Large Scale Data Management

Relational Data and Data Warehouses

Silviu **Maniu**, LIG, Univ. Grenoble Alpes

Relational Data and DBMS

What are Databases / DBMS

IT applications:

- Compute simulations, train models, predict
- Manage devices, communicate
- **Store and process data**

Databases (DB)

Organized collection of data. This collection must have a regular structure, the data is meant to capture a subset of the "real" world; the collections are connected to each other and deal with a same subject. The data are organized in a way that facilitates their manipulation (access, updates).

Database Management System (DBMS)

Dedicated *software* to manage databases. Organizes data storage, handles access to data: queries, updates. . .

Database Uses

Enterprise management:

- Accounting
- Supply chain management (inventory...)
- staff management
- orders, reservations

Scientific data (medicine, biology...).

Sensor data

Administrative data (Healthcare, demographics, economy)

Back-end for everyday applications:

- websites: CMS (e.g., Joomla, Wordpress)
- emails/contacts/certificates (Thunderbird, Firefox)

Everyday life use cases:

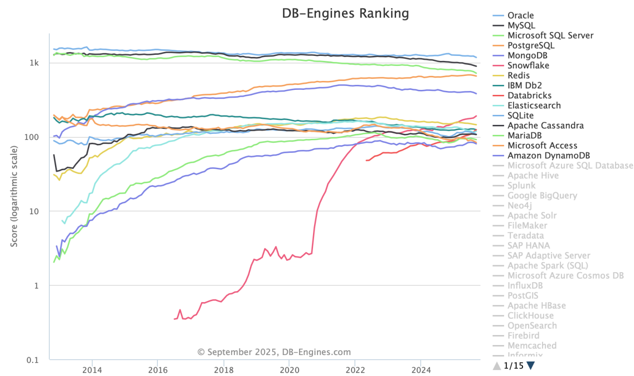
- Booking hotel, train tickets
- Ordering on e-commerce
- Banking account

DBMS Vendors

open source:

DBMS Popularity

https://db-engines.com/



DMBS Functionalities

present in all

- Persistent storage of data in files (terabyte level) *disks, buffer, dictionnaries*
- Performance for queries   indexes, query optimization...
- Multi-user (concurrent access) *transactions, locks, MVCC*
- Security (access control) *access rights*
- Ease to use (queries, maintenance, evolution)
- Reliability as a system (recovery after failure...) *logs, backups*
- Data integrity (consistency) *normalization, integrity constraints (FK)*

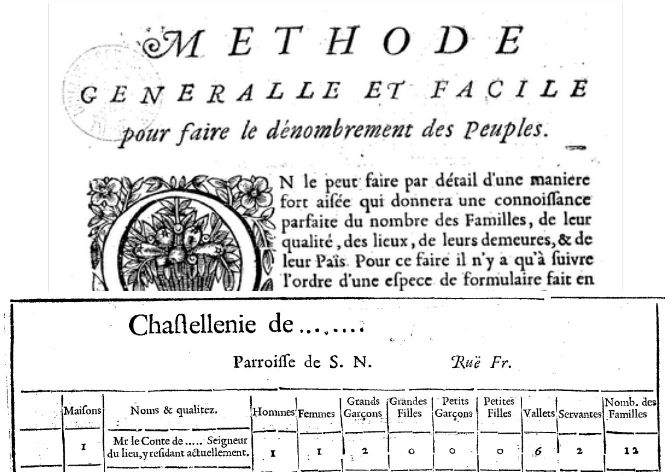
When are they *not* good

NoSQL solutions (or even plain files) may be a better choice when:

- you have too much data: > 1TB
- you need very low latency
- tiny IOT devices with limited memory
- the relational model does not fit your data (graph, time series) and queries
- your software stack integrates better with other data storage technology

Historical Overview

Census systems



METHODE GENERALLE ET FACILE pour faire le dénombrement des Peuples.

ON le peut faire par détail d'une maniere fort aisée qui donnera une connoissance parfaite du nombre des Familles, de leur qualité, des lieux, de leurs demeures, & de leur Païs. Pour ce faire il n'y a qu'à suivre l'ordre d'une espece de formulaire fait en

Chastellenie de ....

Parroisse de S. N. Ruë Fr.

Maisons	Noms & qualitez.	Hommes	Femmes	Grands Garçons	Grandes Filles	Petits Garçons	Petites Filles	Vallets	Servantes	Nomb. des Familles
1	Mr le Conte de ..... Seigneur du lieu, y residant actuellement.	1	1	2	0	0	0	6	2	12

Historical Overview

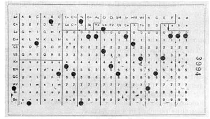
Automatic Census

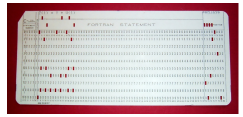
H.Hollerith: mecanography (punch card machines) for 1890 US census.

starts what will become (part of) IBM.

Punch cards: main storage device (data&program) from 1900 to 1950.

1959 Archive center: 2000 cards per box: total in warehouse 4GB.





80 column card (12lines)

Historical Overview

Hardware storage systems (HDDs)

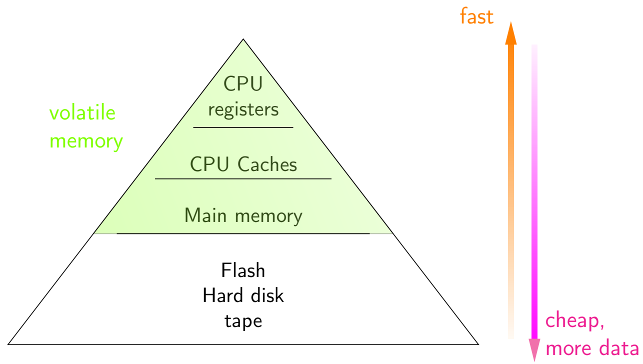
Introduced by IBM in 1956.

Several disks (platters) spinning fast (7200 rpm), coated with thin (10nm) ferromagnetic layer. Concentric tracks on each disk. 1 read/write head per disk (floats on air cushion)

Storage systen for which current DBMS have been optimised

Historical Overview

Hardware memory hierarchy



Historical Overview

History of DBMS (1)

- 1960: First DBMS (accounting, logistics):
  - hierarchical (IMS, by IBM)
  - network (IDS for General Electric)

  Main idea: high performance

- 1970: relational model by Codd at IBM San Jose (Almaden)

  Simple model. Goal: increase productivity.

  Avoid redundancies/inconsistencies. Theory behind (logics) attracts academic research and industry adoption

  SystemR (IBM,74), Ingres (UC Berkeley,75), Oracle (79)

Historical Overview

History of DBMS (2)

- 1980-90: relational model dominates the market.

  SQL gets richer, new data types (images, text), huge volume (1TB) analytical queries (DataWarehouses). Simplified installation (PC). Prototypes: Object-oriented.

- 2000: datacenters, digital economy.

  Prototypes: XML DB, multimedia. Keyword search, recommender systems. Federated DB: integrate information from multiple (heterogeneous) sources.

- 2010-20: Web as a major data source. Cloud storage.

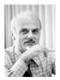
  New hardware (FPGA, GPGPU, SSD).

  Storage/data processing revisited:
  - in-memory DB (columns...)
  - **NoSQL**: massively parallel architectures (Map/Reduce, key-value systems...).
  - widespread adoption of AI and big data technology

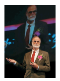
Historical Overview

Turing Awards

Bachman 1973 For his outstanding contributions to database technology
Designed Integrated Data Store(IDS) by 1963. Ideas: store data in single place. Data Manipulation Language. Highly efficient.



Codd 1973 For his fundamental and continuing contributions to the theory and practice of database management systems.
Early parallel programming. relational model: data modeling, normalization, relational algebra, connection to logics. OLAP.



Gray 1998 For seminal contributions to database and transaction processing research and technical leadership in system implementation.
Foundations of transaction processing. GIS. Fault-tolerant DBMS.

Stonebraker 2014 For fundamental contributions to the concepts and practices underlying modern database systems.
INGRES. Postgres. Vertica. VoltDB. SciDB.

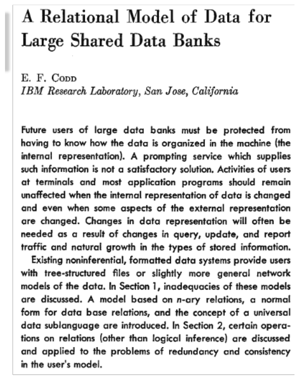
DBMS Fundamentals

Data model: The relational model

Relational model = data represented by tables

- Prevailing DB model
- Formalised by E.F. Codd (@IBM, Turing'81)
- table = relation

id	name	address	city
23	Maniu	38000	Grenoble
67	Sen	38100	Grenoble
98	Letellier	38000	Grenoble
58	Idani	75015	Paris



A Relational Model of Data for Large Shared Data Banks

E. F. Codd
IBM Research Laboratory, San Jose, California

Future users of large data banks must be protected from having to know how the data is organized in the machine (the internal representation). A prompting service which supplies such information is not a satisfactory solution. Activities of users at terminals and most application programs should remain unaffected when the internal representation of data is changed and even when some aspects of the external representation are changed. Changes in data representation will often be needed as a result of changes in query, update, and report traffic and natural growth in the types of stored information.

Existing noninferential, formatted data systems provide users with tree-structured files or slightly more general network models of the data. In Section 1, inadequacies of these models are discussed. A model based on *n*-ary relations, a normal form for data base relations, and the concept of a universal data sublanguage are introduced. In Section 2, certain operations on relations (other than logical inference) are discussed and applied to the problems of redundancy and consistency in the user's model.

Communications of the ACM, 13(6), 1970

DBMS Fundamentals
Query language: SQL

SQL = structured query language used on relational model

- introduced in 1974, standardized afterward
- successive versions enriched the language (last version: SQL 2016)
- prevailing query language for DBs.
- declarative : describes expected output, not computation process

```
select name from persons where address='38000'
```

id	name	address	city
23	Maniu	38000	Grenoble
67	Sen	38100	Grenoble
98	Letellier	38000	Grenoble
58	Idani	75015	Paris

name
Maniu
Letellier

The result of a query on a relation is a relation!

DBMS Fundamentals
Schema vs Instance

Schema: describes data organization.

In the relational model: table schema provides the format of each column in each table

e.g.: `Persons(id:int, namee:string, address:string, city:string)`

N.B. this description of data is itself (meta)data, stored in the DB and accessible to user queries.

Instance: the actual data in DB (that must be organized according to the schema).

One can view the instance as the current state of relation

DBMS Fundamentals
Constraints and Normalisation

Constraints: define rules for the database to be consistent

- Unicity (key): ensures values are not repeated
- Foreign key: ensures values are taken from valid values present in other tables

Normalisation: technique for eliminating redundancies in databases

- Uses the concept of table normal forms, introduced by Codd in the original model, extended afterwards (1NF, 2NF, 3NF, BCNF, …)

id	name	address
23	Maniu	38000
67	Sen	38100
98	Letellier	38000
58	Idani	75015

key

foreign key

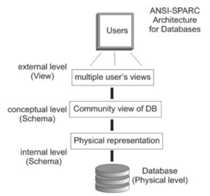
address	city
38000	Grenoble
38100	Grenoble
75015	Paris

DBMS Fundamentals
Three-level architecture

external: manages how apps access data; defines what each user sees of the DB

logical: defines the structure of data (schema, constraints)

physical: defines data storage and organization on disk (files, indexes)



ANSI-SPARC design standard, introduced in 1975, widespread

goal: containing the impact of modifications within a level

DBMS Fundamentals

ACID Properties in DBMS

Atomicity - each statement in a transaction (to read, write, update or delete data) is treated as a single unit (entire statement is executed, or none of it is executed)

Consistency - transactions can only occur if they keep the database in a consistent state

Isolation - concurrent transactions bring the database in the same state as if they were executed serially

Durability - committed transactions remain committed even when failure occurs

Data Warehouses: Introduction

What is a Data Warehouse

"society is data rich but information poor"

Business intelligence (BI): set of techniques and tools that enable a company to transform business data into into meaningful and useful information for decision making.

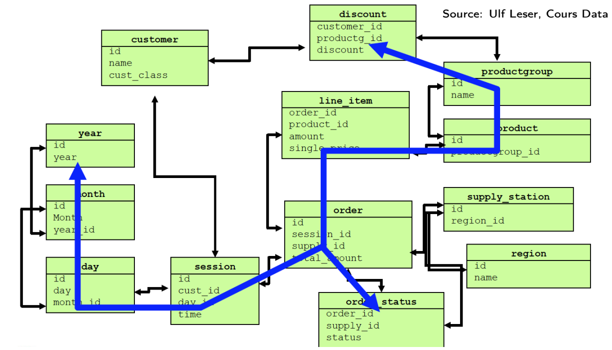
DataWarehouse (DW): repository that stores the data and infrastructure to support analysis.

according to R. Kimball:

> *"a copy of transaction data specifically structured for query and analysis"*

Motivation: example

"How many sales completed in December before Christmas per group of product and discount?"



(Relational) DBMS solution: large relations (millions of orders, sessions), numerous tables

Motivation: example

"How many sales completed in December before Christmas per group of product and discount?"

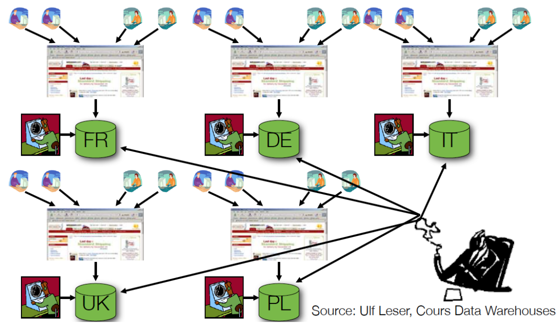
```
SELECT Y.year, PG.name, DI.disc, count(*)
FROM year Y, month M, day D, session S,
  line_item I, order O, product P, productgroup PG, discount DI, order_status OS
WHERE M.year_id = Y.id and
  D.month_id = M.id and
  S.day_id = D.id and
  O.session_id = S.id and
  I.order_id = O.id and
  I.product_id = P.id and
  P.productgroup_id = PG.id and
  DI.productgroup_id = PG.id and
  O.id = OS.order_id and
  D.day < 24 and
  M.month = 12
  and OS.status='FINISHED'
GROUP BY Y.year, PG.name, DI.discount
ORDER BY Y.year, DI.discount
```

Source: Ulf Leser, Cours Data Warehouses

(Relational) DBMS solution: large relations (millions of orders, sessions), numerous tables

Motivation: example

"How many sales completed in December before Christmas per group of product and discount?"



Amazon (US, FR, UK, …)

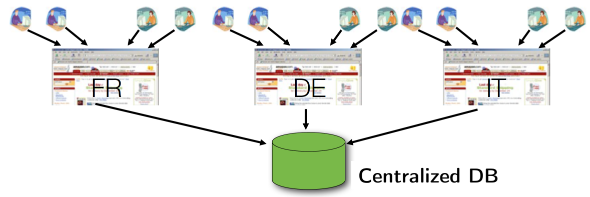
Motivation: example

"How many sales completed in December before Christmas per group of product and discount?"

1. Heterogeneity
   - the DB schemas are modified from time to time
   - some country-specific properties (VAT, shipping costs, etc.)
   - different data semantics (measurements, etc.)
2. Data volume
   - complex query = we can reuse the results of similar queries (views)
   - avoid transferring large volumes of data over the Internet
3. Divergent needs
   - no need to keep historical data to process transactions (delete processed orders)
   - analysts do not need the full details (customer name, supplier...)

Motivation

Limitations of the possible solutions

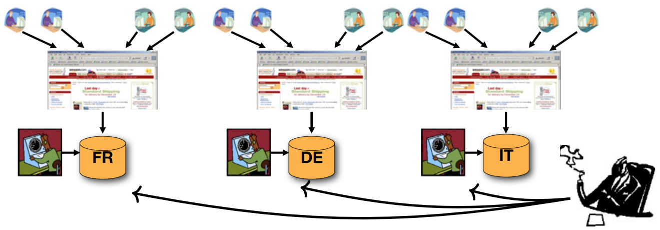


Solve the heterogeneity issue, but

- Each branch must connect across the network
- Long delay for operations
- Does not solve the problem of the data volume.

Motivation

Limitations of the possible solutions

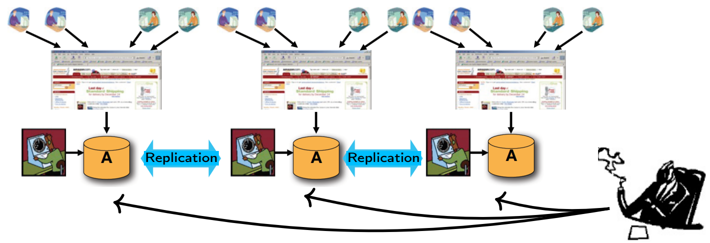


Reduced time for operational transactions but

- Does not solve heterogeneity
- Long delay for analytical queries

Motivation

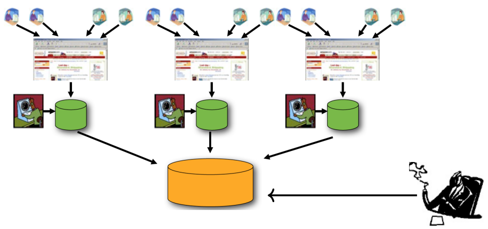
Limitations of the possible solutions



Reduced time for analytical requests but

- Very voluminous relations in local DBs
- Extended time for operational transactions

Data Warehouse Solution



- Redundant data
- DW stores only selected and transformed data
- Specific data model
- Asynchronous update of warehouse data
- Specific tools (preparation, data visualization)

Data Warehouses

Definition

William H.Inmon's definition ['92]

A data warehouse is a *subject oriented*, *integrated*, *time varying*, *non-volatile* collection of data in support of management's decision making process.

- subject oriented: the DW organized according to one or several domains determined by the analysts' requirements
- integrated: the content results from the integration of data from multiple sources
- time-varying: keeps track of data changes so that reports show evolution over time
- non-volatile: new data can be added, but data is almost never deleted nor updated

OLAP vs. OLTP

OLTP (OnLine Transaction Processing) : “Standard”, operational/transactional databases

OLAP (OnLine Analytical Processing) : Data warehouses, decision-making support

OLAP vs. OLTP

Aspect	Operational DB	DW
User	clerk	manager
Concurrency	huge (thousands)	limited (hundreds)
Interaction	short (s)	long analyses (min,h)
Type of interaction	Insert, Update, Delete	Read,periodically (bulk) inserts
Type of query	many simple queries	few, but complex queries (typically drill-down, slice. . .)
Query scope	a few tuples (often 1)	many tuples (range queries)
Data source	single DB	multiple independant DB. . .
Schema	query-independant (3NF)	based on queries
Data	original, detailed, dynamic	derived,consolidated, integrated,historicized,partially aggregated,stable
Size	MB,GB	TB,PB
Availability	crucial	not so crucial
Architecture	3-tier (ANSI-SPARC)	adapted to data integration

Data Warehouse Applications

- Retail
- e-business
- Banks, finance
- Insurances
- Telecoms
- Logistics, Travels, Hotels
- Health
- (Life,...) Science
- Public Administrations

Data Warehouse Providers

Commercial:

ORACLE SAP SQL Server

IBM TERADATA snowflake

Open Source:

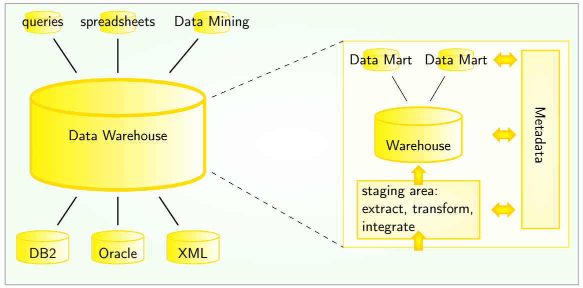
pentaho A Hitachi Data Systems Company talend CloverETL

600 M US $ acq. 06/2015

Data Warehouses: Architecture

Data Warehouse Architecture

Bird's eye view



Birds' view on DW architecture

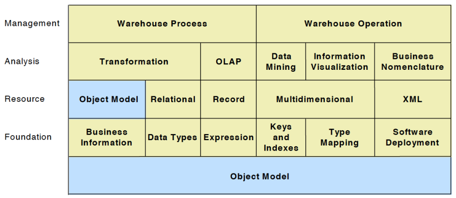
Metadata

"data that describes data"

Aims:

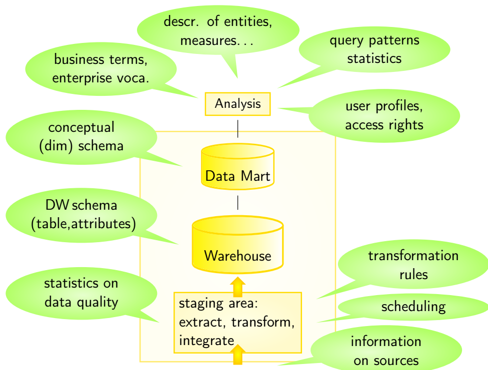
- monitor and understand processes
- avoid erroneous interpretations
- describe technical aspects of the DW

Multiple proprietary models, some standardization effort: Common Warehouse Metamodel from OMG. (other standards: IRDS, OIM...)



Metadata

Examples



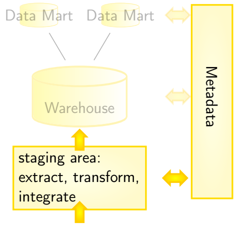
Data Staging: ETL

Extract-Transform-Load

Traditionally, data physically transported in two steps:

- from sources to staging area: extract the data, differential updates ⇒ minimize load on OLTP DB
- from staging area to base database: cleaning, filtering and integration ⇒achieve data quality (consistency…)

ETL considered most demanding in DW development (up to 80% of effort)!



Data Staging: ETL

Extract-Transform-Load

ETL

- *Extraction:* selecting relevant data from sources
- *Transformation:* adapting data to the target schema, meet quality requirements
- *Loading:* feeding the data from staging area into target database

Challenges for ETL:

- multiplicity of sources
- heterogeneity
- volume
- complex transformations
- recovery from failure

2 major alternatives for extraction:

- snapshot extraction (whole data)
- incremental extraction (requires source cooperation)

Data Staging: ETL

Data Heterogeneity - Schema Integration

Source 1

Produit P1

p_id	nom	prix	type
1	Matrix	9.99	Blue Ray
2	Der Pate	$12.45	DVD
3	Dirty Dancing	7.50	Video
...	...	...	

Vente

p_id→P1.p_id	panier_id	nombre	date
1	234	2	1/1/2010
2	234	1	1/1/2010
2	456	0	2/2/2010
3	456	1	3/3/2011
...	...	...	...

Source 2

Produit P2

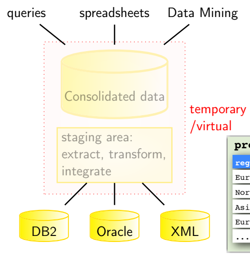
ident	description	dvd	video	blueray
P1	The Matrix	non	non	oui
P2	The Godfather	oui	non	non
P3	Moulin Rouge	non	non	oui
...	...	...		

Stock

sid	p_ident→P2.ident	stock	dernier_restockage
1	P1	50	1.4.10
1	P2	20	1.3.10
2	P2	-10	1.1.10
2	P3	30	1.2.10
...	...	...	...

Datamarts: Views over original data

Datamarts: specialised views over the initial database, created for a specific application domain



product_sales_per_region

region	product_name	product_class	income
Europe	The Matrix	Blue Ray	50,000
North America	The Matrix	Blue Ray	100,000
Asia	The Matrix	Blue Ray	90,000
Europe	The Godfather	DVD	150,000
...	...	...	...

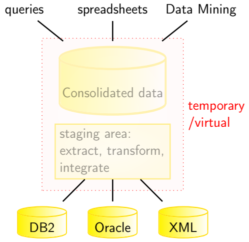
customer_class_per_product

customer_class	product_name	product_class	count
Class 1	The Matrix	Blue Ray	5000
Class 2	The Matrix	Blue Ray	2000
Class 1	Dirty Dancing	Video	1000
Class 2	Dirty Dancing	Video	6000
...	...	...	...

Datamarts: Views over original data

Materialised vs. Virtual

- Datamarts may be virtual or materialized
- Metadata is persistent
- Staging area typically temporary
- DW typically persistent except in virtual integration



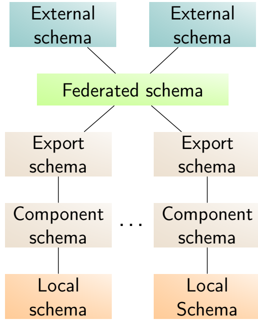
Datamarts: Views over original data

Materialised vs. Virtual

Solution to provide common access to autonomous sources.

- Logical schema of source is mapped into component schema (solves heterogeneity issues)
- Export schema is a portion of component schema (access control...)
- Federated schema integrates export schemas, is aware of data repartition. Might be large.
- External schema implements access control, simplifies federated schema, hides schema evolutions.

Metadata helps!

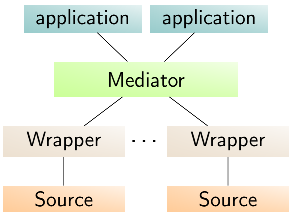


Federated: autonomous DB sharing their data (pairwise, minimal centralisation)

Mediator-Wrapper

Solution to provide common access to heterogeneous independent sources (not relying on metadata).

- wrapper is buffer level between sources and mediator; solves heterogeneity issues (interfaces, data model, semantics)
- mediator integrates the data *but mediator schema derived from applications and not from source integration*



Multidimensional Data Model

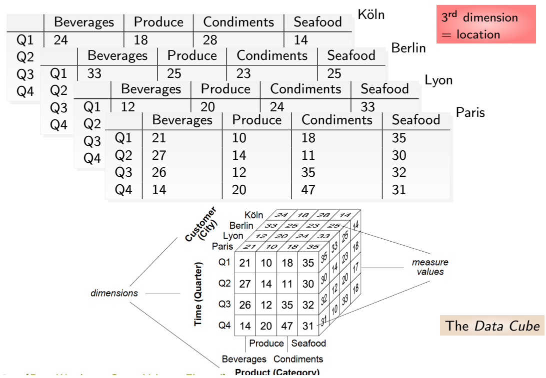
Why a Different Data Model?

- 3NF relational schema too complex for BI queries
- ... and suffers from slow query performance

Multi-dimensional model

- is easy to understand for Business users
- (OLAP exploration, generalizes spreadsheets...)
- delivers fast query performance
- schemas will not need to be reorganized too much over time

From Spreadsheet to Cube

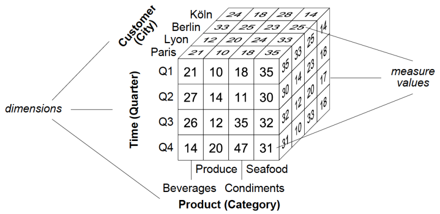


The Multidimensional Model

Data viewed as *n* dimensional cube (here n = 3).

Associated to the cube are:

- Dimension: perspective used to analyze the data
- Cell: the intersection of dimension values
- Fact: non-empty cell
- Measure: numeric value of the cells



The Multidimensional Model

Facts

- concept relevant for the analysis: represents an event (typically models a set of events taking place in the company)
- non-empty cell in the cube
- has a granularity = level of detail
- determined by the value of its dimension coordinates

Transaction fact

- Represent physical world events (at most detailed level)
- Exactly one fact per event
- Typically, events are independent and will occur at any time

Transaction fact example

- One fact for each sale of a wine bottle (detail)
- One fact for each day where at least one bottle sold (aggregation)
- sale fact tied to specific time/place

Snapshot fact

- Measures the current state of a process (possibly the result of a serie of events)
- Generally not independent
- Evaluated at specific interval/time

Snapshot fact example

- Inventory level per product,month,store
- a same product may appear in several inventories snapshots

The Multidimensional Model

Dimensions

- analysis perspective
- axis of the cube
- described by attributes

Dimensions can be used to define more than one cube.

Dimension attributes form a hierarchy of subsets (containment hierarchy): each level describes one degree of detail for the analysis on this dimension.

			Beverages	Produce	Condiments	Seafood
2010	Q1	Jan	4	5	0	6
		Feb	16	3	28	4
		Mar	4	10	0	8
	Q2	Apr	5	5	5	4
		May	12	5	5	10
		Jun	13	5	5	4
	Q3	Jul	20	10	15	7
		Aug	2	2	5	6
		Sept	3	3	10	4
	Q4	Oct	8	10	5	2
		Nov	3	10	20	8
		Dec	4	5	20	8
2011	Q1	Jan	10	25	40	10
				...		

The Multidimensional Model

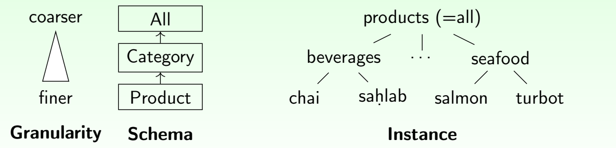
Dimensions

Dimension Schema

The classification schema of a dimension is a partially ordered set of category attributes: $(D.k_1, \dots, D.k_n, \mathrm{Top}_D, \rightarrow)$.

- $\rightarrow$ is the functional dependancy
- Top_D is the maximal attribute regarding $\rightarrow$: $\forall i, D.k_i \rightarrow \mathrm{Top}_D$
- there exists a (unique) minimal attribute: $\exists i, \forall j \neq i, D.k_i \rightarrow D.k_j$. $D.k_i$ describes the finest granularity of the dimension.

Classification schema and instance of Product dimension



The Multidimensional Model

Measures

- analysis perspective
- axis of the cube
- described by attributes

Dimensions can be used to define more than one cube.

Dimension attributes form a hierarchy of subsets (containment hierarchy): each level describes one degree of detail for the analysis on this dimension.

			Beverages	Produce	Condiments	Seafood
2010	Q1	Jan	4	5	0	6
		Feb	16	3	28	4
		Mar	4	10	0	8
	Q2	Apr	5	5	5	4
		May	12	5	5	10
		Jun	13	5	5	4
	Q3	Jul	20	10	15	7
		Aug	2	2	5	6
		Sept	3	3	10	4
	Q4	Oct	8	10	5	2
		Nov	3	10	20	8
		Dec	4	5	20	8
2011	Q1	Jan	10	25	40	10
				...		

The Multidimensional Model

Measures

- describes a fact
- consists of two functions:
  1. numeric property for each fact
  2. function to compute measure at coarser aggregation levels

Several measures can be associated to a fact.

A measure can depend on (other) measures on other facts.

Measure examples:

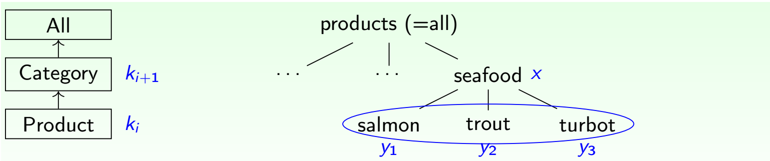
- number of units in stock
- value per unit
- quantity $\times$ price $\times$ turnover
- ...

The Multidimensional Model

Measures

Aggregation by levels:

- Standard aggregations: SUM, COUNT, AVG
- Can be additive, semi-additive, non-additive
- Can be distributive, algebraic, holistic



The Multidimensional Model

Measures

Cube

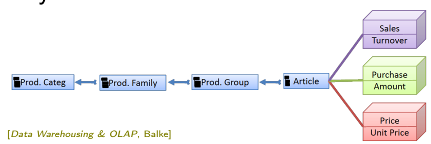
Cube Schema: set of n dimension schemas, and m measures $(\{D_1, \ldots, D_n\}, \{M_1, \ldots, M_m\})$.

Cube: instance of the cube schema= set of cells in $\mathrm{dom}(D_1) \times \cdots \times \mathrm{dom}(D_n) \times \mathrm{dom}(M_1) \times \cdots \times \mathrm{dom}(M_m)$

The cube contains *all* cells from its domain, not only the non-empty ones.

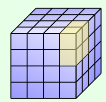
Can be observed at several granularities (determined by one level on each dimension)

Several cubes may share a dimension.



OLAP Operations

OLTP vs. OLAP Queries

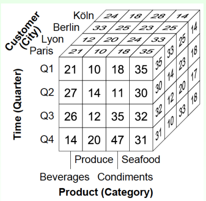
Operational DB	DW
transactions	analytical queries
INSERT, UPDATE, SELECT	SELECT, bulk insert
query deals with few tuples (often 1)	drill-down, slice. . .)
`UPDATE Customers` `SET City='Heidelberg'` `WHERE CustomerName='H. Plattner';`	

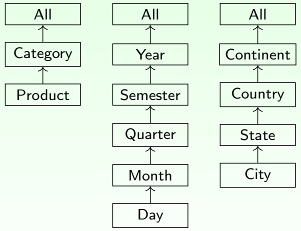
Multidimensional Model

Example cube

Consider a cube of 3 dimensions, one single measure (sales quantity)

Cube, and dimension schemas





OLAP Operations

Typical OLAP operations (cube navigation):

- Roll-up
- Drill-down
- Slice and Dice

Typical OLAP operations (rearranging the cube):

- Pivot
- Sort

Advanced OLAP operations:

- Drill-across
- Drill-through

Other common operations: aggregate functions, ranking functions...

Roll-up

Summarise data by navigating upward

Roll-up/Drill-up/Consolidate

Let C a cube with schema $(\{D_1, \ldots, D_n\}, \{M_1, \ldots, M_m\})$ at granularity $G = (l_1, \ldots, l_n)$, where $1 \leq l_i < m+1$ is the level in dimension D_i : $(D_i.k_1, \ldots, D_i.k_m, \mathrm{Top}_{D_i}, \rightarrow)$.
A Roll-up operation navigates toward a coarser granularity: some dimension $Dim^{up} \subset \{D_1, \ldots, D_n\}$ are rolled-up;
The new granularity is $G' = (l'_1, \ldots, l'_n)$.

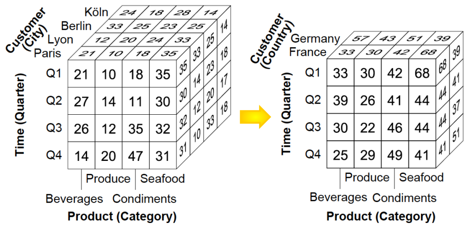
- $\forall D_i \in Dim^{up}, l_i < l'_i \leq m+1$
- $\forall D_i \notin Dim^{up}, l_i = l'_i$

This definition allows

- to zoom-out on multiple dimensions at once
- to roll-up a dimension to the TopD level

Roll-up

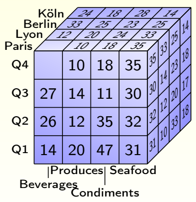
ROLLUP(Cube, Dimension->Level, Agg(Measure))



Roll-up to the Country level on Customer dimension:
ROLLUP(Sales, Customer→ Country, SUM(Quantity))

Roll-up

Draw the result of the operation.



ROLLUP(Sales, Time→ Semester, SUM(Quantity))

ROLLUP(Sales, Customer→ Country,Time→ Top_{Time}, SUM(Quantity))

Drill-down

Zoom-in on the data

Drill-Down

Let C a cube with schema $(\{D_1, \ldots, D_n\}, \{M_1, \ldots, M_m\})$ at granularity $G = (l_1, \ldots, l_n)$, where $1 \leq l_i < m + 1$ is the level in dimension D_i : $(D_i.k_1, \ldots, D_i.k_m, \text{Top}_{D_i}, \rightarrow)$.

A Drill-drown operation navigates toward a finerer granularity: some dimension $Dim^{down} \subset \{D_1, \ldots, D_n\}$ are drilled-down;

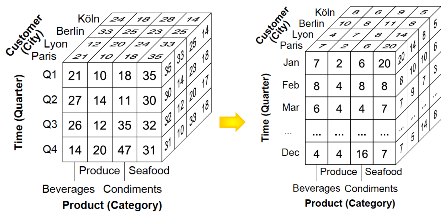
The new granularity is $G' = (l'_1, \ldots, l'_n)$.

- $\forall D_i \in Dim^{down}, 1 \leq l'_i < l_i$
- $\forall D_i \notin Dim^{down}, l_i = l'_i$

It's obvious but… you cannot drill down if you do not have the finer-grained data!

Drill-down

DRILLDOWN(Cube, Dimension->Level)



Drill-down to the Month level on Time dimension:
DRILLDOWN(Sales, Time→ Month)

Drill-down / Roll-up Summary

Navigate between granularities.

- Roll-up: fewer details, measure may be computed from input cube
- Drill-down: more details, measure computed from the finest-detailed data

…assuming measures have not (yet) been materialized at other granularities.

The number of dimensions remains the same (except when at top level of dimension).

Slide and Dice

Conditions on the cube dimensions

Slice and Dice definitions

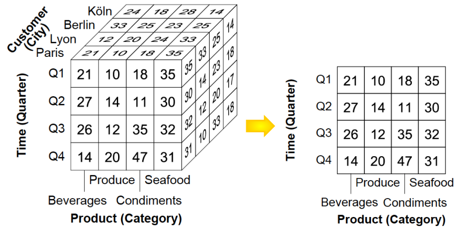
Slice: returns a "slice" of the cube by selecting a *single* value on *one* of the dimensions.

$\Longrightarrow$ corresponds to SQL's WHERE with equality selection.

Dice: returns a "dice" of the cube by selecting for each dimension a boolean combination of range or value conditions one one dimension *single* value.

Slice

SLICE(Cube, Dimension, Level=value)

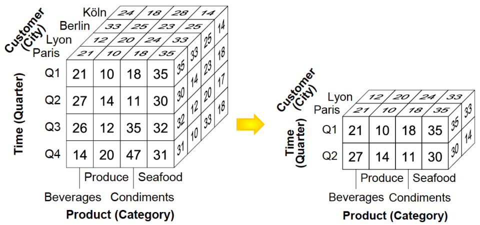


Slice on City='Paris':

SLICE(Sales, Customer,City='Paris')

Slice

DICE(Cube, *cond*)



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

DICE(Sales,(Customer.City='Paris' OR Customer.City='Lyon')

AND (Time.Quarter='Q1' OR Time.Quarter='Q2'))

Other Operations

Other typical operations on data:

- Sorting the cube
- Pivoting
- Joining cubes

No official standard for OLAP queries! (unlike SQL)

Cross-tabulations

Corsstab, pivot table, contingency table

Pivoting = selecting 2 dimensions to aggregate some measure

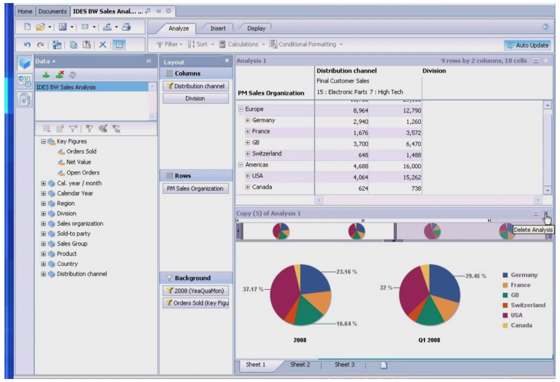
Can be computed in Excel, and using SQL operators in DBMS implementations.

Sales per city and day

	Mon	Tue	Wed	Thu	Fri	Sat	Row totals
Paris	10	20	30	40	30	10	140
Lyon	40	20	20	30	50	30	190
Lille	50	20	30	20	10	0	130
Col totals	100	60	80	90	90	40	460

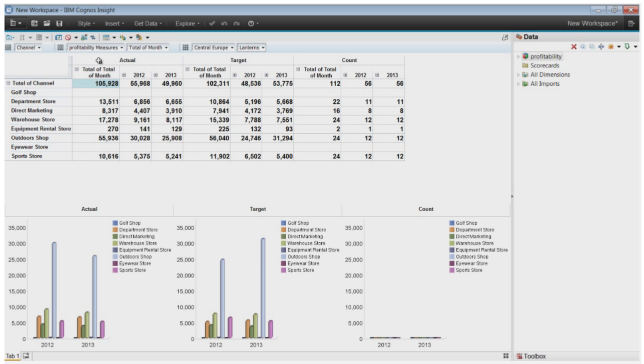
OLAP Visualisation in Practice

SAP BusinessObject



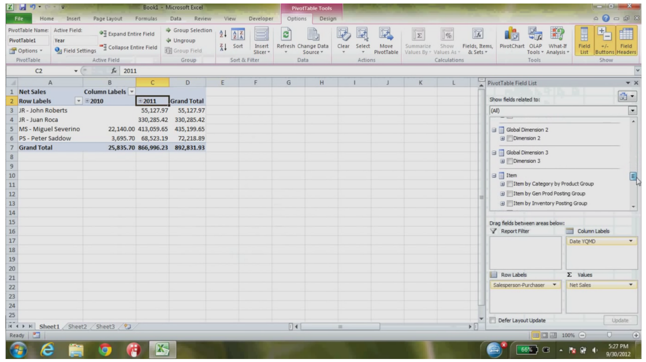
OLAP Visualisation in Practice

IBM Cognos



OLAP Visualisation in Practice

Excel



Implementing OLAP Cubes

Implementing the Cube

ROLAP: store the cube in a relational DBMS; extend SQL for implementing OLAP operations

MOLAP: store the cube in specific data structure (multi-dimensional array)

HOLAP: combine the 2 approaches

ROLAP

Advantages:

- Store only non-empty cells
- Dimensions are small tables; can store in-memory
- Takes advantage of relational optimisation

Issues:

- Translating into SQL can be costly

MOLAP

Advantages:

- No need to store cell coordinates
- Random access
- Fast aggregates
- Some analytical queries may not be available in SQL
- Allows fine-tuning of access control

Issues:

- No standard, vendor dependent
- Storing aggregates adds to cost
- Storage optimisation limited w.r.t. DBMS
- Scales worse with more dimensions

ROLAP

Objective: implementing the cube (dimensions, measures) into an DBMS

- Keep the semantic information (dimensions hierarchy)
- Maintenance should be easy
- Allow easy OLAP queries
- Allow DBMS functionalities: optimisation, concurrency, security

ROLAP
Properties

Stores data in relations

Separates structure and data:

- One relation for facts
- One relation for dimension attributes
- The fact relation stores measure and reference dimension table(s)

Only stores existing facts (no empty cells!)

ROLAP
Schemas

Achieved by different schema types:

- Star schema
- Snowflake schema
- Galaxy schema
- Starflake schema

Star Schema
Definition

A **star schema** is defined a by a fact table and a set of dimension tables:

- Each dimension $(D_i . k_1, \ldots, D_i . k_l, \text{Top}_D, \rightarrow)$ is a relation having schema $(\underline{\text{PK}}, k_1, \ldots, k_l)$
- The fact relation has schema $(\underline{D_1 . \text{PK}, \ldots, D_n . \text{PK}}, M_1, \ldots, M_m)$

Un attribute per level in the schema. The combination of all foreign keys is the key of the fact table. One attribute per measure.

Star Schema

Properties

Low redundancy due to 2NF dimension tables

Dimension tables contain few tuples compared to fact table => redundancy not costly

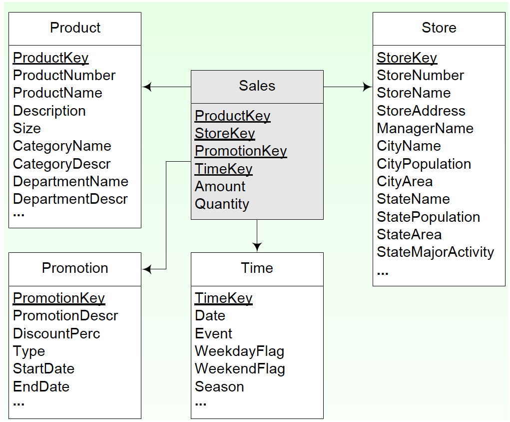
Fact table in 3NF

Generally, need to create artificial key for dimensions

Efficient querying

Star Schema

Example



Snowflake Schema

Definition

A **snowflake schema** is defined a by a fact table and a set of dimension tables:

- Each dimension $(D_i.k_1, \ldots, D_i.k_l, \mathrm{Top}_D, \rightarrow)$ is a set of relations $D_i^1, \ldots, D_i^l$ where D_i^j has schema $(\underline{\mathrm{PK}}, A_1, A_2, \ldots, D_i^{j+1}.\mathrm{PK})$
- The fact relation has schema $(\underline{D_1.\mathrm{PK}, \ldots, D_n.\mathrm{PK}}, M_1, \ldots, M_m)$

An set of attributes per level in the schema. The combination of all foreign keys of the lowest level is the key of the fact table. One attribute per measure.

Snowflake Schema

Properties

No redundancy due to full normalisation of dimension relations

Still (relatively) small size of dimension tables

n levels in a dimension hierarchy => *n* dimension tables

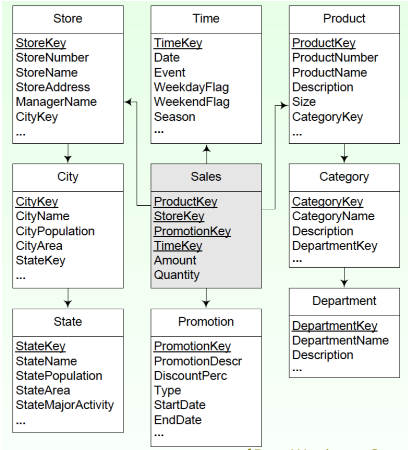
Fact table in 3NF

Fact tables references the finest granularity

More joins in queries, lower performance

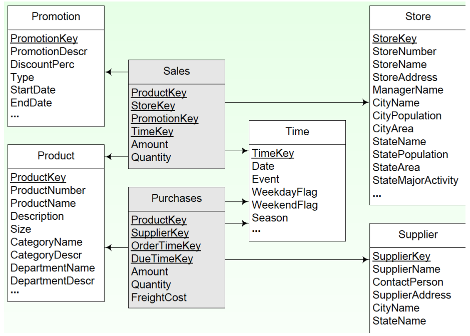
Snowflake Schema

Example



Galaxy Schema

Multiple fact tables = "fact constellation"



Starflake Schema

Combine snowflake and star dimension tables