Data-Warehouse-Technologien

Exercise sheet 8

Assignment 1: **What is vertical and horizontal partitioning of database tables? What is the difference regarding allocation?**

Assignment 2: **Given the following dates, compare the different requirements for storing the cube in MOLAP- or ROLAP:**

1. **1 fact; 3 dimensions with each 1000 values; filling degree 20%; 1 attribute = 8 bytes**
2. **1 fact; 5 dimensions with each 1000 values; filling degree 20%; 1 attribute = 8 bytes**
3. **1 fact; 3 dimensions with each 1000 values; filling degree 50%; 1 attribute = 8 bytes**
4. **1 fact; 5 dimensions with each 1000 values; filling degree 50%; 1 attribute = 8 bytes**

Assignment 3: **Create the dwarf for the following example data:**

Region	Customer	Product	Price
Saxony-Anhalt	Müller	Mobile phone	30
Saxony	Schmidt	TV	30
Saxony-Anhalt	Schneider	TV	20
Saxony	Fischer	Mobile phone	45

Which advantages do dwarfs have?

Assignment 4: **Discuss important properties of row- and column-stores concerning the following aspects:**

1. **Usability for Online Analytical Processing**
2. **Compression techniques**
3. **Query execution**

Assignment 5: Given is the following SQL-query:

SELECT shipdate, linenum
FROM lineitem
WHERE shipdate = '12-30-1995' AND linenum = 12

Based on this query, discuss the different materialization strategies for Column-Stores.

 Good Luck!

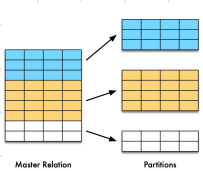
Assignment 1

a) vertical and horizontal partitioning of database tables?

Partitioning = separate large relations into smaller subrelations (partition, fragments)

↳ size depends on request and update characteristics

Horizontal partitioning (sharding)



- divides a table into multiple smaller tables
- each table is a separate data store
- contains same number of colums, but fewer rows

→ split table by rows

Forms of splitting

- Range partitioning: each partition is defined by selection criterion
  - usecase of ordered columns (eg. integers, longs, ...) to separate rows
  - eg. [0...999, 1000...1999]

  eg. Range UID 1 – 5 (Partition 1) UID 6 – 8 (Partition 2)

  - equal size of rows per table
  - exploit range predicates

- Hash partitioning:
  - Hash determines to which partition a tuple belongs
  - Tuples with same hash value are in the same partition

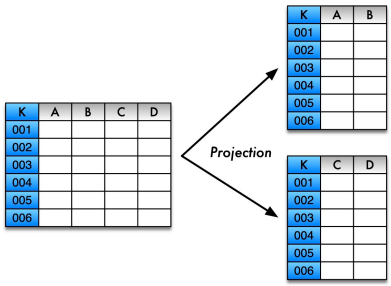
  (Hash function is applied to one or more columns)

  [h(x)=1, h(x)=2, h(x)=3, ...]

  - fixed number of partitions → evenly large partitions
  - load balancing

Vertical partitioning

- some columns are moved to new tables (split table by columns)
- each table contains same number of rows but fewer columns
- ! For reconstructing master relation there must be the same attribute in two partitions (usually primary key) → each partition needs primary key



Mini Dimensions as special case of vertical partitioning
- frequently used columns are stored together

b) Difference regarding allocation ?

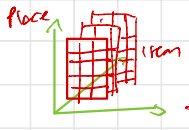
Allocation = process of distributing different segments of database across multiple physical locations
(used to improve performance, reliability, data consistency)
(physical distribution refers to how data is stored, replicated and managed across various physical storage systems or locations)
→ decide where and how to store data

Horizontal: allocates partitions to nodes based on row-level filters, helps balance data volume accross nodes

Vertical:
- Allocates partitions to nodes based on column usage
- optimizes query performance for specific workloads

Assignment 2

Comparison for different requirements for storing the cube in MOLAP or ROLAP



MOLAP: = Multidimensional Storage

- uses different data structures for data cube and dimensions
- storage of cube: as array (multidimensional arrays)
- ordering dimensions: for addressing the cube cells necessary

→ Mehrdimensionales Datenmodell, für einfache Datenanalyse
→ Daten werden vorab berechnet, zsm.gefasst und in MOLAP gespeichert
→ changing dimension elements, changes whole array (rebuild array → work intense)

ROLAP: = Relational storage

- Implementation of star or snowflake schema to relations
- Common form of storing DW tables
  - large fact tables (relational tables for storing the cube)
  - multidimensional access
  - update characteristics

PlaceID	ItemID	TimeID	Fact
0	0	1	x
0	1	1	y
-	-	-	-

↳ changing dimensions/elements lead to adding/deleting rows → not as costly as rebuilding

Difference:

S.NEIN	ROLAP	MOLAP
1.	ROLAP steht für **Relational Online Analytical Processing.**	**Wobei MOLAP für Multidimensional Online Analytical Processing** steht .
2.	ROLAP wird für große Datenmengen verwendet.	Es wird zwar für begrenzte Datenmengen verwendet.
3.	Der Zugriff auf ROLAP ist langsam.	Während der Zugriff auf MOLAP schnell ist.
4.	In ROLAP werden Daten in Beziehungstabellen gespeichert.	In MOLAP werden Daten in einem mehrdimensionalen Array gespeichert.
5.	In ROLAP werden Daten aus dem Data Warehouse abgerufen.	Während des MOLAP-Betriebs werden die Daten aus der MDDB-Datenbank abgerufen.
6.	In ROLAP werden komplizierte SQL-Abfragen verwendet.	Während in MOLAP eine dünn besetzte Matrix verwendet wird.
7.	In ROLAP wird eine statische mehrdimensionale Datenansicht erstellt.	Während in MOLAP eine dynamische mehrdimensionale Ansicht der Daten erstellt wird.

Storage

MOLAP:
- dense data storage
- data is stored in compressed, multi-dimensional arrays
- storage depends on total potential case size (number of cells) and filling degree (percentage of non-empty cells)

− empty cells are included in structure

ROLAP:
- sparse data storage

+ data stored in relational tables → only non empty cells
- storage proportional to number of non-empty cells and size of each record (fact and dimensions combined)

A) 1 Fact, 3 Dimensions, 1000 values each, 20% Filling, 1 Attribut = 8 Bytes

- Total case size: 1000 × 1000 × 1000 = 1 Billion cells
  non-empty cells (20%): 0,2 × 1 Billion = 200 Millionen cells → gives us only fact
  Per cell storage: 8 Bytes (fact) + dimension key

MOLAP: approximate storage: 1 Billion cells × 8 Bytes = 8 GB
ROLAP: 200 Million * (3 + 1) → 1 = fact (always 1) = 6,4 GB
approximate storage 200 Mio × 8 Bytes

B) 1 fact, 5 dimensions, 1000 values each, 20% Filling, 1 Attribute = 8 Bytes

- Total case size: 1000^5 = 1 trillion
- nonempty cells (20%): 0,2 · 1 billion = 200 billion cells

MOLAP: 1 billion cells × 8 Byte = 8 PB
ROLAP: (200 billion cells) * (5 + 1) = 9.6 PiB

↳ more Dimensions Rolap gets worse

c) 1 fact, 3 dimensions, 1000 values each, 50% Filling, 1 Attribute = 8 Bytes

- Total Cube Size: 1 Billion cells → MOLAP 8 GiB
- 500 Million cells * (3+1) → ROLAP 16 GiB

d) 1 fact, 5 dimensions, 1000 values each, 50% Filling, 1 Attribute = 8 Bytes

- Total cube size: 1 Trillion cells → MOLAP 8 PiB
- non-empty cells (50%): 500 Billion cells * (5+1) → ROLAP 24 PiB

Result

- MOLAP: efficient for small, dense datasets due to compression
  → inefficient for sparse datasets with many dimensions and low filling bc. empty cells are still included in structure
  → more dimensions and higher filling degree

- ROLAP: scales better with high-dimensional and sparse datasets as it stores only non-empty cells
  → less efficient than MOLAP for dense datasets, with high filling degree
  → less dimensions and less filling degree

HOLAP: Hybrid OLAP

- higher granularities → MOLAP (dense)
- finer granularities → ROLAP (sparse)

> partition cube to store data in different formats