14 Data Warehouses in a nutshell

Kemper / Eickler: chap.17.2 , Bernstein, Kifer et. al.: chap 17
Melton: chap. 12,

14.1 Introduction OLTP versus OLAP

Freie Universität Berlin

10^n users per day

Online Transaction processing (OLTP)

Online Analytical processing (OLAP)

Response Time of shopping transaction? Schema extension for new products, backup?

How many English books in Germany? Revenue for electronics per country?

The naive Bookshop DB

Freie Universität Berlin

How many books in English books sold during Christmas season in Southern Europe, Western Europe, Eastern Europe?

```
Order_item
Book_id
amount
Order_id
```

```
Books
Book_id
price
title
language
genre
```

```
Orders
Order_id
Date
Cust_id
Total_amount
```

```
Customer
Cust_id
adress
country
```

SELECT Count(*), country
FROM orders o JOIN
order_item oi
JOIN books b
JOIN customer c
WHERE o.date...
AND b.language = ..
Group by country....

Suited for Online shop, less suited for analytical queries

Analytical queries on OLTP DB

Freie Universität Berlin

"Difficult" queries

Many aggregations

UNIONs

How to express the "European regions" in the example, if only "country" attribute available?

Strange predicates

"Christmas season" = October 15 - Dec 23 + Dec. 27-Jan.10

Data are available, but querying and aggregating difficult

Transform schema in order to support analytical queries. **Another data modeling task!**

Data Warehouse Modeling

Freie Universität Berlin

Which **dimensions** used to organize facts in DB?
Example: Time, Customers, book types
Which **level** of summarization for each dimension?
Time: Year, month, day and season
Customers: Region, income, gender, ...
Book types: Genre (fiction, hobby,...), language
Which **hierarchies** – organize levels within a dimension?
Language / Genre within book type etc.
Facts from the DB?
Unit price, orders, amount per order, ...

TPC-H

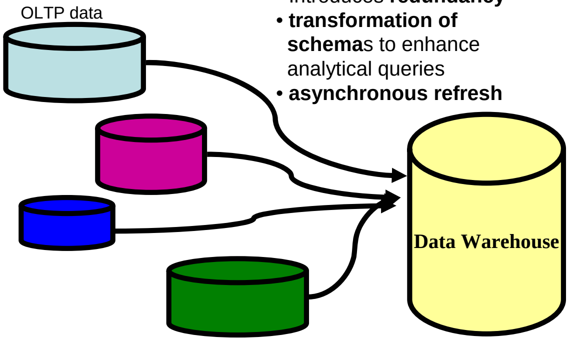
Freie Universität Berlin

Benchmark for DWH: sample query (functional)

```
select
n_name,
sum(l_extendedprice * (1 - l_discount)) as revenue
from customer, orders, lineitem, supplier, nation, region
where
c_custkey = o_custkey
and l_orderkey = o_orderkey
and l_suppkey = s_suppkey
and c_nationkey = s_nationkey
and s_nationkey = n_nationkey
and n_regionkey = r_regionkey
and r_name = '[REGION]'
and o_orderdate >= date '[DATE]'
and o_orderdate < date '[DATE]' + interval '1' year
group by
n_name
order by
revenue desc;
```

DWH as a centralized data resource

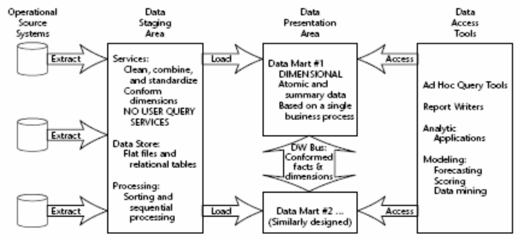
- Introduces **redundancy**
- **transformation of schemas** to enhance analytical queries
- **asynchronous refresh**



ETL : Extract – Transform - Load

14-DWH-7

Elements of a DWH



from R. Kimball: The Data Warehouse Toolkit

14-DWH-8

OLTP vs OLAP

OLTP

- Very l**arge number of transaction records**
- Typically **more than one DB** --> amazon.de, .fr, .com
- Slightly (?) **different schemas**
  - Amazon US versus Amazon D ?
- Must be **operational ~ 99.999 %** of the year
- **No long running queries**

OLAP

- Very **sophisticated ad hoc queries**
- Data have to be **homogenized**
- Ad hoc queries, but structurally similar (**aggregation**)

14-DWH-9

DWH / OLAP : Tools for Decision support

DWH exist since the 70ies

- Decision support systems
- Management Information Systems

No general methodology

Not very successful, why?

Infrastructure was missing

- networks
- high performance DBS
- high performance computer systems
  - cycles, main memory, disks

Data resources tremendously increased

14-DWH-10

Large databases

SBC Communications (US Telecommunication)

- 57.000.000 customers (phone, DSL)
- 360 Terabyte
- 12.000 tables
- 300.000 logins / day

Deutsche Telecom

- 100 Terabyte
- IBM, Oracle

An ERP application:

```
SQL> select count(*) from dba_tables
  2    where owner ='SAPR3';

     COUNT(*)
      ----------
      25885
```

14-DWH-11

DWH and Data mining: "Business Intelligence"

Data Mining

- Analyze data statistically (more or less) and **find out regular patterns**
- **Prediction** of values by
  - **association rules** ("Customers who buy beer buy cigarettes in 70% of all transactions")
  - **classification** "Good customer / bad customer"
- and more....

Which kind of **regularities are hidden in the data**?
Can be use them to look into the future?
(**Forecast, prognosis**)

14-DWH-12

DWH

In contrast to data mining data ware house technology aims at a *retrospective* analysis of (large volumes) of data.
Appropriate Decisions to be taken by people.

Definitions:
A data warehouse is a copy of transaction data specifically structured for query and analysis. (Kimball)

Ein Data Warehouse ist ein physischer Datenbestand, der eine integrierte Sicht auf die zugrunde liegenden Datenquellen ermöglicht. (Zeh, in Inf. Forschung u. Entw. 18(1), 2003)

Many more exist....

DWH versus Data mining

DHW:

Data have different "**dimensions**" : time, location, product group, ...

Aggregate dataset according to one or more dimensions, levels, hierarchies.

Some dimensions typical for a large class of problems: *time*, *location*.

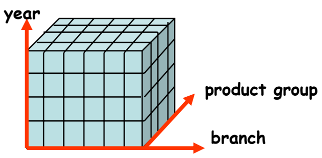
What happened in the past? Make business decisions based on what happened.

14.2 DWH methodology

Basic modeling

Facts → Book ISBN 3456 sold 2005.6.10 by .de-branch to Customer 12789

Facts: points in a multidimensional space

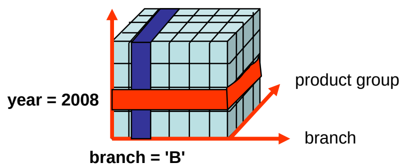


Dimensions: categories for describing facts: time, space, groups — time, branch, product group

Classification hierarchies — day -> month -> quarter -> year

DWH-Model

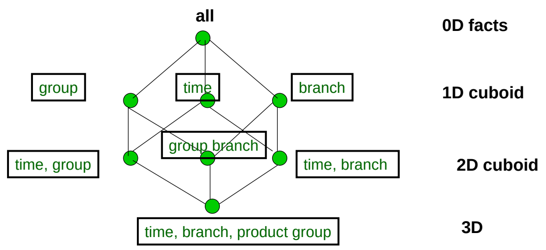
- DWH is based on a multi dimensional data model
- Each subset of dimensions defines a cuboid



Each cuboid represents a restriction and an aggregation.

Cuboids

Cuboids of n-dimensional hypercube arranged as a lattice:

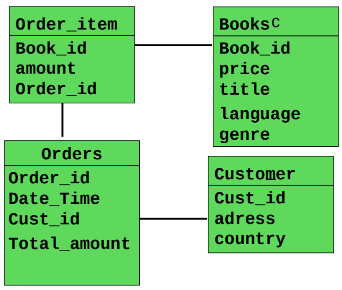


Base cuboid, an n-dim. point – a fact !

Number of cuboids?

14.3 Stars and stripes

How should a DWH be modelled?
OLTP schema **not satisfactory** for DHW operations



Dimensions?
What can be **omit**ted?
What kind of **redundancy** should be introduced?

Design of a DWH schema

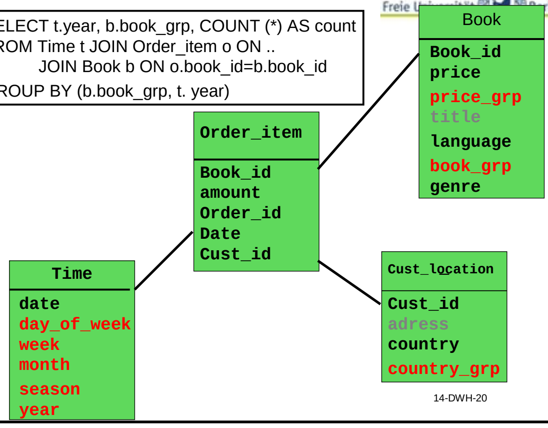
In the **bookstore example**

- the **orders** ("shopping carts") of clients are less interesting than **order items**
- Timestamps of orders should be mapped into a **time dimension hierarchy**: *day, month, season, year* ?
- *title* attribute of books not important (?) but *book group* and *price group*. Have to be introduced as new attributes.
- Countries should be **grouped** (South Europe, …)

⇒ …

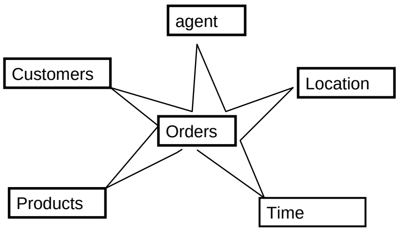
Star scheme of "Naïve Bookstore"

```
SELECT t.year, b.book_grp, COUNT (*) AS count
FROM Time t JOIN Order_item o ON ..
        JOIN Book b ON o.book_id=b.book_id
GROUP BY (b.book_grp, t. year)
```



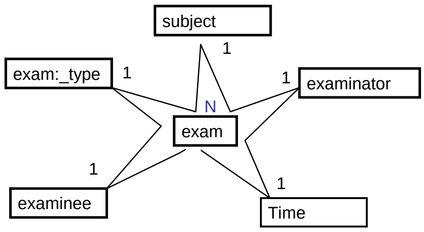
Methodology for DHW Building

Why "star"?



Fact table plus n non-normalized dimension tables

Another star

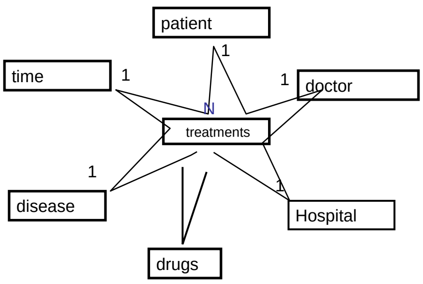


ER-pattern: n-ary relationship between dimension entities

Typical situation: **one "central" table with many "dimensions"**

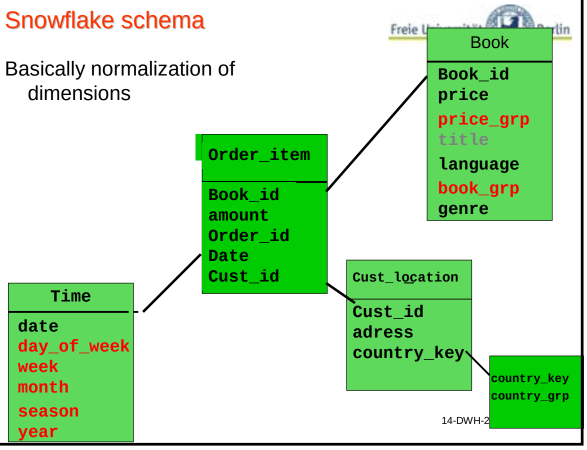
- 1:N relationship
- **foreign key** or attribute
- not normalized!

Medical DWH



Snowflake schema

Basically normalization of dimensions



Normalization / denormalization

OLTP

Normalized tables make sense: no update inconsistencies

Functional dependencies can be checked easily
(check for duplicate key, very efficent operation)

OLAP

Normalized relations ⇒ many joins when slicing
⇒ denormalized tables

Extract, transform, load

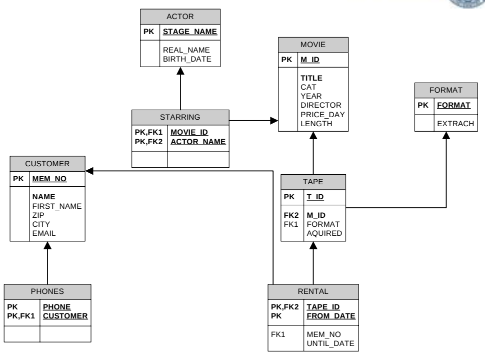
ETL tools transform operational OLTP DB into data warehouse

Must not necessarily be reversible:

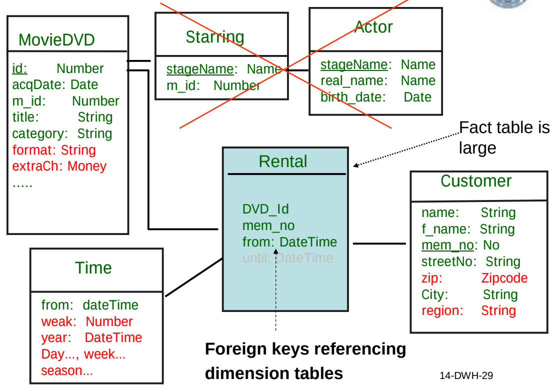
- **drop unimportant attributes**
- **aggregate values**
- introduce **classification hierarchie**
  ("time" -> month ->quarter -> year, see below)

Tools make life easier, but the creative part is the design of the DWH!

Movie DB OLTP schema



MovieDB: possible OLAP schema



Star schema tables

Time table attributes depend on resolution of time dimension

Time

date	dd	mm	y	q	week	season
12.3.2005	12	3	2005	1	10	spring
14.1.2005	17	1	2005	1	3	winter
23.8.2004 …….						

Rental

Mem_no	tid	from
531	1730	11.1.2005
278	311	23.8.2005
23.8.2004 …….		

Movie_DVD

customer

?

Star scheme ⇨ star join :

```
SELECT c.name, t.season, m.category
FROM customer c JOIN  time t on  (…)
           JOIN  movie_DVD m on (…) JOIN rental on(…)
WHERE c.zipcode LIKE '14%'
 // zipcode supposed to be string, if not: transform
```

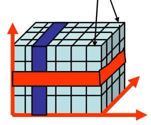

14.4 OLAP operators

OLAP queries do not ask for individuals (mostly...):

```
SELECT count(*) t.season, m.title
FROM customer c JOIN  time t on  (…)
     JOIN  movie_DVD m on (…)
     JOIN rental on(...)
WHERE c.zipcode LIKE '14%'
```

Result set: customers who…

A row (hypercube of dim n-2) is one group



Typical: grouping and aggregation

```
SELECT t.season, m.genre, count(*)
FROM customer c JOIN  time t on  (…)
     JOIN  movie_DVD m on (…)
     JOIN rental on(...)
WHERE c.zipcode LIKE '14%'
GROUP by t.season, m.genre
```

Bad design?!

Video OLAP DB

Better solution:

```
SELECT t.season, m.genre, a.zip_12, count(*)AS ct
FROM customer c JOIN  time t on  (…)
     JOIN  movie_DVD m on (…)
     JOIN rental on(...)
GROUP by t.season, m.genre, a.zip_12
```

adr_id	zipcode	zip_i	zip_12,...
10	14219	1	14,...
53	21984	2	21
13	14195	1	14

...

Mem_no	adr_id	tid	from
531	10	1730	11.1.2005
278	11	311	23.8.2005

23.8.2004 ....

customer

Roll up and Drilldown

Drill down: "drill deeper into hypercube"

more grouping attributes, less records in group
⇨ less aggregation ⇨ lower dimensional HC

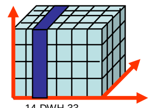
```
SELECT ... c.name,...
GROUP by t.season, m.genre, c.name
```

Roll up

less grouping attributes ⇨ more aggregation

```
SELECT t.season, count(*) FROM...
GROUP by t.season
```

… aggregates *all* DVD loans over each season



ROLLUP operator: motivation

```
SELECT m.category, t.season, count(*)
FROM customer c JOIN  time t on  (…)
          JOIN  movie_DVD m on (…)
          JOIN rental r on (…)
WHERE c.zipcode LIKE '14%'
GROUP by m.genre, t.season
```

Result table....

```
action spring  2388
action summer  2115
action fall    2917
action winter  3012
....
comedy summer  3527
....
thriller winter 5418
```

Missing:
aggregate value for groups e.g.

```
action    _    10422
...
thriller  -    12317
```


ROLLUP operator

```
SELECT m.category, t.season, count(*)
FROM customer c JOIN  time t on  (…)
          JOIN  movie_DVD m on (…)
          JOIN  rental r on (…)
WHERE c.zipcode LIKE '14%'
GROUP BY ROLLUP  (m.category, t.season)
```

Result:

```
action spring  2388
action summer  2115
action fall    2917
action winter  3012
action    _    10422
....
comedy summer   3527
....
thriller winter 5418
thriller  _    15317
_         _    93717
```

Super-aggregates

n Rollup attributes
⇨ n+1 groupings

ROLLUP BY (a1,...,an)
⇨
GROUP BY a1,...an
GROUP By a1,..a(n-1)
...
GROUP BY a1
[GROUP BY _]
i.e. aggregate over all rows

ROLLUP and GROUPING

Slight problem:

Suppose season has a NULL in some row
⇨ Result looks like

```
action spring  2388
action summer  2115
action fall    2917
action    _    0008
action    _   10422
....
comedy summer  3527
....
thriller winter 5418
thriller  _    15317
_         _    93717378
```

Confusing!
Want to distinguish NULL values in rows from super aggregates

GROUPING

```
SELECT m.category, t.season, count(*)AS ct,
       GROUPING (t.season)AS S, GROUPING (m.category) C
FROM customer c JOIN  time t on  (…)
          JOIN  movie_DVD m on (…)
          JOIN rental r on (…)
WHERE c.zipcode LIKE '14%'
GROUP BY ROLLUP  (m.category, t.season)
```

```
category season  ct      S  C
action spring  2388      0  0
action summer  2115      0  0
action    _      3012    0  0
action    _     10422    1  0
....
comedy summer  3527      ...
....
thriller winter 5418
thriller  _    15317     1  0
_         _    93717378  1  1
```

this line: not a superaggregate

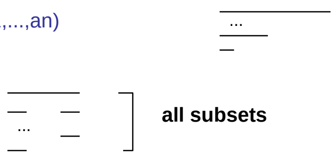
The CUBE operator

n attributes in GROUP BY (a1,...,an)
⇨ 2^n -1 nonempty subsets of {a1,..an}

GROUP BY CUBE (a1,...,an) :
UNION of 2^n -1 groupings together (UNION)
with aggregation of all rows, i.e. 2^n groupings

Obvious extension of ROLLUP a1,a2,....an

GROUP BY CUBE (a1,...,an)



CUBE example

```
SELECT m.category, t.season count(*)
FROM customer c JOIN  time t on  (…)
          JOIN  movie_DVD m on (…)
          JOIN rental r on (...)
WHERE c.zipcode LIKE '14%'
GROUP BY CUBE  (m.genre, t.season)
```

```
action spring  2388
action summer  2115
action fall    2917
action winter  3012
action   _    10422
....
comedy summer  3527
....
thriller winter 5418
thriller  _    15317
  _    spring 19317
...
  _    winter 27381
  _      _    93717
```

4 more rows for the second super aggregation over one attribute (season, 4 values)
But only 2 group attributes;
⇒ no combinatorial explosion, this time.

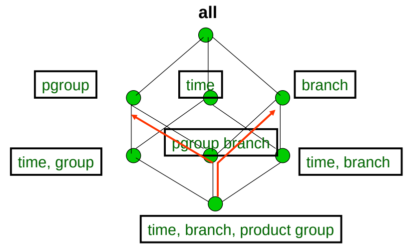
14.5 ROLAP

Reuse of temporary results -> materialization
Aggregation on m(onth) can be used for
aggregating over y(ear) ⇨ reuse

Materialized views

store aggregates which may be used frequently
combinatorial explosion prevents to store all of them
redundancy is NOT the problem in OLAP

Materialisation




```
INSERT INTO ProdGroupBranch
 (SELECT branch, pgroup, count(*) AS ct
  FROM sales JOIN time on ..
  GROUP BY branch, pgroup)
```

Reusable for GROUP BY branch or GROUP BY pgroup

Dimensions..

... are nonlinear in general.

time transformed into attributes
d → m → season → year *
↘ christmas_season ↗

Classification is a **partial order (lattice)** in general, e.g. time:
christmas_season = {Yes, NO} with semantics January 1..August 15 = NO,
August 16 – Dec 31 = YES...

Consequence: c_season cannot be aggregated from month (e.g.)

MOLAP

Efficiency is a big problem in the DWH context
CUBE over 3 or more attributes is heavy stuff

⇨ first solution: materialization
and specific index structures ("Bitmap index")

⇨ second solution: use completely different
data structures instead of tables

Buzz words:

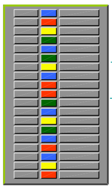
ROLAP : <u>R</u>elational **O**n**L**ine **A**nalytical **P**rocessing

MOLAP : <u>M</u>ultidimensional OLAP

Bitmap index

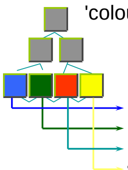
Bitlist index for attribute 'c':

For **each value v of c** store a bitlist B_c^v, and $B_c^v[i] = 1 \Leftrightarrow$ attribute c of record i has value v



File 3
Block 10
Block 11
Block 12

Index for attribute 'colour'



key	start ROWID	end ROWID	bitmap
<Blue,	10.0.3,	12.8.3,	1000100100010010100>
<Green,	10.0.3,	12.8.3,	0001010000100100000>
<Red,	10.0.3,	12.8.3,	0100000011000001001>
<Yellow,	10.0.3,	12.8.3,	0010001000001000010>

14-DWH-44

Bitmap operations

Efficient implementation of set operations

Example:

```
SELECT x,y,z FROM people
WHERE (color = 'Blue' OR color = 'Red' )
AND      sex = 'm'
```

```
(  <Blue, 10.0.3, 12.8.3,  1000100100010010100>
   <Red, 10.0.3, 12.8.3,   0100000011000001001> )   OR
   <male 10.0.3, 12.8.3,   1010101001001001010>     AND

   <RESULT                 1000100001000001000>
```

14-DWH-45

Bitmap operations

Grouping

```
SELECT color AS HairColor, count(*)AS CT
FROM people
GROUP BY color
```

```
<Blue, 10.0.3, 12.8.3, 1000100100010010100> ----> 6
<Red, 10.0.3, 12.8.3,  0100000011000001001> ----> 5
<pink 10.0.3, 12.8.3,  1010101001001001010> ----> 8
```

HairColor,	CT
Blue	6
Red	5
pink	1

14-DWH-46

Bitmap eval

Advantage

- If few values and many rows e.g. sex, marital status,..
- Compression of bit lists saves space compared to standard idx
- Efficient processing of OR / AND, COUNT queries,

Disadvantage

- Updates expensive.... Why? Not a problem in DWH
  - bitmaps must be locked during update (why?)
  - all blocks (and all rows) in a segment have to be locked

```
CREATE BITMAP INDEX cust_bidx1 ON Customer (hairColor)
    TABLESPACE myTBS PCTFREE 10;  -- Oracle style
```

14-DWH-47

Join indexes

Join index: JI(R-id, S-id) where R (R-id, …) ⋈ S (S-id, …)

- Indices map attribute value to a list of record ids
- Speeds up search for join candidates — but rather costly operation large relations: many lists have to be checked
- Data warehouses: join always links values of the **dimensions** **to rows in the fact table**. (star join)

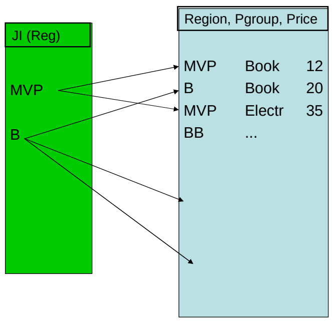
E.g. fact table: *Sales* and two dimensions *region* and *productGroup*

- A join index on *region* maintains for each distinct *region* a list of R-IDs of the tuples recording the *Sales* in that *region*.

- Join indices can span multiple dimensions

14-DWH-48

Join index: example




```
Select region,
        sum(price)
FROM Sales
GROUP BY region
```

14-DWH-49

Summary

- OLAP important for **strategic planning**
- **Transforms operational data** into "Data Warehouse"
- **Analyzes data by aggregation** and (simple) statistical operations
- OLAP = **data analysis** in a **multidimensional** space
- **ROLLUP, CUBE** etc part of **SQL-3**
- Implemented in most commercial system (Oracle, DB2)
- Workbench, e.g. AWM (Analytical Workspae Mgr., Oracle)
- OLAP functions may are based on RDBMS (ROLAP) or multidimensional data structures (MOLAP)