ISIT 312 Big Data Management

Exam Contents:

Skipped chapters: 01, 04, 06, 08, 12, 13,

1. Hadoop architecture (HDFS, YARN, MapReduce)

Hadoop receives near to linear scalability which means after a set amount of nodes, it is better to have another cluster.

HDFS is designed for:

- Large files
- Stream data access
- Commodity hardware

Not for:

- Low-latency data access
- Lots of small files
- Multiple writers, arbitrary file modifications

Key components:

NameNode:

- HDFS Master node
- Manage metadata pertaining to the file system
- Manage user access to the data files
- Map the data blocks to the datanodes in the cluster
- Perform file system operations (eg. opening and closing of files)
- Provide registration service and periodic heartbeats for datanodes

DataNode:

- HDFS SlaveNode
- Manage block storage and access for reading, writing data and block replication
- One DataNode contains multiple HDFS blocks
- Provides block storage by storing blocks on slave's file system
- Fulfill read and write request
- Replicate data across cluster
- Send heartbeat to show that it is healthy and alive as well as block report to show the blocks managed by the datanode to the namenode

Yet Another Resource Negotiator - YARN

It is an OS and a cluster resource management

It is a core subsystem in Hadoop that is responsible for governing, allocating and managing the distribution of resources available on a Hadoop cluster

YARN provides core services via two types of long running daemons

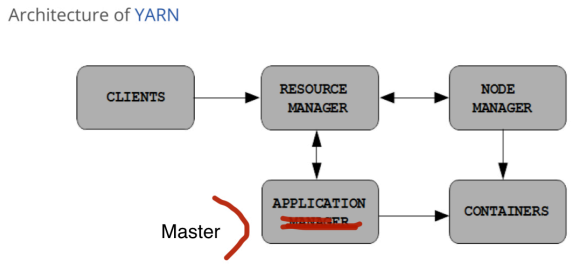
- Resource Manager:
    - Runs on master node

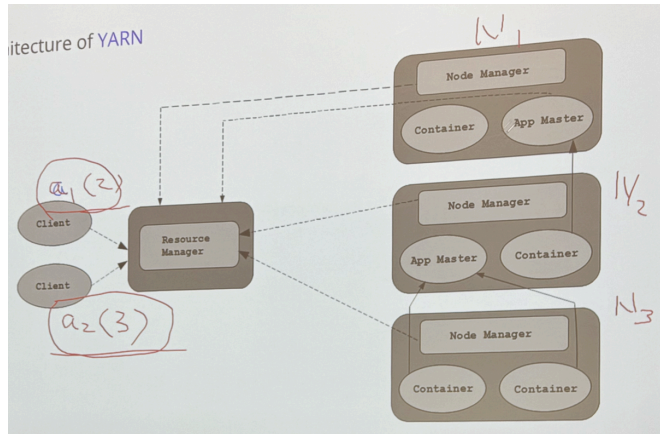
    - One per cluster to manage use of resource
    - Grants computing resources to applications running in the cluster
    - Tracks heartbeats from Node Managers to manage Data Nodes
    - Runs scheduler to determine resource allocation among cluster
    - Manage cluster level security
    - Manages the resource request from ApplicationMasters
    - Monitors the status of ApplicationMaster and restarts that container upon its failure
    - Deallocates the containers when the application completes or after they expire

- Node Manager:
    - Runs on slave nodes in the cluster
    - Launch and monitor containers
    - Application Master: an application that submits an extra resource request to run the application when the resource manager estimated wrongly the amount of containers needed for the application. It is usually run on the first container allocated by the resource manager for each application
    - Communicates with resource manager through heartbeats and container status notification
    - Register and starts application processes
    - Launches both application master and the rest of an application's resource containers(map and reduce tasks that run in the containers) on request from application master
    - Oversees the lifecycle of the application containers
    - Monitors, manages and provides information regarding the resource consumption (CPU/memory) by the containers
    - Provides auxiliary services to YARN applications, such as shuffle and sort operation

A client is the program that submits job to the cluster which may also be the gateway machine that the client program runs on
A job, also called an application, contains one or more tasks. A task in MapReduce can be either a mapper or a reducer task.
Each mapper and reducer task runs within a container

- It is a logical construct that represents a specific amount of memory and other resources
- May also refer to the running environment of an application





MapReduce

It is the most important processing framework in Hadoop

Many processing languages are abstractions of MapReduce, eg. Pig and Hive or heavily influenced by MapReduce concepts eg. Spark

MapReduce is also a platform and language independent programming model at the heart of most big data

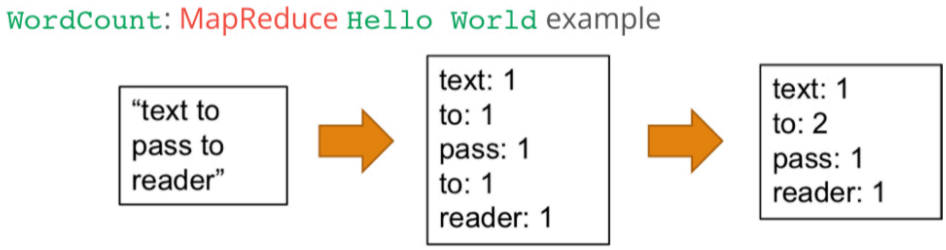
There is a map phase and reduce phase

Limitations of early distributed computing and grid computing framework:

- Complexity in parallel programming
- Hardware failures
- Bottlenecks in data exchange
- Scalability problem
- MapReduce uses key value pairs for processing data

Word count is to count the occurrence of each word in a sentence:

Input > key value pair (where word is the key and value is the occurrence) > reduced phase



MapReduce will happen in every node and then be followed by a merging process

MapReduce frees the user from the low level communication and coordination of nodes and processes. With implementation of Hadoop 2, programmers can focus on MapReduce implementation and configuration. It also reduces the overhead of network transmission.

2. MapReduce Model

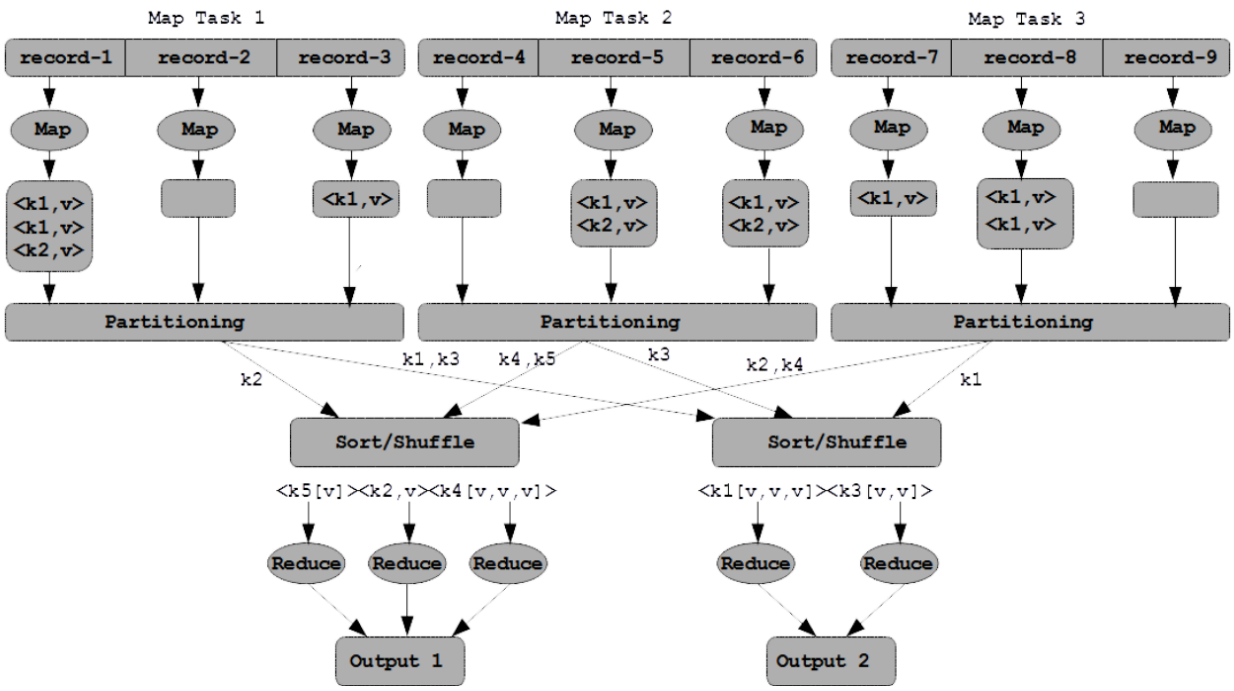
Key-Value pairs: MapReduce basic data model
Key is an identifier, eg. name of an attribute

- In MapReduce, key is not required to be unique

Value is a data associated with a key

- It may be simple value or complex object

MapReduce data processing model flows in a sequence:
Map → Partition → Shuffle and Sort → Reduce
Map phase and Reduce phase is implemented by the developer while Partition as well as Shuffle and Sort are handled by the platform.



Map phase -
Uses input format and record reader function to derive records in the form of key-value pair for the input data.
Huge data input will be segmented into portions called blocks and if there are n blocks, there will be n map tasks.
Each map task operates on one filesystem block
In the image above, record-1, record-2, record-3, all have their own map task.
It then outputs data in key value pair, “<k1,v>, <k1,v>, <k2,v>”

Reduce phase -

The input of the Reduce phase is the output of the Map phase (via shuffle and sort)

The output for each reduce() function is zero or more key-values. It can also be an input to another Map phase in a complex multistage computational workflow.

In a simple reduce function:

```
reduce (k, list ) =
        {
          sum = 0
          for int i in list :
            sum + = i
          emit (k, sum)
        }
```

This function reduces (k1, [1,1]) to (k1, 2).

Reducer can be also done with some predefined methods such as max() and avg().

Shuffle and Sort -

It is a process where data are transferred from Mapper to Reducer

The purpose is to minimize data transmission through a network.

So after the Map phase, the data is then shuffled and sorted to group and merge together. Through this process, it can greatly reduce the amount of data that needs to be transmitted to the Reduce phase

Combine phase -

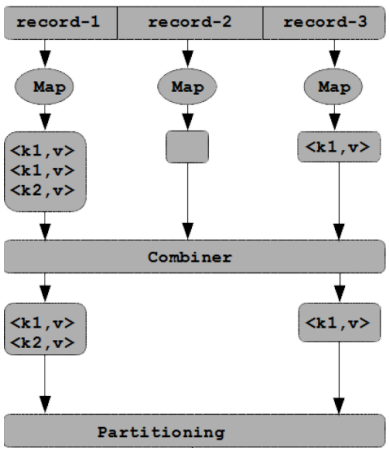
When the Reduce function is commutative and associative then it can be performed before the Shuffle and Sort phase.

In that case, the Reduce function is called a Combiner function.

Sum and count is commutative and associative but average is not.

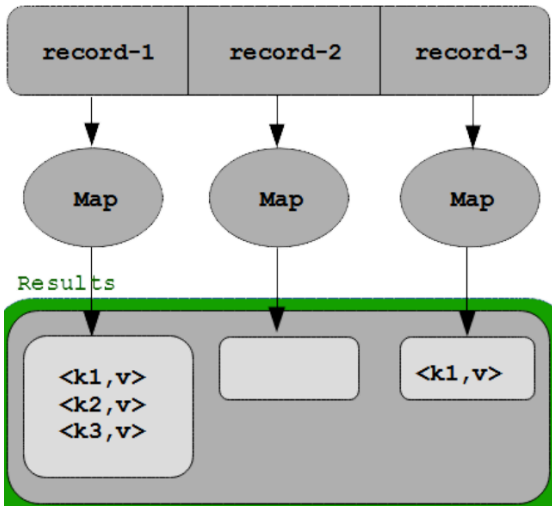
Combiner can further reduce the amount of data transferred to reduce phase to reduce the data transmitted.

Structure of Combine Phase:



MapReduce application may contain zero Reduce tasks, which is known as Map-Only application. (eg. image processing, file format conversion)

Map-only MapReduce:



Example for MapReduce:

For a database of 1 billion people, compute the average number of social contacts a person has according to age

In SQL like language, can finish in 3 lines:

```
SELECT age, AVG(contacts)
FROM social.person
GROUP BY age
```

If the records are stored in different data nodes then in Map and Reduce function is the following:

```
function Map is
  input: integer K between 1 and 1000, representing a batch of 1
  million social.person records
  for each social.person record in the K-th batch do
    let Y be the person age
    let N be the number of contacts the person has
      produce one output record (Y,(N,1))
  repeat
end function
```

```
function Reduce is
   input: age (in years) Y
   for each input record (Y,(N,C)) do
     Accumulate in S the sum of N*C
     Accumulate in C_new the sum of C
   repeat
   let A be S/C_new
   produce one output record (Y, A)
 end function
```

Expected results
(18, ?),
(19, ?),
(20, ?),
⋮
(80, ?)

K = age
=18 - example.
in the let statement,
Y = key
N = number of contact followed by the occurance.
data eg.: key, value
p1 (18, 4) → (18, (4, 1))
p2 (18, 10) → (18, (10, 1))
p3 (20, 11) → (20, (11, 1))
P4 (18, 4) → (18, (4, 1))

partitioning and shuffle and sort

⇓

Partition 1 { (18, (4, <1, 1>)
(18, (10, <1>))

Partition 2 { (20, (11, <1>))

* Partitioning will try balance the to make full use of the resources.

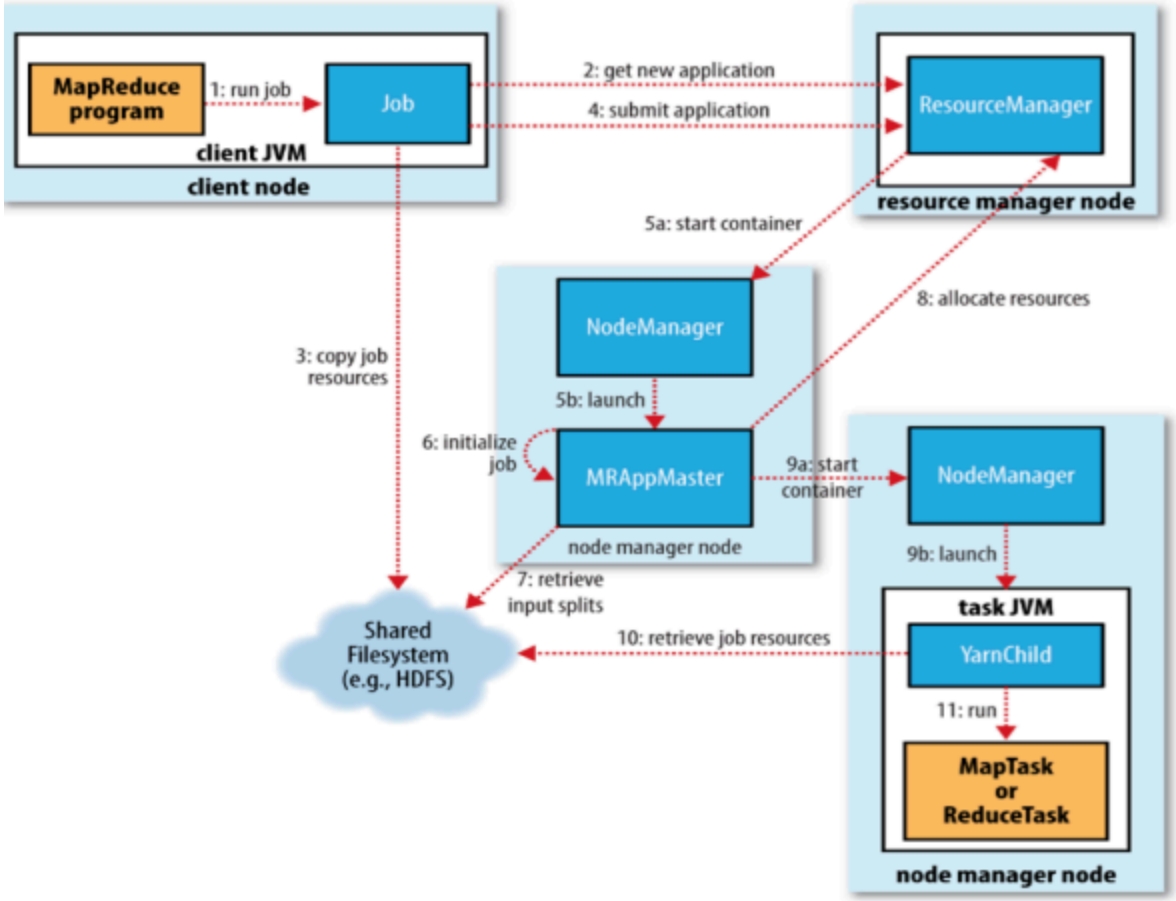
⇓ reducer, one for each partition

Partition 1 { (18, (4, <2>))
(18, (10, <1>)) } ⇒ go through reduce function

$\left(18, \frac{4 \times 2 + 10 \times 1}{2+1} = 6\right)$

Ans = (18, 6)

Partition will balance the load so that all the Reduce functions can finish around the same time. This will reduce the use of resources as well as increase the overall performance.



3. Basic Hadoop commands

4. Hive concept, structure

It is a software system that provides a tabular view of data stored in HDFS and SQL-like methods for manipulating data in HDFS.
Contrary to Pig, Hive provides SQL-like abstractions on top of MapReduce
Language used is called Hive Query Language (HQL) which almost implements SQL-92 standard.
HQL provides a tabular view of data and it can be used to access data located in HDFS
Hive frees data analysts from Java MapReduce programming skills (not completely)
HQL statements are parsed by the Hive client and translated into a sequence of Java MapReduce operations, which are later on processed by Hadoop.

Process of HQL queries:
Hive → HQL statement → Parsing HQL (Using Hive metastore), Create processing plan, Submit job, monitor progress ←→ MapReduce ←→ HDFS

Deployment and Configuration
Hive is available on all commercial distributions of Hadoop
Relational embedded database system, Derby, is used for implementation of metastore, but it is possible to use other relational database systems for implementation of metastore like MySQL.
To use Hive, Hadoop and HDFS must be up and running.
A top level view of data provided by Hive consists of databases and tables

Metastore -
- Contains the mapping of tables to the directory locations in HDFS
- Relational database read and written by Hive client.
- Uses input/output formats and SerDe (Serialization/Deserialization) functions to understand how to read and write data to and from input data files.
- Input format tells Hive how to read records from the data file and Output format tells Hive how to write records to the data file

Interfaces -
Hive provides command line interface (CLI) that accepts and parses HQL commands.
Hive also provides JDBC/ODBC connector (drivers) to work with existing database systems such as:
- Beeline (CLI)
- Oracle SQL Developer (GUI)
- Talend Open Studio (Data Extraction, Transformation, Loading (ETL) and integration tools)
- Jasper reports, QlikView (Business intelligence reporting tools)
- Microsoft Excel (Analysis), Tableau (Visualization)

Hive provides a storage handler mechanism to integrate with HBase
HUE (Hadoop User Experience) a website to interact with HDFS and Hive
HCatalog enables the use of a combination of tools like Pig, MapReduce, Hadoop, Hive

Hive vs relational DBMS
Similarity - (Most similarity is based on the representation of the data)
- Tabular view of data objects in HDFS
- Directories and files viewed as tables
- Types of columns in table
- Access to Tables through HQL very similar to SQL
- API interface the same as JDBC programming interface

Differences - (Significant difference is
- Load and read-only data management system based on implementation of HDFS
- It is still possible to access data visible in tabular format in Hive directly through HDFS
- Hive stores coarse grained information, DBMS stores fine grained information
- Hive work on top of HDFS while DBMS focuses on database
- Hive does not store the data in the table and does not change the format of the data when stored into HDFS

Databases
It is a collection of conceptually related tables and is implemented as a folder/directory in HDFS. A default database is located at /user/hive/warehouse and a new database creates a subfolder in /user/hive/warehouse

Internal Table:
When you drop the internal table it deletes data, structure and file for managed tables
Stores under the hive metastore warehouse directory
Acid table is only available to be used when it is an internal table
Fully manages the life cycle (add/delete data, create/drop table) of internal tables and data in the internal tables

External Table:
When it is dropped, deletes the metadata but the file on hdfs is not deleted
Stored externally on HDFS
Should use an external table by default as other applications like Spark and Pig can still access the data as they reference the same paths. It can also be retrieved back in Hive with the same command.

5. Basic HQL commands

6. Data warehouse concept and design

Online transactional processing (OLTP)

It is behind many everyday transactions from ATMs to instore purchases to hotel reservations as well as password changes and text messages.

Traditional database systems designed and tuned to support the day to day operation:
- Ensure fast concurrent access to data
- Transaction processing the concurrency control
    - Allows simultaneous access to multiple users and processes
    - Multi user access to the same data
- Focus on online update data consistency
- Does a lot of transaction
- Rapid processing
- Provide indexed datasets for rapid searching, retrieval and querying
- Supports the company infrastructures

OLTP database characteristics:
- Detailed data
- Does not include historical data
- Highly normalized

- Poor performance on complex queries including joins and aggregation (sum, count, avg)

Online analytical processing (OLAP)

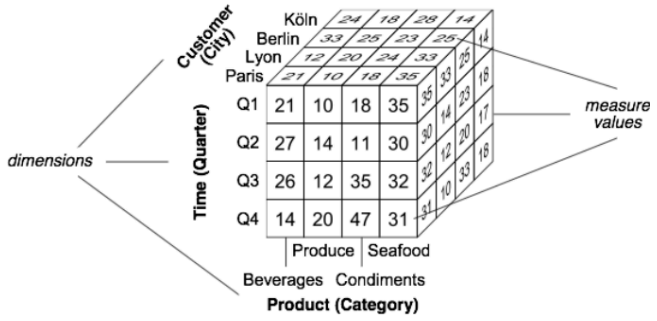
OLAP is a system for performing multi-dimensional analysis at high speeds on large volumes of data. It is ideal for data mining, business intelligence and complex analytical calculations as well as business reporting functions like financial analysis, budgeting and sales forecasting.

OLAP characteristics:

- Focuses on analytical queries
- Normalization not good for analytical queries, reconstructing data requires a high number of joins
- Supports a heavy query load
- Oriented to access few records
- Not efficient in using indexing techniques
- Queries usually include aggregation
- OLAP data cube (Multidimensional data model)
- Data warehouses to support OLAP
    - Large repositories that consolidate data from different sources and updated offline
    - Follows multidimensional data model
    - Designed and optimize to support OLAP queries

7. Multidimensional data model

A data cube is the view of data in n-dimensional space.
It is composed of dimensions and facts
Dimensions is the perspective used to analyze the data
Eg. Three dimensional cube for sales data:



Facts (associated numeric values called measures)

- Total distance travel
- Total number of trips

Measures (what is filled in the cells of the cube, eg quantity, kilometer, money, count of trips)
Dimensions (perspective to analyze the data, attributes describe dimensions eg product num and unit price)

Hierarchies (levels in the dimension)

Cube shows:

- X is product dimension, Y is customer city dimension, Z is time dimension
- Attributes describes dimensions
    - Product dimension may have attributes ProductNumber and UnitPrice (not shown in figure)
- Cells or facts of a data cube have associated numeric values called measures
    - Eg. total sales is measures
- Each cell of the data cube represents the quantity of units sold by category, quarter and city.
- Instances of a dimensions are called members
    - Eg. Seafood and produce are members of the product at the granularity category

Data granularity is the level of detail at which measures are represented for each dimension of the cube.
The data cube may be sparse or dense where sparse means that there are a lot of empty cells without information.

Hierarchies will allow data to be viewed at several granularities

- Define a sequence of mappings relating lower-level, detailed concepts to higher-level ones
- The lower level is called the child and the higher level is called the parent
- The hierarchical structure of a dimension is called the dimension schema
- A dimension instance comprises all members at all levels in a dimension

In the image, each dimension can show finer granularity by going down the hierarchies of the sections.
Eg.

- Time: All time → Yearly → Semesterly → Quarterly → Monthly → Daily
- Customer: All → Continent → Country → State → City → Individual customer

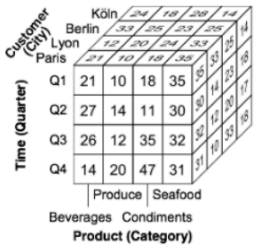
Aggregation of measure changes the abstraction level at which data in a cube are visualized
Measures can be:

- Additive: Can be meaningfully summarized along all the dimensions, using addition
    - Sales revenue can be added across all dimensions (Region, Time, Product etc.)
- Semi Additive: Can be meaningfully summarized using addition along some dimensions
    - Inventory can be added across some dimensions (Product and Inventory)
- Non Additive measures: Cannot be meaningfully summarized using addition across any dimensions
    - Profit margin (percentage) cannot be added

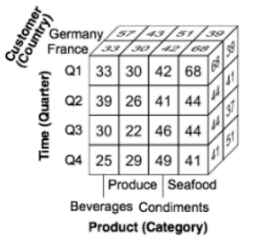
- Distributive: If application is distributed, then it can be combined before shuffle and sort. Functions like count, sum, min, max are distributive where the results will be the same even if the set of numbers is broken up to a few smaller sets.
- Algebraic: Aggregation function which is formed when two functions are used to form a single function like the average function which requires the sum function and count function

OLAP Functions:

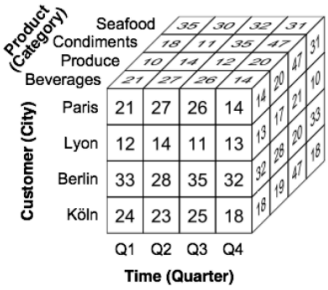
- Roll-up
    - Making data coarse
- Drill-down
    - Making data finer
- Sorting
    - Sort product by category section etc.
- Pivoting
    - Swap the axis of the cubes without affecting granularities
- Slicing
    - Slice piece by piece, removing one axis
- Dicing
    - Cuts out a part of the cube based on the dimensions selected



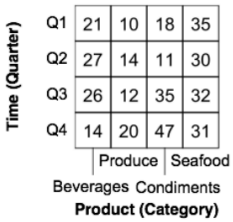
Original cube



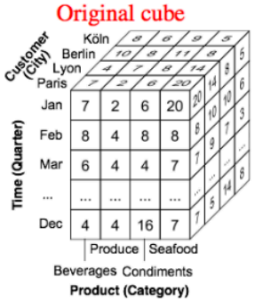
Roll-up to the Country level



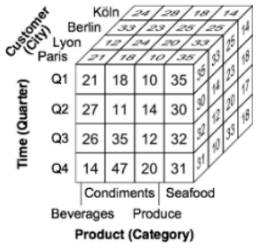
Pivot



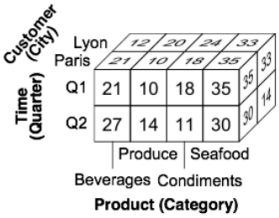
Slice on City='Paris'



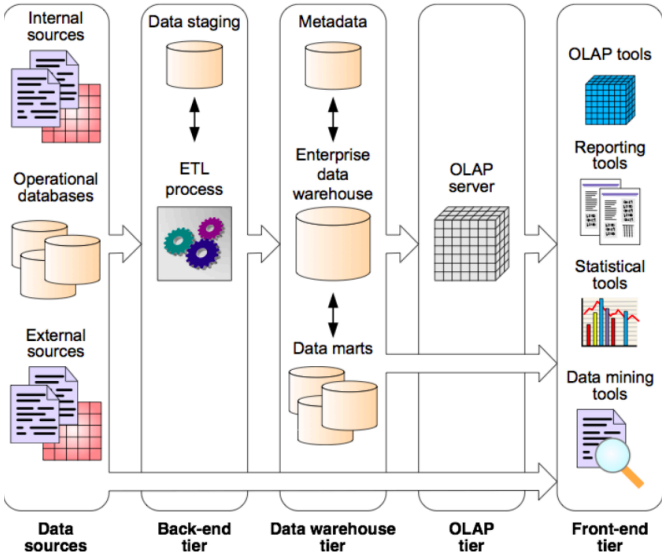
Drill-down to the Month level



Sort product by name



Dice on City='Paris' or 'Lyon' and Quarter='Q1' or 'Q2'



Data from multiple sources are sent to the back end tier.
The data will then be extracted from different resources and the transformation will transform the data from the original format to the target format. If the original data is unstructured data and is not normalized, it would transform to become structured and normalized. It will also filter out the noisy data. Last step in this process is the loading which loads the data into the data repository. The ETL process is usually performed offline and to maximize the computational power of the data warehouse, which means the load process is performed in off peak hours. ETL has data staging to record the current stage of the ETL process.
The data warehouse tier can either be a huge data warehouse or contains several small databases and it may contain metadata which is the indexing of the data in the big repositories. The data is then send to the computer clusters to perform the OLAP operations to perform the online data analysis

Conceptual data models

- Allow better communication between designers and users to understand application requirements
- More stable than implementation-oriented (logical) schema, which changes with the platform
- Provide better support for visual user interfaces

No well-established conceptual model for multidimensional data

Several proposals based on UML, on the ER model, or using specific notation

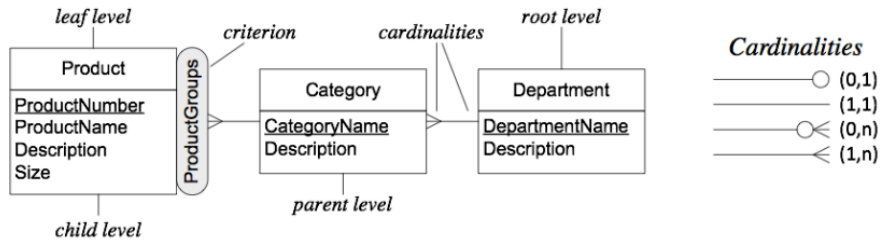
Problems:

- Cannot express complex kinds of hierarchies
- Lack of a mapping to the implementation platform

Currently, data warehouses are designed using mostly logical models (star and snowflake schemas)

- Difficult to express requirements (Technical knowledge required)

A graphical notation used for a sample hierarchy



Dimension: level or one or more hierarchies

Hierarchy: several related levels

Level: entity type

Member: every instance of a level

Child and parent levels: the lower and higher levels

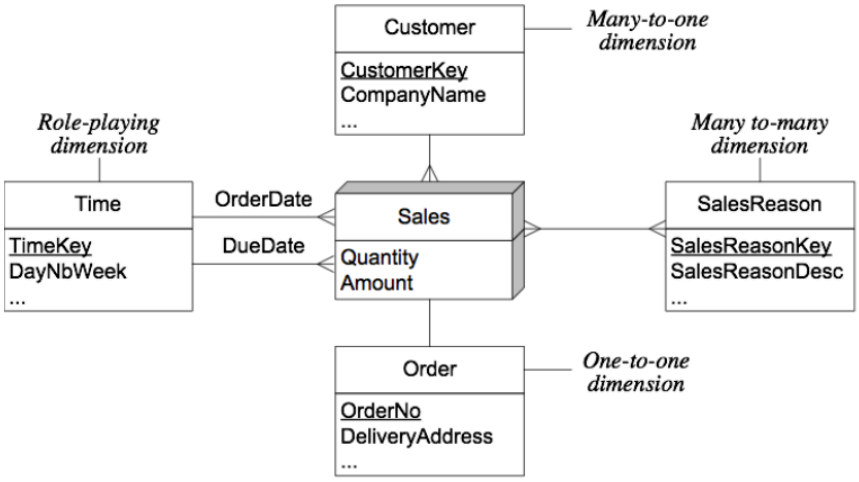
Leaf and root levels: first and last levels in a hierarchy

Cardinality: minimum/maximum numbers of members in a level related to members in another level

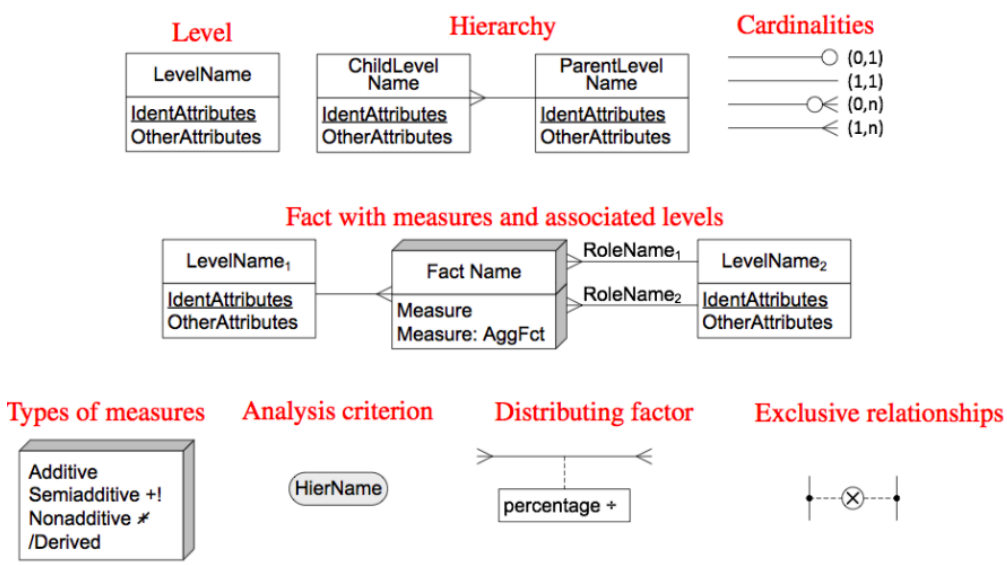
Criterion: expresses different hierarchical structures used for analysis

Key attribute: indicates how child members are grouped

Descriptive attributes: describe characteristics of members



Fact: relates measures to leaf levels in dimensions
There is 5 dimensions as there are 5 lines connected to the fact table
Dimensions can be related several times to a fact with different roles (OrderDate, DueDate)



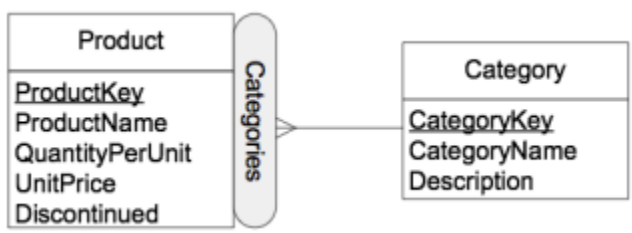
Child level in the conception model is the level close to the fact table and the parent level is further away from the fact table

Exclusive level enables the use of either path. Eg. country, state, and city, but Singapore does not have a state.

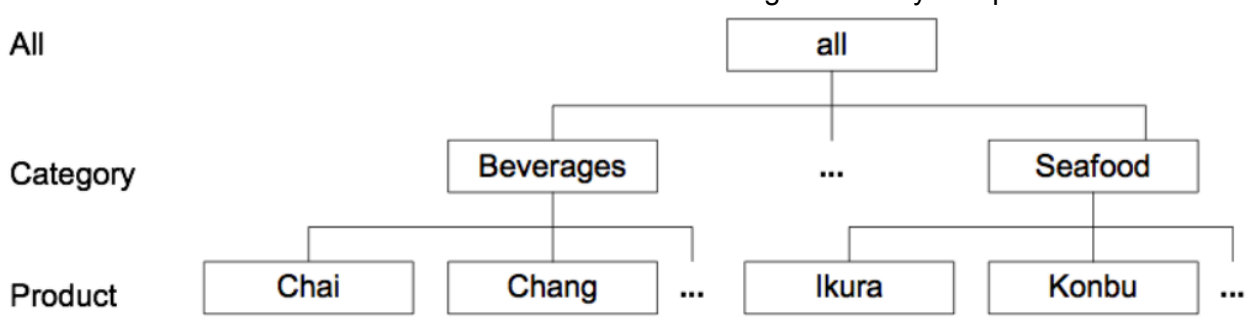
Hierarchies

Balanced Hierarchies

At schema level: One path where all parent-child relationships are many to one and mandatory

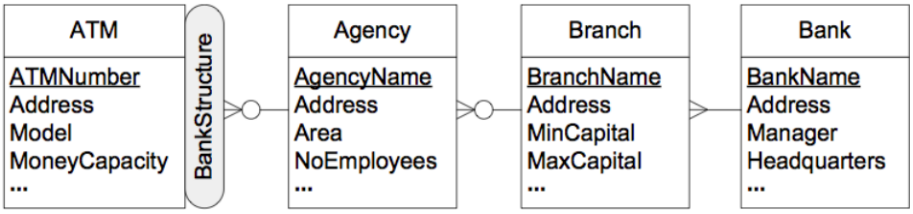


At instance level: Members form a balanced tree (all branches have same length) All parents members have at least one child member and a child belongs to exactly one parent

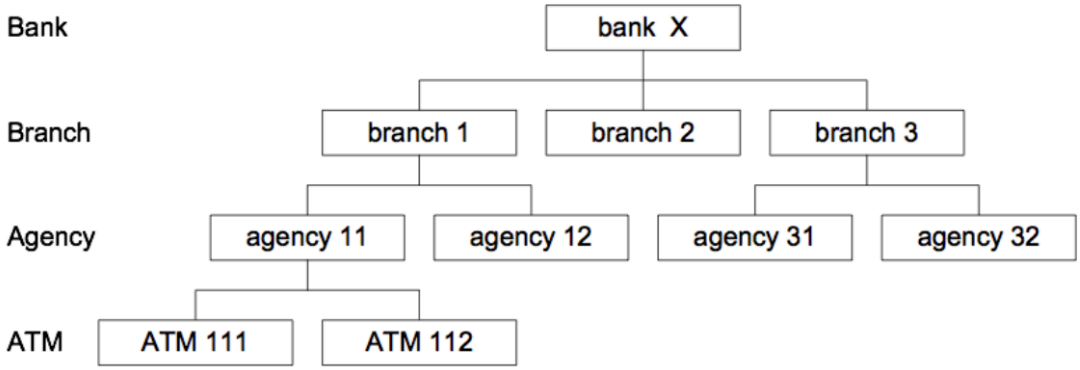


Unbalanced Hierarchies

At schema level: One path where all parent-child relationship are many to one but some are optional

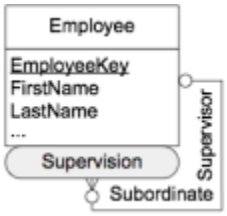


At instance level: members form a unbalanced tree

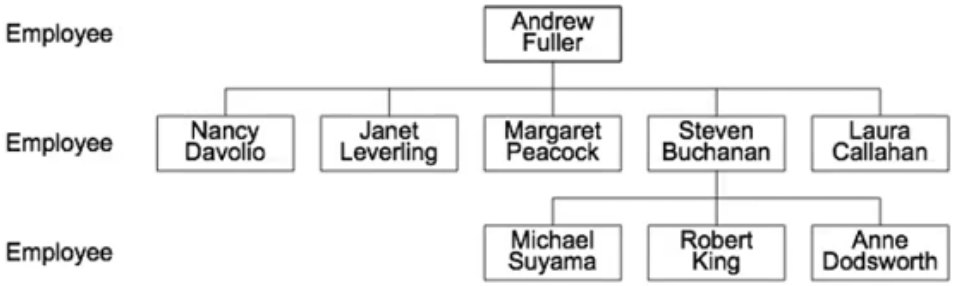


Recursive Hierarchies

At schema level: Special case of unbalanced hierarchies. The same level is linked by the two roles of a parent-child relationship. Characteristics of parent and child are similar or the same



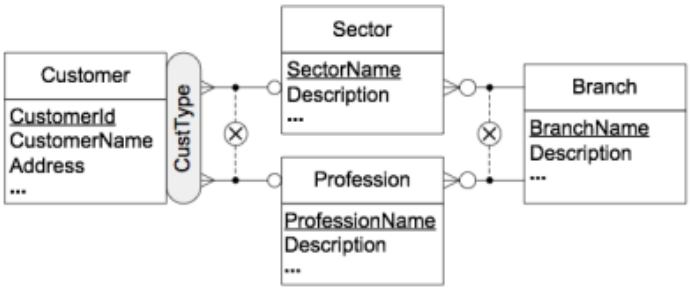
At instance level:



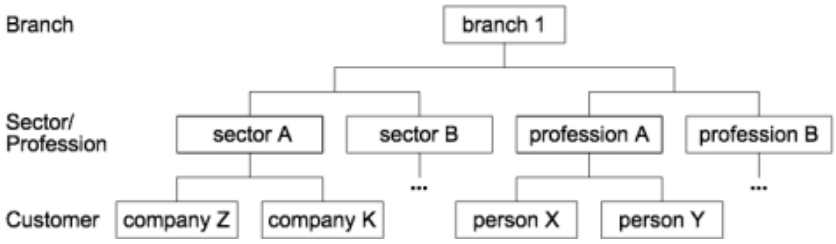
Generalised Hierarchies

At schema level: Multiple exclusive paths sharing at least the leaf level and may share other levels as well.

Two paths, one for each type of customer. Either branch → sector → customer or branch → profession → customer

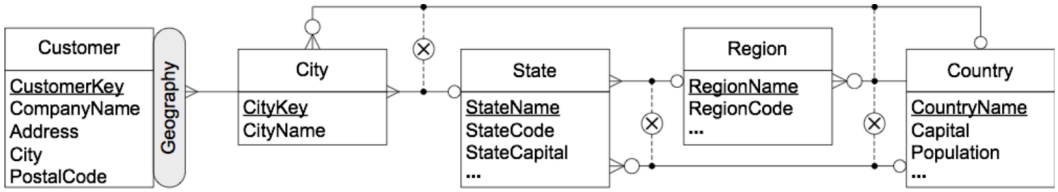


At instance level: Each member belongs to only one path

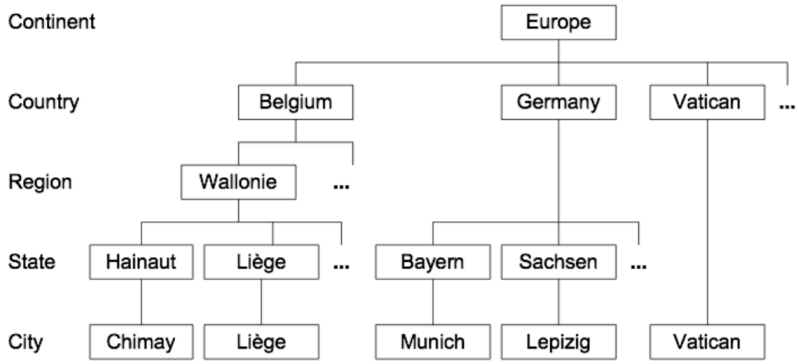


Non covering Hierarchies (Ragged or level skipping hierarchies)

At schema level: Alternate paths are obtained by skipping one or several intermediate levels



At instance level: Path length from the leaves to the same parent can be different for different members



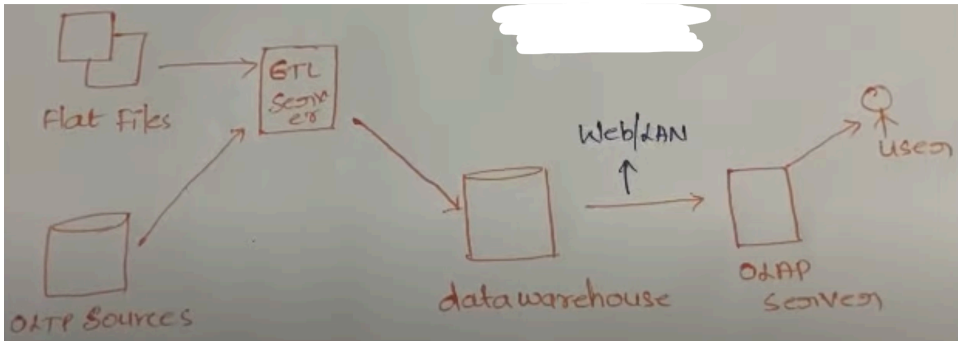
8. Relational data model

OLAP Technologies

Relational OLAP (ROLAP)

Stores data in relational database and supports extension to SQL and special access methods to efficiently implement the model and its operations

- Extension of RDBMS (Presents data in tables) (Fact table and dimensional table are stored in the form of relations)
- Capable of large amount of data analysis
- Can store and analyze large changing amount of data

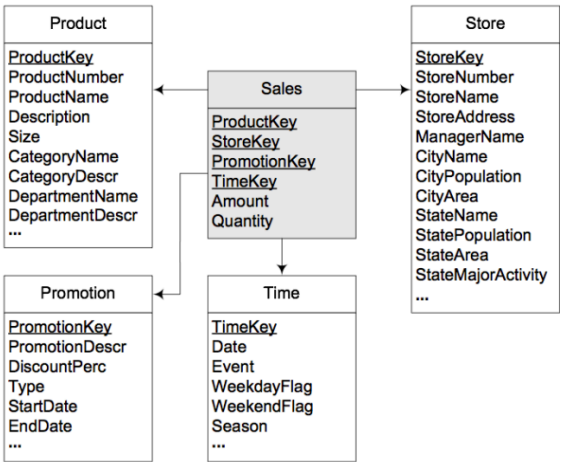


Flat files and OLTP sources go to the ETL servers then to the data warehouse and transmits data using web/lan to OLAP servers.

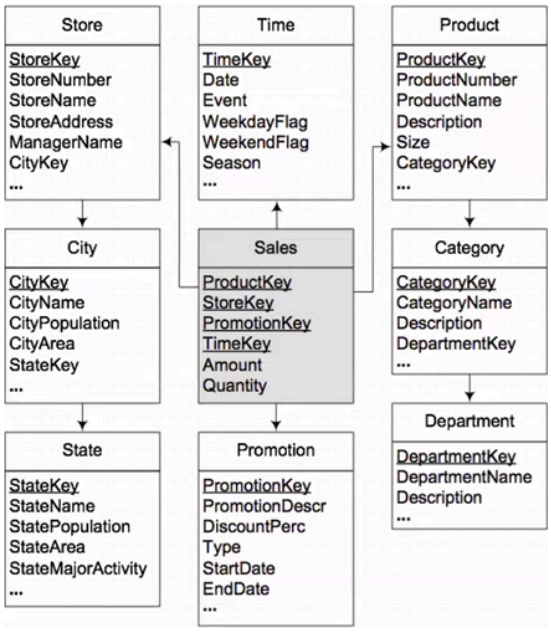
Relational Data Warehouse Design -

Specialized structures:

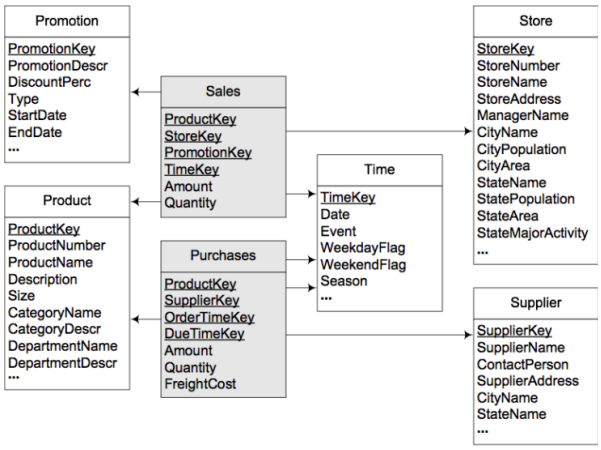
- Star Schema: One fact table and a set of dimension table



- Snowflake Schema: Avoids redundancy of start schemas by normalizing dimension tables



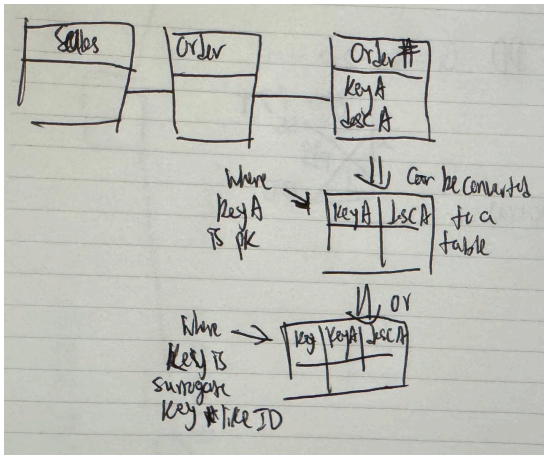
- Starflake Schema: Combination of the star and snowflake schemas, some dimensions normalized and others not.
- Constellation Schema: Multiple fact tables that share dimension tables



Set of rules to translate conceptual model (MultiDim model) into relational model:

Rule 1: A level L , provided it is not related to a fact with a one to one relationship, is mapped to a table TL that contains all attributes of the level

- A surrogate key may be added to the table, otherwise the identifier of the level will be the key of the table
- Additional attributes will be added to this table when mapping relationships using Rule 3



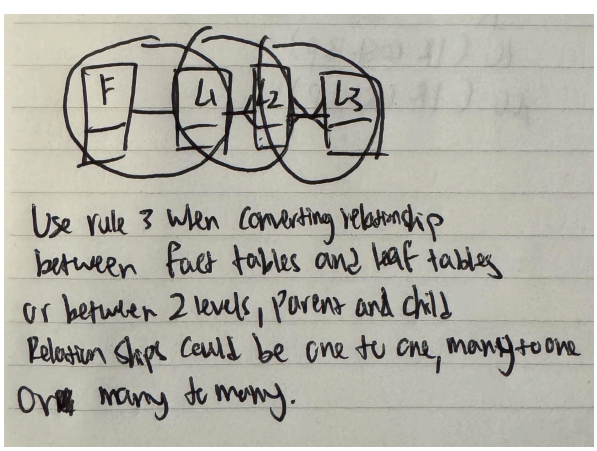
Rule 1 states that the level can be converted to a table straight away.
As keyA might be modified, it is better to define a surrogate key to prevent issues in implementation

Rule 2: A fact F is mapped to a table TF that includes as attributes all measures of the fact

- Surrogate key may be added to the table
- Additional attributes will be added to this table when mapping relationship using Rule 3

Rule 2 is the same as rule 1 where the fact table can be converted to a table but a surrogate key should be added to the fact table when converting to a relational table to prevent issues

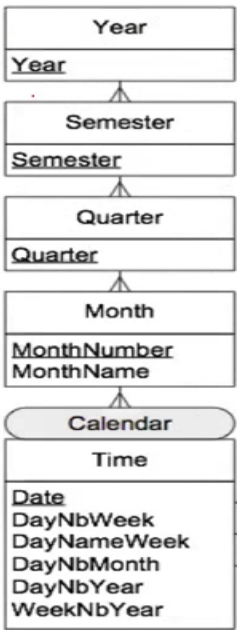
Rule 3: Relationship between either a fact F and a dimension level L, or between dimensions levels LP and LC (parent and child level respectively) can be mapped in three different ways depending on its cardinalities:



Higher level means it is the parent. Eg, lvl 1 will contain product id, but lvl 2 will have the description

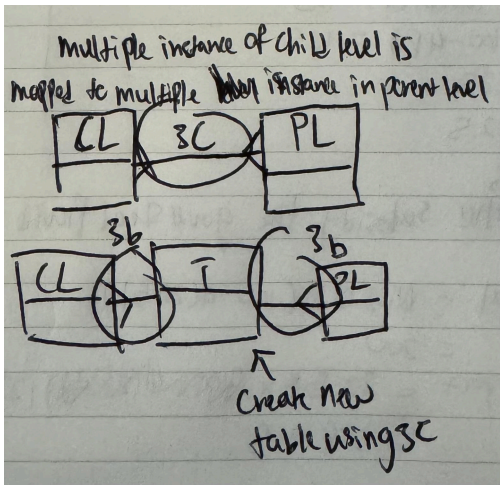
Rule 3a: If the relationship is one-to-one, the table corresponding to the fact or to the child level is extended with all the attributes of the dimension level or the parent level respectively. It means to merge the one to one relationship tables and it will reduce normalization but increase redundancy data/repeated data

Rule 3b: if the relationship is one-to many, the table corresponding to the fact or to the child level is extended with the surrogate key of the table corresponding to the dimension level or the parent level respectively, that is, there is a foreign key in the fact or child table pointing to the other table. Merge tables with one to many relationships and create a surrogate key as PK.

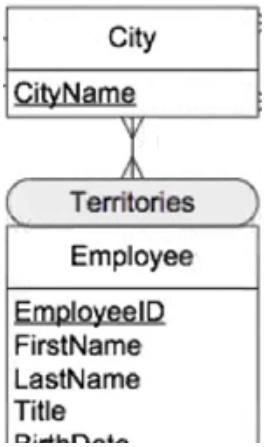


<- converts into this table using rule 3b ->

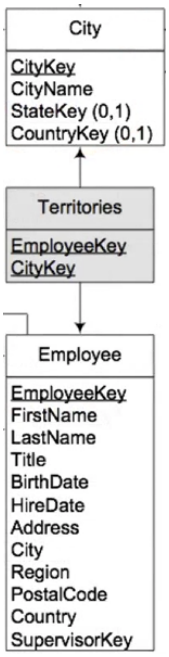
Rule 3c: If the relationship is many-to-many, a new table TB (standing for bridge table) is created that contains as attributes the surrogate keys of the tables corresponding to the fact (TF) and the dimension level (TL), or the parent (TP) and child levels (TC), respectively. If the relationship has a distributing attribute, an additional attribute is added to the table to store this information.



eg.

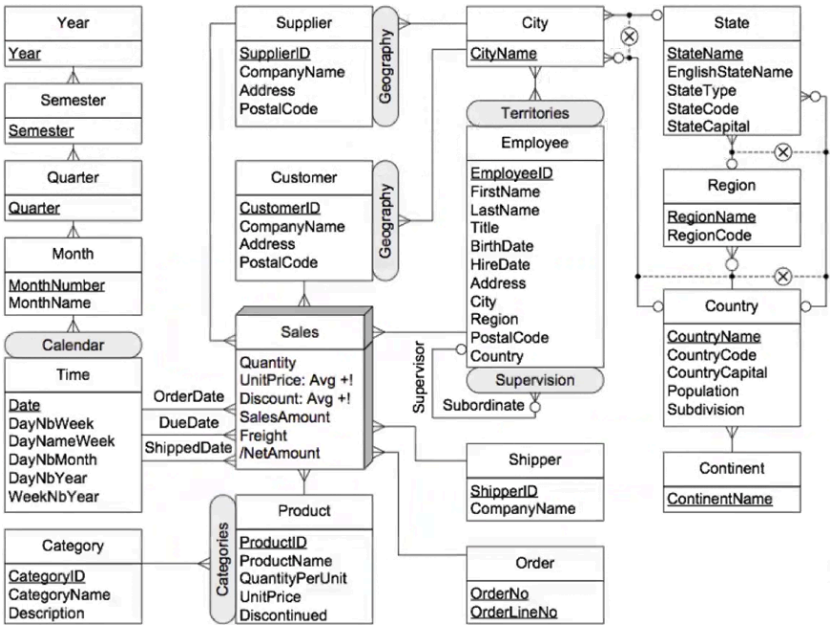


<- convert ->



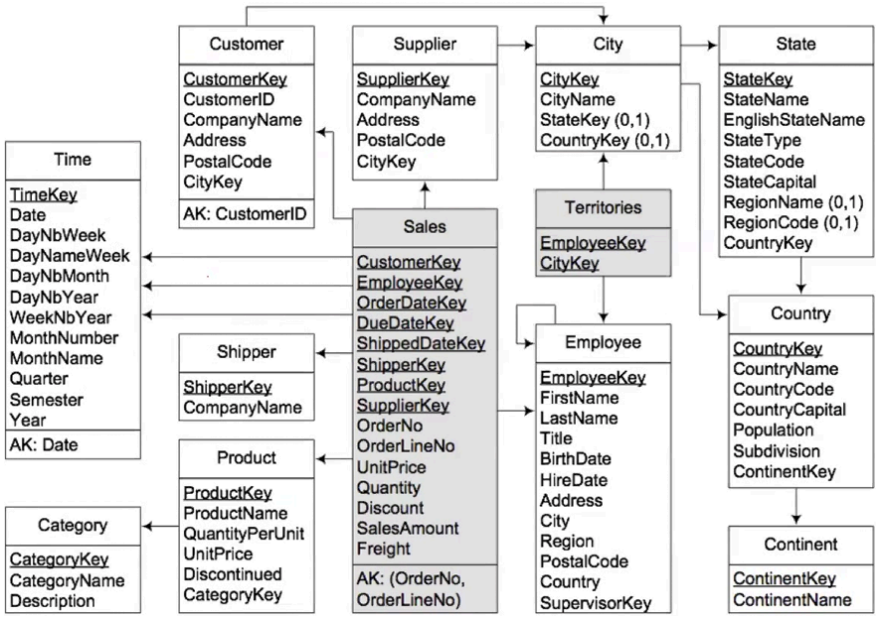
Territories table is created in rule 3c

Conceptual schema



Converts into:

Relational representation



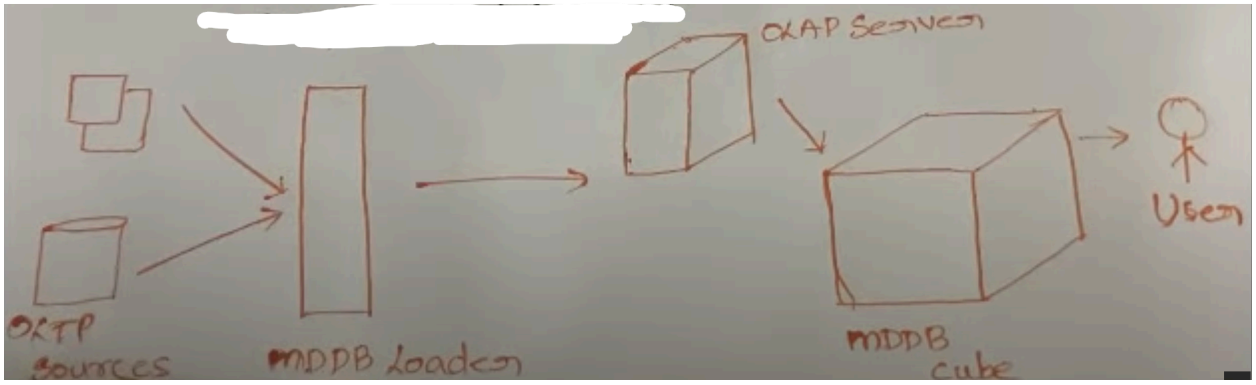
Redundant information in the Time table is alright as it is going to be accessed frequently so that there will be fewer join functions, but city state can be normalized to save space as it is not accessed frequently.

Time dimension is present in almost all data warehouses.
In a star or snowflake schema, time is included both as foreign key in a fact table and as a time dimension containing the aggregation levels.

The granularity of the time dimension varies depending on their use. The number of tuples depends on the amount of data variation needed.

Multidimensional OLAP (MOLAP)
Stores data in special data structures (eg. arrays) and implementing OLAP operations in these structures

- Better performance than ROLAP for query and aggregation but less storage capacity than ROLAP
- Limited amount of data
- Uses arrays
- Fast information retrieval as lesser coarser data is stored
- Able to do complex calculation
- Easy to use
- Stores data in data cube

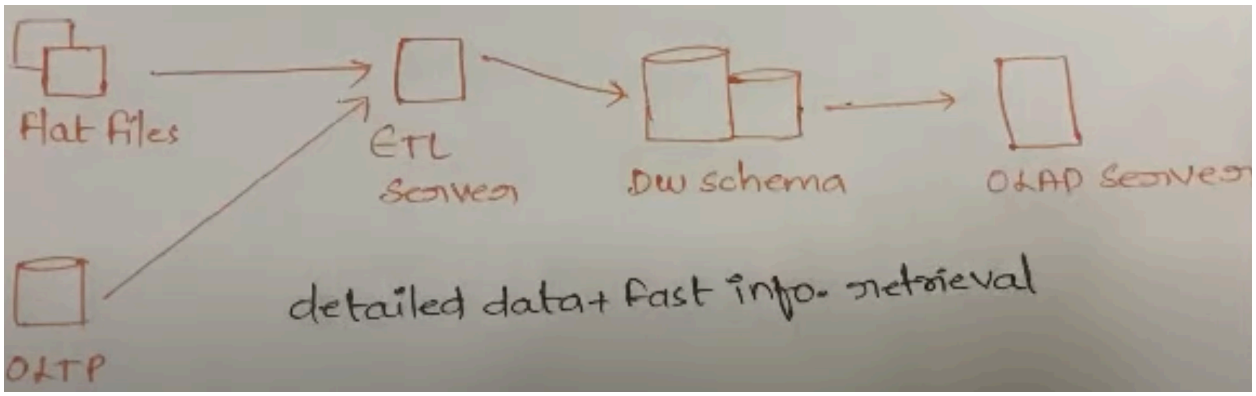


Sources go to multidimensional database loader (Extract, transform and load) to OLAP servers to multidimensional database cube to user

Hybrid OLAP (HOLAP)

Combines both technologies

Eg. detailed data stored in relational databases aggregations kept in separate MOLAP store, apply OLAP operations, store data in special data structure, etc



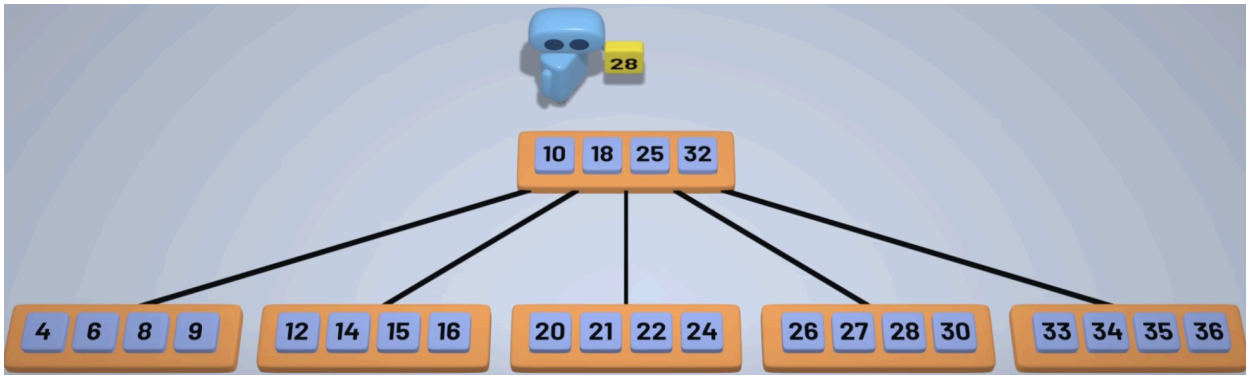
Sources sent to ETL servers to data warehouse schemas and to OLAP servers. It is able to have large amount of data and fast retrieval information as it is a combination of relational and multidimensional

Techniques for Physical Data Warehouse Design

- **Materialized Views**
    - A view physically stored in the DB
        - Stores the table and the query so that it can fetch the table faster when called. Does not run the query when the materialized view is called again.
    - Created and used to reduce the amount of time needed to compute select statements
    - Two ways how materialized view can be used:
        - Brute force method (Use queries to search result based on the materialized view)

        - Transparency query rewrite (Hive will analyze the query and compare with the existing result in the materialized view. If it found that the query is similar to the query kept by the materialized view, it will search the results from the existing view rather than submitting the query to hadoop.)
    - Drawbacks: how to maintain the views.
- **Indexing**
    - Provides a quick way to locate data of interest
    - Used in data warehouse together with materialized views
    - Specific for data warehouse: bitmap and join indexes
    - Drawbacks:
        - Update on indexed attribute requires an index update
        - Too many index will degrade performance
    - Indexing techniques:
        - B*-tree index implementation
        - Bitmap index

B*-tree index implementation



Usually travels from top to bottom until it finds the search result. It can start either from the left or right of each leaf. It will start comparing and see that 28 is more than 10 and less than 32. It is between 25 and 32 so it will search the 4th branch.
B tree needs the data on each sides of the leafs to be balanced so that the performance will be good

Bitmap index

ProductKey	ProductName	QuantityPerUnit	UnitPrice	Discontinued	CategoryKey
p1	prod1	25	60	No	c1
p2	prod2	45	60	Yes	c1
p3	prod3	50	75	No	c2
p4	prod4	50	100	Yes	c2
p5	prod5	50	120	No	c3
p6	prod6	70	110	Yes	c4

Product dimension table

	25	45	50	70
p1	1	0	0	0
p2	0	1	0	0
p3	0	0	1	0
p4	0	0	1	0
p5	0	0	1	0
p6	0	0	0	1

Bitmap index for attribute QuantityPerUnit

	60	75	100	110	120
p1	1	0	0	0	0
p2	1	0	0	0	0
p3	0	1	0	0	0
p4	0	0	1	0	0
p5	0	0	0	0	1
p6	0	0	0	1	0

Bitmap index for attribute UnitPrice

This is a table of products and the bitmap index that is created by the system
To find products that is between 45 and 55 pieces per unit and with a unit price of between 100 and 200

	45	50	OR1
p1	0	0	0
p2	1	0	1
p3	0	1	1
p4	0	1	1
p5	0	1	1
p6	0	0	0

OR for QuantityPerUnit

	100	110	120	OR2
p1	0	0	0	0
p2	0	0	0	0
p3	0	0	0	0
p4	1	0	0	1
p5	0	0	1	1
p6	0	1	0	1

OR for UnitPrice

	OR1	OR2	AND
p1	0	0	0
p2	1	0	0
p3	1	0	0
p4	1	1	1
p5	1	1	1
p6	0	1	0

AND operation

The bitmap table is then sliced to focus on the parts that fit the requirement.
By performing the OR functions, it is able to create a new table to ensure that both requirements are fulfilled by using the AND function.

Requirements for it to be effective:

Symmetric partial match queries
- All dimensions of the cube should be symmetrically indexed, to be search simultaneously

Indexing at multiple levels of aggregation
- Summary tables need to be indexed as well if it gets too large at each level

Efficient batch update
- The refreshing time of a data warehouse must be considered when designing the indexing schema

Sparse data
- Most data cube is sparse and some cells may not have value
- Consider the design to make sure that index can be performed efficiently to handle the sparse and non sparse data

Star Queries

Queries over star schemas are called star queries

Eg.

```
SELECT ProductName, CustomerName, SUM(SalesAmount)
FROM Sales S, Customer C, Product P
WHERE S.CustomerKey = C.CustomerKey AND S.ProductKey = P.ProductKey AND
      P.Discontinued = 'Yes'
GROUP BY C.CustomerName, P.ProductName;
```

Product Key	Product Name	...	Discontinued	...
p1	prod1	...	No	...
p2	prod2	...	Yes	...
p3	prod3	...	No	...
p4	prod4	...	Yes	...
p5	prod5	...	No	...
p6	prod6	...	Yes	...

Product table

Yes	No
0	1
1	0
0	1
1	0
0	1
1	0

Bitmap for Discontinued

Customer Key	Customer Name	Address	Postal Code	...
c1	cust1	35 Main St.	7373	...
c2	cust2	Av. Roosevelt 50	1050	...
c3	cust3	Av. Louise 233	1080	...
c4	cust4	Rue Gabrielle	1180	...

Customer table

Product Key	Customer Key	Time Key	Sales Amount
p1	c1	t1	100
p1	c2	t1	100
p2	c2	t2	100
p2	c2	t3	100
p3	c3	t3	100
p4	c3	t4	100
p5	c4	t5	100

Sales fact table

c1	c2	c3	c4
1	0	0	0
0	1	0	0
0	1	0	0
0	1	0	0
0	0	1	0
0	0	1	0
0	0	0	1

Bitmap for CustomerKey

p1	p2	p3	p4	p5	p6
1	0	0	0	0	0
1	0	0	0	0	0
0	1	0	0	0	0
0	1	0	0	0	0
0	0	1	0	0	0
0	0	0	1	0	0
0	0	0	0	1	0

Bitmap for ProductKey

Yes	No
0	1
0	1
1	0
1	0
0	1
1	0
0	1

Bitmap join index for Discontinued

3 steps to evaluate the query:

- Evaluation of the join conditions
    - Bitmap CustomerKey and ProductKey are created with reference from Sales fact table
    - Need p6 in bitmap table even when there is no p6 in sales table as there is p6 in product table
- Evaluate the selection conditions over the dimension (Logic AND)
- Aggregation of the tuples that passed the filter (SUM in eg.)

Answer: (cust2,prod2,200) and (cust3,prod4,100)

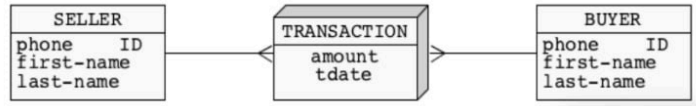
- **Partitioning**
    - Partitioning or fragmentation divides the table into smaller data sets
    - Divides the contents of a relational table into several files
    - Horizontal and vertical partitioning
    - Applied to tables and indexes
    - Vendors provide several different partitioning methods
    - <u>Vertical partitioning</u> splits the attributes of a table into groups that can be stored independently
        - Eg. most often used attributes are stored in one partition and less used in another
        - More records fit into main memory, reducing processing time

- <u>Horizontal partitioning</u> divides the table into smaller tables with the same structure of the full table
    - Eg. If some queries require the most recent data, partition horizontally according to time
- Partition pruning is a typical way of improving query performance
    - Eg. sales fact table can be partitioned by months
- Joins are also enhanced by using partitioning:
    - When the two tables are partitioned on the join attributes (Dont need to join the whole table)
    - When the reference table is partitioned on its primary key
    - Large join is broken down into smaller joins

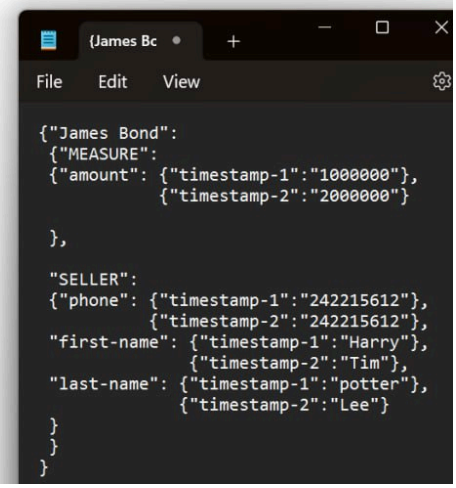
Partitioning Strategies:

- Range partitioning
    - Maps records to partitions based on range of values of the partitioning key
    - Time dimension is a natural candidate
    - Easiest way to create partition
- Hash partitioning
    - Uses hashing algorithm over the partitioning key to map records to partition
        - Distributes row among partitions in a uniform fashion, yielding, ideally, partitions of the same size
        - Typically used when the partitions are distributed in several devices and when data are not partitioned based on time.
    - MapReduce uses hash partitioning
        - Consider the key value and pull all the items with the same key in the same partition as well as balance the workload between different reducers.
    - It is the best in terms of overall performance
        - But the amount of reducers is determined automatically by the partition
- List partitioning
    - List partitioning specifies a list of values for the partitioning key
        - If the data contains the value from the list, it will be put in the partition
        - Eg. Product with same brand can be put in the same list
        - List needs to be manually specified
    - Some vendors support notion of composite partitioning
        - Combine basic data distribution methods
    - Might create unbalanced partitioning so hash partitioning is best as it allows the framework to determine the total number of partitions and reducers automatically.

9. Hbase model (basic commands)




```
{"1234567":
            {"MEASURE":
                      {"amount": {"timestamp-1":"1000000"}
                      },
             "BUYER":
                      {"phone": {"timestamp-1":"242214339"},
                       "first-name": {"timestamp-1":"James"},
                       "last-name": {"timestamp-1":"Bond"}
                      },
             "SELLER":
                      {"phone": {"timestamp-1":"242215612"},
                       "first-name": {"timestamp-1":"Harry"},
                       "last-name": {"timestamp-1":"potter"}
                      }
            }
```

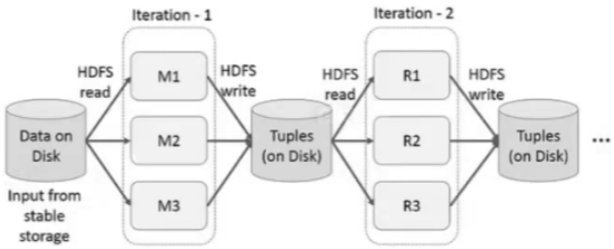
If there is 500 sellers and 100,000 buyers, this will help to reduce the records ^

10.Spark concept, data model, and operations (basic commands)

MapReduce Challenges:

The Java application of MapReduce is still at a low level, that is why other frameworks such as Hive and Pig are used to help make MapReduce easier to use. But all the other frameworks are built on top of the MapReduce framework.

One of the biggest issues of the MapReduce framework is that it contains several stages. Mapping, reducing, etc. The data then needs to be transferred between different nodes. Each data will be split into chunks and each data has 3 copies. Therefore a lot of read and write operations.



When the data is loaded into the HDFS, it will be split into chunks and loaded into the mappers. The intermediate results will then be loaded into HDFS for sharing and transition purposes. The data will then be partitioned and loaded into the reducer. It also includes other series of operations such as object serialization and object deserialization.

This is mainly due to Spark's in-memory computing, although Spark is also more efficient than MapReduce for persistent storage complex applications
Spark covers a wide range of workloads, including batch application, interactive queries, streaming and advanced analytics

- Provides unified environment makes the combination of those different engines easy and inexpensive
- Provides multiple language API (Scala, Python, Java, SQL, R) as well as rich and fast-growing built in libraries

<u>Data Scientist</u> use Spark to:

- Analyze and model data
- Make prediction based on data
- Build data pipelines to fulfill certain tasks

<u>Data Engineers</u> use Spark to:

- Develop data processing application
- Deploy the output of data scientists in production

Features of Spark:

- Support batch computing and interactive, iterative and real time computing
- In-memory cluster computing that increases the processing speed of an application
- It is based on Directed Acyclic Graphs (DAG) and it is more flexible than the MapReduce model
- Reduces the management burden of maintaining separate tools
- Helps to run applications in Hadoop cluster 100 times faster in memory and 10 times faster when running on disk.
    - Achieved by reducing number of read/write operations to persistent storage
    - Stores the intermediate processing data in memory
- Provides built-in APIs in Java, Scala, Python
- Supports MapReduce, SQL queries, Streaming data, Machine Learning (ML) and Graph algorithms

Unlike MapReduce, Spark uses memory cluster computing which is to create different memory blocks called distributed memory and it is located in different nodes. They keep the data in the distributed memory and the data sharing and transaction happens between memories. This reduces the IO overhead greatly.

Spark supports 3 fundamental data structures:

- RDD (Resilient Distributed Data). It was introduced in 2011. RDD consist of collection of objects which is capable of storing data partitioned across the multiple nodes of the cluster and also allows them to process in parallel. Column cannot be mutated
- Dataframes are the distributed collection of the data points but here the data is organized into the named columns. They allow developers to debug the code during runtime which was not allowed with RDD. Dataframe can read and write data into

various formats like CSV, JSON, AVRO, HDFS, and Hive table. It is optimized to process large datasets for most of the preprocessing tasks so that we do not need to write complex functions on our own

- Dataset is an extension of datafames API with the benefits of both RDD and Dataframes. It is fast as well as provides a type-safe interface. The type safety means that the compiler will validate the datatypes of all the columns in the dataset while compilation only will throw an error if there is any mismatch in the data type

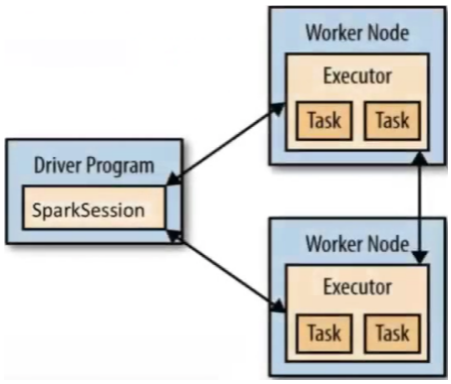
Spark supports two types of operations

- Transformation: This operation creates a new database from an existing one. For example, ‘map is a transformation that passes each dataset element through a function and returns a new RDD representing the results
- Actions: this operation returns a value to the drive program after running a computation on the dataset. For example, ‘reduce’ is an action that aggregates all the elements of the RDD using some function and returns the final result to the driver program.

Spark Architecture

Spark is built on top of Hadoop so most of the components used by Hadoop are supported by Spark. Eg. HDFS for data storage purposes, YARN for resource management.

Spark standalone mode can work without the use of YARN and should use YARN when using the distributed mode (cluster).



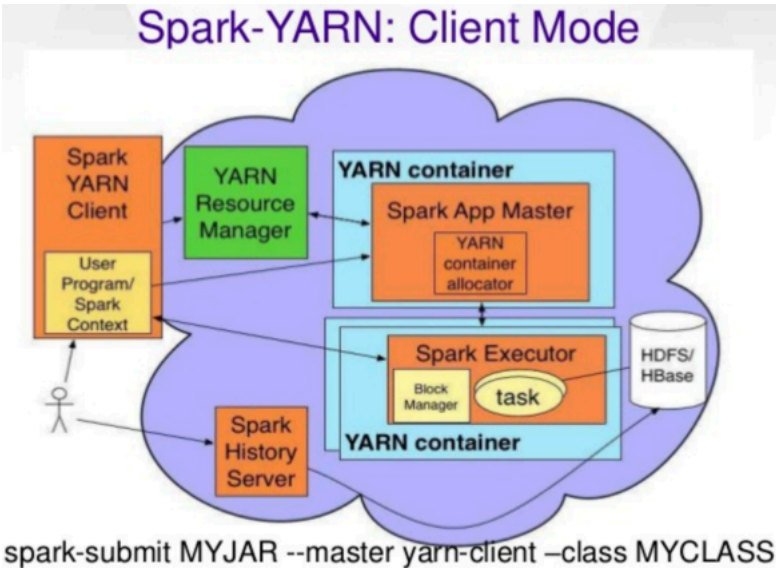
Driver program

- Main Spark application that consists of the data processing logic
- Sets up the spark application
- Submits application to HDFS

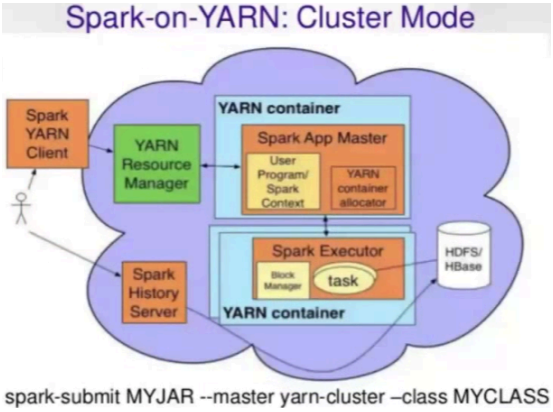
Executor

- JVM process that runs on each worker node and processes the jobs that the driver program submits
- Resource application master
- Each executor session contains different tasks and work on different data nodes to handle the data and do some analysis

Task is a subcomponent of a data processing job



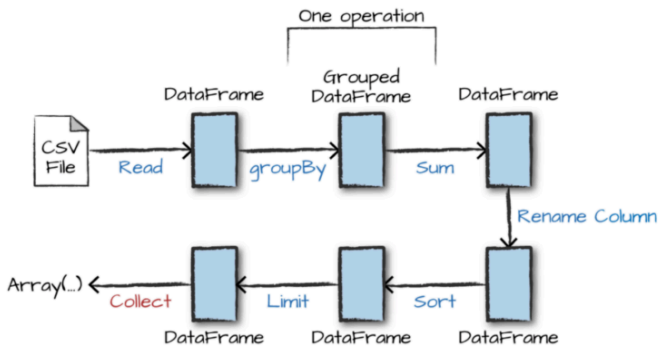
There are 2 ways for users to submit the Spark application. First one is through the client where the user submits the program and data together. The program will be compiled and the data will be loaded into the YARN resource manager. Containers will be allocated to each spark application. It is called the Spark application master and a similar process as MapReduce will be used there.



This is the second method. It is built very similarly but the function is already uploaded into the cluster so the client is able to choose the function. It is able to support multiple user which make sit easier for data and program sharing

High-level API of Spark eases the development of a data processing pipeline

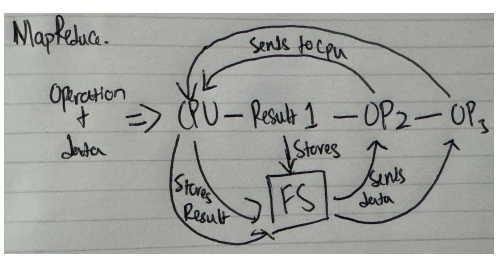
- No need to confirm to a specific pattern, eg. MapReduce



This shows the Spark stream. When the data comes through the stream, the different operations are applied. The task will be completed in one operation.

Eg. The word count example. The stream will help to read the data from the input to the inbuilt memory. In the inbuilt memory, the data will be separated into different partitions by using structures such as data frame or data set. The data frame and data sets are further grouped based on different criterias. The aggregation can be performed straight away in the memory. Calculating the sum, analyzing the value, modifying the data structure. Once the results are out, it can be easily sorted and ranked based on the request and return the results.

All the operations are sent to the CPU at the same time, eg. sum, avg, sort. The data will be sent to the CPU with all the operations lined up into the cpu. And the CPU will then generate the results.



The operation and data is sent to the CPU and when the results are generated, it would be first stored before going on to the next operation. The data will be sent to the operation and both would be sent to the CPU together and the cycle just repeats.

Spark Components

- Spark Core
    - Execution engine for Spark platform that all other functionality is built on
    - Provides In-Memory computing and referencing datasets in external storage system
- Spark SQL
- Spark Streaming
    - Leverages Spark Core fast scheduling capability to perform streaming analytics
- SparkML
    - Distributed machine learning framework above Spark because of the distributed memory-based Spark architecture
- GraphX
    - Graph processing framework on top of Spark
    - Provides API for expressing graph computation and provides an optimized runtime for this abstraction
- SparkR
    - R package that provides a light-weight frontend to use Spark from R

Spark Data Model

Resilient Distributed Dataset (RDD):

- Lowest level used to represent data
- Immutable (Cannot be modified)
- Partitioned collection of data (Distributed)
- Can be used in parallel computing
- RDD is like a table and every different row is a Java object
    - RDD is a collection of serialized objects
- Does not need schema
- Can be applied in different application but makes manipulation of data more complicated
- Programmers need to specify their own functions (SUM, AVG, etc.)

Characteristics:

- Partition of the data
- Contains function for computing each partitions
- Contains a list that indicate interdependencies between RDDs
    - Most of the times, objects may not be able to work by themselves
    - Need to try apply parallel computing which indicates which RDD depends on other RDDs
    - Can specify key-value for RDDs
    - Can specify location preferred to compute each RDD
- Properties of RDD determine how Spark schedules and processes the applications
- Different RDD may use different properties depending on the required outcomes

DataFrames:

- It is a table of data where data may not be stored in the database
- It can span over many systems in the cluster
- It can be used in different programming language
- DataFrames can be placed in the same or different partitions
- Data is divided into chunks called partitions
- When operating, Spark simultaneously processes all relevant partitions
- Shuffle operation is performed to share data among the system in the cluster

A DataFrame can be created in the following way

Creating a DataFrame

```
val dataFrame = spark.read.json("/bigdata/people.json")
```

The contents of a DataFrame can be listed in the following way

Listing results

```
dataFrame.show()

+----+-------+
| age| name  |
+----+-------+
|null|Michael|
| 30 | Andy  |
| 19 | Justin|
+----+-------+
```

If the JSON file is edited, it will show the updated results when the command is executed again.

SQL Tables/View:

- It can be used to operate on DataFrames
- DataFrames can be registered as table or view and SQL queries can be applied onto the DataFrame
- SQL table and DataFrame's only difference is that SQL has a real database while DataFrame is just the data in different formats being represented as a table.
- SQL Tables are logically equivalent to DataFrames

DataFrame as SQL Table

```
dataFrame.createOrReplaceTempView("people")
val sqlDF = spark.sql("SELECT * FROM people")
sqlDF.show()
```

This creates a materialized view

Managed table is a table that stores data and metadata, equivalent to an internal table in Hive

Unmanaged tables is a table that stores only data, it is equivalent to an external table in Hive

Datasets:

- It is a distributed collection of data
- It is similar to DataFrame
    - DataFrame is a type of row in a Dataset
- The use of Dataset is more strict than DataFrame
    - Can only be used in Scala and Java
- Can be constructed from Java Virtual Machine (JVM) and manipulated using inbuilt functions
    - Can also be used with Scala as it is a simplified version of Java
- Dataset supports all operation of DataFrame but the DataFrame does not support all operations available in Dataset
- Dataset uses a specialized Encoder to serialize the objects for processing and transmitting

Lecture notes
Ch 1
What is a computer cluster?
It is a collection of computers which are called nodes which are connected through the internet or intranet. They will work together to simulate a much more powerful computer system. Each node can work on different versions of the same task in parallel and is controlled by their own OS. They might have varying hardware.
Computer cluster is used to speed up parallelization of data processing on the nodes of the cluster.
It provides high availability through automatic replacement if a failed node with a replica node. An advantage is that there is faster processing speed, larger storage capacity, better data integrity, greater reliability and wider availability of resources
Upload data through master nodes so that it can manage and allocate data with the slave nodes.
Cluster computing is the process of sharing computation tasks among computers included in a computer cluster

What does big data mean?
It is so big that it cannot be stored on the persistent storage devices attached to a single computer system. It also means an infinite amount of data. And it is usually generated consistently by machines, humans and business processes

Example:
Sensor data
Click stream (what ppl click on website)
Call centre data
Email and instant messaging
Unstructured data
Geographic data
Satellite data
Image data
Temporal data
And more

Big data is characterised in 3Vs:
Volume: size, billion or millions of rows and columns
Variety: complexity of data types and structures
Velocity: speed of new data creation and growth
Additional Vs:
Veracity: able to process and represent uncertain and imprecise data
Value: it is the driving force of the next generation business
Viability: benefits we can potentially have from data analysis

Traditional data architecture
Contains a central data storage called data warehouse connected by other databases which uses 3 dimension analysis that analyses the data. It may or may not use data cluster

Strengths of traditional data architecture:
Centralised governance of data repository
Light fast inquiries
Optimisation of OLTP (can be done every 24 hour to update to the warehouse) and OLAP (hard to do so)
Security and access control
Fault tolerance and backup

Challenges:
New type of data such as unstructured data and semi structured data
Increasing amount if data flowing into organisation
New computational paradigms use non-traditional no-sql databases to rapidly mine and analyse large data sets.
Increasing cost of storing and analysing the large amounts of data
Increasing use of data analytics which requires significant storage and process capabilities

Data lake architecture is not a centralised repository which doesn't have that many nodes. Data repositories have their own specified tasks

Centralised data repository (not fault tolerant) vs multiple data repository (they can be in different location which needs high speed network to connect) is how traditional data architectures can scale

Hadoop features:
Capability to handle large data sets, eg simple scalability and coordination
File size range from gigabytes to terabytes
High fault tolerance
Supports data replication
Supports streaming access to data
Supports batch processing
Supports interactive, iterative and stream processing
Implements a data consistency model of write-once-read-many access model
Run on commodity hardware and not high performance computers
Inexpensive
Can be deployed on premises or in the cloud

Core components of Hadoop
Different data processing frameworks
YARN: OS (cluster resource management) (help allocate data and resource to nodes)

HDFS: file system
Master nodes only hold metadata and wont contain any data. Only slave data will have data.
Metadata can be used to find the data
Hadoop receives near to linear scalability which means after a set amount of nodes, it is better to have another cluster.

What is a database cluster?
Collection of databases that is managed by a single instance of a running database server.
A very large database in a database cluster is partitions over a number of smaller databases each located on a separate node of a computer cluster
It requires replication and sharding and improves performance, availability and scalability.
Kubernetes (K8) is a container or microservice platform that orchestrates computing, networking and storage infrastructure workloads.
It is an orchestration platform to manage ant containerised application
It consists of a single master node and multiple corresponding worker nodes.
Benefits (same as hadoop)

Ch3
Hadoop architecture
HDFS is designed for large files, stream data access, commodity hardware.
Not for:
multiple writers, arbitrary file modification
Low latency data access
Lots of small files
Contains key components

NameNode:
Master node
SecondaryNameNode (expedited file system metadata recovery) and standby namenode (increased availability)

Datanode:
Slave node
Manage block storage and access for reading, writing data and block replication

Name node functions:
Maintain the metadata pertaining to the file system
Manage user access to the data files
Map the data blocks to the datanodes in the cluster
Perform file system operations (eg opening and closing of files)
Provide registration service and periodic heartbeats for datanodes

Datanodes function:
Provide block storage by storing blocks on the local file system

Fulfil read write request
Replicate data across cluster
Send heartbeat to show that it is healthy and alive as well as block report to show the blocks managed by the data node to name node

Yet Another Resource Negotiator (YARN) is a resource manager
Core subsystem in Hadoop responsible for governing, allocating and managing the distribution of resources available on a Hadoop cluster

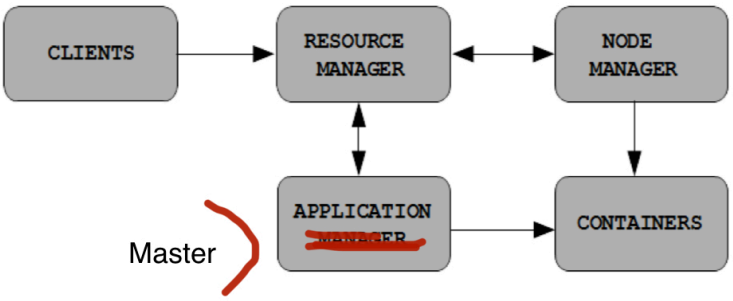
YARN provides core services via two types of long running daemons

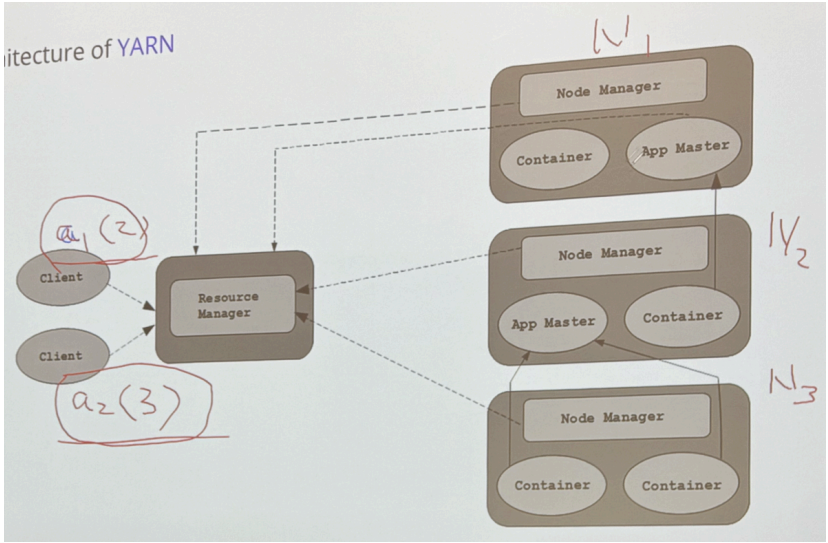
- Resource Manager:
    - One per cluster to manage use of resource
    - Runs on master node
    - Grants computing resources to applications running in the cluster
- Node Manager:
    - Runs on nodes in the cluster to launch and monitor containers
    - Manages containers on slave node
    - ApplicationMaster: the first container allocated by the ResourceManager to run on a NodeManager for each application

A client is the program that submits jobs to the cluster which may also be the gateway machine that the client program runs on
A job, also called an application, contains one or more tasks. A task in MapReduce can be either a mapper or a reducer task
Each mapper and reducer task runs within a container

- It is a logical construct that represents a specific amount of memory and other resources
- May also refer to the running environment of an application.

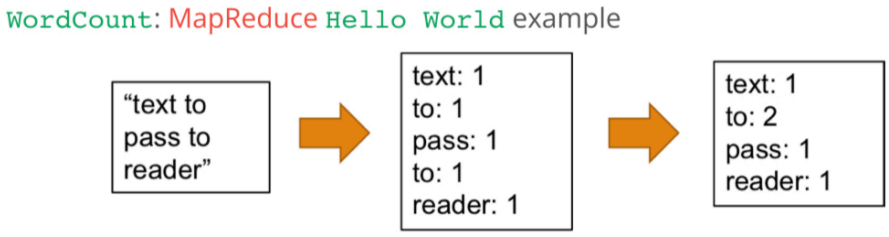
Architecture of YARN





Resource manager will estimate how many containers are needed for the application. And when the estimate is wrong, the application master will submit an extra resource request to run the application.

Chap 2
MapReduce framework
It is important processing framework
Many processing language are abstractions of MapReduce, eg pig and hive or heavily influenced by MapReduce concepts eg spark
MapReduce is also a platform and language independent programming model at the heart of most big data
There is a map phase and reduce phase
Limitations of early distributed computing and grid computing framework.
Complexity in parallel programming
Hardware failures
Bottlenecks in data exchange
Scalability problem
MapReduce uses key value pairs for processing data
Word count is to count the occurrence of each word in a sentence.
Input > key value pair (where word is the key and value is the occurrence) > reduced phase



It will happen in every node and then the reduced phase will be followed by another merging process.

Hadoop MapReduce frees the user from the low level communication and coordination of nodes and processes. With implementation of Hadoop 2, programmers can focus on MapReduce implementation and configuration. It also reduce the overhead of network transmission
Hadoop is
Cost effective fault tolerant storage (HDFS)
Scalable
Data that is ingested mat be interpreted at run time
Low cost in storing unstructured and semistructured data
Fast transfer if data into storage
Separation of programming logic and scheduling/management

Ch4
HDFS Interface
Codes^

Ch5
Key value pair
In MapReduce, Key dont need to be unique.
Value is data associated with key
It is usually done in a sequence
Map > partition > shuffle and sort > reduce

Map phase
Uses input format and record reader function to derive records in the form of key-value pair for the input data
The huge data input will be segmented into portions called as blocks and if there are n blocks, there will be n map tasks.
Each map task operates on one filesystem block

Reduce phase is output of the map phase through shuffle and sort.

Combine phase is when the reduce function is commutative and associative which allows it to be performed before the shuffle and sort phase (For example, sum and count is commutative and associative, but average is not)

Ch6
Miss part 1

Word count the hello, world of map reduce

Ch7 Hive

It is a software system that provides tabular view of data stored in HDFS and sql methods for manipulating data

Contrary to Pig, Hive provides SQL like abstractions on top of Mapreduce

Language used is HQL, hive query language.

Provides tabular view of data and can be used to access data located in HDFS

Hive frees data analyst from Java MapReduce programming skills

Metastore contains mappings of tables to the directory locations in the HDFS

It is a relational database read and written by the Hive client

Metastore also includes the input and output formats for the files represented by the table objects.

It does not move the data from the datanodes

Hive interface provides a command line interface that accepts and parses HQL commands

It also provides JDBC/ODBC connector (drivers) to work with other tools

- Beeline (CLI)
- Oracle sql developer (GUI)
- Talend open studio (data extraction, transformation, loading (ETL) and integration tools
- Jasper reports, qlikview (business intelligence reporting tool
- Microsoft excel (data analysis) and tableau (data visualisation tools)

Provides storage handler mechanism to integrate with HBase

HUE(hadoop user experience) provides unified web interface to HDFS and hive

HCatalog provides metadata management system for Hadoop, Pig, Hive and MapReduce

Hive vs relational DBMS

Similarity

- Tabular view of data objects in HDFS
- Directories and files viewed as ables
- Typrs of columns in tables
- Access to tables through HQL which is similar to SQL
- API interface the same as JDBC programming interface

Differences

- Load and read only datamanagement system based on impleemtnation of HDFS
- Possible to access data visible in tabular format in Hive directly through HFS
- Update supported as coarse grained transformation instead of fine grained transformation in relational DBMS
- No transaction processing system
- No verification of consistency constraints such as PK, FK, domain constraints etc.

OLAP (online **Analytical**) vs OLTP (online **Transactional**)
OLTP systems are behind many everyday transactions from ATMs to in store purchases to hotel reservations as well as password changes and text messages
Traditional database systems designed and tuned to support the day to day operation:

- Ensure fast concurrent access to data
- Transaction processing the concurrency control
- Focus on online update data consistency
- Does a lot of transactions
- Rapid processing
- Multi User access to the same data
- Provide indexed datasets for rapid searching, retrieval and querying
- Does all the infrastructure stuff

OLTP database characteristics:

- Detailed data
- Do not include historical data
- Highly normalised
- Poor performance on complex queries including joins an aggregation

OLAP is a system for performing multi-dimensional analysis at high speeds on large volumes of data. It is ideal for data mining, business intelligence and complex analytical calculations, as well as business reporting functions like financial analysis, budgeting and sales forecasting.

OLAP characteristics

- OLAP focused on analytical queries
- Normalisation not good for analytical queries, reconstructing data requires a high number of joins
- OLAP databases support a heavy query load
- OLTP indexing techniques are not efficient in OLAP
- OLAP Cube

Uses OLTP to provide data to OLAP

Data granularity:
Level of detail at which measures are represented for each dimension of the cube

Datacube may be sparse or dense. Sparse meaning that there are a lot of empty cells without information

Hierarchies will allow data to be viewed at several granularities

Multidimensional measures can be

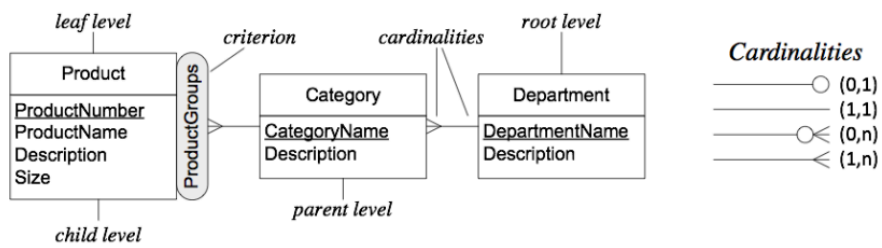
- Additive: Can be meaningfully summarised along all the dimensions, using addition
- Semi additive: Can be meaningfully summarised using addition along some dimension.

- Non additive: cannot be meaningfully summarised using addition across any dimensions
- Distributive: If application is distributed, then it can be combined before shuffle and sort. Functions like count, sum, min, max are distributive where the results will be the same even if the set of numbers is broken up to a few smaller sets.
- Algebraic: aggregation function which is formed when two functions are used to form a single function like the average function which requires the sum function and count function.

Conceptual model

- Allows better communication between designers and users to understand application requirements
- More stable than implementation oriented (logical) schema which changes with the platform
- Provide better support for visual user interface

A graphical notation used for a sample hierarchy



Dimension: level or one or more hierarchies

Hierarchy: several related levels

Level: entity type

Member: every instance of a level

Child and parent levels: the lower and higher levels

Leaf and root levels: first and last levels in a hierarchy

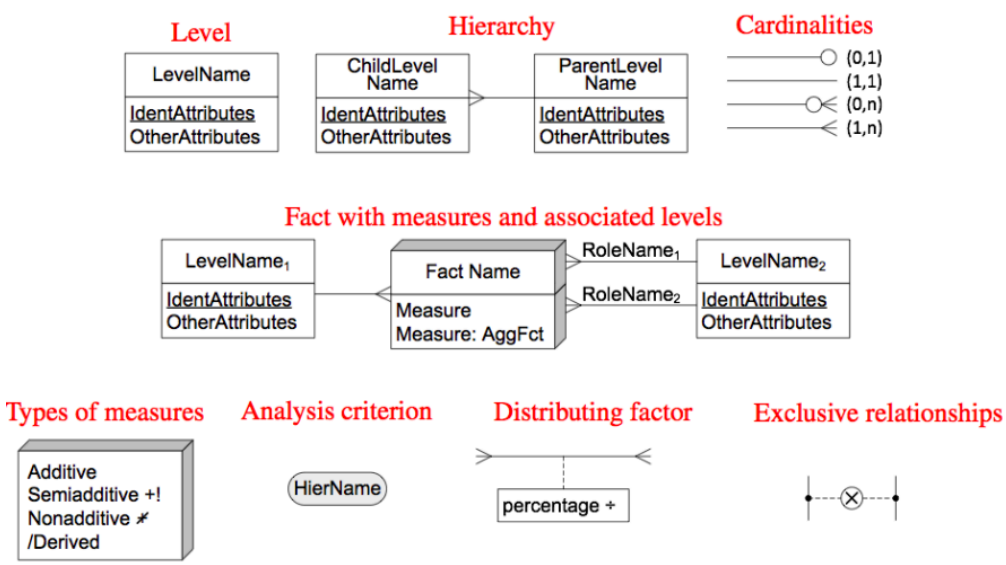
Cardinality: minimum/maximum numbers of members in a level related to members in another level

Criterion: expresses different hierarchical structures used for analysis

Key attribute: indicates how child members are grouped

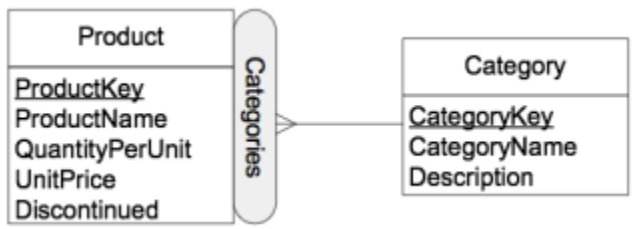
Descriptive attributes: describe characteristics of members

Dimension can be related several times to a fact with different roles (Time related to sales but one is duedate one is orderdate)

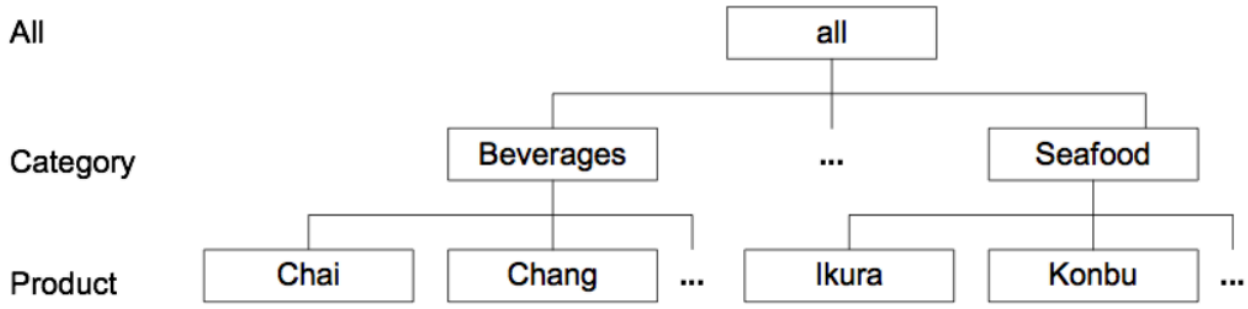


Balanced Hierarchies

At schema level: One path where all parent-child relationships are many to one and mandatory

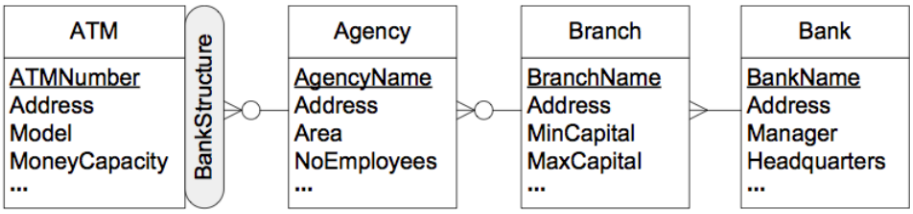


At instance level: Members form a balanced tree (all branches have same length) All parents members have at least one child member and a child belongs to exactly one parent

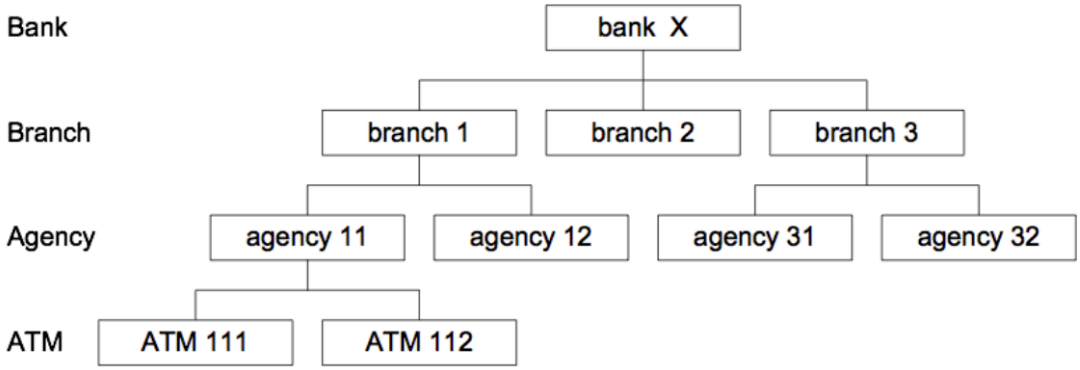


Unbalanced Hierarchies

At schema level: One path where all parent-child relationship are many to one but some are optional

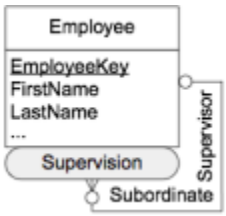


At instance level: members form a unbalanced tree

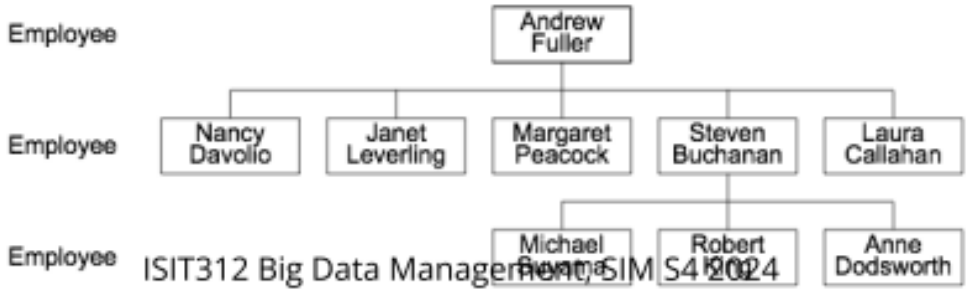


Recursive Hierarchies

At schema level: Special case of unbalanced hierarchies. The same level is linked by the two roles of a parent-child relationship. Characteristics of parent and child are similar or the same



At instance level:

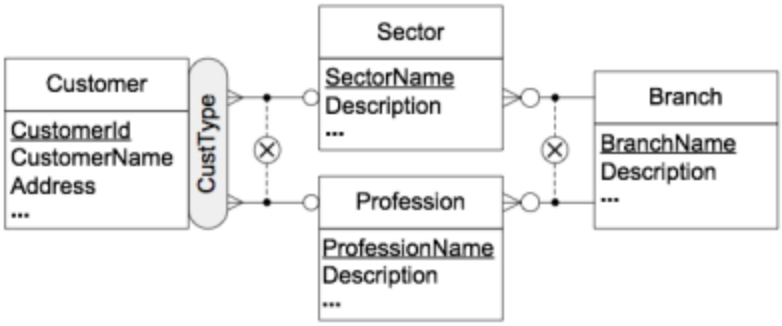


ISIT312 Big Data Management, SIM S4 2024

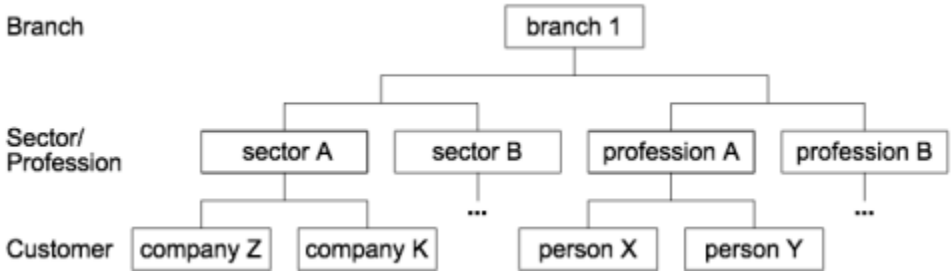
Generalised Hierarchies

At schema level: Multiple exclusive paths sharing at least the leaf level and may share other levels as well.

Two paths, one for each type of customer

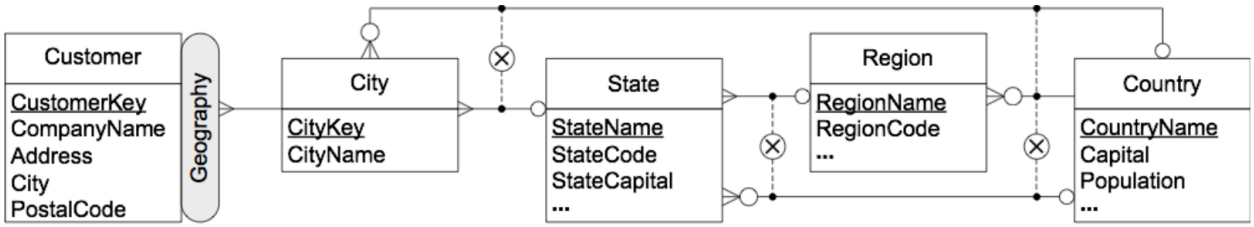


At instance level: Each member belongs to only one path

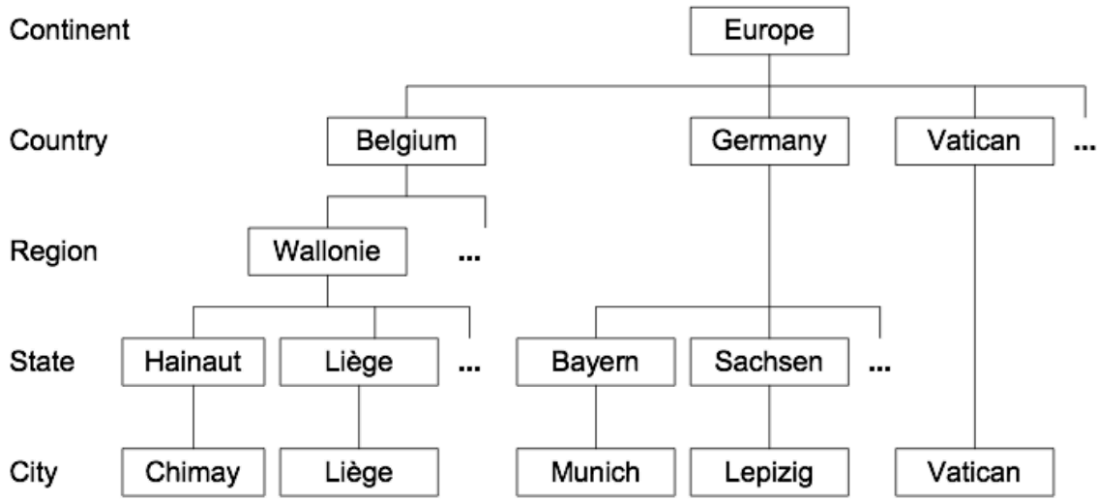


Non covering Hierarchies (Ragged or level skipping hierarchies)

At schema level: Alternate paths are obtained by skipping one or several intermediate levels



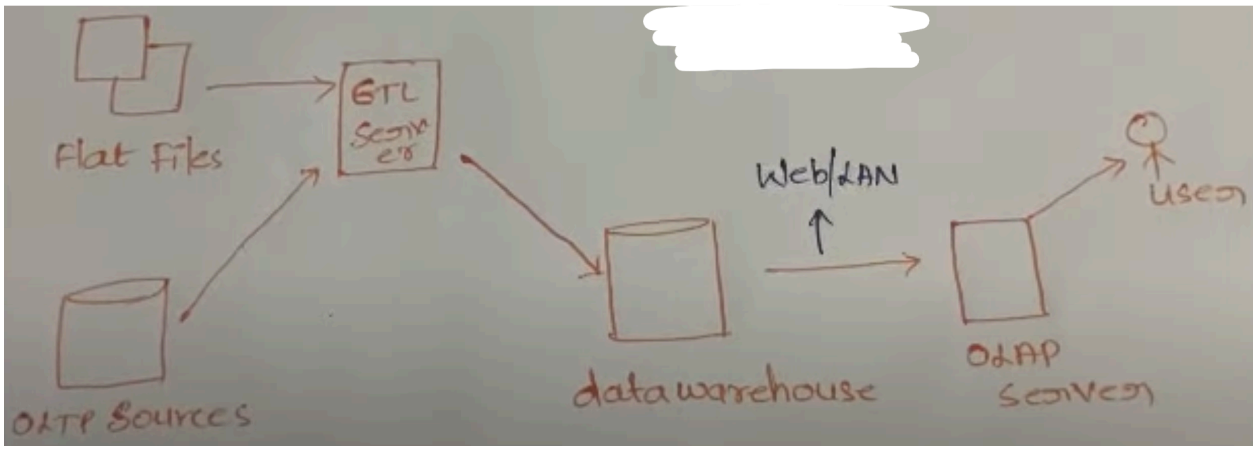
At instance level: Path length from the leaves to the same parent can be different for different members



OLAP

Relational OLAP (ROLAP): Stores data in relational database and supports extension to SQL and special access methods to efficiently implement the model and its operations

- Extension of RDBMS (presents data in tables) (Fact table and dimensional table are stored in the form of relations)
- Capable of large amount of data analysis
- Can store and analyse large changing amount of data

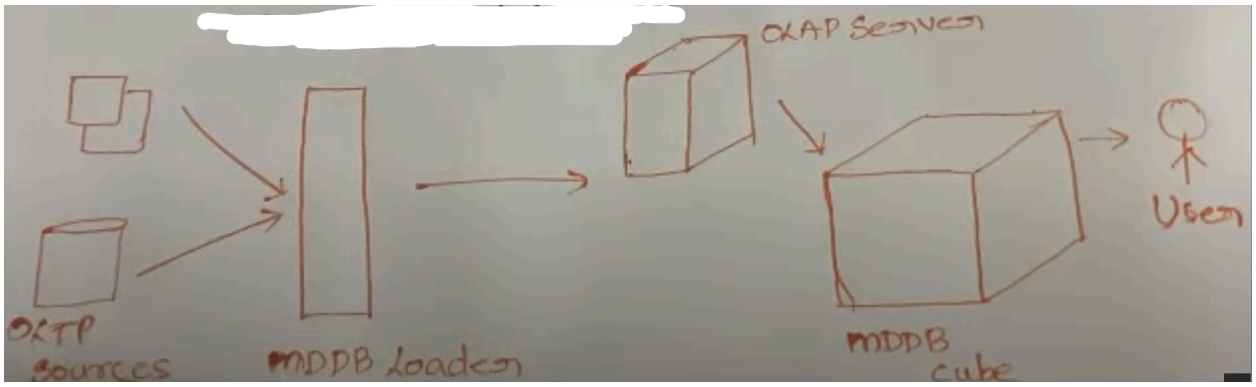


Flat files and OLTP sources goes to the ETL servers then to the data warehouse and transmits data using web/lan to OLAP servers.

Multidimensional OLAP (MOLAP): Stores data in special data structures (eg. arrays) and implement OLAP operations in these structures

- Better performance than ROLAP for query and aggregation but less storage capacity than ROLAP
- Limited amount of data
- Uses arrays
- Fast information retrieval as less data

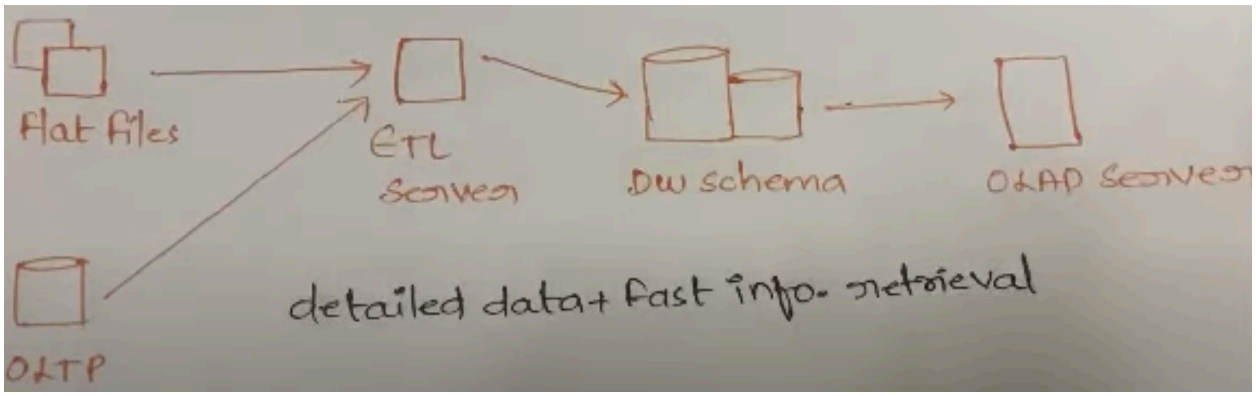
- Complex calculations
- Easy to use



Sources go to multidimensional database loader to OLAP servers to multidimensional database cube to user

Hybrid OLAP (HOLAP): combines both technologies

- Eg. detailed data stored in relational databases aggregations kept in separate MOLAP store, apply OLAP operations, store data in special data structure, etc

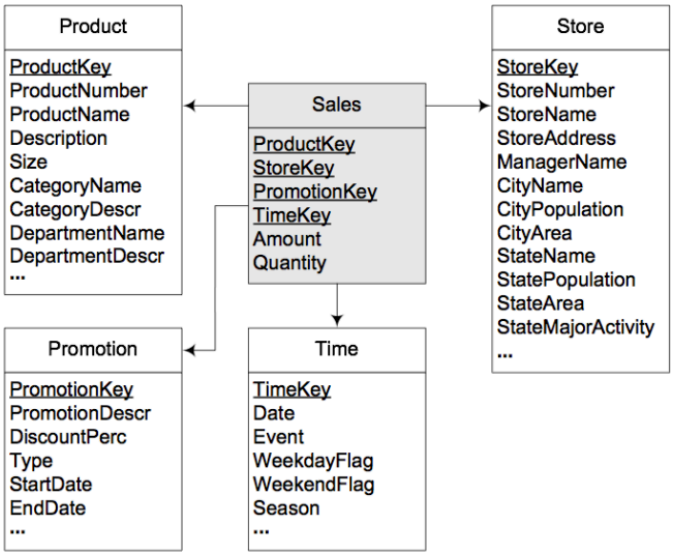


Sources sends to ETL servers to data warehouse schemas and to OLAP servers. It is able to have large amount of data and fast retrieval information as it is a combination of relational and multidimensional

ROLAP systems organizes tables in specialized structures:

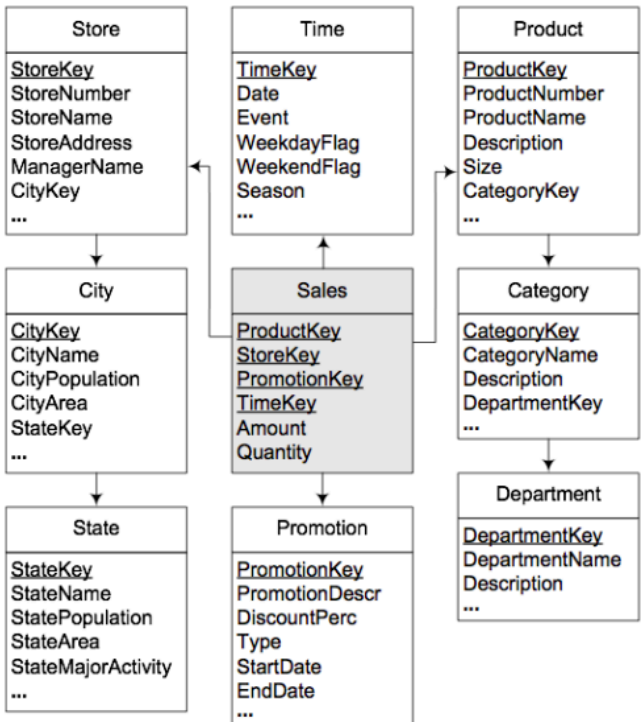
Star schema: One fact table and a set of dimension table

- Referential integrity constraints between fact table and dimension tables
- Dimension tables might contain redundancy in the presence of hierarchies
- Dimension tables denormalized, fact tables normalized



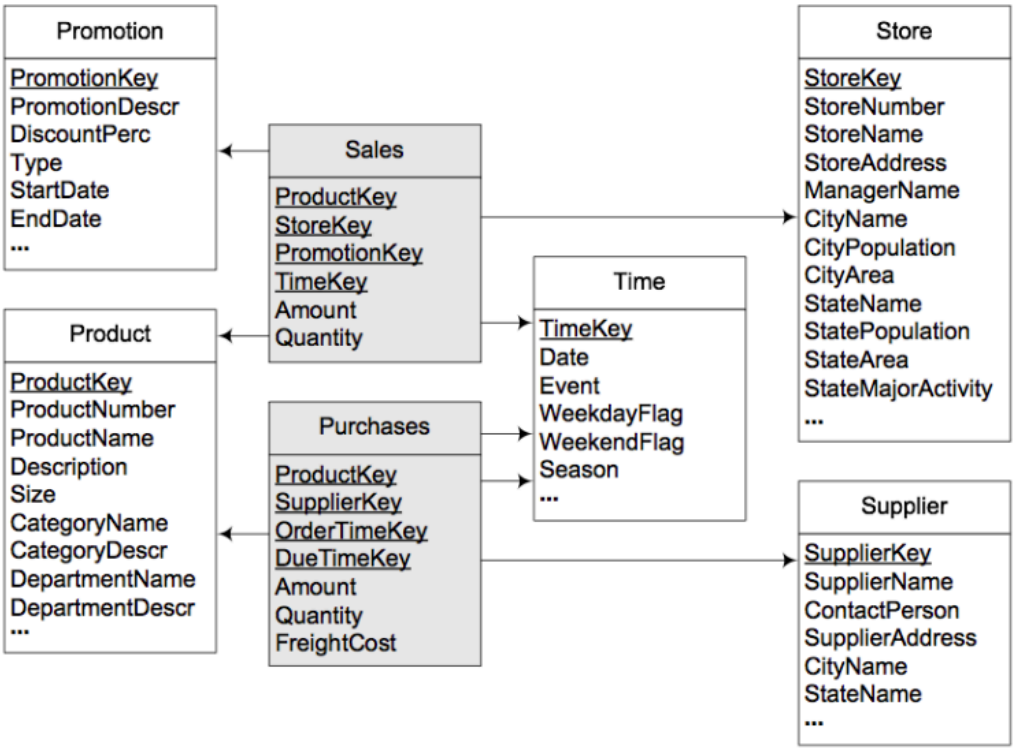
Snowflake schema: Avoids redundancy of star schemas by normalizing dimension tables

- Normalized tables optimize storage space but decrease performance



Starflake schema: Combination of the star and snowflake schemas, some dimensions normalized, other not

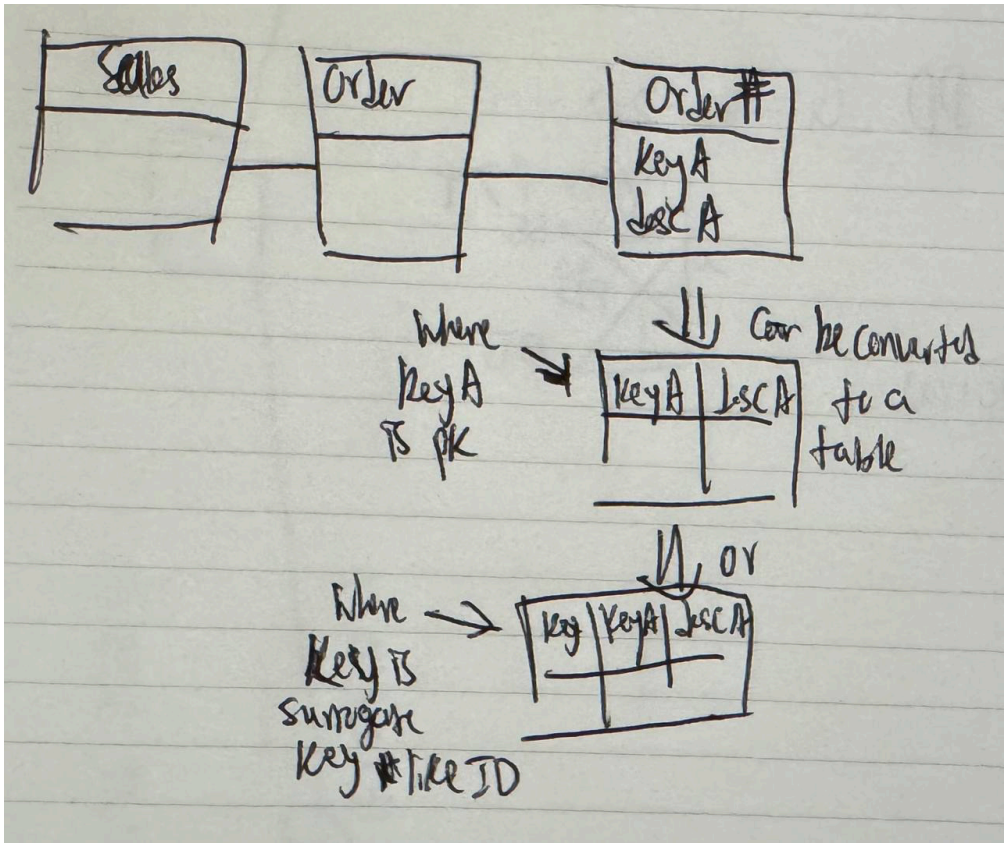
Constellation schema: Multiple fact tables that share dimension tables



Relational implementation of the conceptual model starts from 2:22:20 lecture 4

Rule 1: A level L, provided it is not related to a fact with a one to one relationship, is mapped to a table TL that contains all attributes of the level

- A surrogate key may be added to the table, otherwise the identifier of the level will be the key of the table
- Additional attributes will be added to this table when mapping relationships using Rule 3



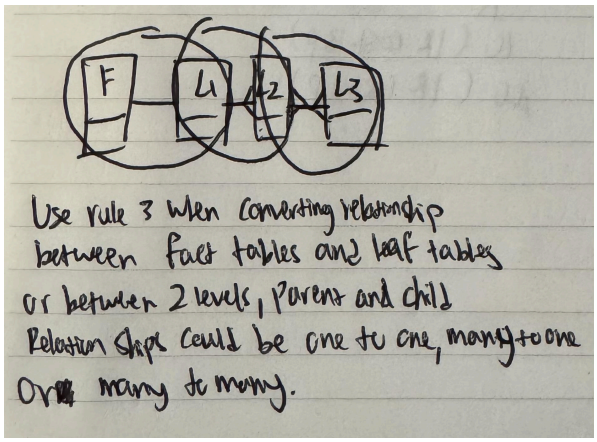
Rule 1 states that the level can be converted to a table straight away.
As keyA might be modified, it is better to define a surrogate key to prevent issues in implementation

Rule 2: A fact F is mapped to a table TF that includes as attributes all measure of the fact

- Surrogate key may be added to the table
- Additional attributes will be added to this table when mapping relationships using Rule 3 below

Rule 2 is the same as rule 1 where a surrogate key should be added to the fact table when converting to a relational table to prevent issues

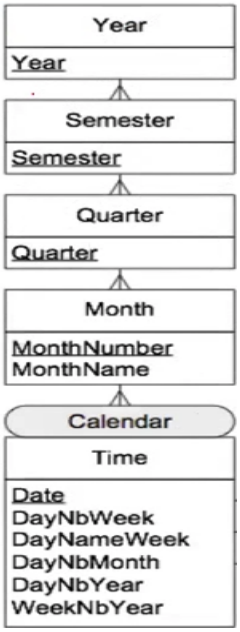
Rule 3: Relationship between either a fact F and a dimension level L, or between dimensions levels LP and LC (parent and child level respectively) can be mapped in three different ways depending on its cardinalities:



Higher level means it is the parent. Eg, lvl 1 will contain product id, but lvl 2 will have the description

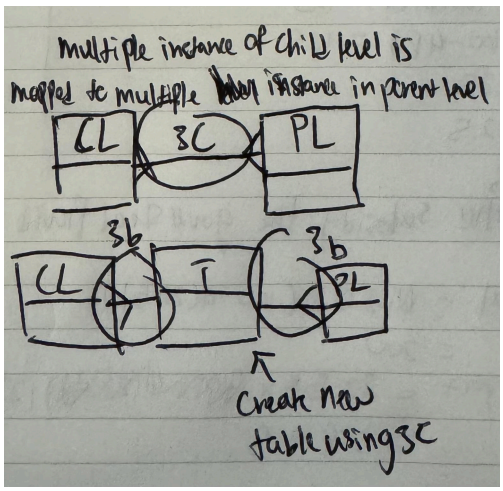
Rule 3a: If the relationship is one-to-one, the table corresponding to the fact or to the child level is extended with all the attributes of the dimension level or the parent level respectively. It means to merge the one to one relationship tables and it will reduce normalization but increase redundancy data/repeated data

Rule 3b: if the relationship is one-to many, the table corresponding to the fact or to the child level is extended with the surrogate key of the table corresponding to the dimension level or the parent level respectively, that is, there is a foreign key in the fact or child table pointing to the other table. Merge tables with one to many relationships and create a surrogate key as PK.

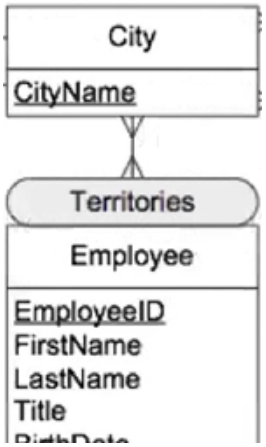


<- converts into this table using rule 3b ->

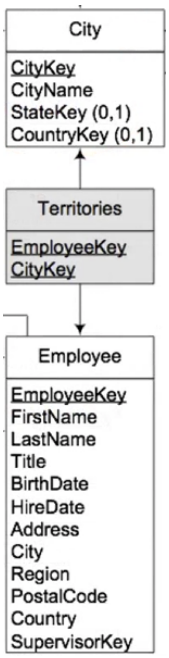
Rule 3c: If the relationship is many-to-many, a new table TB (standing for bridge table) is created that contains as attributes the surrogate keys of the tables corresponding to the fact (TF) and the dimension level (TL), or the parent (TP) and child levels (TC), respectively. If the relationship has a distributing attribute, an additional attribute is added to the table to store this information.



eg.

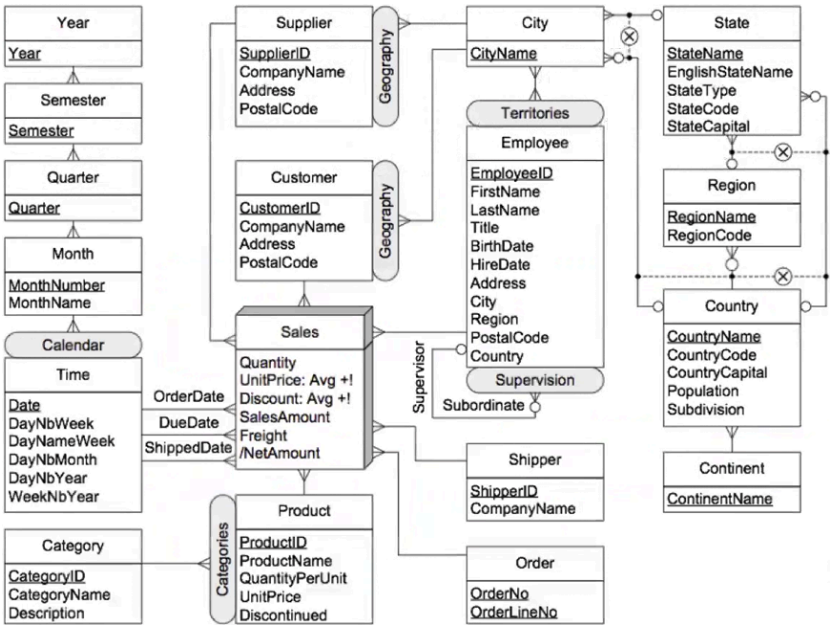


<- convert ->



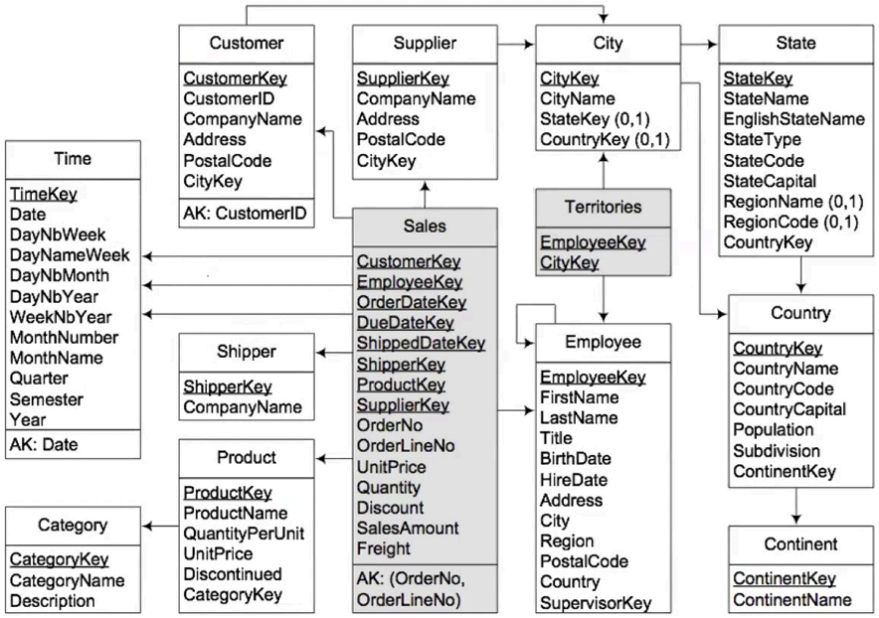
Territories table is created in rule 3c

Conceptual schema



Converts into:

Relational representation



Redundant information in the Time table is alright as it is going to be accessed frequently so that there will be fewer join functions, but city state can be normalized to save space as it is not accessed frequently.

Time dimension is present in almost all data warehouses.
In a star or snowflake schema, time is included both as foreign key in a fact table and as a time dimension containing the aggregation levels.

The granularity of the time dimension varies depending on their use. The number of tuples depends on the amount of data variation needed.

File 15-19 not written.