Spatial Data Management – Advanced Topics 3 – NoSQL and Blockchain

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

Big Data

- There is much more data – and lots of it is spatial!
  - Twitter, Facebook
  - Sensors e.g. Crossrail vibration sensors for tunnelling, EveryAware Air Quality sensors
  - Traffic/Congestion cameras
  - Online shopping/delivery services
  - Bicycle hire

Big Data

- What is Big Data …
- https://www.youtube.com/watch?v=Hv397JnNWYc (from IBM but covers the basic principles well)

Overview

- Managing Big Data in a Relational Database
  - Distributing the Data and Replication
  - Adjusting Block Size
- Beyond Relational Databases
  - NoSQL
  - NoSQL and Spatial Data
  - Blockchain

Managing Big Spatial Data

- The slowest operation in any computer is the time taken to read data from a hard drive and to write data to a hard drive
- Therefore, a good part of optimising database performance focusses on minimising the number of reads/writes that take place

Managing Big Spatial Data

- One way to do this is to use multiple disks, so that the read/write operations can take place in parallel (at the same time).
- If you use one disk for the data and another for the indexes, the system can be reading data at the same time as it is reading indexes, which is more efficient

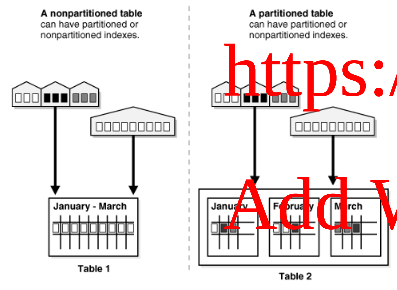
Managing Big Spatial Data

- You can even distribute your data onto more than two disks if you have them.
  - Think about which tables are read more often and put them on separate disks.

Managing Big Spatial Data

- “Partitioning allows a table, index, or index-organized table to be subdivided into smaller pieces, where each piece of such a database object is called a partition. Each partition has its own name, and may optionally have its own storage characteristics.” (https://docs.oracle.com/cd/B28359_01/server.111/b32024/partition.htm)

Managing Big Spatial Data


A nonpartitioned table can have partitioned or nonpartitioned indexes.
A partitioned table can have partitioned or nonpartitioned indexes.
January - March
Table 1
January
February
March
Table 2

https://docs.oracle.com/cd/B28359_01/server.111/b32024/partition.htm

Managing Big Spatial Data

- **Horizontal partitioning**
  - Put **different rows** from the same table onto different disks as different tables:
    - for example customers with postcodes beginning with A-H are stored in one table and those from I onwards in another
  - A view (which creates a ‘fake’ table by joining two or more sub tables) can then be created to make the two tables
  - A view appears to SQL as a table
    - However it runs at RUN TIME so the join query to merge the two tables may take a little more time, although the data retrieval will be quicker as the two hard drives can operate in parallel
- https://www.itprotoday.com/sql-server/horizontal-and-vertical-partitioning

Managing Big Spatial Data

- Horizontal partitioning
  - Put all the temperature data from Pearson sensors in 1 table and from Chadwick in another table

```
create table assets.temperature_values_pearson (
Temperature_value_id serial,
Temperature_sensor_id integer
Date_and_time date,
Value_degrees_c numeric (5,2));

create table assets.temperature_values_chadwick (
Temperature_value_id serial,
Temperature_sensor_id integer
Date_and_time date,
Value_degrees_c numeric (5,2));
```

Managing Big Spatial Data

- Horizontal partitioning
  - Then create a view to link the two (a view is a ‘stored’ SQL statement that can be treated like a table

```
CREATE VIEW assets.temperature_values AS
SELECT * FROM assets.temperature_values_pearson
UNION ALL
SELECT * FROM assets.temperature_values_chadwick;

SELECT * FROM assets.temperature_values;
```

Managing Big Spatial Data

- **Vertical partitioning**
  - Split a table into two or more tables, by placing some of the columns on one disk and the remainder on another.
  - This goes beyond normalisation (which splits tables into smaller ones to reduce data duplication) and actually splits the fully normalised data again.
  - You could chose to put rapidly changing columns on a faster hard drive than columns where data does not change often.
  - A view can then be used to re-unite the split tables

- http://cloudgirl.tech/data-partitioning-vertical-horizontal-hybrid-partitioning/

Managing Big Spatial Data

- Replication
  - Rather than just split the data/tables, you make COPIES of the entire database on different servers
  - It is sometimes difficult to keep all the data synchronised, in particular where you have a very high level of inserts/updates/deletes per second
  - However this is very useful when there is high demand for the data
    - And also serves as a backup

Managing Big Spatial Data

- Types of Replication:
  - *Active replication* is performed by processing the same request at every replica
    - i.e. if you delete a row it is deleted simultaneously on all servers
    - Useful if you want to maintain a high level of service as if one replica fails the others still exist
    - However, needs more resources

Managing Big Spatial Data

- Types of Replication:
  - *Passive replication* involves processing each single request on a single replica and then transferring its resultant state to the other replicas
    - This is sometimes known as mirroring
    - In a basic system one server receives all the requests and then transmits changes to the backups
    - In a more advanced system, each server receives requests and transmits changes to all the others

Managing Big Spatial Data

- Distributed load systems
  - A consequence of replication is a distributed database in which users can access data relevant to their tasks without interfering with the work of others.

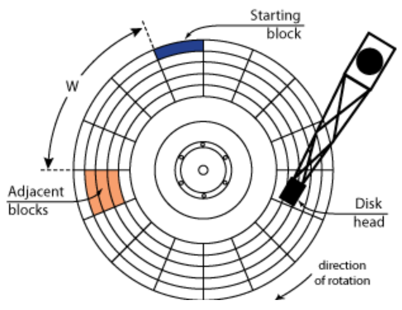
Overview

- Managing Big Data in a Relational Database
  - Distributing the Data and Replication
  - Adjusting Block Size
- Beyond Relational Databases
  - NoSQL
  - NoSQL and Spatial Data
  - Blockchain

Managing Big Spatial Data

- Block Size
  - Is the size on disk of one ‘block’ of data – i.e. the amount of data that is read by one ‘read’ operation of the hard disk

Managing Big Spatial Data



https://www.usenix.org/legacy/event/fast05/tech/schlosser/schlosser_html/disk-adjacent-blocks.png

Managing Big Spatial Data

- Block Size
  - Differs between different databases
    - In PostgreSQL, block size is set up when you install the software.
    - In Oracle you can set it up when you create a database and different parts of the database can have different block sizes.

Managing Big Spatial Data

- Why is block size important?
  - As mentioned above, the slowest part of any computer operation is the time taken to read data from a hard drive into the computer’s memory (where it can then be used for querying).
  - The larger the block the fewer of these read operations are needed …

Managing Big Spatial Data

- Block size
  - The standard block size in PostgreSQL is 8kB (kilo-bytes). (A maximum of 32kB can be set)
  - This is particularly important for spatial data, because some spatial objects may be quite large
  - 1 double number takes 8 bytes of storage -> i.e. one x, y or z coordinate requires 8 bytes ..

Managing Big Spatial Data

- Block Size
  - So you can only have 1000 coordinates in your 8KB disk read operation
    - That is 500 coordinate pairs, which for many geometry objects is low.
  - Of course, this is assuming that you don’t also want to see the attributes: each character also requires 1 byte of storage

Managing Big Spatial Data

- Block Size
  - However, there are some situations – e.g. financial transactions on a bank account – where the data required to be read is very small, and a 2KB block size would be more appropriate
    - You can have 250 numbers in your 2KB disk read operation which is still too many for one trasaction

Managing Big Spatial Data

- Block Size
  - A balance needs to be found between differing uses of your database:
    - OLTP databases – on line transaction processing – such as banks have many read/write operations onto disk per second
      - Thus, block size should be smaller
    - Decision Support Systems – DSS – read large quantities of data and then perform analysis on the data.
      - This requires larger block sizes so that several rows of data can be read at the same time.

Managing Big Spatial Data

- Block Size
  - NB: Be careful with spatial data, as 'row chaining' – where a single row of data does not fit into one block – can happen.
    - If it does, you require more than one hard disk read operation to get the data into memory which can be very slow!

Overview

- Managing Big Data in a Relational Database
  - Distributing the Data and Replication
  - Adjusting Block Size
- Beyond Relational Databases
  - NoSQL
  - NoSQL and Spatial Data
  - Blockchain

Managing Big Spatial Data

- Some data is structured in traditional relational databases
  - Most of this module has been about how to do that
- But some data – e.g. documents, videos, pictures and so forth is *unstructured*
  - *E.g. web pages can have text and images anywhere, word documents don't all have the same headers and sub sections*

Managing Big Spatial Data

- Why did NoSQL evolve
  - You are a search engine company and you realise that you have access to huge reams of data – all the web pages on the internet
  - These are unstructured so difficult to manage and monetize (i.e. make profit for your company)
  - So you need new approaches to doing this

Managing Big Spatial Data

- NoSQL databases have evolved to help with this challenge
- NoSQL = 'not only SQL'
- NoSQL databases are still DBMS – so authentication, backup, security etc still valid

Managing Big Spatial Data

- NoSQL Databases
  - Don't structure data in a relational format, the way we have seen so far
  - Rather, items are grouped into more useful groupings

Managing Big Spatial Data

- NoSQL Databases
  - Are designed for maximum access and speed of response
  - Are able to run on very large clusters of low-powered computers
  - Do not adhere to ACID principles (see next slides)

Managing Big Spatial Data

- Terminology – ACID
  - Atomicity
    - if one part of a transaction fails, it all fails
  - Consistency
    - Any change to data will adhere to all the rules in the database (primary keys, foreign keys, constraints)

Managing Big Spatial Data

- Terminology – ACID
  - Isolation
    - If two transactions are executed at the same time, the result would be the same as if they are executed one after the other
  - Durability
    - Once the transaction is complete it doesn't change even if power is lost

What is NoSQL

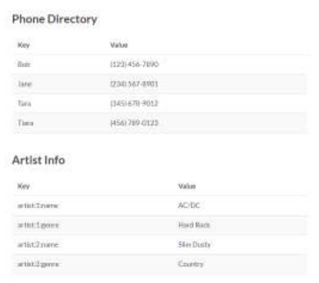
- https://www.youtube.com/watch?v=qUV2j3XBRHc

Managing Big Spatial Data

- Types of NoSQL Databases
  - Key Value – use a hash table to store a key with a pointer to a particular item of data
  - Simple to implement
    - Very simple data structure, always 2 columns

- http://rebelic.nl/2011/05/28/the-four-categories-of-nosql-databases/

Key Value Database



Phone Directory

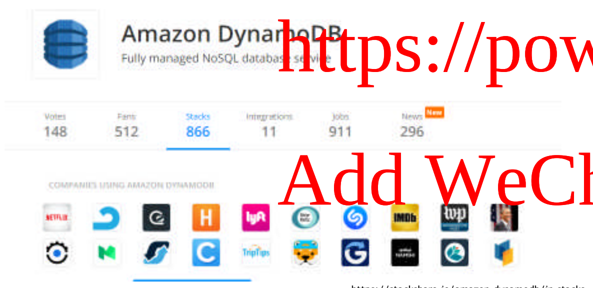
Key	Value
Bob	(123) 456-7890
Jane	(234) 567-8901
Tara	(345) 678-9012
Tiara	(456) 789-0123

Artist Info

Key	Value
artist:1:name	AC/DC
artist:1:genre	Hard Rock
artist:2:name	Slim Dusty
artist:2:genre	Country

https://database.guide/what-is-a-key-value-database/12

Key Value Database



https://stackshare.io/amazon-dynamodb/in-stacks

Managing Big Spatial Data

- Types of NoSQL Databases
  - Document Databases
    - Similar to key/value stores but consist of hierarchies of key/value pairs (nested key/value pairs)
    - The semi-structured documents are stored in formats such as JSON
    - Support more efficient querying than key/value pairs
      - You can drill down through the structure

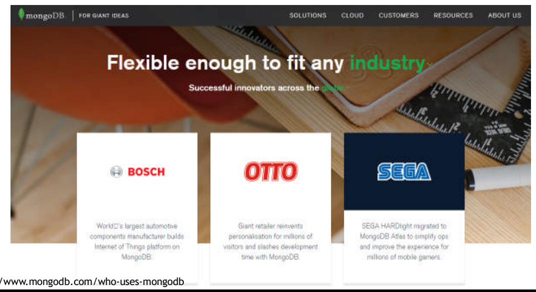
- http://rebelic.nl/2011/05/28/the-four-categories-of-nosql-databases/

Managing Big Spatial Data

JSON Example

```
{
    "myName": "Fred",
    "lastName": "Sanger",
    "address": {
        "streetAddress": "25 Hinxton Hall",
        "city": "London",
        "Country": "GB",
        "postalCode": W2 1PG
    },
    "phoneNumbers": [
        "44 0208 3345456",
        "44 0207 876789"
    ]
}
```

Document Databases



World□'s largest automotive components manufacturer builds Internet of Things platform on MongoDB.

Giant retailer reinvents personalisation for millions of visitors and slashes development time with MongoDB.

SEGA HARDlight migrated to MongoDB Atlas to simplify ops and improve the experience for millions of mobile gamers.

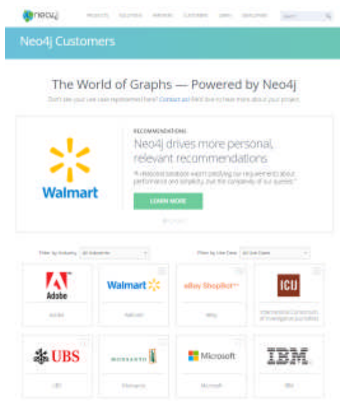
https://www.mongodb.com/who-uses-mongodb

Managing Big Spatial Data

- Types of NoSQL Databases
  - Graph Databases
    - A graph model is used (like a road network) to link data together
    - Used to store information about networks, such as social connections

- http://rebelic.nl/2011/05/28/the-four-categories-of-nosql-databases/

Graph Databases



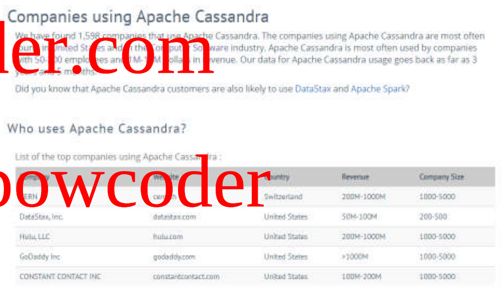
https://neo4j.com/customers/

Managing Big Spatial Data

- Types of NoSQL Databases
  - Column Family Store
    - Also uses keys, but the keys point to multiple columns of data, which may be arranged across multiple machines
      - So, 1 key, multiple values
    - Like a database table, but each row can have a different number of columns

- http://rebelic.nl/2011/05/28/the-four-categories-of-nosql-databases/

Column Store



Companies using Apache Cassandra

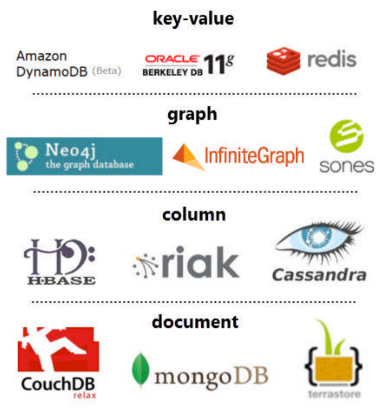
We have found 1,598 companies that use Apache Cassandra. The companies using Apache Cassandra are most often found in United States and in the Computer Software industry. Apache Cassandra is most often used by companies with 50-200 employees and 1M-10M dollars in revenue. Our data for Apache Cassandra usage goes back as far as 3 years and 5 months.

Did you know that Apache Cassandra customers are also likely to use DataStax and Apache Spark?

Who uses Apache Cassandra?

List of the top companies using Apache Cassandra :

Company	Website	Country	Revenue	Company Size
CERN	cern.ch	Switzerland	200M-1000M	1000-5000
DataStax, Inc.	datastax.com	United States	50M-100M	200-500
Hulu, LLC	hulu.com	United States	200M-1000M	1000-5000
GoDaddy Inc	godaddy.com	United States	>1000M	1000-5000
CONSTANT CONTACT INC	constantcontact.com	United States	100M-200M	1000-5000



Overview

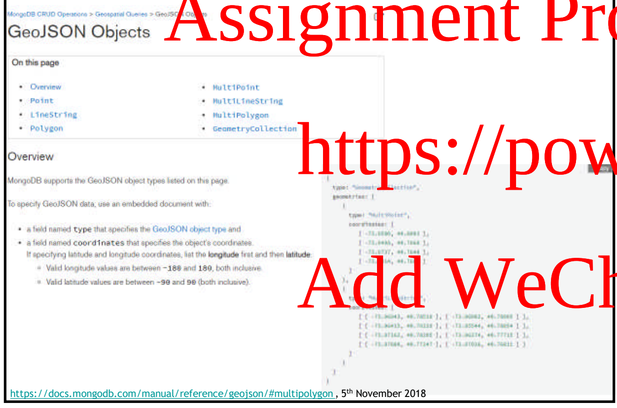
- Managing Big Data in a Relational Database
  - Distributing the Data and Replication
  - Adjusting Block Size
- Beyond Relational Databases
  - NoSQL
  - NoSQL and Spatial Data
  - Blockchain

Managing Big Spatial Data

- NoSQL Databases
  - Don't offer a consistent SQL-type query interface - each query interface is different
  - Relatively new to market, so fewer tools such as PGAdmin 4 or FME to handle the data
    - Most of the work is done at command line
  - However, if GeoJSON can be created (served) then it is possible to visualise spatial data
    - GeoJSON is a de-facto industry standard for sharing spatial data

Geospatial in MongoDB

- Document database
- Offers 2D coordinate handling and spatial indexing
  - Allows storage of spatial objects as geoJSON
- Also allows some spatial queries
    - No 3D



MongoDB CRUD Operations > Geospatial Queries > GeoJSON Objects

GeoJSON Objects

On this page

- Overview
- Point
- LineString
- Polygon
- MultiPoint
- MultiLineString
- MultiPolygon
- GeometryCollection

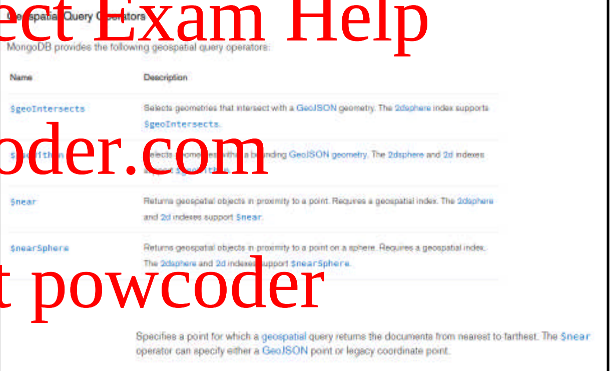
Overview

MongoDB supports the GeoJSON object types listed on this page.

To specify GeoJSON data, use an embedded document with:

- a field named `type` that specifies the GeoJSON object type and
- a field named `coordinates` that specifies the object's coordinates.
  If specifying latitude and longitude coordinates, list the **longitude** first and then **latitude**.
  - Valid longitude values are between `-180` and `180`, both inclusive.
  - Valid latitude values are between `-90` and `90` (both inclusive).

https://docs.mongodb.com/manual/reference/geojson/#multipolygon, 5th November 2018



Geospatial Query Operators

MongoDB provides the following geospatial query operators:

Name	Description
`$geoIntersects`	Selects geometries that intersect with a GeoJSON geometry. The 2dsphere index supports `$geoIntersects`.
`$geoWithin`	Selects geometries within a bounding GeoJSON geometry. The 2dsphere and 2d indexes support `$geoWithin`.
`$near`	Returns geospatial objects in proximity to a point. Requires a geospatial index. The 2dsphere and 2d indexes support `$near`.
`$nearSphere`	Returns geospatial objects in proximity to a point on a sphere. Requires a geospatial index. The 2dsphere and 2d indexes support `$nearSphere`.

Specifies a point for which a geospatial query returns the documents from nearest to farthest. The `$near` operator can specify either a GeoJSON point or legacy coordinate point.

https://docs.mongodb.com/manual/geospatial-queries/#id1, 5th November 2018

Geospatial in MongoDB

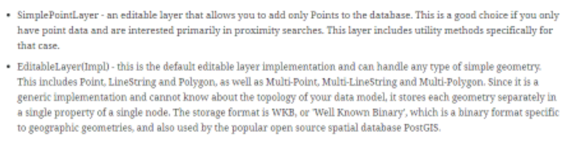
- Coordinate reference systems are sort-of handled - but not the wide range you find in RDBMS
  - Geometry can either be 'on the plane' (i.e. flat, projected) or 'on the sphere' (using latitude/longitude)

Geospatial in Neo4J

- Graph database
- Spatial supported via an add-on library (only point data in the core installation)
  - Not clear if being updated frequently but last release on 7th August this year

Geospatial in Neo4J

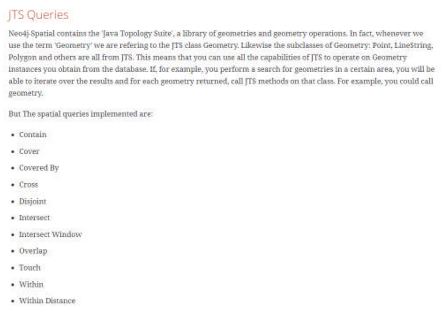
- Data types - points and WKT



- SimplePointLayer - an editable layer that allows you to add only Points to the database. This is a good choice if you only have point data and are interested primarily in proximity searches. This layer includes utility methods specifically for that case.
- EditableLayer(Impl) - this is the default editable layer implementation and can handle any type of simple geometry. This includes Point, LineString and Polygon, as well as Multi-Point, Multi-LineString and Multi-Polygon. Since it is a generic implementation and cannot know about the topology of your data model, it stores each geometry separately in a single property of a single node. The storage format is WKB, or 'Well Known Binary', which is a binary format specific to geographic geometries, and also used by the popular open source spatial database PostGIS.

https://neo4j-contrib.github.io/spatial/0.24-neo4j-3.1/index.html#spatial-indexing , 5th November 2018

Geospatial in Neo4J



JTS Queries

Neo4j-Spatial contains the 'Java Topology Suite', a library of geometries and geometry operations. In fact, whenever we use the term 'Geometry' we are refering to the JTS class Geometry. Likewise the subclasses of Geometry: Point, LineString, Polygon and others are all from JTS. This means that you can use all the capabilities of JTS to operate on Geometry instances you obtain from the database. If, for example, you perform a search for geometries in a certain area, you will be able to iterate over the results and for each geometry returned, call JTS methods on that class. For example, you could call geometry.

But The spatial queries implemented are:

- Contain
- Cover
- Covered By
- Cross
- Disjoint
- Intersect
- Intersect Window
- Overlap
- Touch
- Within
- Within Distance

https://neo4j-contrib.github.io/spatial/0.24-neo4j-3.1/index.html#_jts_queries , 5th November 2018

Geospatial in Neo4J

- Core system seems to support WGS84 and projected data
- Import functionality for shapefiles and OSM data
- Also links to GeoServer (spatial data management and publication software)

Overview

- Managing Big Data in a Relational Database
  - Distributing the Data and Replication
  - Adjusting Block Size
- Beyond Relational Databases
  - NoSQL
  - NoSQL and Spatial Data
  - **Blockchain**

Blockchain

- “The blockchain is an incorruptible digital ledger of economic transactions that can be programmed to record not just financial transactions but virtually everything of value.”
  Don & Alex Tapscott, authors
  Blockchain Revolution (2016)

https://blockgeeks.com/guides/what-is-blockchain-technology/

Blockchain

- “Blockchain is a public electronic ledger that can be openly shared among disparate users and that creates an unchangeable record of their transactions, each one time-stamped and linked to the previous one.”

https://www.computerworld.com/article/3191077/security/what-is-blockchain-the-most-disruptive-tech-in-decades.html

Blockchain

- "A **block chain** is a type of database that takes a number of records and puts them in a block (rather like collating them on to a single sheet of paper). Each block is then 'chained' to the next block, using a cryptographic signature.
- This allows block chains to be used like a **ledger**, which can be shared and corroborated by anyone with the appropriate permissions."
  - A ledger is a record of transactions – e.g. in accounts

https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/492972/gs-16-1-distributed-ledger-technology.pdf

Blockchain Features

- Decentralized data – more than one copy of the data
- Distributed ledger
  - No middle man
  - Tamperproof (very difficult to hack)
- Write once, append only (no delete unless you control the network)

Blockchain

- The real novelty of block chain technology is that it is more than just a database — it can also set rules about a transaction (business logic) that are tied to the transaction itself.
- This contrasts with conventional databases, in which rules are often set at the entire database level
  - In an RDBMS one set of rules applies for the entire schema
- Accuracy of the ledger can be checked by consensus (the term 'mining' is used for a variant of this process in the cryptocurrency Bitcoin)
-

https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/492972/gs-16-1-distributed-ledger-technology.pdf

Blockchain

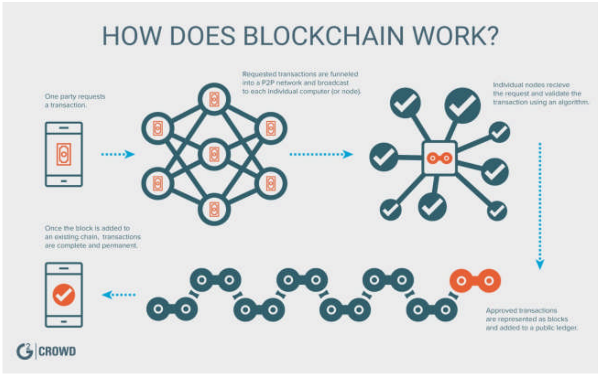
- If the consensus process is open to everyone, the ledger is **unpermissioned**
  - **Unpermissioned ledgers** such as Bitcoin have no single owner
- The purpose of an unpermissioned ledger is to allow anyone to contribute data to the ledger and for everyone in possession of the ledger to have identical copies.
- No one can prevent a transaction from being added to the ledger
  - Full, open, consensus process

https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/492972/gs-16-1-distributed-ledger-technology.pdf

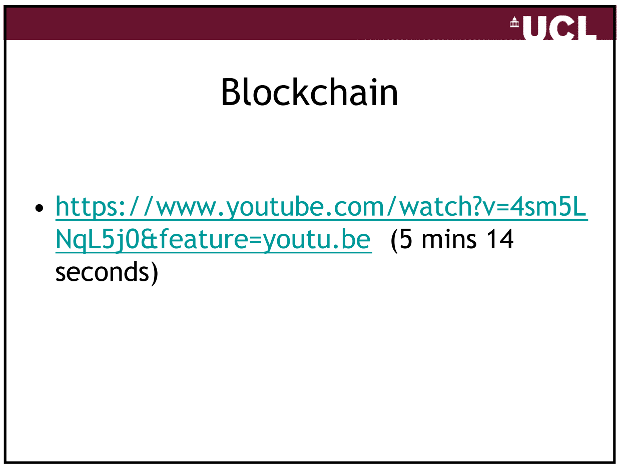
Blockchain

- If the consensus process is not open to everyone, the ledger is permissioned
  - Permissioned ledgers may have one or many owners.
- For a permissioned ledger, entries are checked by trusted authorities — government departments or banks
- Permissioned block chains provide highly-verifiable data sets because the consensus process creates a digital signature, which can be seen by all parties.
- Requiring government departments to validate a record could give a high degree of confidence in the record's security
  - Current data sharing using paper is very open to forgery

https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/492972/gs-16-1-distributed-ledger-technology.pdf

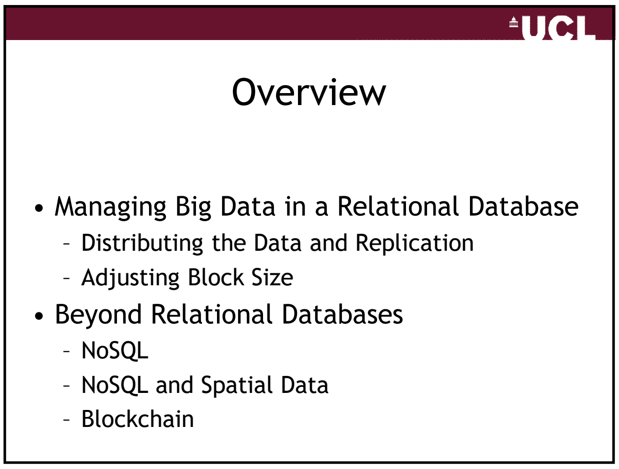

HOW DOES BLOCKCHAIN WORK?
One party requests a transaction.
Requested transactions are funneled into a P2P network and broadcast to each individual computer (or node).
Individual nodes recieve the request and validate the transaction using an algorithm.
Approved transactions are represented as blocks and added to a public ledger.
Once the block is added to an existing chain, transactions are complete and permanent.
G2 CROWD

https://www.g2crowd.com/categories/blockchain



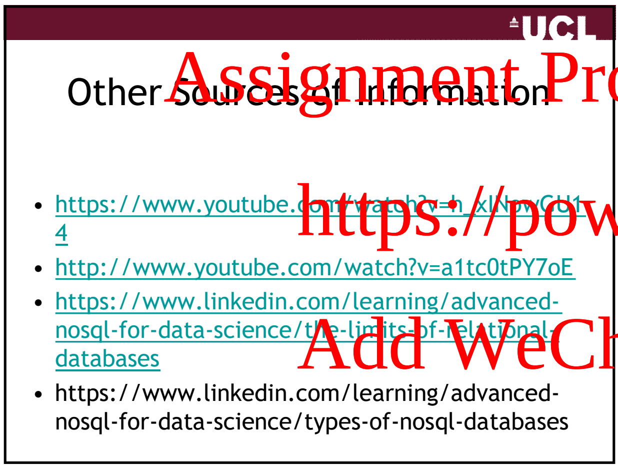
Blockchain

- https://www.youtube.com/watch?v=4sm5LNqL5j0&feature=youtu.be (5 mins 14 seconds)



Overview

- Managing Big Data in a Relational Database
  - Distributing the Data and Replication
  - Adjusting Block Size
- Beyond Relational Databases
  - NoSQL
  - NoSQL and Spatial Data
  - Blockchain



Other Sources of Information

- https://www.youtube.com/watch?v=h_xlNowCU14
- http://www.youtube.com/watch?v=a1tc0tPY7oE
- https://www.linkedin.com/learning/advanced-nosql-for-data-science/the-limits-of-relational-databases
- https://www.linkedin.com/learning/advanced-nosql-for-data-science/types-of-nosql-databases