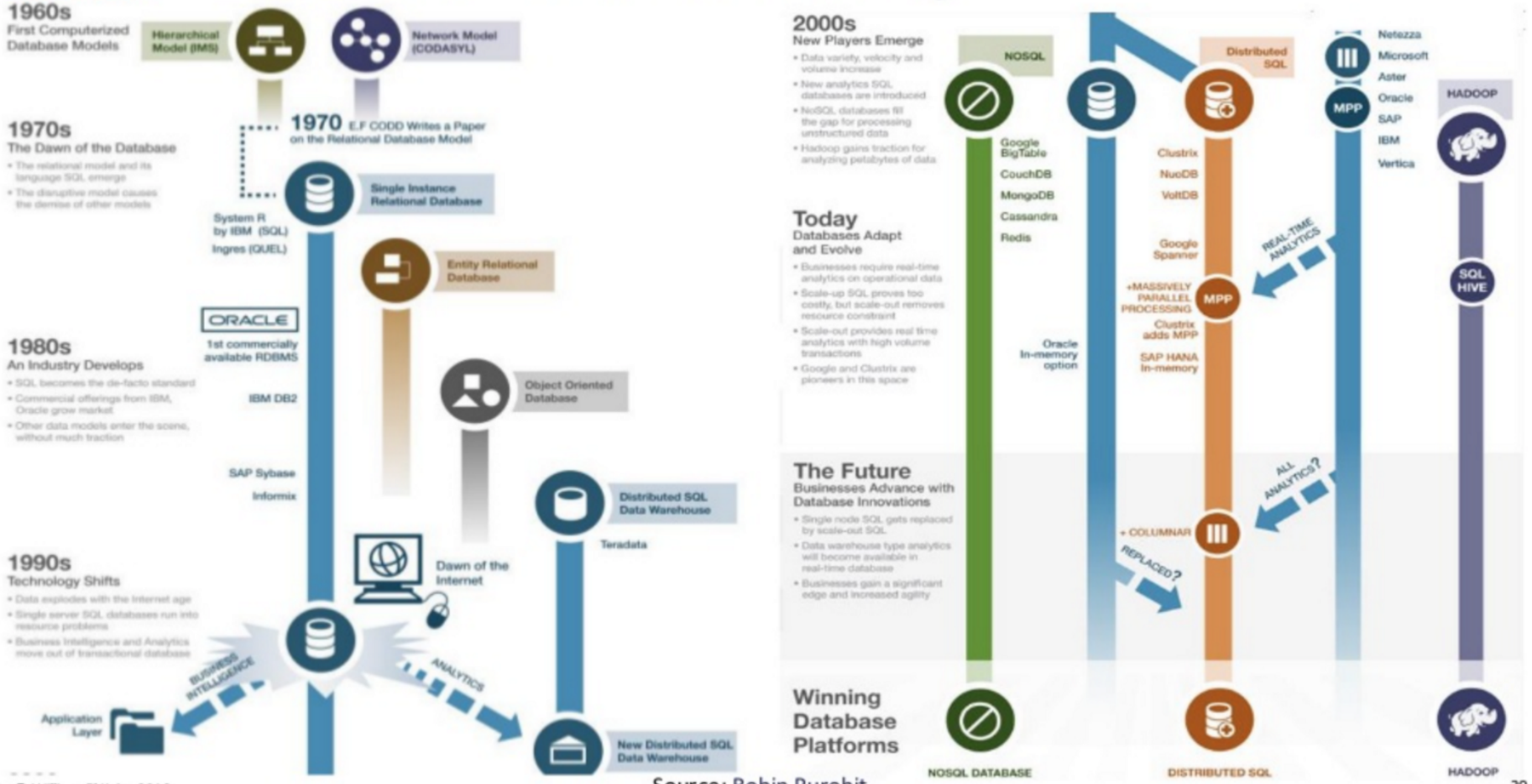
“One Size Fits All”: An Idea Whose Time Has Come and Gone

(M. Stonebraker)

Database Evolution History



Source: Robin Purohit

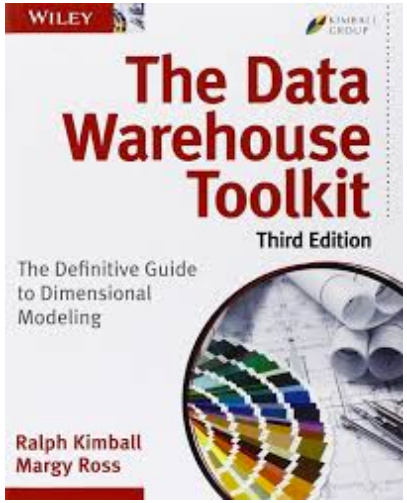
Today's Questions

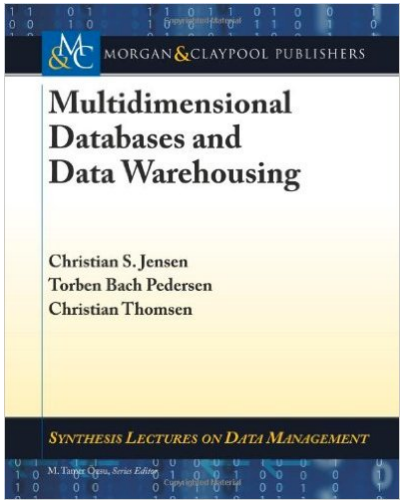
From Relational Databases to Data-Warehouses

- why RDB are insufficient for big-data analysis ?
- why do we need new models to *analyze* data ?
- which models do we need and how to conceive them ?

Resources

- Entrepôts de données, guide pratique de modélisation dimensionnelle. R.Kimball, M.Ross



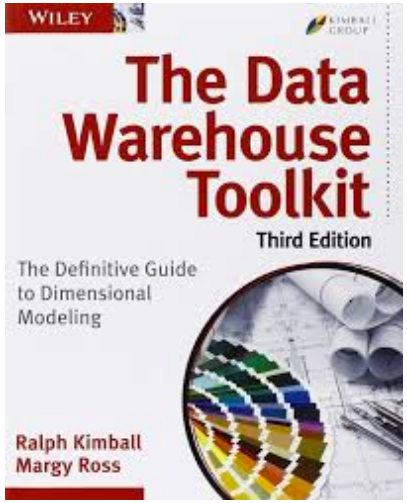


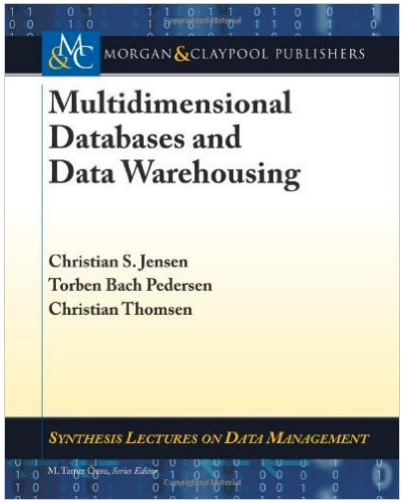
- Multidimensional databases and Data Warehousing C.S. Jensen, T.B.Pedersen and C.Thomsen

Resources

these slides cannot replace the textbooks by any means !

- Entrepôts de données, guide pratique de modélisation dimensionnelle. R.Kimball, M.Ross

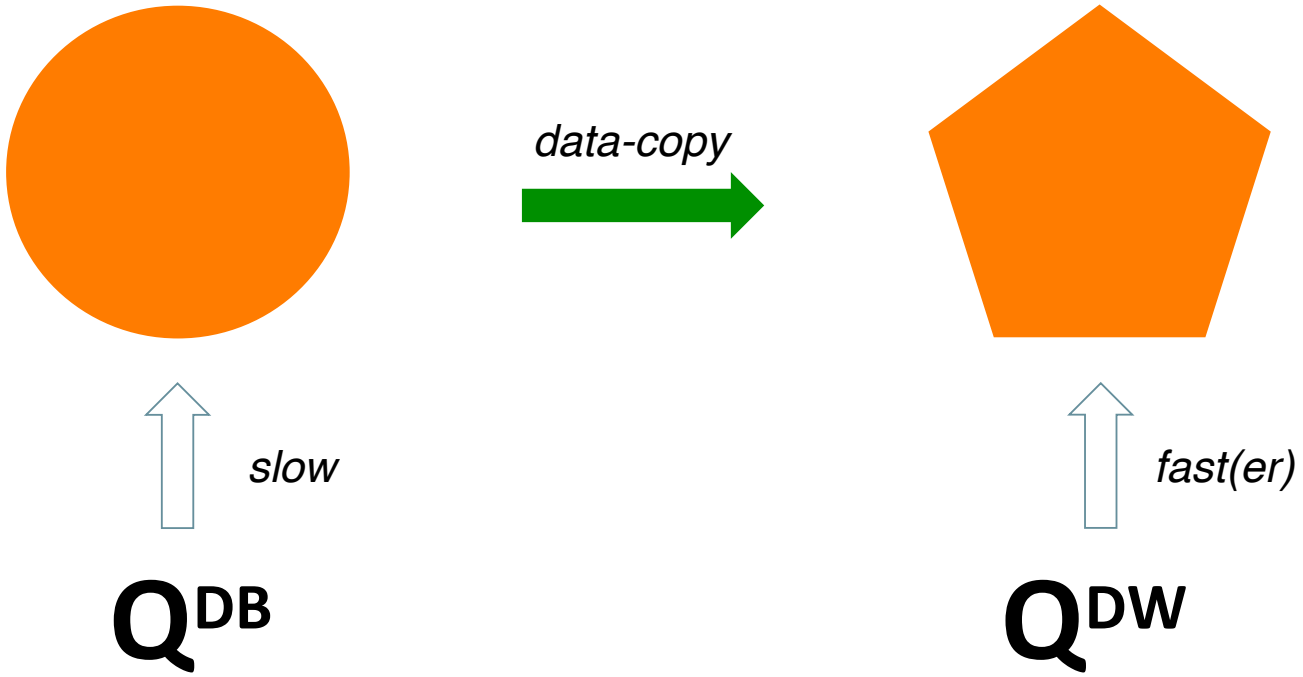




- Multidimensional databases and Data Warehousing C.S. Jensen, T.B.Pedersen and C.Thomsen

Datawarehouse

"a copy of transaction data specifically structured for analytical queries"



Why bothering with replicas of data ?

What are analytical queries and why are they different ?

Motivation Story

http://philip.greenspun.com/sql/data-warehousing.html

- '90 Bentonville (Arkansas)

- **Walmart**:''*I want to keep track of sales in all of my stores simultaneously.*"

- **Sybase** : "*You need our wonderful RDBMS software. You can stuff data in as sales are rung at cash registers and simultaneously query data right here in your office.*

  That's the beauty of concurrency control."

Motivation Story

http://philip.greenspun.com/sql/data-warehousing.html

- Walmart buys a $1M HP multi-CPU server and a $0.5M Sybase license, and builds a **normalized** database

- `Sales( product_id, store_id, quantity_sold, date_time_of_sale)`

- `Products( product_id, product_name, product_category, manufacturer_id)`

- `Stores(store_id, city_id, store_address, phone_num)`

- `Cities(city_id, city_name, state, population)`

Motivation Story

http://philip.greenspun.com/sql/data-warehousing.html

product

Products(product_id, product_name, product_category, manufacturer_id)

city

Cities(city_id, city_name, state, population)

store

Stores(store_id_city_id, store_address, phone_num)

sale

Sales(product_id, store_id, quantity_sold, date_time_of_sale)

Concurrency Control

sale

```
Sales( product_id,
store_id, quantity_sold,
   date_time_of_sale)
```

Concurrency Control

```
INSERT INTO Sales
(1,1,10,12345678)
```

WRITE

```
INSERT INTO Sales
(1,2,10,12345758)
```

WRITE

```
SELECT COUNT(*)
FROM Sales
```

READ

```
INSERT INTO Sales
(3,4,10,12345768)
```

WRITE

sale

```
Sales( product_id,
store_id, quantity_sold,
   date_time_of_sale)
```

Concurrency Control

```
INSERT INTO Sales
(1,1,10,12345678)
```

WRITE

```
INSERT INTO Sales
(1,2,10,12345758)
```

WRITE

```
SELECT COUNT(*)
FROM Sales
```

READ

```
INSERT INTO Sales
(3,4,10,12345768)
```

WRITE

can we **WRITE** while we **READ** on the same table?

sale

```
Sales( product_id,
store_id, quantity_sold,
   date_time_of_sale)
```

Concurrency Control

```
INSERT INTO Sales
(1,1,10,12345678)
```

WRITE

```
INSERT INTO Sales
(1,2,10,12345758)
```

WRITE

```
SELECT COUNT(*)
FROM Sales
```

READ

```
INSERT INTO Sales
(3,4,10,12345768)
```

WRITE

NO! **COUNT**(*) may become inconsistent : table locking

sale

```
Sales( product_id,
store_id, quantity_sold,
   date_time_of_sale)
```

Motivation Story

- Some time after…

- Walmart executive asks: *"I noticed that there was a Colgate promotion recently, directed at people who live in small towns.*
  How much Colgate toothpaste did we sell in those towns ***yesterday****?*
  And how much on ***the same day a month ago****?"*

- **Sybase** : *"Let me write the query!"*

Motivation Story

```
SELECT SUM(sales.quantity_sold)
FROM sales, products, stores, cities
WHERE products.manufacturer_id = 68  -- Colgate_id
AND products.product_category = 'toothpaste'
AND cities.population < 40000
AND sales.datetime_of_sale::date =
                                'yesterday'::date
AND sales.product_id = products.product_id
AND sales.store_id = stores.store_id
AND stores.city_id = cities.city_id
```

Motivation Story

http://philip.greenspun.com/sql/data-warehousing.html

product

Products(product_id, product_name, product_category, manufacturer_id)

city

Cities(city_id, city_name, state, population)

store

Stores(store_id_city_id, store_address, phone_num)

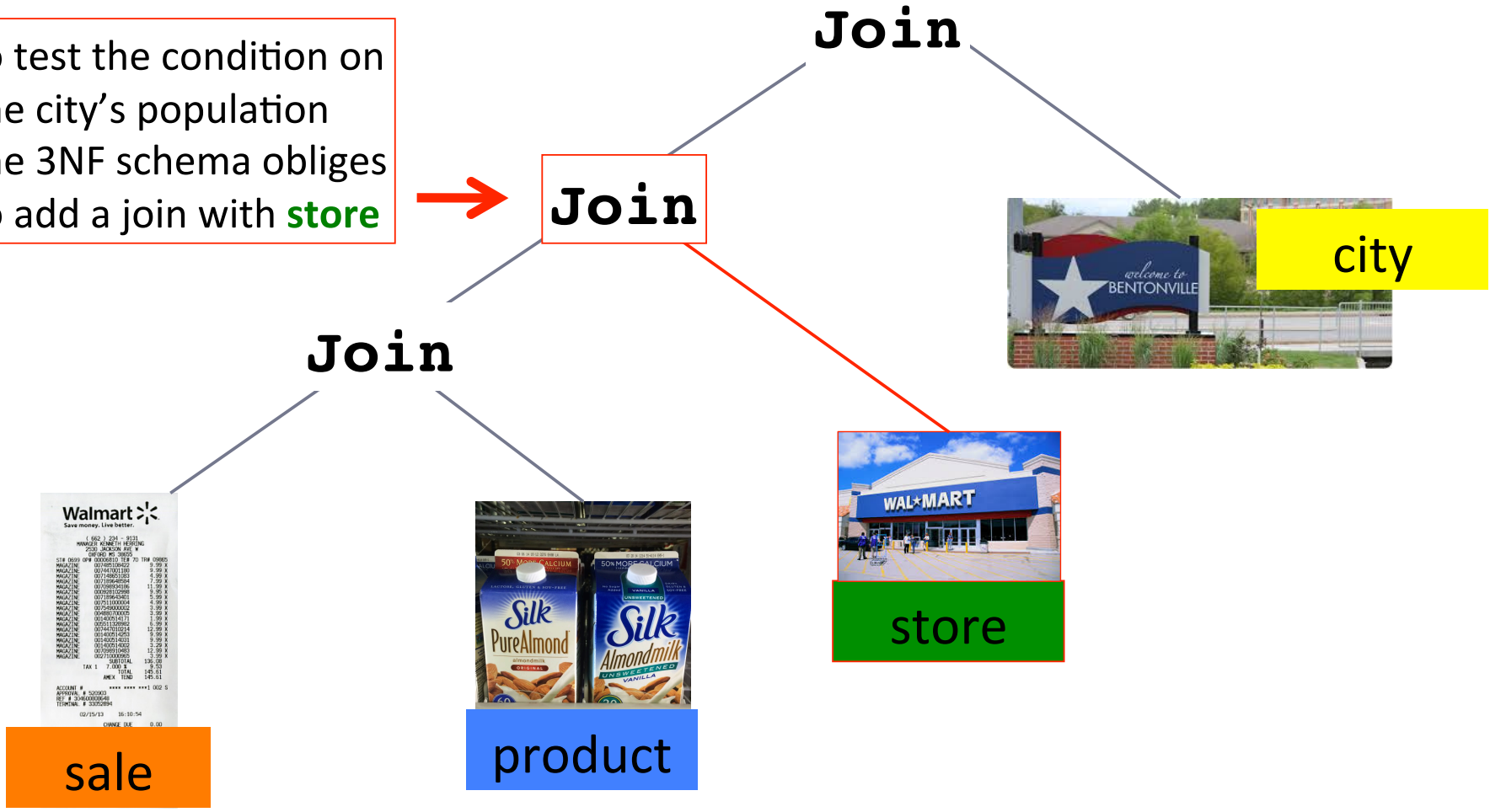
sale

Sales(product_id, store_id, quantity_sold, date_time_of_sale)

Motivation Story

http://philip.greenspun.com/sql/data-warehousing.html

to test the condition on the city's population the 3NF schema obliges to add a join with **store**



Motivation Story

```
INSERT INTO Sales
(1,1,10,12345678)
```

WRITE

```
INSERT INTO Sales
(1,2,10,12345758)
```

WRITE

```
SELECT sum(sales.quantity_sold)
FROM sales, products, stores, cities
WHERE products.manufacturer_id = 68  -- Colgate_id
AND products.product_category = 'toothpaste'
AND cities.population < 40000
AND sales.datetime_of_sale::date = 'yesterday'::date
AND sales.product_id = products.product_id
AND sales.store_id = stores.store_id
AND stores.city_id = cities.city_id
```

READ

Motivation Story

```
INSERT INTO Sales
(1,1,10,12345678)
```

WRITE

```
INSERT INTO Sales
(1,2,10,12345758)
```

WRITE

lock on 4 tables :
sales, **products**, **stores**, **cities**

```
SELECT sum(sales.quantity_sold)
FROM sales, products, stores, cities
WHERE products.manufacturer_id = 68  -- Colgate_id
AND products.product_category = 'toothpaste'
AND cities.population < 40000
AND sales.datetime_of_sale::date = 'yesterday'::date
AND sales.product_id = products.product_id
AND sales.store_id = stores.store_id
AND stores.city_id = cities.city_id
```

READ

Motivation Story

- What can happen when you run joins on big-data : the query returns after **20**mins.

- Cash **registers over the country cannot process the sales** when the toothpaste query is run.

Motivation Story

- **Walmart**: *"We type in the toothpaste query and our system wedges."*

- **Sybase**: *"Of course it does!*
  You built an on-line transaction processing system.
  You can't ask analytic queries & expect things to work!"

- **Walmart**: *"But I thought the whole point of SQL and your RDBMS was that users could query and insert simultaneously."*

Motivation Story

- **Sybase**: *"Uh, not exactly. The system prevents simultaneous Writes and Reads to guarantee coherent information: this is called "pessimistic locking".*

- **Walmart**: *"Can you fix your system so that it doesn't lock up?"*

- **Sybase**: *" No.*
  But we made a great loader tool to copy everything from your transactional system into a separate decision support system at 100 GB/hour."

Analytical Query

Basically, a query which needs to access a

very large portion

of a database

for the sake of data analysis

(ex. compare sales per region on July)

Note : analytical query are also possible outside DW but as the example shows, they can just be inefficient

Main SQL construct : Group By (ex. Monoprix)

date	product	store	city	amount
1/12/2018	P1	Gare	Montpellier	2000
1/12/2018	P1	Antigone	Montpellier	3400
1/12/2018	P2	Port Marianne	Montpellier	1280
		.....		

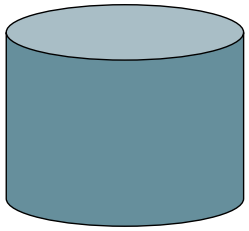
```
SELECT    date, store, SUM(amount)
FROM      ventes
GROUP BY  date, store
```

Business Intelligence

(Big-Data before 2010)

Make strategic fact-based decisions

Aggregate Data



Database, Data Mart, Data Warehouse, ETL Tools, Integration Tools

Present Data

Reporting Tools, Dashboards, Static Reports, Mobile Reporting, OLAP Cubes

Enrich Data

Add Context to Create Information, Descriptive Statistics, Benchmarks, Variance to Plan or LY

Inform a Decision

Decisions are Fact-based and Data-driven

Transactional Vs **A**nalytical Systems

Aspect	Operational DB	DW
User	clerk	manager
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)
Concurrency	huge (thousands)	limited (hundreds)
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

cost of redundancy

Applications of Datawarehouses

Domains

- Retail
- E-business
- Banks
- Telecoms
- Logistics
- Travels
- Hotels Insurances
- Health
- (Life,...)
- Science
- Public Administration
- …

Retail (Walmart)

[Data Warehousing: Using the Wal-Mart Model,Westman]

- Pioneer: one of the largest (retail) warehouses since. Teradata solution.

	92	2001	2004	2008
size	1 TB	70 TB	>500 TB	2.5 PB

- $20M Prototype DW to analyze sales launched in '90
  - **investment paid back in 6 months**
- A study over analysts estimated ROI per query at $12K

Retail (Casino)

Sources: http://www.mycustomer.com/topic/technology/casino-group-upgrades-teradata-data-warehouse http://fr.teradata.com/newsrelease.aspx?id=12338
http://www.lemagit.fr/actualites/2240197570/
Pour- harmoniser- ses- calculs- de- marge- Casino- sengage- dans- la- refonte- de- son- decisionnel

- One of the earliest DW in France. Teradata solution

	94	2002	2009
size	80 GB	10 TB	18 TB
users	50	1,500	3500
queries/day		25,000	600,000

- Saved millions when realized that Coca-Cola stocks were often low :
  - order more ! (better price (negotiate) / logistics)
  - sell more ! (reduce missed sales)

Retail (Casino)

Goals:

- improve sales
- product offerings
- optimize supply chain
- optimize promotions
- redesign store layouts
- customer retention

fidelity-cards are good or bad ???

Main challenge : Making a datawarehouse is not science, is an art.

Datawarehouse Making

"Art" = combination of a set of best-practices

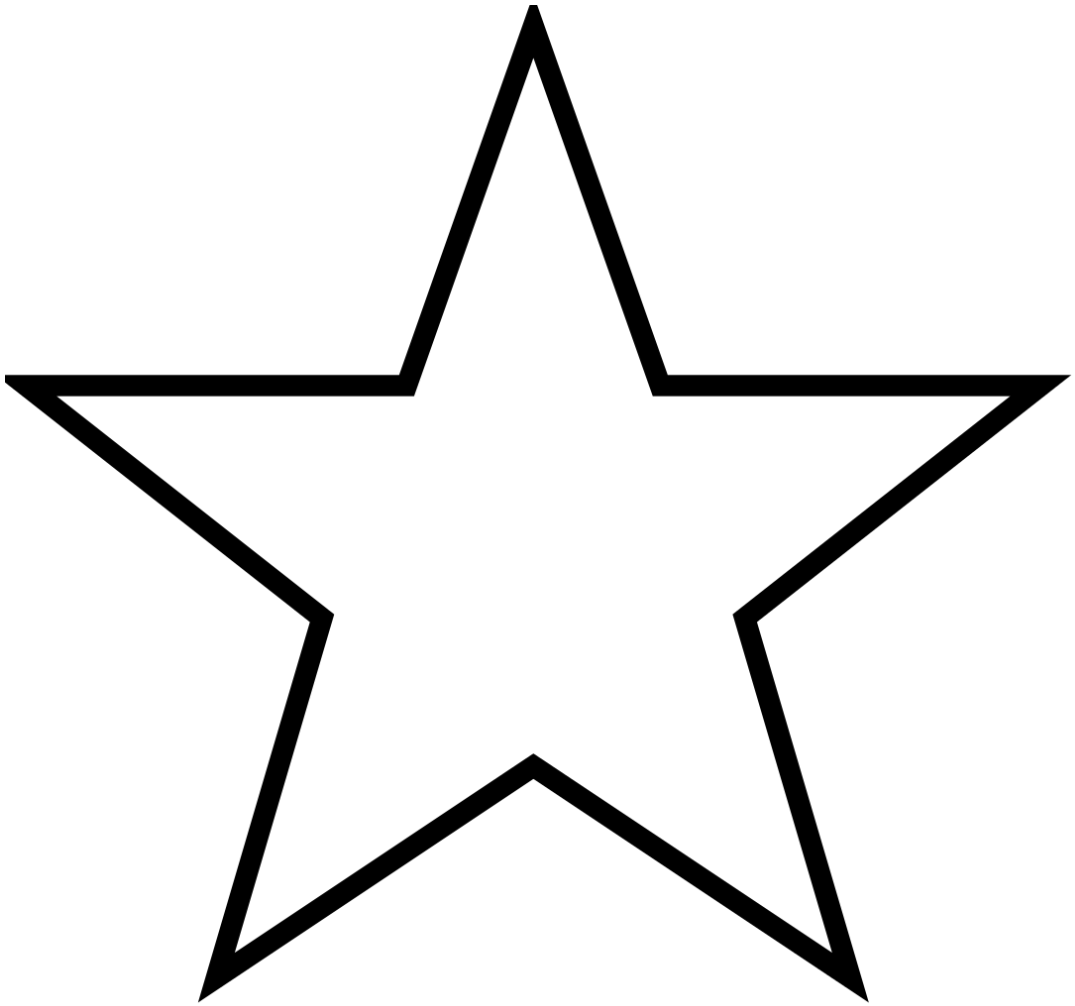
- Totally driven by the "industrial" case considered
- Neglects sensible parts of rel. theory : eg Normal Forms
- User-driven : make analytic **user-**queries easy

Problem #1 : optimizing joins on large tables

- Datawarehouses employ redundant relational schemas

(that violate 3NF, for example) in order to ease joins !

Relational Models for DW



Star Schemas

product

store

sale

city

date

Star Schemas

product

store

sale

city

date

Basic terminology : Fact

Fact = recording of an event occuring in the real world
(the center of the star)

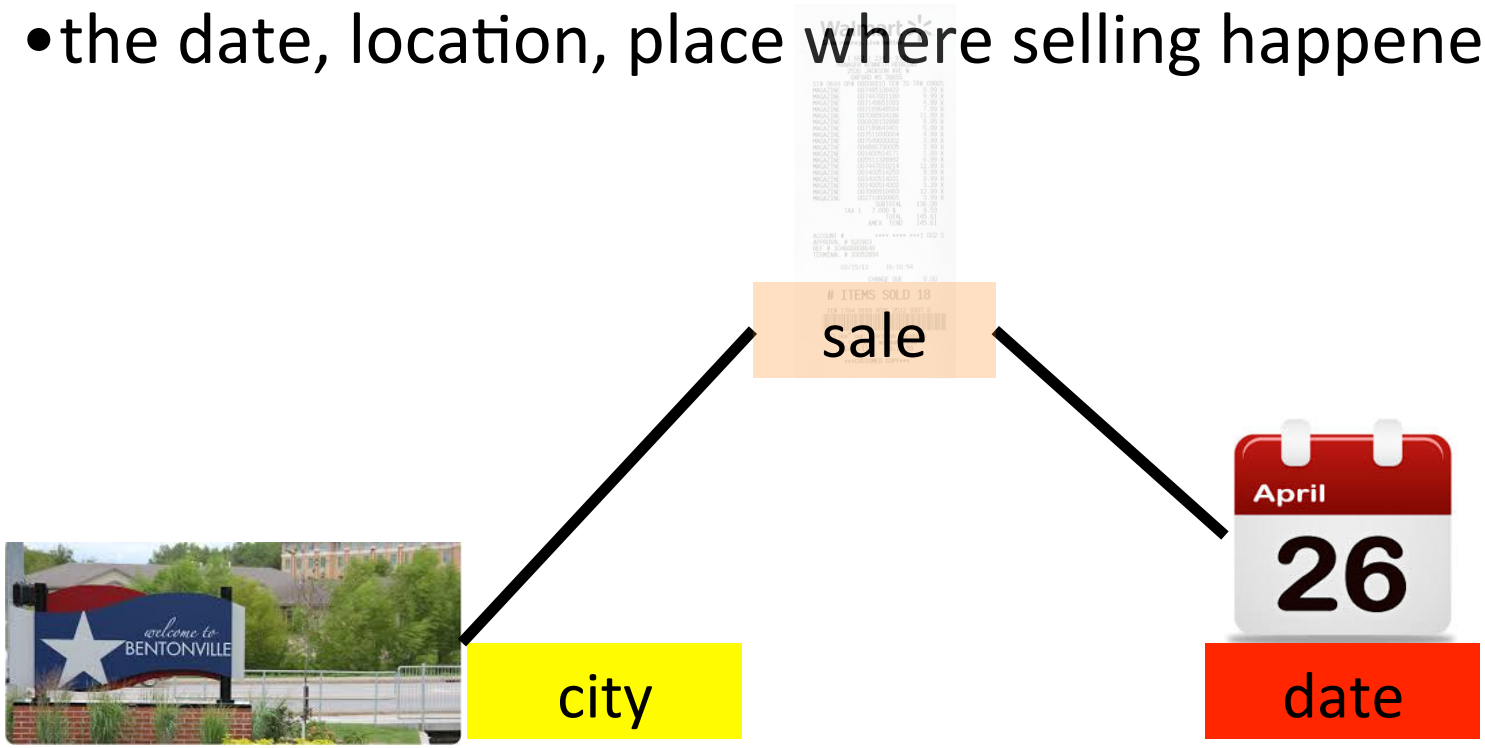
- I sell an item

sale

Basic terminology : Dimension

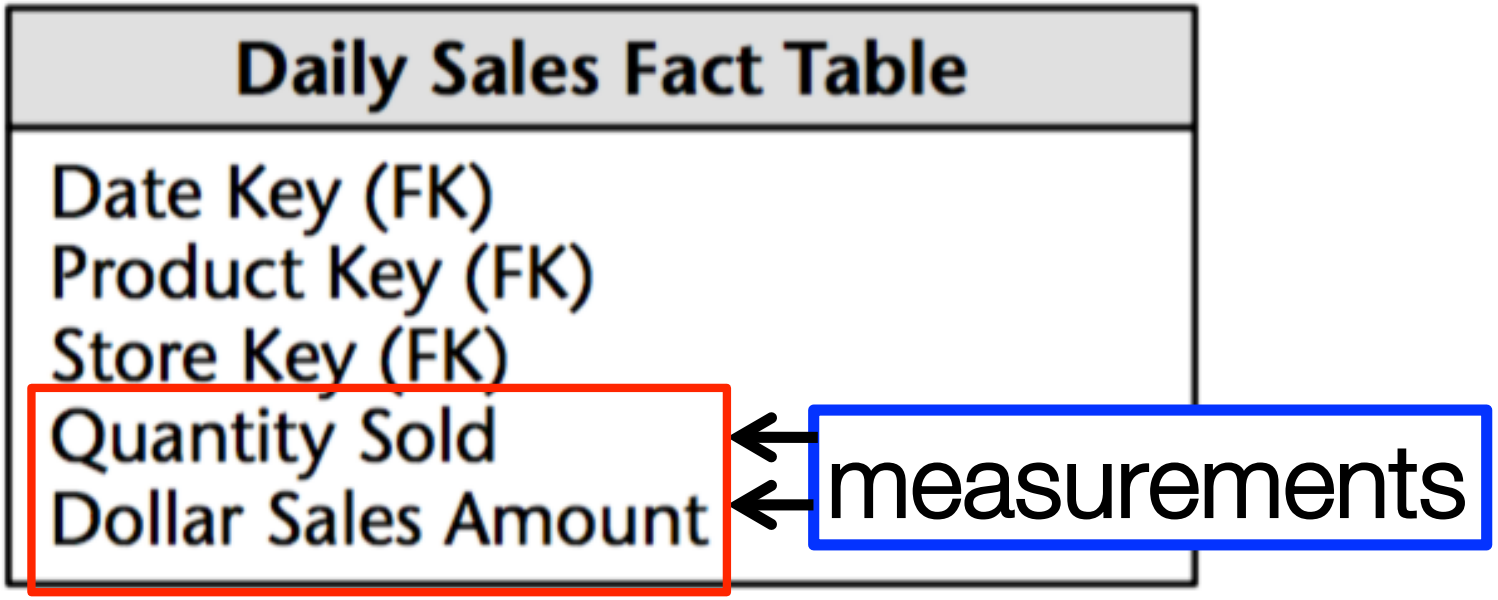
Dimension = a set of attribures describing the fact
(a vertex of the star)

- the date, location, place where selling happened



Fact Table

- Principal table in a datawarehouse
- Stores performance **measurements** of the business



Fact Table

- Principal table in a datawarehouse
- Stores performance **measurements** of the business

date	product	store	quantity	amount($)
1	3	1	11	45
1	21	2	65	1200
1	47	3	2332	15000
1	710	4	53	75

Numeric and Additive Facts

- most useful **measurements** are **numeric** and **additive**:
- **Additivity is crucial**

date	product	store	quantity	amount($)
1	3	1	11	45
1	21	2	65	1200
1	47	3	2332	15000
1	710	4	53	75

- **Text facts are rare**

measurements

Numeric and Additive Facts

- most useful **measurements** are **numeric** and **additive**:

- **Additivity is crucial** because if data warehouse retrieves 10^6 of fact at a time, most useful thing is *add them up*

date	product	store	quantity	amount($)
1	3	1	11	45
1	21	2	65	1200
1	47	3	2332	15000
1	710	4	53	75

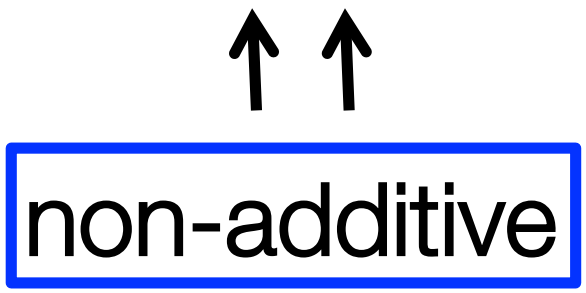
- **Text facts are rare**
  Unpredictable content
  makes it nearly impossible to analyze.

measurements

Non-Additive Facts

- **%** (gross margin), **ratios** (movie rating) are nonadditive

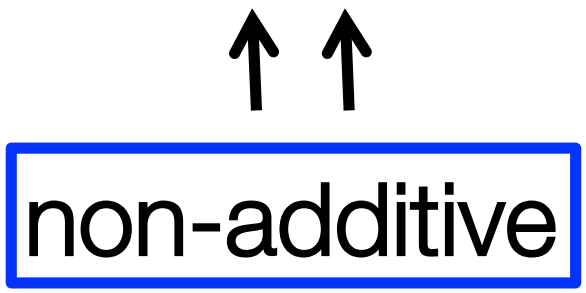
date	product	store	quantity	amount($)	margin(%)
1	3	1	11	45	15
1	21	2	65	1200	35
1	47	3	2332	15000	20
1	710	4	53	75	50



Non-Additive Facts

- **%** (gross margin), **ratios** (movie rating) are nonadditive

date	heure	film	note (num)	note (den)
1	11	1	1	5
1	23	1	4	5
1	47	3	5	5
1	70	4	5	5



ratios best practice for DW:
numerator & denominator
separately in fact table
ratio calculated on the fly
by the query

Fact Table

Fact tables usually make up **90% of the DW**
(can be billions of lines)

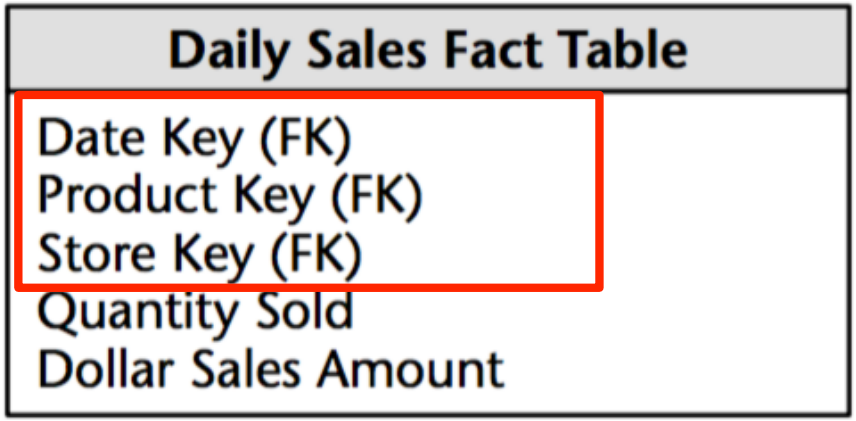
- **Deep** in the #rows (facts)
- **Narrow** in the #columns (dimensions)

NO sales activity (day, store, product) = NO rows

- Zero(s) saying nothing happening = space waste

Referential Integrity

- **Fact tables foreign keys** connect to the **dimension** tables primary keys (this makes the "star")



Referential Integrity

- **Fact tables foreign keys** connect to the **dimension** tables primary keys (this makes the "star")

date	product	store	quantity	amount($)
1	3	1	11	45
1	21	2	65	1200
1	47	3	2332	15000
1	710	4	53	75

- Fact table => has ***composite key*** made of foreign keys (a priori, no need for ROWID)

Fact-row Uniqueness

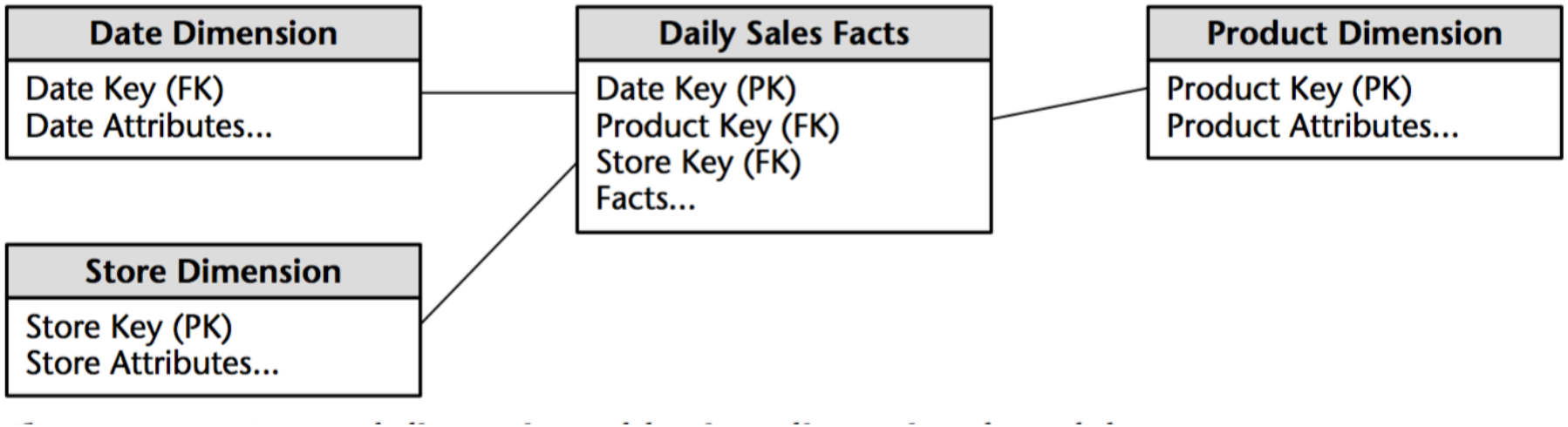
- A subset of the components in the fact table composite key typically needed to guarantee row uniqueness

Example invoice number and the product key, do not require the date/time dimension for row uniqueness

- Consequence: no advantage to have a unique ROWID key to serve as the primary key in the fact table.

- Except if allowing multiple identical rows in fact table

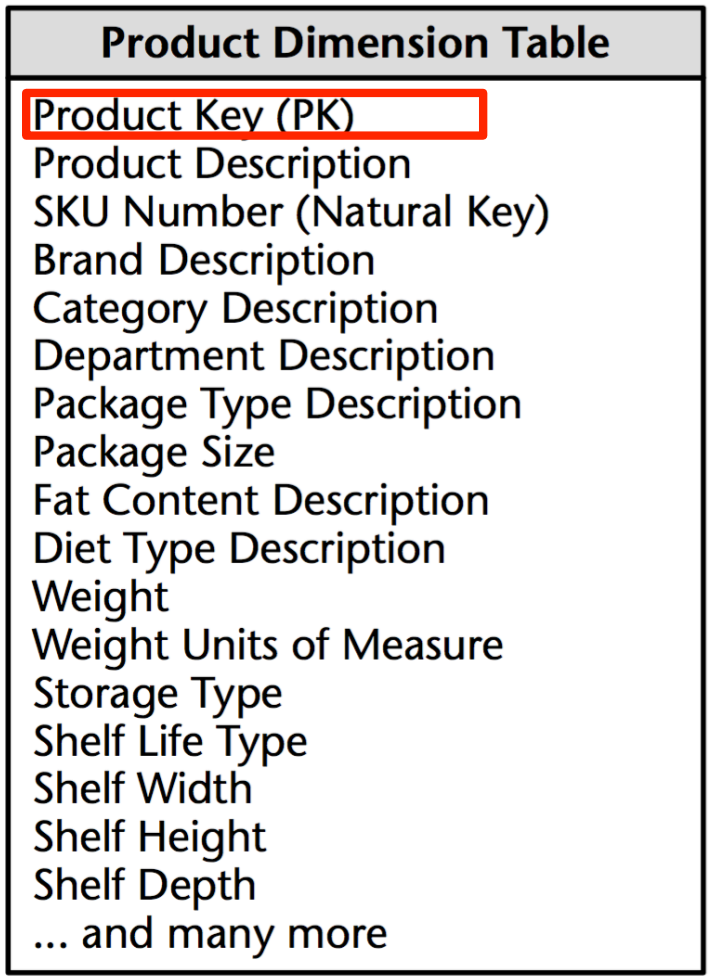
Marrying Facts and Dimensions (into star-schemas)



A list of dimensions defines the *grain* of the fact table.
All measurements in a fact table are at the same grain.

Dimension Table

- Has a single-attribute PK
- Contain as many textual descriptors of the business as possible (**50-100 : OK**)
- Usually shallow in terms of the number of row,
  has many large columns



Textual & Discrete Dimension = 

- No cryptic abbreviations
  Include a short description (10 to 15 characters),
  a long description (30 to 50 characters),
  a brand name,
  a category name,
  a packaging type,
  size (behaving like a a discrete and constant descriptor)

Product Key	Product Description	Brand Description	Category Description	Department Description	Fat C
1	Baked Well Light Sourdough Fresh Bread	Baked Well	Bread	Bakery	Redu
2	Fluffy Sliced Whole Wheat	Fluffy	Bread	Bakery	Regu
3	Fluffy Light Sliced Whole Wheat	Fluffy	Bread	Bakery	Redu

Dimension Table

- Entry points into the fact table
- Example : first I choose a a brand (eg. Colgate), then I look for sellings of corresponding items

Product Dimension Table
Product Key (PK)
Product Description
SKU Number (Natural Key)
Brand Description
Category Description
Department Description
Package Type Description
Package Size
Fat Content Description
Diet Type Description
Weight
Weight Units of Measure
Storage Type
Shelf Life Type
Shelf Width
Shelf Height
Shelf Depth
... and many more

Dimension Table

10% of the DW **but, data analysis power** directly proportional to quality and depth of the dimension attributes

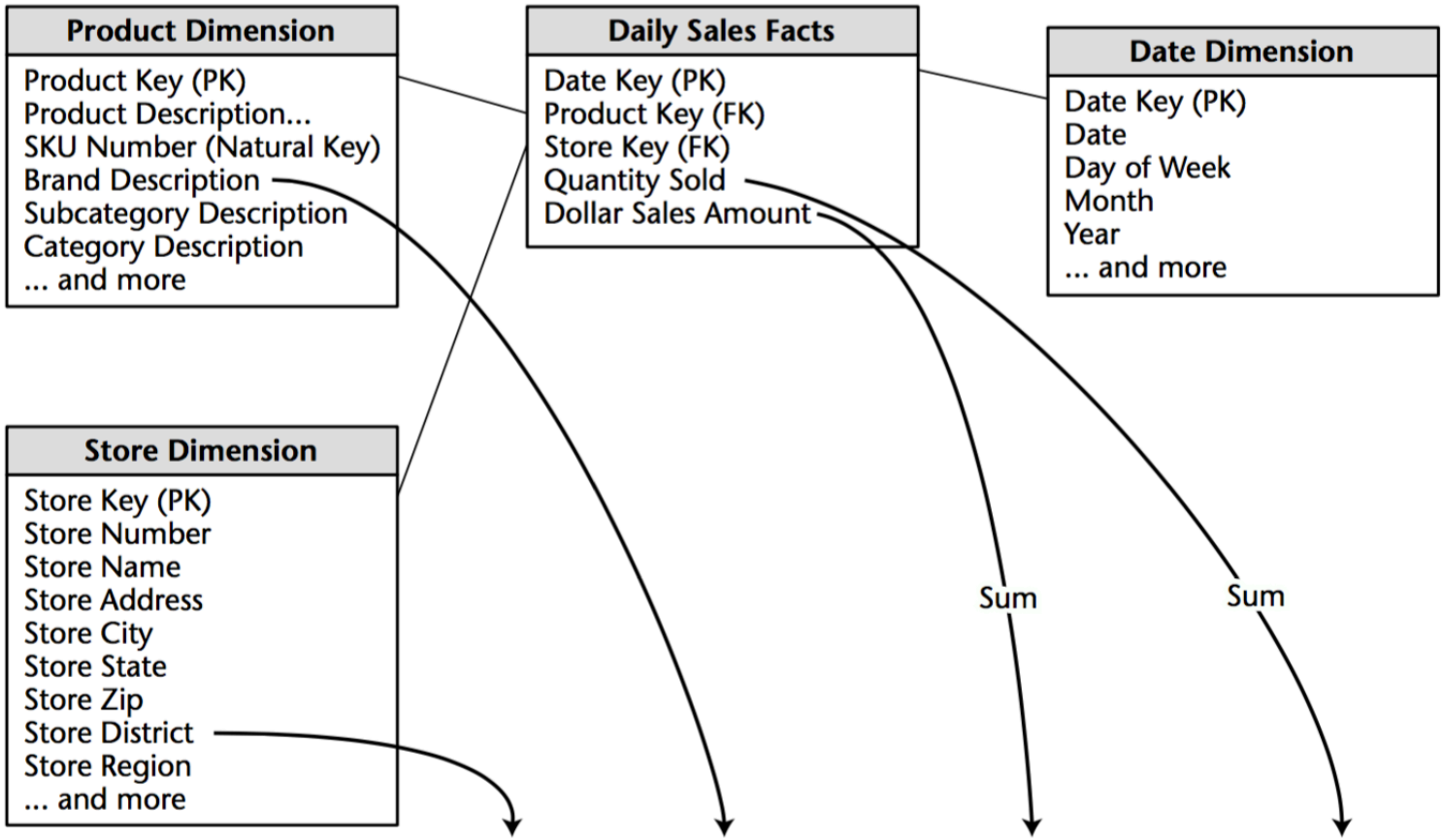
Compared to a **fact table** is :

Large(r) in the #colums (attributes)

Shallow(er) in the #rows (possible values) (<< 10^6 rows)

SQL ANALYTICAL QUERIES

Ultimate Goal : Analytical Reports



District	Brand	Dollar Sales Amount	Quantity Sold
Atherton	Clean Fast	1,233	1,370
Atherton	More Power	2,239	2,035
Atherton	Zippy	848	707
Belmont	Clean Fast	2,097	2,330
Belmont	More Power	2,428	2,207
Belmont	Zippy	633	527

New Relational Operators

- `ROLLUP` operator

- `CUBE` operator

Group By

departement	employee	manager	salaire
D1	E1	M1	2000
D2	E2	M2	3400
D1	E3	M3	1280

```
SELECT    departement, AVG(salaire)
FROM      employees
GROUP BY  departement
```

Group By

departement	AVG(salaire)
D1	1640
D2	3400

```
SELECT    departement, AVG(salaire)
FROM      employees
GROUP BY  departement
```

ROLLUP

- `ROLLUP` is an extension to the `GROUP BY` clause.

- `ROLLUP` produces cumulative aggregates

Group By

departement	employee	manager	salaire
D1	E1	M1	2000
D2	E2	M2	3400
D1	E3	M3	1280

```
SELECT   departement, manager, AVG(salaire)
FROM     employees
GROUP BY departement,manager
```

ROLLUP

departement	**manager**	**AVG(salaire)**
D1	M1	2000
D1	M3	1280
D2	M2	3400

```
SELECT   departement, manager, AVG(salaire)
FROM     employees
GROUP BY departement,manager
```

ROLLUP

departement	**manager**	**AVG(salaire)**
D1	M1	2000
D1	M3	1280
D1		1640
D2	M2	3400
D2		3400
		2222.6

```
SELECT   departement, manager, AVG(salaire)
FROM     employees
GROUP BY   ROLLUP(departement,manager)
```

ROLLUP

```
SELECT    departement, manager, AVG(salaire)
FROM      employees
GROUP BY  departement, manager
                       ***
SELECT    departement, AVG(salaire)
FROM      employees
GROUP BY  departement
                       ***
SELECT    AVG(salaire) FROM     employees
```

~

```
SELECT    departement, manager, AVG(salaire)
FROM      employees
GROUP BY    ROLLUP(departement,manager)
```

CUBE

- `CUBE` is an extension to the `GROUP BY` clause.

- `CUBE` takes aggregation even further

```
SELECT   departement, manager, AVG(salaire)
FROM     employees
GROUP BY   CUBE(departement,manager)
```

CUBE

departement	manager	AVG(salaire)
		2226.6
	M1	2000
	M2	3400
	M3	1280
D1		1640
D1	M1	2000
D1	M3	1280
D2		3400
D2	M2	3400

```
SELECT   departement, manager, AVG(salaire)
FROM     employees
GROUP BY   CUBE(departement,manager)
```

CUBE

departement	manager	AVG(salaire)
		2226.6
	M1	2000
	M2	3400
	M3	1280
D1		1640
D1	M1	2000
D1	M3	1280
D2		3400
D2	M2	3400

```
SELECT   departement, manager, AVG(salaire)
FROM     employees
GROUP BY   CUBE(departement,manager)
```

CUBE

```
SELECT    departement, manager, AVG(salaire)
FROM      employees
GROUP BY  departement,manager
                              ***
SELECT    manager, AVG(salaire)
FROM      employees
GROUP BY  manager
                              ***
SELECT    departement, AVG(salaire)
FROM      employees
GROUP BY  departement
                              ***
SELECT    AVG(salaire) FROM     employees
```

~

```
SELECT   departement, manager, AVG(salaire)
FROM     employees
GROUP BY   CUBE(departement,manager)
```

Summing up

- DW : **dedicated** systems for running analytical queries
  (heavy GROUP BY)
- Modelization : star-schemas
  (reduce joins; but don't use universal relations)
- Facts (hold numeric, mostly additive measurements)
- Dimensions (hold rich textual attribute information)