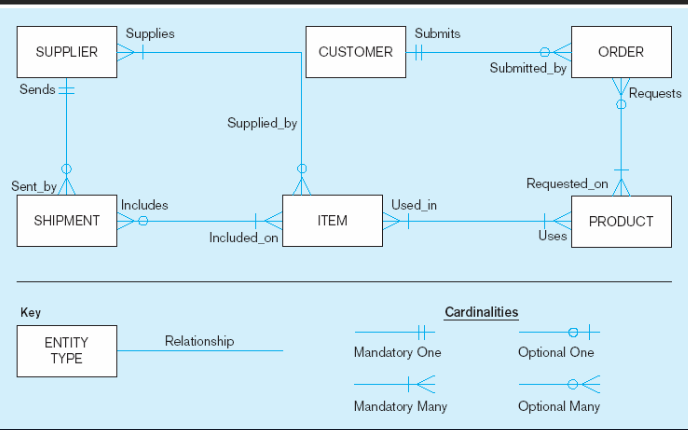
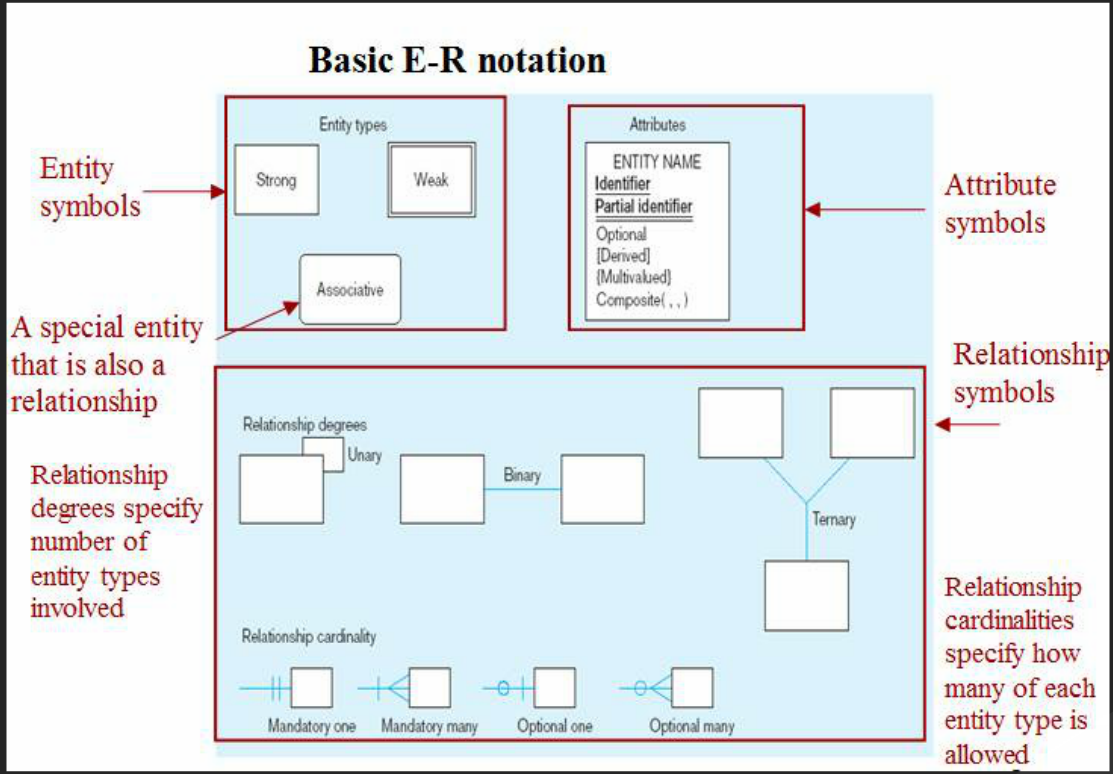


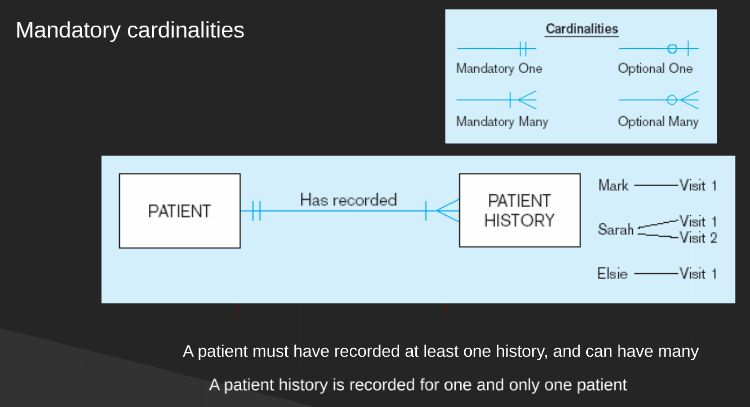
We apply the normalization of tables because it will fix any anomalies that will arise in the table.  
  
1NF

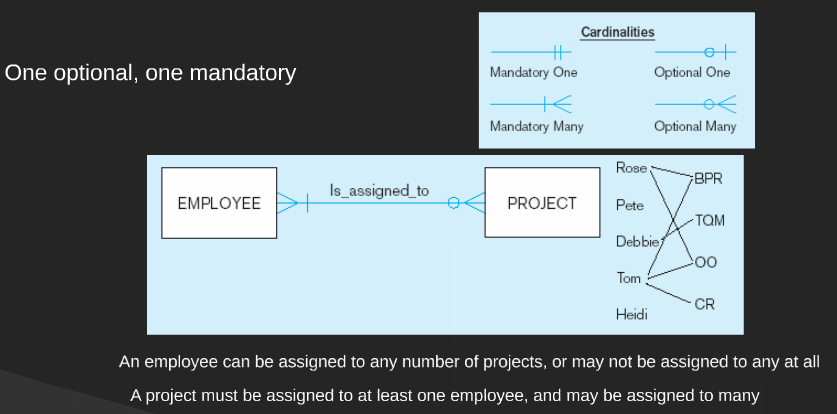
- Has no repeating columns (storing the same information in the same table)  
- Values of each column are atomic (no multi-values)  
- Each table has a Primary Key (minimal set of attributes which can uniquely identify a record).

2NF  
- All requirements of 1NF must be met.  
- Redundant data across multiple rows of a table must be moved to a separate table  
- Resulting tables must be related to each other by use of FK

3NF  
- All requirements of 2NF must be met  
- Eliminate fields that do not depend on the primary key (those that are not dependent not only to PK but other fields must be moved to other table)  
  
Data Anomalies – inconsistencies in the data stored in a database as a result of an operation such as update, insertion or deletion. It could happen when data stored is saved in different places but updated only in few.  
  
  
**ENTITY DIAGRAMS**

 **Entity SHOULD BE:**   
- Object that will have many instances in the DB   
- Has a lot of attributes   
- Object that we are trying to model  
Individual objects are entities, group of same type objects are entity types. Represented by round or square corners

**Attributes:** Property of characteristic of an entity or relationship type.  
-required vs optional  
-simple vs composite  
- singe-valued vs multi-valued  
- stored vs derived  
- identifier  
  
**Keys** – are data item that allows us to uniquely identify individual occurrences or an entity type  
candidate key is an attribute or set of attributes that identify individual occurrences or entity type.  
**Entity type** – may have one or more keys, the one selected is primary. Composite key is a candidate key. Consists of two or more attributes.  
Names of attributes are **underlined  
  
Relationships – meaningful association between entities, where one is included from each type   
Types are represented by different lines in the diagram.  
  
Cardinality – crow’s foot represents many, and 1 represent single.  
one-to-one – each entity in the relationship will have exactly one related entity  
many-to-many – one side will have many entities but an entity on the other side will have a max of one entity (example – a movie may have many stars and each star may act in many movies)  
one-to-many (example: A doctor may have many patients, but a patient is assigned to one doctor)**

 **Strong entities – exist independently of other types of entities, with unique key and with a single line  
Weak entities – depend on a strong entity, no unique key, double lined, box has double line too ,links strong entities to weak**

**SQL**

SQL – special programming language used for managing data held in a relational DBMS   
String types  
CHAR(n) – fixed length char data, max len = 2000bs  
VARCHAR2(n) – variable length char data, max = 4000bs  
LONG – variable length char, up to 4GB max 1perTable  
DATE – two digits for day, 3 for month , 2 for the year   
NUMBER(L,R) – general purpose numeric   
L – max number of decimal digits , R – number of digits from the decimal point to the least   
example NUMBER(5,2) – max value 999.99 , NUMBER(5) – max value 99999  
INTEGER – signed int  
  
NOT NULL – columns must contain value, must be specified at column level  
PRIMARY KEY   
ALTER is used to delete edit or add a statement to a table   
- adding column (example: ADD address VARCHAR2(30))  
- modify column name or type (example: MODIFY supplier\_name CHAR(100) NOT NULL;)  
- drop table (example: DROP COLUMN supplier\_name)   
- rename column (example: RENAME COLUMN supplier\_name TO sname;)  
- rename table (example: RENAME TO new\_table;)  
  
INSERTING DATA:  
**INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)  
VALUES (1, 'Ramesh', 32, 'Ahmedabad', 2000.00 );**

**Populate one table using another table   
INSERT INTO** first\_table\_name [(column1, column2, ... columnN)]   
 **SELECT** column1, column2, ...columnN   
 **FROM** second\_table\_name  
 [WHERE condition];

**TO RENAME**   
SELECT surname, first\_name as “First Name”  
FROM employees;

CREATE QUERIES:  
SELECT grade\_level, min\_salary, max\_salary  
FROM jobgrades;   
or if we want for all   
SELECT \* FROM jobgrades;

WHERE CustID = 10;  
WHERE min\_sal> 35000.00;  
WHERE Country = “UK” and City = “Coventry”;  
WHERE Country = “ UK” and (City= “Coventry” OR City= “Sunderland”)  
  
INITAP (Column/expression) – Converts most left character into Upper Case.  
ORDER BY {col1,expr,position\_of\_col} [ASC|DESC]];  
examples:   
ORDER BY surname ASC, first\_name DESC;  
  
UPDATE employees  
SET salary = 25000  
WHERE surname = “Smith”;  
or  
WHERE salary BETWEEN 30000 and 45000;

**BIG DATA**

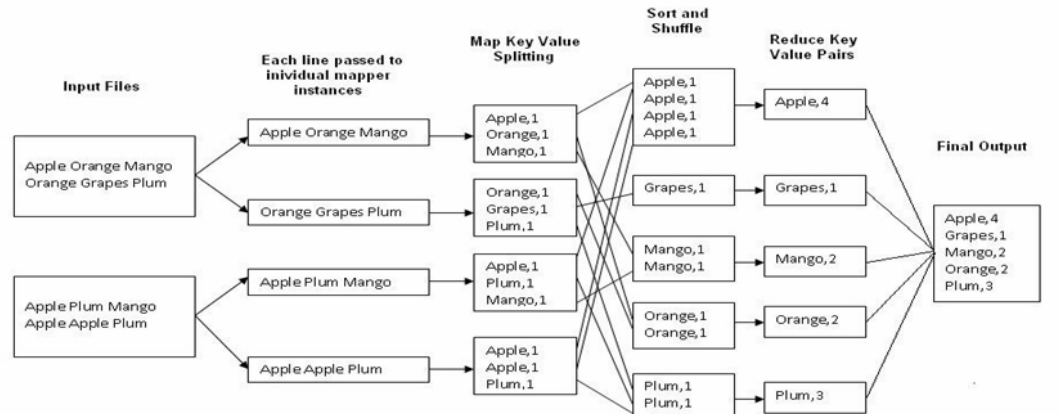
Big data – sets which size is beyond the ability of typical DB software tools to capture, store or manage and analyze.  
Structured data – data placed in specifically designed DB according to specific schema. Everything else is unstructured. (Most is UNSTRUCTURED)

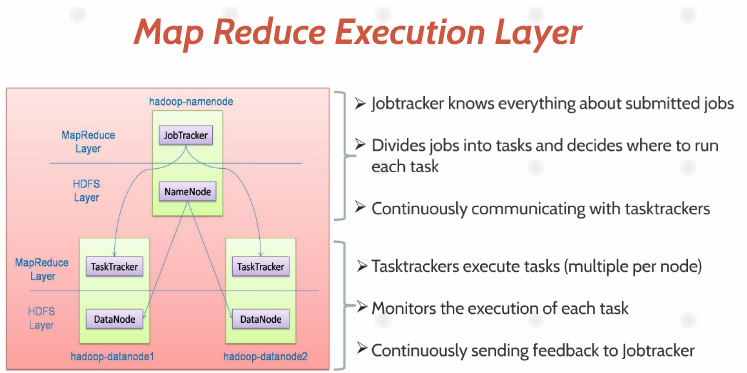
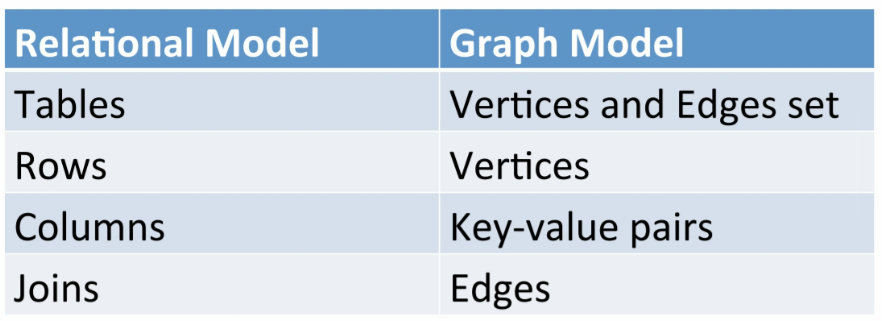
THE FOUR “V’s”   
- Volume – the smarter it gets the more data it will gather  
- Velocity – even tho the information is small the income of data is huge  
- Variety – new data is provided every day and is any kind unstructured and structured   
- Value – locating is a lot of work  
  
ACQUIRE => ORGANIZE => ANALYZE => DECIDE

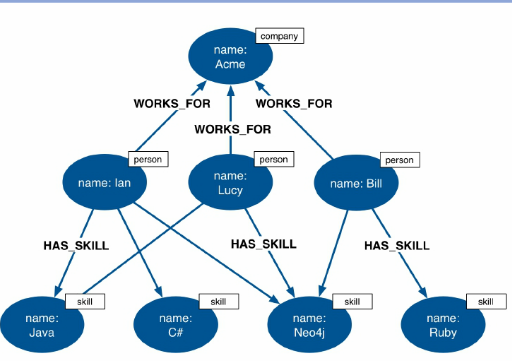
Discovering a relationship, Discovering Behavior, Discovering something new

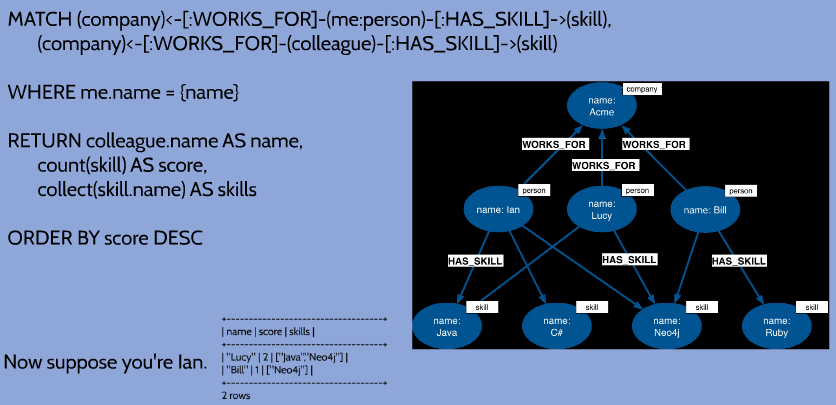
**MAP REDUCE** – Designed for simple data analysis. Uses parallel programing techniques, model and framework. Designed with scalability and fault tolerance. Cost effective and easy to use  
-Functionality-   
@Split the data into smaller chunks  
@Map according to the mapping key.   
@Reduce and merge all related data associated with the key  
PROS: Simplicity, Applicability, Scalability   
CONS: Restricted, does not provide solutions to graphs   
 **HADOOP – open source framework for processing data sets and running applications on clusters of computers. Consists of MapReduce + Hadoop distributed file system (HDFS).**It provides massive storage for any kind of data, processing power and the ability to handle virtually limitless tasks or jobs

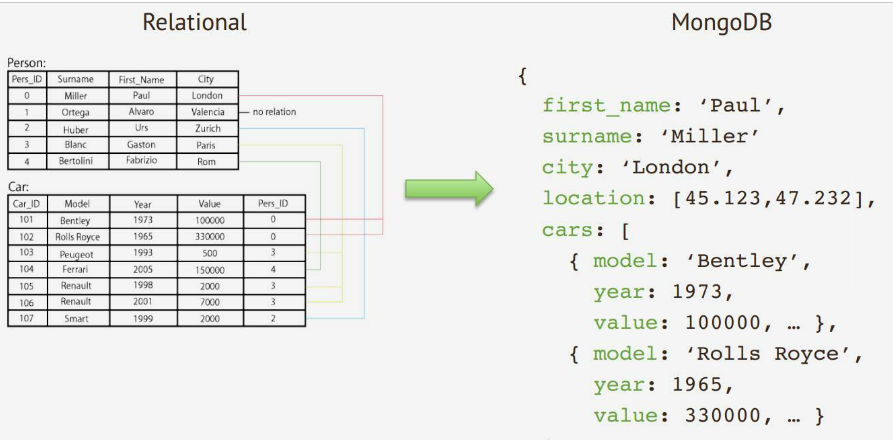
Used for: Long data analysis (write once & read often), data warehouse modernization, Fraud detection, risk modeling, image classification, graph analysis, social sentiment analysis.  
<https://www.youtube.com/watch?v=xJHv5t8jcM8>

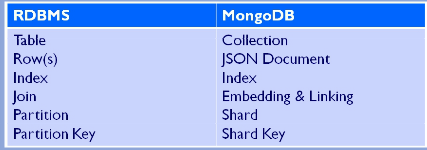
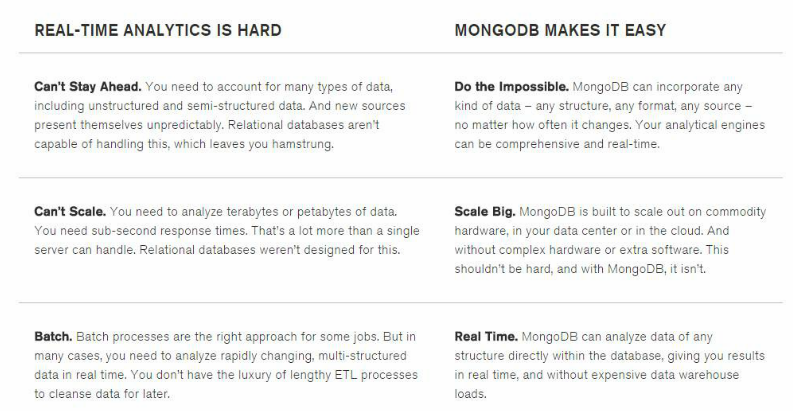


Data is distributed around the network. Every PC can host data and is replicated in case of fault. It is centralized.   
  
HADOOP ARCHITECTURE:  
Namenode: Filenames, permissions, information and central manager for the file system  
Datanode: provide storage for the objects and communication to other nodes for replication  
TaskTracker: accept and runs map, reduce and shuffle. Provides a number of slots for tasks (logical CPUs) if all slots are occupied, run the task on the same rack  
JobTracker: secondary backup of jobs, manager for running MapReduce.  
  
HADOOP STEPS  
STEP\_1 : MapReduce splits files into pieces (64-256mb) copies it on other machines on the cluster.  
STEP\_2: Mapping tasks. Reads and puts into key/value pairs. Applies map operation to each pair. Stores result on the local disk. File is then sorted on output key, then partitioned based on the key values. Locations are forwarded to Master and then they are send to relevant reduce workers   
STEP\_3: Fetch input – Sort by key – for each key, apply reduce operation - write result on file – return location of result files to Master.  
HADOOP Related projects: Avro , Chukwa , Hive, Pig , Spark and etc.  
  
  
  
**GRAPH Database – a graph is a collection of nodes and edges (relationships) that connect pairs of nodes together. Graph DB can be thought of as a key-value store, with full support for relationships.   
Relationships connect two nodes and both nodes and relationships can hold an arbitrary amount of key-value pairs.**

nodes – instances of entities  
entties – unique conceptual identity, change attribute value but identity remains the same  
  
Every relationship must have a start node and end node.  
Graphs can always grow by adding more nodes and new subgraphs to an existing structure. Huge flexibility.and speed due the structure.  
  
Navigating is done traversal (moving from one to another) can’t move to two different nodes following directions. From specific nodes to others. Search accordingly. Depth or Breadth first  
Depth – follows one path to its end then returns and follows the second, third and etc.  
Breadth – follows all the first steps first then lists the depth 2 paths and so on.  
CYPHER – query declarative language  
use “MATCH” operator to find answers to query  
use “WHERE” to find relationships between nodes. And “CREATE” example (a)-[r.Likes]->(b)  
MERGE – ensures that the graph pattern exists , by creating new nodes and relationships or reusing old ones.  
DELETE – remove nodes, relationships and properties  
SET – Set values  
FOREACH – Performs an updating action for each element in a list.  
UNION – merges results from 2 queries  
START – specified onr or more starting points  
  
example -   
(company) <- [:WORKS\_FOR]-(person)-[:HAS\_SKILL]->(skill)  
  
Which people, who work for the same company as me, have similar skills to me?  


CREATE (a:Person { name: “Tom Hanks”, born:1956})-[r:ACTED\_IN {roles: [“Forrest”]}] -> (m.Movie {title:” Forrest Gump” , released:1994}]  
  
CREATE (alice:User {username:”Alice”}],  
 (bob:User {username:”Bob”}],  
 (edward:User {username:”Edward”}]  
 (davina:User {username:”Davina”}]  
(alice)-[:ALIAS\_OF]->(bob)  
MATCH (bob:User {username:”Bob”}]  
 (Davina:User {username:”Davina”}]  
  
CREATE (bob)-[:EMAILED]->(Edward),  
(bob)-[:CC]->(Davina)  
(bob)-[:BCC]->(Edward)  
  
MONGO DB – modeling data as document based solutions to big data. Scales by adding more server.  
stores data in binary JSON (java script object notation). Each document has an ID generated from timestamp , machine and process ID’s and a counter



hosts databases > DBs hold set of collections > collection holds a set of docs > a doc is a set of key-value pairs  
  
change or add a new field – db.t.update({ln:'Smith'},{$set:{age:30}})  
Removing things from tables - db.t.remove({fn:'John'})  
locate a key or value - db.t.find({fn:'John'})  
locate from/where - db.t.find({fn:'John',ln:'Smith'})  
  
SQL  
SELECT country FROM t  
WHERE fn='John'  
ORDER BY age DESC;  
  
MongoDB  
db.t.find(  
{fn:'John'},  
{country:1, \_id:0}  
).sort({age:1})  
  
  
PROPERTIES  
Key-value queries returning results baesd on any field in the doc.  
Range queries returning results on definded values (greater than , less than , between)  
Text Search queries returning results in revelance order based on text arguments using Boolean  
• Document-Oriented storage  
• Full Index Support  
• Replication and High Availability  
• Auto-Sharding  
• Querying  
• Fast In-Place Updates  
• Map/Reduce functionalit

* Aggregation and MapReduce queries
* Native Hadoop integration for offline analytics

SHARDING – the process of storing data records across multiple machines and mongodb approach to meetin in demands of data growth.  
  
As the size of the data increases, a single machine may not be enough to store the data or provide an acceptable read and write. Sharding solves the problem with horizontal scaling. With sharding, you add more machines to support data growth and the demands of read and write operations.

NoSQL – just a term for systems which are not SQL but still some use SQL. They can be faster and analyze big chunks of data. RDBMs can be made to run over clusters, but they were not designed for it.  
  
Different records might need to contain different fields or different number of fields.   
A field might need to hold several values. Availability and low latency may be more important than absolute integrity.  
  
NoSQL Family  
Neo4J , HyperGraphDB (Graph DB’s)  
MongoDB , CouchDB (Document Oriented)  
Cassandra , HBaseDB ( Column Family)

DB ON CLUSTERS Acts like one big machine , can do backup , check points and logs. Single point of failure if a system or the network goes down. Resilence achieved with replication.  
  
  
Synchronous replication - All replicas are updated on every write. Reads are up to date. Works best if fewer writes are made.  
Asynchronous – reads can be out of date, eventual consistency, works best if reads , peer to peer , writes are prop-gated asap, but reads do not have to wait.  
Consistency is better reading from the primary  
  
Peer-2-Peer Replication – allows all nodes to accept read and write, fast and resillent, problems when 2 peers receive conflicting updates  
  
Range Based Keys – divdes the data set into ranges determined by the shard key values to provide range based partititoning. Chunks are range of value from min value to max  
Hash Based Keys – two docs with close shard key values are unlikely to be part of the same chunk. This ensures random distribution of a collection in the cluster

Data Mining – discovering patterns in a large data set or using methods to make something understandable from data info.   
It comes from from similarities to searching for valuable business info in a large DB , and mining to find a valuable info  
  
Cassandra DB – Apache column family DB , Peer to peer distributed data store.  
Random partitioning – based on md5 hash of key value , adjacent keys not kept togheter  
Order partitioning – based on raw key order , good if you want to access rows in adj groups , bad as causes “hotspots”.  
  
Works better when denormalized, has no joins. Leads to poor performance as you have to look up both UserID and ProductID for each row.  
Syntax  
SELECT cols FROM col\_family  
WHERE condition   
**(cols can be comma seprarated list, or range)**

INSERT INTO col\_family ( col, col, … ) VALUES (val, val, ..)  
USING option  
(values: can be literal, a list , set or a JSON array of literals