

Data Warehousing

Databases are great, but for business: Too many of them; Everybody got what was best for them; Eventually this re-created the problem databases were meant to solve (duplicated, inaccessible, inconsistent)

=> An integrated way of getting the ENTIRE organisational data: **Informational Database**: a single database that allows all of the organisations data to be stored in a form that can be used to support organisational decision processes

Introduction to Data Warehousing

Warehouse: An Informational Database

Data Warehouse: a single repository of organisational data; integrates data from multiple sources; Makes data available to managers/users and Supports analysis and decision-making

Involve a large data store (Terabytes, Petabytes...)

Informational: DW 专门用来应对处理大量信息的

Analytical Queries

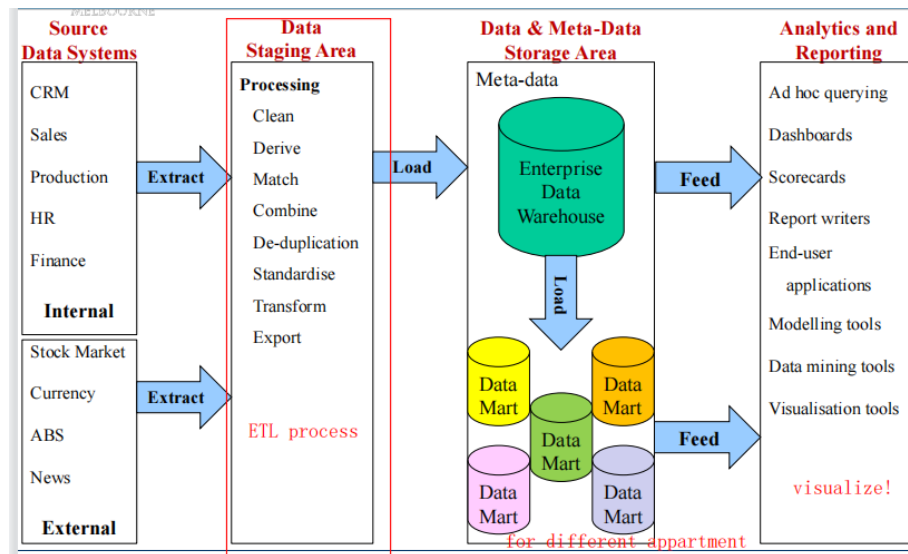
1. Numerical aggregations (How many? what's the average? what's the total cost?)
2. Understanding dimensions

DW can help answer these questions

Characteristics of a DW

1. **Subject oriented** (e.g. sales, customers, products)
2. **Validated, integrated data**
data converted to a common format => allow comparison and consolidation of data from different sources
data are validated before storing in a data warehouse
3. **Time variant**
Historical data: Data consists of a series of snapshots / timestamps
Required by trend analysis, which is crucial for decision support
4. **Non-volatile**
Users have read access only--all updating done automatically by ETL process and periodically by a DBA;
ET: Extract data from source system, transforms, loads into the warehouse

Architecture



Dimensional Modelling

AKA **star schema** design

Based on the multi-dimensional model of data and designed for retrieval-only databases.

Very simple, intuitive and easily-understood structure

Consists of: Fact table, several dimensional tables, (sometimes) hierarchies in the dimensions

Fact Table



Contains

1. the actual business measures (additive, aggregates), called facts
2. foreign keys pointing to dimensions
3. Granularity or level of detail is a key issue
 - Finest level of detail for a fact table determined by the finest level of each dimension

Dimension Table

Captures a factor by which a fact can be described or classified.

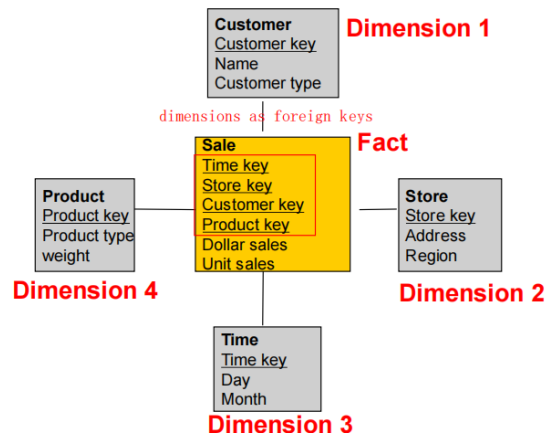
Each dimension can contain hierarchies (name, type, group)

Star Schema

Two table joins can get all dimensions (one layer)

Fact table is an intersection table

Star schemas are organized around facts (business measures) and dimensions that help with managerial decision making. What's more, they are denormalised, making it faster to aggregate and query data.



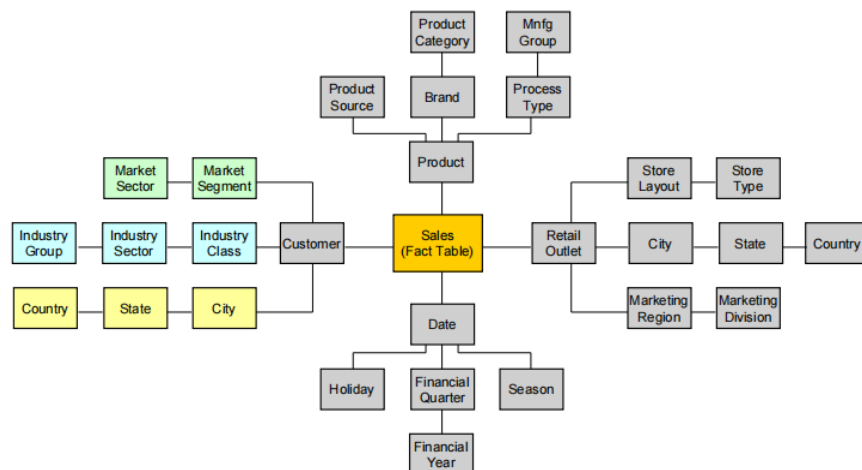
Designing Steps (Kimball)

1. Choose a business process
2. Choose the measured facts (numeric, additive quantities)
3. Choose the granularity of the fact table
4. Choose the dimensions
5. Complete the dimension tables

Briefly

1. identify business process & measured facts
2. choose the granularity
3. Fact table
4. Dimension table

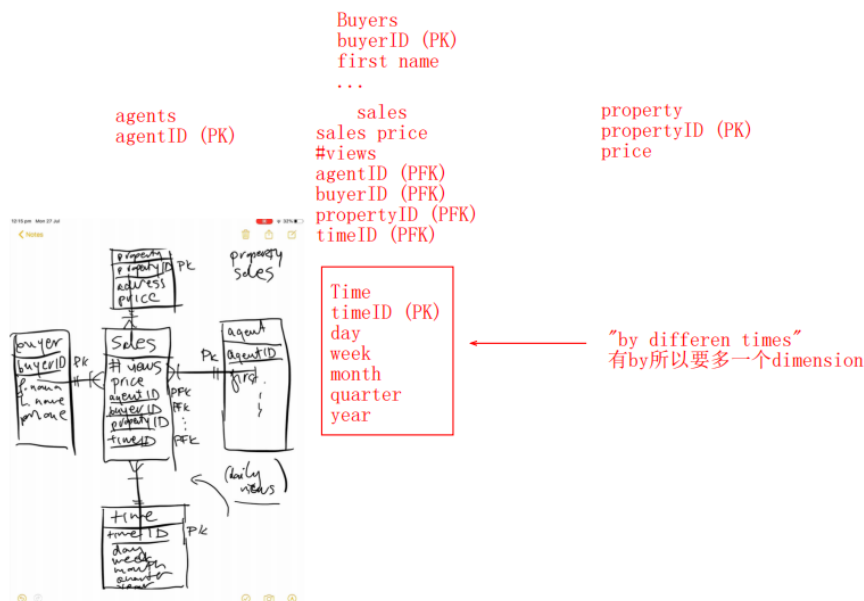
Snowflake Schema: hierarchy in dimensions



Design Outcomes

- Normalisation
Eliminates redundancy; Storage efficiency; Referential Integrity
- Denormalisation
Fewer tables (fewer joins); Fast querying; Design is tuned for end-user analysis

- We are making a data warehouse for a real estate agency. The company wants to track information about the **selling** of their properties. This warehouse keeps information about the **agents** (license#, first name, last name, phone #), **buyers** that come in (buyer id, first name, last name, phone #), and **property** (property#, property address, price). The information managers want to be able to find is **the number of times a property is viewed**, **sales price**. The information needs to be accessible **by rental agent, by buyer, by property** and **for different time** (day, week, month, quarter and year).
- Draw a star schema to support the design of this data warehouse.



Distributed Database

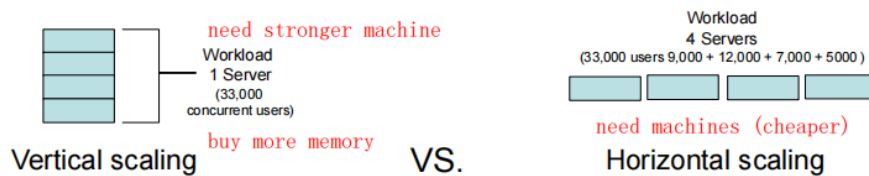
Definitions

- Distributed Database**
a single **logical** database physically spread across multiple computers in multiple locations that are connected by a data communications link
appears to users as though it is one database
- Decentralized Database**
a collection of independent databases which are not networked together as one logical database
appears to users as though many databases

Pros and Cons

Advantages of DDBMS

- Good fit for geographically distributed organizations/users** (utilize the internet)
- Data located near site with greatest demand** (e.g. Facebook data)
- Faster data access** (to local data)
- Faster data processing** (workload split amongst physical servers)



5. **Allows modular growth** (add new servers as load increases-horizontal scalability)
6. **Increased reliability and availability** (less danger of a single-point of failure (SPOF) if data is replicated)
7. **Supports database recovery** (when data is replicated across multiple sites)

Disadvantages of DDBMS

1. **Complexity of management and control** (db & application must stitch together data across sites)
2. **Data integrity** (what if two users in two locations update the record at the same time? => **Transaction Manager/Master-slave design**)
3. **Security** (Many server sites -> high change of breach)
4. **Lack of standards** (different DDBMS vendors use different protocols)
5. **Increased training & maintenance costs** (disk storage, inter network infrastructure, clustering software, Network speed)
6. **Increased storage requirements** (replication model)
7. **Location transparency**
A user accessing data do not need to know the location of the data in the network of DBMS
Automatically forwarded by the system to the site or sites related to the processing request.
A single query can join data from tables in multiple sites
8. **Local autonomy** (local users can continue use if network's lost)
Being able to operate locally when connections to other databases fail.
Users can administer their local database (control local data, administer security, log transactions, recover when local failures occur, provide full access to local data)

Functions of a DDBMS

1. Locate data with a distributed catalog (meta data)
2. Determine location from which to retrieve data and process query components
3. DBMS translation between nodes with different local DBMSs (middleware)
4. Data consistency (multiphase commit protocols)
5. Global primary key control
6. Scalability
7. Security, concurrency, query optimization, failure recovery

Distribution options

Data replication-a process of duplicating data to different nodes **Data partitioning**-a process of partitioning data into subsets that are shipped to different nodes

Many real-life systems use a combination of two (partition data and keep some replicas around, 3)



Replication: Pros and Cons

Advantages

1. High reliability due to redundant copies of data
2. Fast access to data at the location where it is most accessed
3. May avoid complicated distributed integrity routines (Replicated data is refreshed at scheduled intervals)
4. Decoupled nodes don't affect data availability (Transactions proceed even if some nodes are down)
5. Reduced network traffic at prime time (if updates can be delayed)

This is currently popular as a way of achieving high availability for global systems (e.g. Most SQL & NoSQL databases offer replication)

Disadvantages

1. Need more storage space (copy)
2. Data integrity (High tolerance for out-of-date data may be required; updates may cause performance problems for busy nodes; Retrieve incorrect data if updates have not arrived)
3. Takes time for update operations
4. Network communication capabilities (Updates can place heavy demand on telecommunications/networks; High speed networks are expensive)

Partitioning

Split data into chunks, store chunks in different nodes; A chunk can be a set of rows or columns =>

Horizontal partitioning & Vertical partitioning

Horizontal Partitioning

Different rows of a table at different sites

Pros

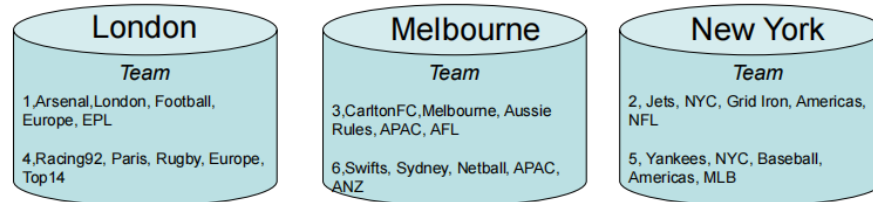
1. Data stored close to where it is used--efficiency
2. Local access optimization--better performance
3. Only relevant data is stored locally--security
4. Unions across partitions--ease of query

Cons

1. accessing data across partitions--inconsistent access speed
2. No data replication--backup vulnerability (SPOF)

ID	Team	City	Code	Region	League
1	Arsenal	London	Football	Europe	EPL
2	Jets	NYC	Grid Iron	Americas	NFL
3	Carlton FC	Melbourne	Aussie Rules	APAC	AFL
4	Racing92	Paris	Rugby	Europe	Top14
5	Yankees	NYC	Baseball	Americas	MLB
6	Swifts	Sydney	Netball	APAC	ANZ

Horizontal Partitioning based on Region






Vertical Partitioning

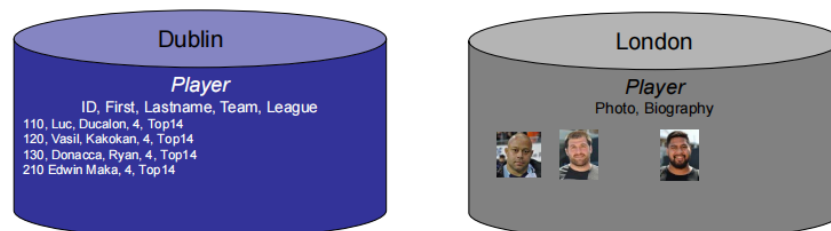
Different columns of a table at different sites

Pros and cons are the same as for horizontal partitioning except

- Combining data across partitions is more difficult, because it requires joins (instead of unions)

ID	Firstname	Lastname	Team	League	Photo	Biography
110	Luc	Ducalon	4	Top14		Ipsa locum
120	Vasil	Kakokan	4	Top14		Ipsa locum est
130	Donacca	Ryan	4	Top14	<null>	
210	Edwin	Maka	4	Top14		

Vertical Partitioning based on column requirements

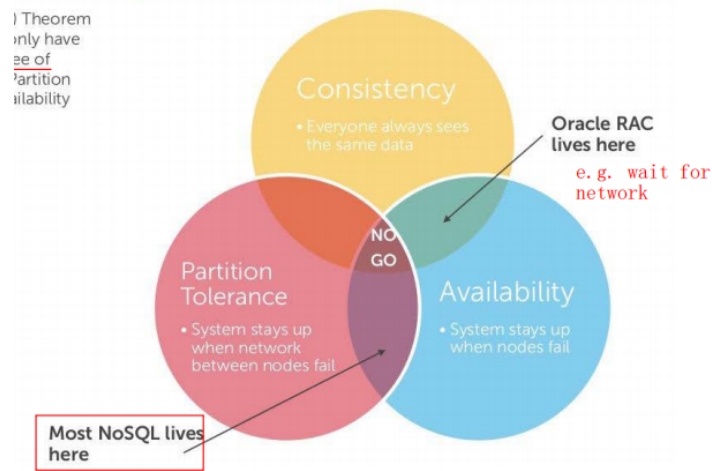


Trade-offs with DDBMS

- Availability vs Consistency => CAP theorem
- Synchronous vs Asynchronous updates (changes immediately visible everywhere or later propagated?)

The CAP theorem

You can only have two out of three of Consistency, Availability, and Partition Tolerance



Synchronous vs Asynchronous updates

Synchronous-Data is continuously kept up to date

Pro: Ensures data integrity and minimizes the complexity of knowing where the most recent copy of data is located

Con: Can result in slow response time and high network usage

(DBMS spends time checking that an update is accurately and completely propagated across the network; the committed updated record must be identical in all servers)

Asynchronous-Some delay in propagating data updates to remote databases

(Some degree of at least temporary inconsistency is tolerated; may be ok it is temporary and well managed)

Pro: Acceptable response time

(updates happen locally and data replicas are synchronized in batches and predetermined intervals)

Con: May be more complex to plan and design

Which update to use depends on the information systems:

e.g. Commerce/finance systems-Synchronous; Social media-Asynchronous

The CAP theorem has three key components:

- **Consistency** – All the servers hosting the database will have the same data, so that anyone accessing the data will get the same copy regardless of which server is answering the query. (This is a different thing to “consistency” in the context of the ACID principles, which refers to data integrity.)
- **Availability** – The system will always respond to a request even if it is not the latest data or consistent across the system.
- **Partition tolerance** – A “partition” in the context of the CAP theorem refers to a disruption in network access so that some servers are unable to access other servers. The system continues to operate as a whole even if individual servers fail or can’t be reached.

The CAP theorem states that, at any given point in time, a system can achieve *two* out of three principles, while it is theoretically impossible to achieve three at the same time. In case of NoSQL databases, the choice is between AP or CP, as the biggest advantage of NoSQL databases is partition tolerance when compared to relational DBMSs:

- **AP:** The database always answers, but possibly with outdated or wrong data, hence ensuring *availability* instead of consistency. On systems that allow reads before updating all the nodes, high availability is achieved. Such systems eventually achieve consistency as well. For example, Google and Facebook enforce eventual consistency such that different servers might have inconsistent views depending on how many servers are updated at a given time.
- **CP:** The database stops all the operations until the latest copy of data is available on all nodes. On systems that lock all the nodes before allowing reads, high consistency is achieved. Such systems become available after the consistency is achieved. Most NoSQL databases choose AP over CP to ensure continuous availability and eventual consistency.