14 Data Warehouses in a nutshell

14.1 Introduction OLTP vs. OLAP
14.2 DWH methodology
14.3 Stars and Stripes
14.4 OLAP operators: Roll up and Drill down, SQL operators ROLLUP and CUBE
14.5 ROLAP and MOLAP ... and more

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

14.1 Introduction OLTP vs. OLAP

Online Transaction processing (OLTP)
Online Analytical processing (OLAP)

10^6 users per day
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

<table>
<thead>
<tr>
<th>Order_item</th>
<th>Order_id</th>
<th>Amount</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Book_id</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Orders

<table>
<thead>
<tr>
<th>Order_id</th>
<th>Date</th>
<th>Cust_id</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Total_amount</td>
</tr>
</tbody>
</table>

Customer

<table>
<thead>
<tr>
<th>Cust_id</th>
<th>Address</th>
<th>Country</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Books

<table>
<thead>
<tr>
<th>Book_id</th>
<th>Price</th>
<th>Title</th>
<th>Language</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

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....

Suites for Online shop, less suited for analytical queries

Analytical queries on OLTP DB

"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

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

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

**OLTP vs OLAP**

**OLTP**

- Very large 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)

**Large databases**

SBC Communications (US Telecommunication)
- 57,000,000 customers (phone, DSL)
- 360 Terabyte
- 12,000 tables
- 300,000 logins / day

Deutsche Telekom
- 100 Terabyte
- IBM, Oracle

An ERP application:

```
SQL> select count(*) from dba_tables
2 where owner = 'SAPR3';
COUNT(*)     
----------
25445
```

**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 used them to look into the future? (Forecast, prognosis)
DWH

In contrast to data mining data warehouse 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)

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: points in a multidimensional space

Dimensions:
categories for describing facts: time, space, groups

Classification hierarchies

day -> month -> quarter -> year

Cuboids

Cuboids of n-dimensional hypercube arranged as a lattice:

Number of cuboids?

0D facts

1D cuboid

2D cuboid

3D

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

14.3 Stars and stripes

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

Dimensions?
What can be omitted? 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, ...)

Methodology for DHW Building

Why "star"?

Fact table plus n non-normalized dimension tables

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 efficient 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 hierarchy

(“time” → month → quarter → year, see below)

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

---

**Movie DB OLTP schema**

![Movie DB OLTP schema](image)

**MovieDB: possible OLAP schema**

![MovieDB: possible OLAP schema](image)

---

**Star schema tables**

Time table attributes depend on resolution of time dimension.

<table>
<thead>
<tr>
<th>Time</th>
<th>Movie_DVD</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.3.2005</td>
<td>12 3 2005</td>
</tr>
<tr>
<td>14.1.2005</td>
<td>17 1 2005</td>
</tr>
<tr>
<td>23.8.2004</td>
<td>……</td>
</tr>
</tbody>
</table>

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%'

---

**14.4 OLAP operators**

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

SELECT 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

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

Result set: customers who...

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

Bad design?!
**Video OLAP DB**

Better solution:

```sql
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 r on (...)
GROUP by t.season, m.genre, a.zip_12
```

<table>
<thead>
<tr>
<th>Mem_no</th>
<th>adr_id</th>
<th>tid</th>
<th>23.8.2004</th>
</tr>
</thead>
<tbody>
<tr>
<td>60</td>
<td>14219</td>
<td>1</td>
<td>14</td>
</tr>
<tr>
<td>228</td>
<td>311</td>
<td>21</td>
<td>23.8.2005</td>
</tr>
</tbody>
</table>

**Roll up and Drill down**

Drill down: "drill deeper into hypercube"
- more grouping attributes, less records in group
  - less aggregation
  - lower dimensional HC

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

Roll up
- less grouping attributes ⇒ more aggregation

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

... aggregates all DVD loans over each season

**ROLLUP operator: motivation**

```sql
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>
<thead>
<tr>
<th>category</th>
<th>season</th>
<th>ct</th>
</tr>
</thead>
<tbody>
<tr>
<td>action spring</td>
<td>2388</td>
<td></td>
</tr>
<tr>
<td>action summer</td>
<td>2115</td>
<td></td>
</tr>
<tr>
<td>action fall</td>
<td>2917</td>
<td></td>
</tr>
<tr>
<td>comedy summer</td>
<td>3527</td>
<td></td>
</tr>
<tr>
<td>thriller winter</td>
<td>5418</td>
<td></td>
</tr>
</tbody>
</table>

Missing: aggregate value for groups e.g.

- action 10422
- thriller 12317

**ROLLUP operator**

```sql
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:

<table>
<thead>
<tr>
<th>category</th>
<th>season</th>
<th>ct</th>
</tr>
</thead>
<tbody>
<tr>
<td>action spring</td>
<td>2388</td>
<td></td>
</tr>
<tr>
<td>action summer</td>
<td>2115</td>
<td></td>
</tr>
<tr>
<td>action fall</td>
<td>2917</td>
<td></td>
</tr>
<tr>
<td>comedy summer</td>
<td>3527</td>
<td></td>
</tr>
<tr>
<td>thriller winter</td>
<td>5418</td>
<td></td>
</tr>
<tr>
<td>thriller</td>
<td>15317</td>
<td>93717</td>
</tr>
</tbody>
</table>

- n Rollup attributes ⇒ n+1 groupings
- ROLLUP BY (a1,...,an)
  - GROUP BY af,...an
  - GROUP BY af,...(n-1)
- GROUP BY af
- i.e. aggregates 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   |     |
| comedy summer  | 3527   |     |
| thriller winter | 5418   |     |

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

<table>
<thead>
<tr>
<th>category</th>
<th>season</th>
<th>ct</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>action spring</td>
<td>2388</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>action spring</td>
<td>2388</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>action</td>
<td>10422</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>comedy</td>
<td>3527</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>thriller</td>
<td>15317</td>
<td>93717</td>
<td></td>
</tr>
</tbody>
</table>

- this line: not a superaggregate

**GROUPING**

```sql
SELECT m.category, t.season, count(*) AS ct,
    GROUPING (t.season) AS S, GROUPING (m.category) AS C
FROM customer c JOIN time t on (...)
    JOIN movie_DVD m on (...)
GROUP BY ROLLUP (m.category, t.season)
```
The CUBE operator

\( n \) attributes in \( \text{GROUP BY} \) \((a_1, \ldots, a_n)\)

\( \Rightarrow 2^n - 1 \) nonempty subsets of \((a_1, \ldots, a_n)\)

\[ \text{GROUP BY CUBE} (a_1, \ldots, a_n) : \]

UNION of \( 2^n - 1 \) groupings together (UNION)

with aggregation of all rows, i.e. \( 2^n \) groupings

Obvious extension of ROLLUP \( a_1, a_2, \ldots, a_n \)

\[ \text{GROUP BY CUBE} (a_1, \ldots, a_n) \]

... all subsets

---

CUBE example

```
SELECT m.genre, 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 winter 3012
action _ 10422
comedy summer 3527
... 
thriller winter 5418
thriller _ 13917
  _ spring 19317
  _ winter 27381
_  _  _  _ 93717
```

4 more rows for the second super aggregation over
one attribute (season, 4 values)

But only 2 group attributes;
\( \Rightarrow \) no combinatorial explosion, this time.

---

14.5 ROLAP

Reuse of temporary results -> materialization

Aggregation on \( n \text{(on(th))} \) can be used for aggregating over \( y \text{(ear)} \) \( \Rightarrow \) 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

---

Dimensions..

... are nonlinear in general.

Classification is a partial order (lattice) in general, e.g. time:

\( \text{christmas season} = \{\text{Yes}, \text{NO}\} \) with semantics January 1..August 15 = NO,

August 16 – Dec 31 = YES...

Consequence: \( c \_ \text{season} \) 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

\( \Rightarrow \) first solution: materialization

and specific index structures (“Bitmap index”)

\( \Rightarrow \) second solution: use completely different

data structures instead of tables

Buzz words:

ROLAP : Relational OnLine Analytical Processing

MOLAP : Multidimensional OLAP
Bitmap index

**Bitlist index** for attribute `v`:
For each value `v` of `c` store a bitlist `B_v`, and `B_v[i] = 1` if attribute `c` of record `i` has value `v`.

**Index for attribute `colour`**
- Start and end positions:
- `key`, `ROWID`, `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>`

Bitmap operations

**Efficient implementation of set operations**
Example:

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

```
<Blue, 10.0.3, 12.8.3, 000100100010010100>
<Red, 10.0.3, 12.8.3, 0100000011000001001>
<male 10.0.3, 12.8.3, 0101010001010101010>
<RESULT 1000100001000001000>
```

**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

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

Join indexes

**Join index:** `JI(R-id, S-id)` where `R (R-id, …)` and `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

```sql
SELECT region, sum(price)
FROM Sales
GROUP BY region
```

```
Region, Pgroup, Price
MVP Book 12
B Book 20
MVP Electr 35
BB ...
```

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