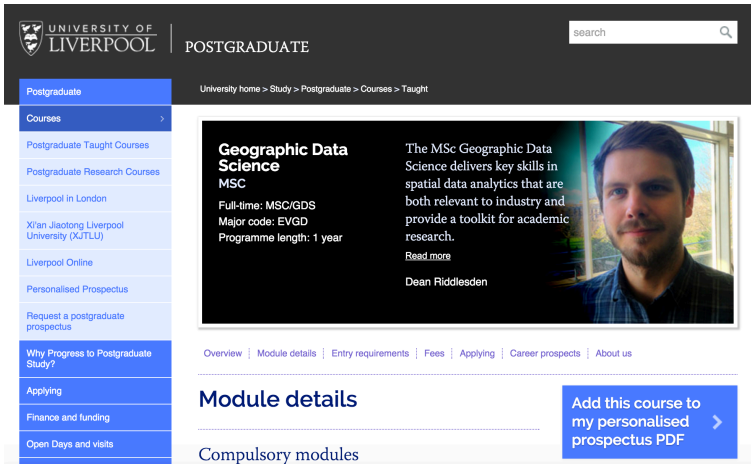
Basic Information

Module Level	Level 4
Nature of Study	Taught Programme
Credits	20
European Credit Transfer Scheme	10
Probable Attendance	25
Location	Hull Campus

This module is not available as a Free Elective
This module is not available as a postgraduate training module
This module is available to Exchange students

Module Rationale

To provide an entry point for those with an AS-level or A-level qualification (or equival
take them in one semester to level B1- as defined by the Common European Framew



UNIVERSITY OF LIVERPOOL

Database Systems

- **Database Management System**
  1. Databases
  2. Tools and programs to store, retrieve, and update the databases
- Database System vs File Systems
  - Single internal vs multiple external applications to manage data
  - Data Redundancy
  - Data Integrity
  - Security

Data Models

- Specify how data are related to each other
- Examples:
  - Relational data model (most popular, SQL)
  - Graph/network model (Linked Data)
  - Document model (MongoDB)
  - Flat model (spreadsheet, tabular data)

Relational Database System

- Data are stored in one or more tables
- Columns are attributes, rows are records
- Each table has a key to differentiate rows (primary key).
- Primary key is used as a reference in other tables (aka foreign key)

Relational Database System
(E.F.Codd, 1970)

- Data are stored in one or more **tables**
  - Example: Customer table
- **Columns** are attributes, **rows** are records
  - Example: names, addresses, etc.
- Each table has a **key** (column or set of columns) to differentiate rows *(primary key*).
  - Example: Customer ID, email address
  - There can be multiple candidate keys, but only one is chosen as primary key.

Customer Table

CustomerName	ContactName	Address	City	PostalCode
Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021
Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023
Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP
Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22
Blauer See Delikatessen	Hanna Moos	Forsterstr. 57	Mannheim	68306
Blondel père et fils	Frédérique	24, place Kléber	Strasbourg	67000

http://www.w3schools.com/sql/trysql.asp?filename=trysql_select_all

Table Relationships

- Primary key of a table is used as a reference in other tables (aka foreign key)
  - Example: CustomerID is stored as foreign keys in Orders table.
- 3 type of relationships:
  - One to One
  - One to Many
  - Many to Many

Table Relationships

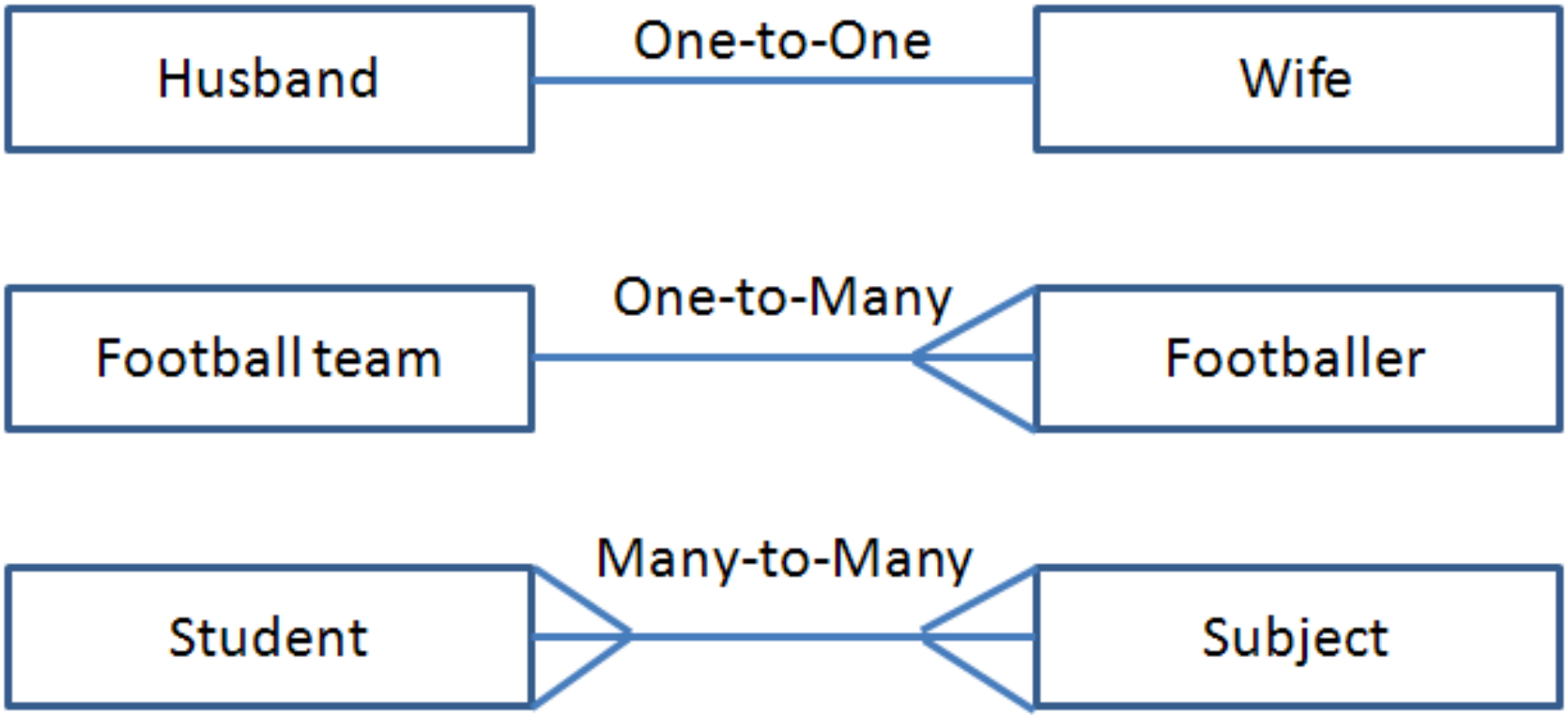


Table Relationship: One to One

- Notion: 1..1
- Examples:
  - Customer table vs. Loyalty card table:
  - Customer table:
    - CustomerID, Name, Address, DOB, etc.
  - Loyalty card table:
    - CardNumber, CustomerID,

Table Relationship: One to Many

- Notion: 1..M (1..∞)
- **Examples**:
  - Customer table vs. Orders table
  - Each order has exactly 1 customer
  - Each customer can have multiple orders
  - Customer table:
    - CustomerID (primary key), Name, Address, DOB, etc.
  - Orders table:
    - OrderID (primary key), CustomerID (foreign key)

Example: Order table

OrderID	CustomerID	EmployeeID	OrderDate
10308	2	7	1996-09-18
10365	3	3	1996-11-27
10355	4	6	1996-11-15
10383	4	8	1996-12-16
10278	5	8	1996-08-12
10280	5	2	1996-08-14
10384	5	3	1996-12-16
10265	7	2	1996-07-25
10297	7	5	1996-09-04

http://www.w3schools.com/sql/trysql.asp?filename=trysql_select_all

Table Relationship: Many to Many

- Notion: M..M (∞..∞)
- A composition of 2 1:M relationships
- **Examples**:
  - Products table vs. Orders table
  - Each order can have multiple products
  - Each product can exist in multiple orders
- This relationship is usually decomposed into 2 1:M relationship
  - Orders vs OrderDetails, OrderDetails vs. Products

Many to Many Relationship

Order Details table

OrderDetailID	OrderID	ProductID	Quantity
1	10248	11	12
2	10248	42	10
3	10248	72	5
4	10249	14	9
5	10249	51	40
6	10250	41	10
7	10250	51	35

1..M

1..M

OrderID	CustomerID	EmployeeID	OrderDate
10308	2	7	1996-09-18
10365	3	3	1996-11-27
10355	4	6	1996-11-15
10383	4	8	1996-12-16
10278	5	8	1996-08-12
10280	5	2	1996-08-14
10384	5	3	1996-12-16
10265	7	2	1996-07-25
10297	7	5	1996-09-04

Orders table

ProductID	ProductName	SupplierID	CategoryID	Unit	Price
1	Chais	1	1	10 boxes x 20 bags	18
2	Chang	1	1	24 - 12 oz bottles	19
3	Aniseed Syrup	1	2	12 - 550 ml bottles	10
4	Chef Anton's Cajun Seasoning	2	2	48 - 6 oz jars	22
5	Chef Anton's Gumbo Mix	2	2	36 boxes	21.35
6	Grandma's Boysenberry Spread	3	2	12 - 8 oz jars	25
7	Uncle Bob's Organic Dried Pears	3	7	12 - 1 lb pkgs.	30
8	Northwoods Cranberry Sauce	3	2	12 - 12 oz jars	40
9	Mishi Kobe Niku	4	6	18 - 500 g pkgs.	97

Products table

SQL – Structured Query Language

- Designed for relational databases
- A declarative programming language
- To manage both schemas (tables) and data (records)
- ***Schema*** Management:
  - Create/delete tables
  - Add/change columns
- ***Data*** Management:
  - Data retrieval
  - Data creation/update/deletion

SQL-based Database Systems

SQL Statements: SELECT

- Retrieve data from the database (read-only)
- **Syntax**:
  - `SELECT [Columns] FROM [Tables] WHERE [Conditions] GROUP BY [Columns] ORDER BY [Columns]`
  - **Compulsory parts**: "`SELECT [Columns] FROM [Tables]`"
  - **Optional parts**:
    - `WHERE [Conditions]`: filtering output
    - `GROUP BY [Columns]`: group results by some cols
    - `ORDER BY [Columns]`: present results as ordered

Example: List all records

- `SELECT * FROM CUSTOMERS;`

Number of Records: 90

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden
6	Blauer See Delikatessen	Hanna Moos	Forsterstr. 57	Mannheim	68306	Germany
7	Blondel père et fils	Frédérique Citeaux	24, place Kléber	Strasbourg	67000	France
8	Bólido Comidas preparadas	Martín Sommer	C/ Araquil, 67	Madrid	28023	Spain

Example: List all customers from the UK

```
SELECT * FROM CUSTOMERS WHERE Country
= 'UK';
```

Number of Records: 7

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
11	B's Beverages	Victoria Ashworth	Fauntleroy Circus	London	EC2 5NT	UK
16	Consolidated Holdings	Elizabeth Brown	Berkeley Gardens 12 Brewery	London	WX1 6LT	UK
19	Eastern Connection	Ann Devon	35 King George	London	WX3 6FW	UK
38	Island Trading	Helen Bennett	Garden House Crowther Way	Cowes	PO31 7PJ	UK
53	North/South	Simon Crowther	South House 300 Queensbridge	London	SW7 1RZ	UK
72	Seven Seas Imports	Hari Kumar	90 Wadhurst Rd.	London	OX15 4NB	UK

Example: List customers count by country

```
SELECT Country, COUNT(*) FROM
CUSTOMERS GROUP BY Country;
```

Number of Records: 21

Country	COUNT(*)
Argentina	3
Austria	2
Belgium	2
Brazil	9
Canada	3
Denmark	2
Finland	2
France	11
Germany	10
Ireland	1
Italy	3
Mexico	5
Norway	1

Example: List customers count by country in decreasing order by country

```
SELECT Country, COUNT(*) FROM Customers GROUP BY Country ORDER BY Country DESC;
```

Number of Records: 21

Country	COUNT(*)
Venezuela	4
USA	13
UK	7
Switzerland	2
Sweden	2
Spain	5
Portugal	2
Poland	1

Example: List customers count by country in decreasing order by country

```
SELECT Country, COUNT(*) FROM Customers GROUP BY Country ORDER BY Country DESC;
```

Number of Records: 21

Country	COUNT(*)
Venezuela	4
USA	13
UK	7
Switzerland	2
Sweden	2
Spain	5
Portugal	2
Poland	1

SELECT from multiple tables

- JOIN clauses: combining 2 tables
- **Syntax**:
  - `SELECT [Columns] FROM [Table1] JOIN [Table2] ON [conditions between the primary key and foreign key]`
- Types of JOINs:
  - **`A INNER JOIN B:`** return all rows from A and B that match.
  - **`A LEFT OUTER JOIN B`**: return all rows from A and matching rows from B
  - **`A RIGHT OUTER JOIN B`**: return all rows from B and matching rows from A
  - **`A FULL OUTER JOIN B`**: return a union of LEFT OUTER and RIGHT OUTER joins

Example: INNER JOIN

```
SESELECT * FROM Customers INNER JOIN Orders
ON Customers.CustomerID = Orders.CustomerID;
```

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country	OrderID
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico	10308
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico	10365
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK	10355
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK	10383
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden	10278
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden	10280

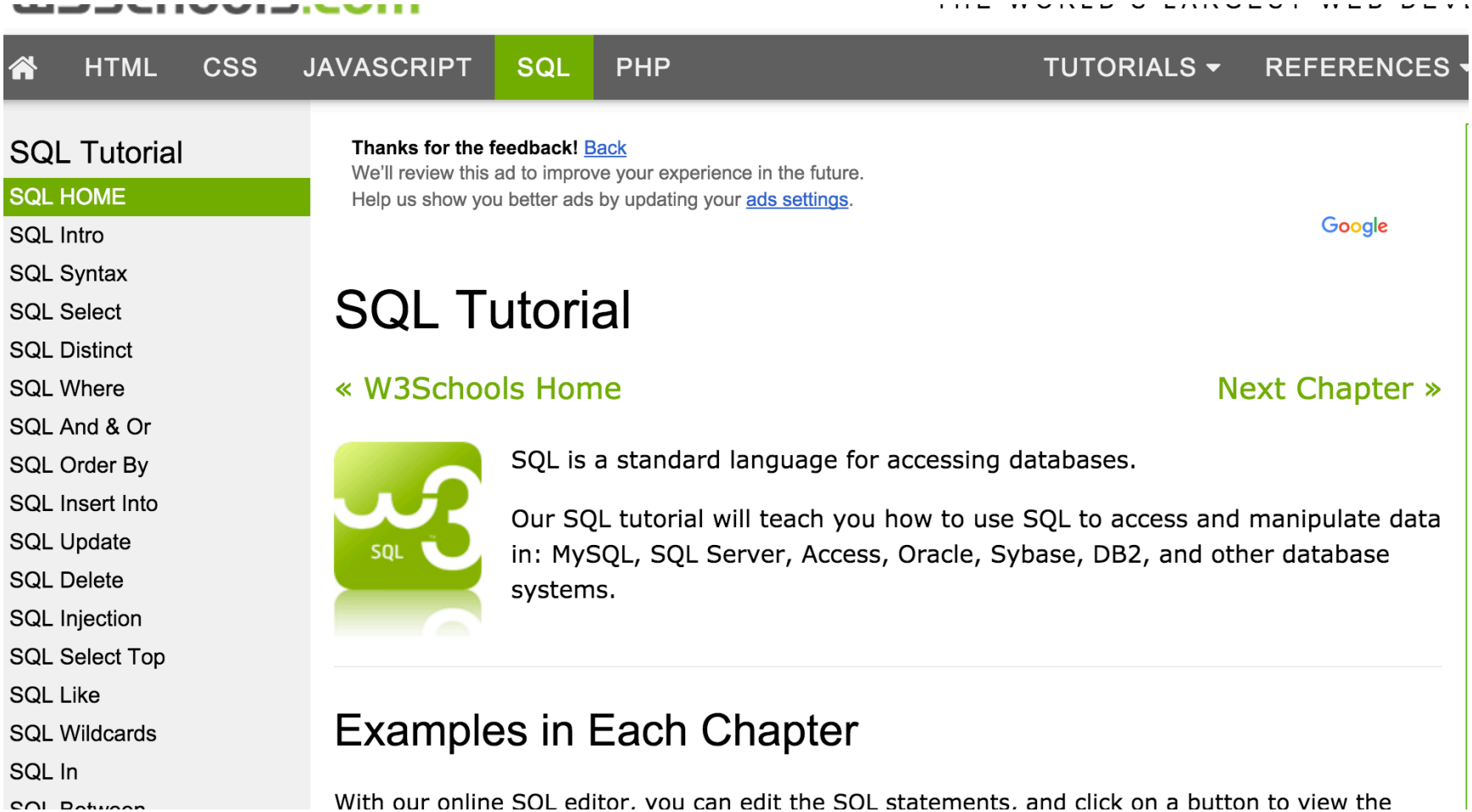
Other Statements for Data Maintenance

- Add data:
  - INSERT [data] INTO [table]
- Modify data:
  - UPDATE [table]

    SET column1=value1,column2=value2,...

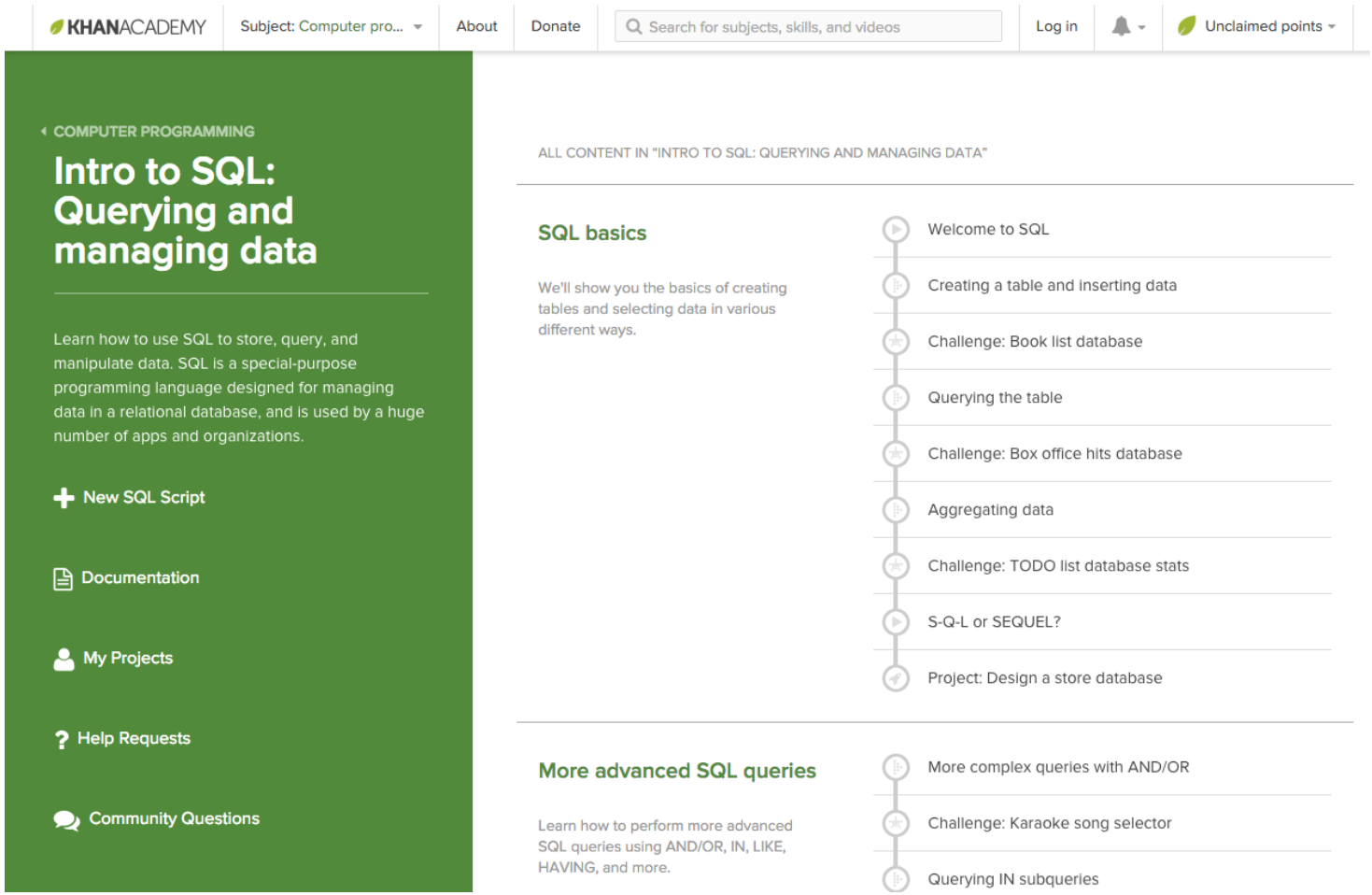
    WHERE [conditions];
- Delete data:
  - DELETE FROM [table] WHERE [conditions]

More on SQL (1)



http://www.w3schools.com/sql/default.asp

More on SQL (2)



UNIVERSITY OF LIVERPOOL

https://www.khanacademy.org/computing/computer-programming/sql

Spatial Databases

- **Spatial databases**: databases that can store spatial data effectively
- **Spatial Database Systems**: systems that can store and support retrieval/management of spatial databases.
- **Examples**:
  - Relational Spatial Database Systems: PostGIS, SQL Server 2008 and higher, MySQL with spatial extension
  - Document-based Database Systems: MongoDB

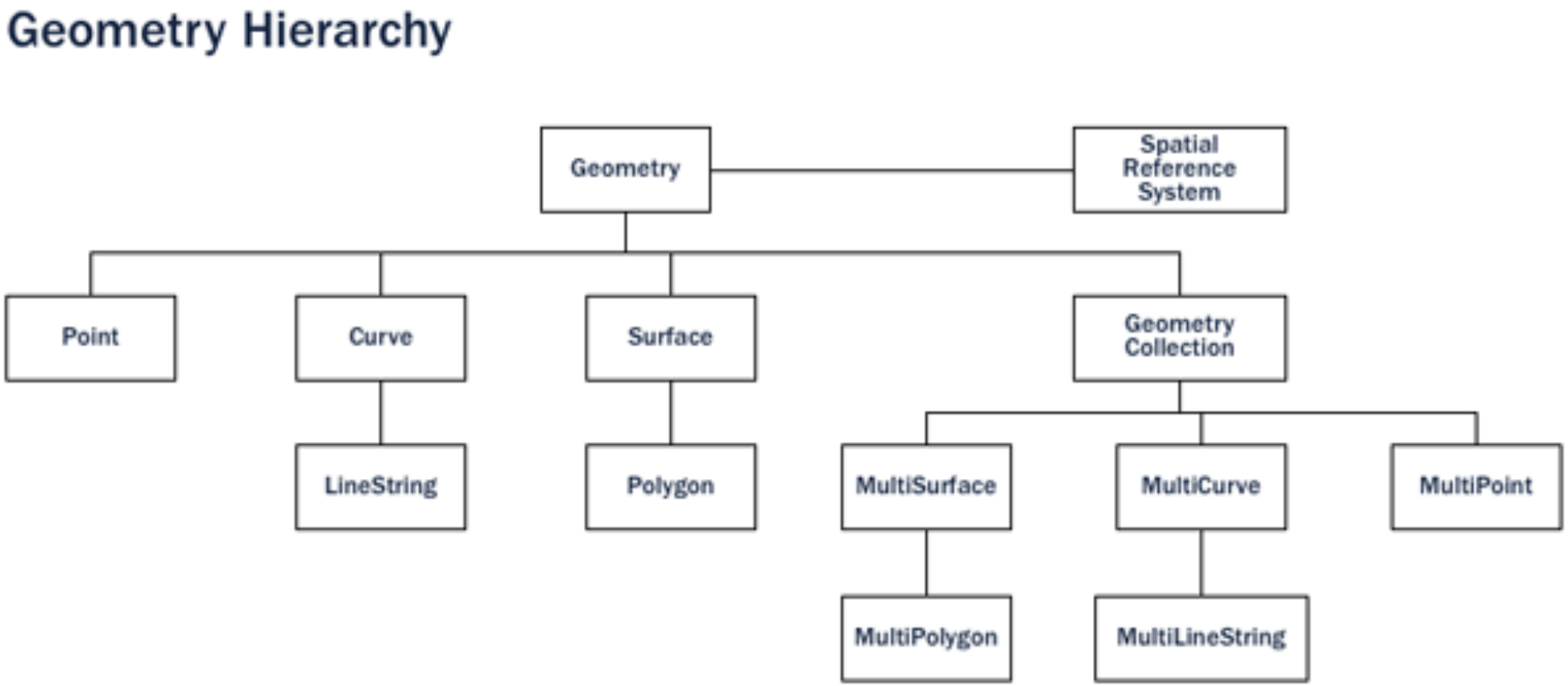
POSTGIS

- **PostGIS** = PostgreSQL + Spatial extensions
- **Support**:
  - **Spatial indexing**: how spatial data can be stored and retrieved
  - **Spatial data types**: points, lines, polygons, etc.
  - **Spatial functions**:
  - creating and modifying spatial objects,
  - retrieve spatial objects' properties and relationships, etc.

Spatial Indexing

- **Indexing**: methods to arrange/sort data in a way that allows fast and efficient retrieval and update.
- **Example:**
  - Phone book ordered by last names.
  - Similar for numeric/string values
- **Spatial indexing:**
  - Given a bounding box, which objects is within this bounding box?
  - Speed up spatial queries by only look at relevant objects

Spatial Geometry Types



Spatial Indexing

- **Indexing**: methods to arrange/sort data in a way that allows fast and efficient retrieval and update.
- **Example:**
  - Phone book ordered by last names.
  - Similar for numeric/string values
- **Spatial indexing:**
  - Given a bounding box, which objects is within this bounding box?
  - Speed up spatial queries by only look at relevant objects

Spatial Functions

- Standard SQL syntax
- 5 main types:
  - **Conversion** functions
  - **Management** functions
  - **Retrieval** functions
  - **Comparison** functions
  - **Generate** functions

Example: Retrieval functions

- List all LADs from the LAD table and their area
- `SELECT LAD_name, ST_Area(geom) FROM LAD`
- **Similar retrieval functions:**
  - ST_LENGTH
  - ST_PERIMETER
  - ST_X
  - ST_Y

Example: Comparison functions

- Used in conditions and joins
- List all streets intersecting an LAD using JOIN
  - `SELECT LAD_name, shop_names FROM LAD INNER JOIN STREETS ON ST_INTERSECTS(STREETS.geom, LAD.geom)`
- List all shops within an LAD using WHERE clause
  - `SELECT LAD_name, shop_names FROM LAD, SHOPS WHERE ST_WITHIN(SHOPS.geom, LAD.geom)`

Other comparison functions

- ST_CONTAINS/ST_WITHIN
- ST_INTERSECTS/ST_DISJOIN
- ST_TOUCHES
- ST_CROSSES

Many thanks…

UNIVERSITY OF LIVERPOOL