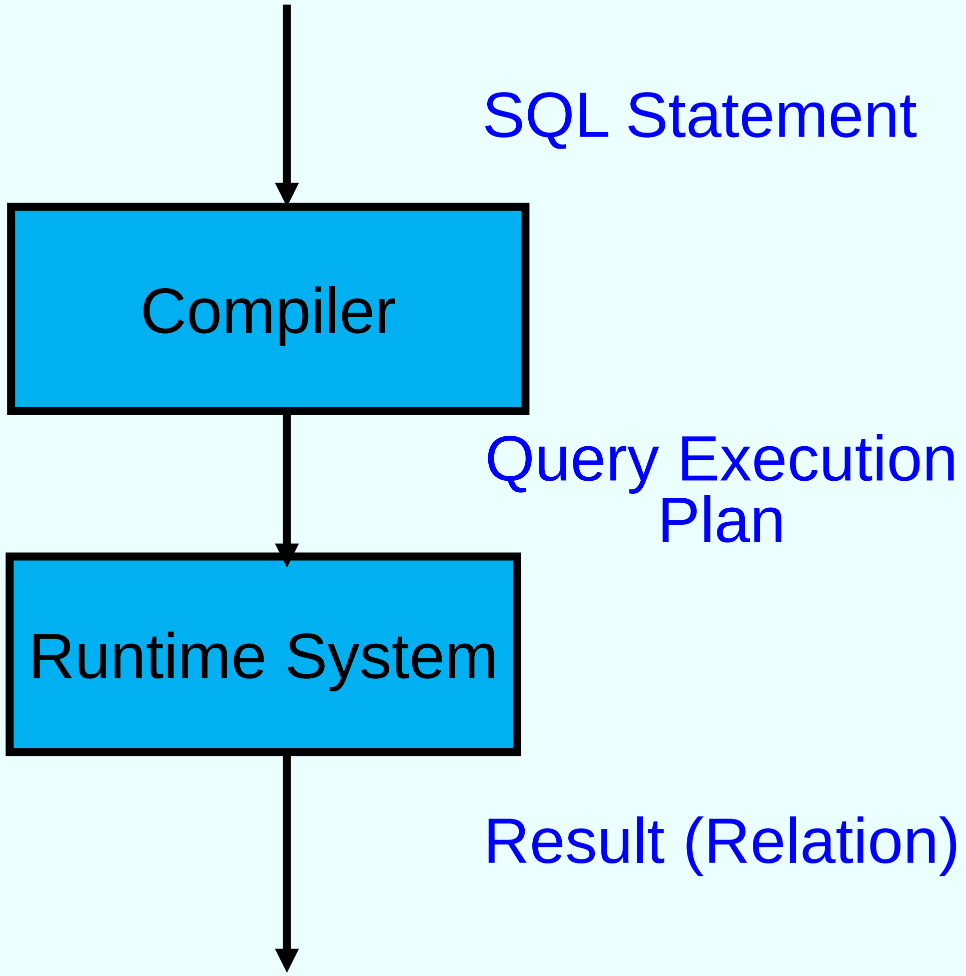
Query Execution



Compiler

- SQL is declarative,
  for runtime system it has to be translated into something procedural
- DBMS first translates SQL into an internal representation
- Common approach is translation into the relational algebra

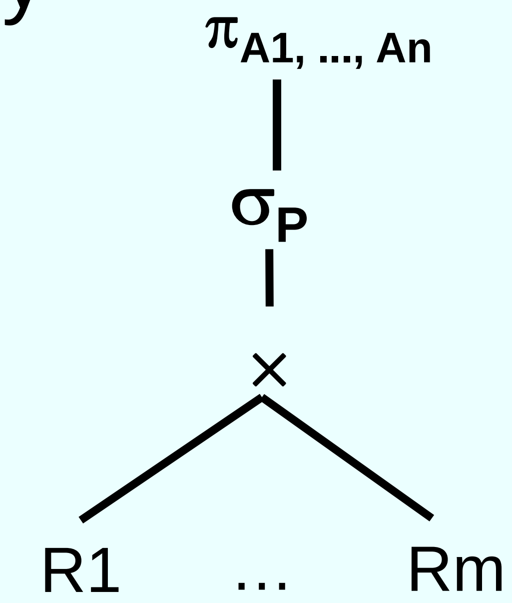
Canonical Translation

- Standard translation of SQL into relational algebra
- Algebra expression are often represented graphically
- Example

  select *A*1*, . . . ,An*

  from *R*1*, . . . , Rm*

  where *p*



Optimizer

- Canonical plan not efficient,
  e.g. cartesian product

- DBMS has optimizer to transform the plan into an efficient form
- Finding the / one optimal plan very difficult:
  still research topic

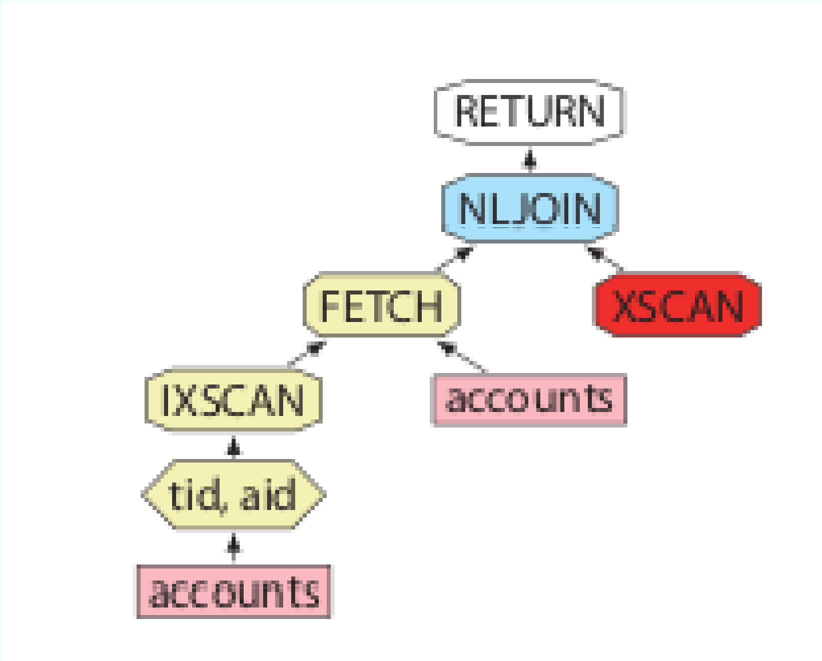
Optimizer (2)

- User ↔ Query optimization?

- Users can
  - see the generated plan
  - analyse the generated plan
  - (where appropriate) reformulate queries or give DBMS hints for query execution

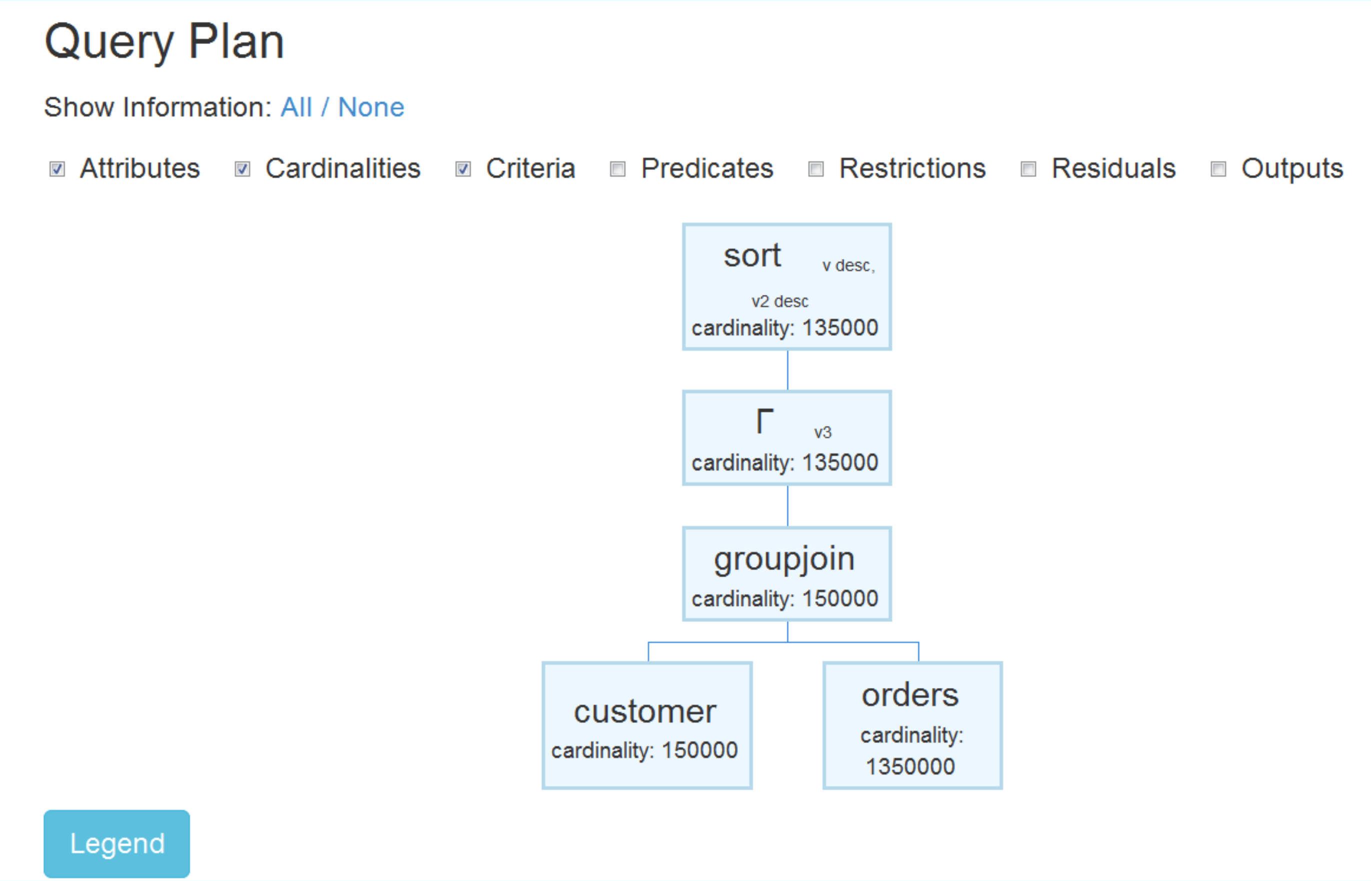
Visualization of plans

e.g. PostgreSQL:



Visualization of plans

e.g. Hyper:



Query Optimization Basics

- Query optimization is cost based
- Execution time estimation with the help of cost models and statistics
- Application of heuristics
  far too expensive to look at all possible plans
- Two layers of optimization:
  - Logical layer
  - Physical layer

Logical Layer

- **Starting point**: expression of the relational algebra (result of the canonical translation)
- **Optimization**: transformation into equivalent expressions (with faster execution time)
- **Goal of the transformations**: Output (result) of the single (algebra) operations preferably small

Logical Layer (2)

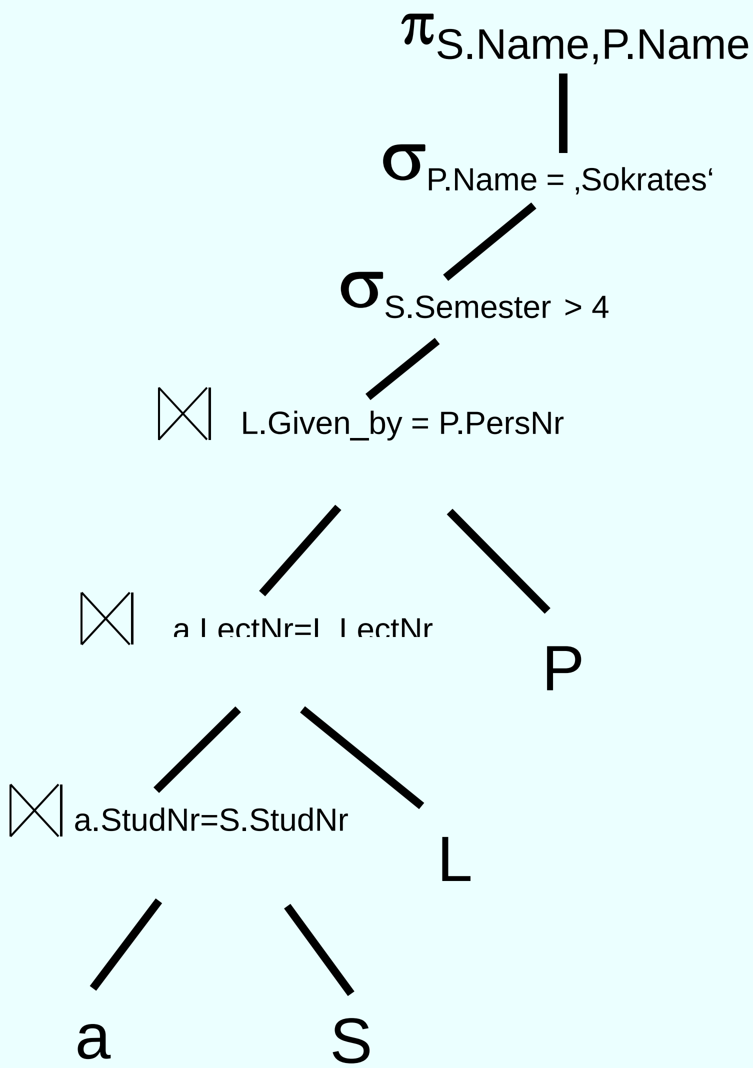
Basic techniques - rules:

- Break selections
- Shift selections ‚down‘ in the plan
- Combine selections and cartesian products to joins
- Determine the join order
- Insert projections
- Shift projektions ‚down‘ in the plan

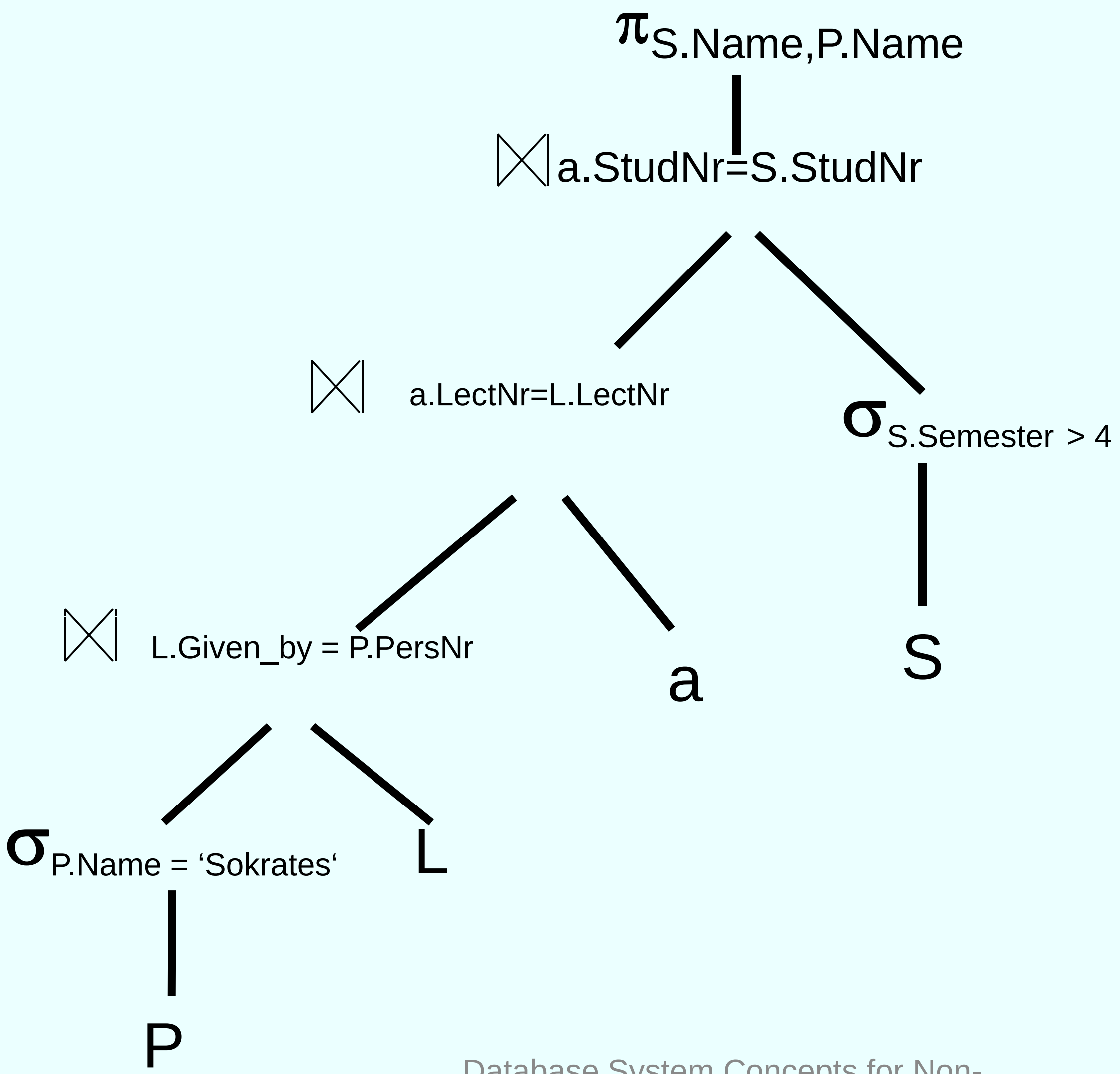
Example Query

```
select S.Name, P.Name
from Students S, attend a, Lectures L,
Professors P
where S.StudNr = a.StudNr
and a.LectNr = L.LectNr
and L.Given_by = P.PersNr
and S.Semester > 4
and P.Name = ’Sokrates’;
```

Query Plan


$\pi_{S.Name,P.Name}$
$\sigma_{P.Name = ,Sokrates‘}$
$\sigma_{S.Semester > 4}$
$\bowtie$ L.Given_by = P.PersNr
$\bowtie$ a.LectNr=L.LectNr
P
$\bowtie$ a.StudNr=S.StudNr
L
a
S

Optimized Query Plan



Physical Optimization

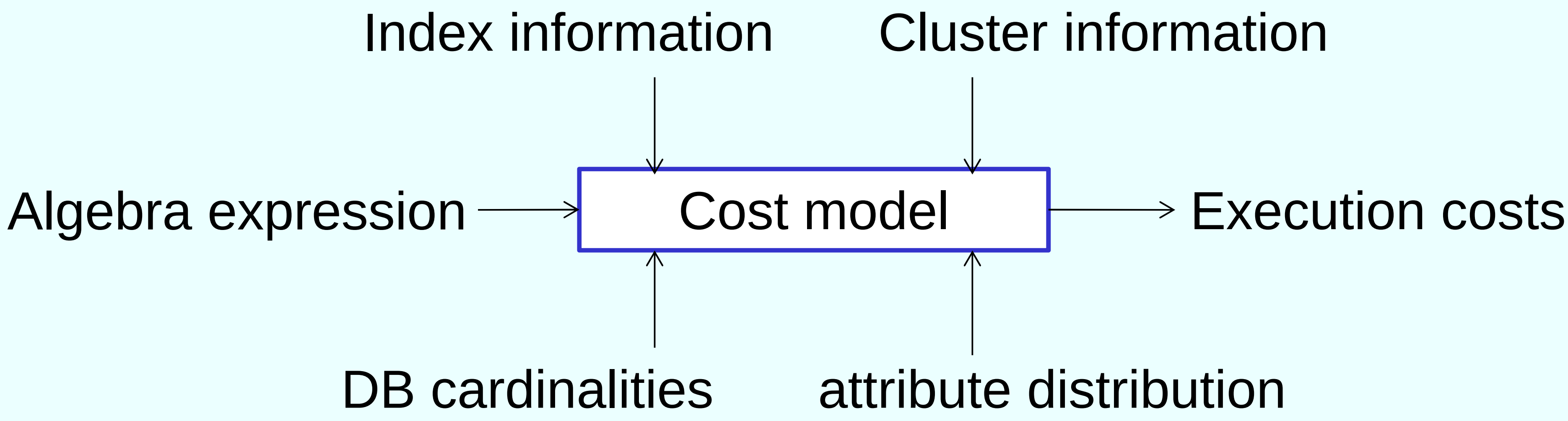
- Distinction between logical and physical algebra operators
- Physical algebra operators are the implementation of the logical ones
- Several physical operators for one logical operator available
- Optimization on the physical layer means:
  - Pick one of the physical operators
  - Decide whether to use indexes
  - Decide whether to materialize intermediate results
  - …

Implementation of Operators

- Selection:
  - o Scan
  - o Indexscan

- Join:
  - o Nested-Loop-Join
  - o (Sort-)Merge-Join
  - o Index-Join
  - o Hash-Join

Cost model


Index information
Cluster information
Algebra expression
Cost model
Execution costs
DB cardinalities
attribute distribution

Cost estimation

- Selectivity

  fraction of qualifying tuples of an operation
  high selectivity means small fraction
- Estimate of selectivity through
  - Formula
  - Sampling
- Selectivity cost
- Join cost
- Join order

→ therefore you need statistics

When to gather statistics?

When to update statistics?
Excerpt of IBM DB2 Manual:

- after data has been loaded into a table and appropriate indexes have been built
- after a new index for a table has been built
- after a table has been reorganized with REORG
- after a table and the corresponding indexes have been changed considerably via UPDATE-, INSERT- or DELETE-operations
- after executing the command REDISTRIBUTE DATABASE PARTITION GROUP

Which kind of statistics are there?

Excerpt of the IBM DB2 Manual:

SYSCAT: read-only view – System catalog

SYSSTAT: updatable view – Statistic data

- Table statistics (SYSCAT/SYSSTAT.**TABLES**)
- Column statistics (SYSCAT/SYSSTAT.**COLUMNS**)
- Statistics for groups of columns (SYSCAT/SYSSTAT.COLGROUPS)
- distribution statistics for columns (SYSCAT/SYSSTAT.COLDIST)
- distribution statistics for groups of columns (SYSCAT/SYSSTAT.**COLGROUPDIST / COLGROUPDISTCOUNTS)**
- Index statistics (SYSCAT/SYSSTAT.**INDEXES**)

Summary

- Query execution and optimization are important tasks of a database system

- Also / even users should know about it as design decisions and query formulation influence the performance of a DBMS

New Developments

- Main Memory Database Systems,

(https://en.wikipedia.org/wiki/List_of_in-memory_databases), e.g.

Times Ten (Oracle)

VoltDB (some database scientists, open source)

Monet DB (CWI, Amsterdam, open source)

SAP HANA

HYPER (Informatics, TUM)

- Column Store Database Systems,

(https://en.wikipedia.org/wiki/List_of_column-oriented_DBMSes), e.g.

C-Store / Vertica (HP)

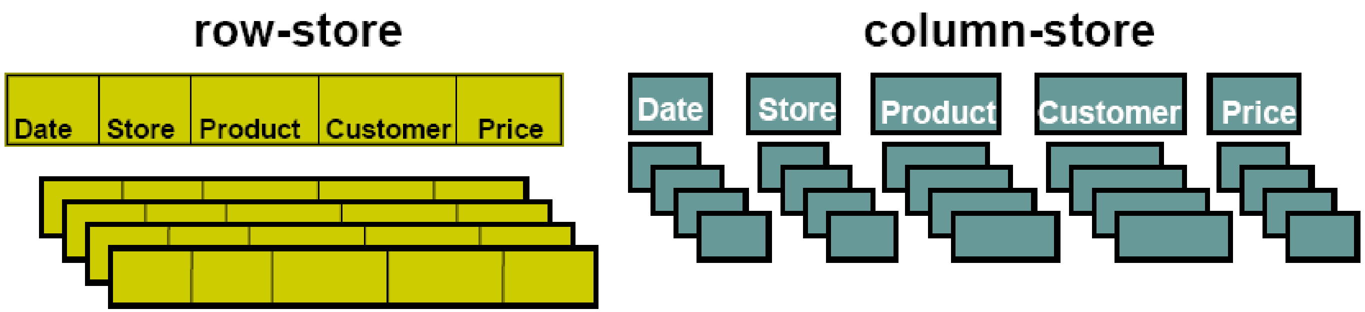
Monet DB (CWI, Amsterdam, open source)

SAP HANA

HYPER (Informatics, TUM)

Column Stores

What is a column-store?



row-store	column-store
+ easy to add/modify a record	+ only need to read in relevant data
- might read in unnecessary data	- tuple writes require multiple accesses

=> *suitable for read-mostly, read-intensive, large data repositories*

Row Store versus Column Store

Sales				
Product	Customer	Price	Branch	…
Mobile	Kemper	345	Schwabing	…
Radio	Mickey	123	Bogenhausen	…
Mobile	Minnie	233	Schwabing	…
Fridge	Urmel	240	Augsburg	…
Beamer	Bond	740	London	…
Mobile	Lucie	321	Bogenhausen	…

Row Store versus Column Store

Product	
ID	Product
0	Mobile
1	Radio
2	Mobile
3	Fridge
4	Beamer
5	Mobile

Customer	
ID	Customer
0	Kemper
1	Mickey
2	Minnie
3	Urmel
4	Bond
5	Lucie

Price	
ID	Price
0	345
1	123
2	233
3	240
4	740
5	321

Branch	
ID	Branch
0	Schwabing
1	Bogenhausen
2	Schwabing
3	Augsburg
4	London
5	Bogenhausen

Compression

In particular possibilities of compression are interesting:

Dictionary	
ID	Word
0	Augsburg
1	Beamer
2	Bogenhausen
3	Mobile
4	Fridge
5	London
6	Radio
7	Schwabing
…	

Product	
ID	Product
0	3
1	6
2	4
3	4
4	1
5	3

Branch	
ID	Branch
0	7
1	2
2	7
3	0
4	5
5	2

NoSQL

NoSQL
No SQL - Not only SQL

Characteristics: Schema-free, web scale (but sacrificing ACID), distributed (scale-out), usually key-value store (hash tables: key and pointer to the value), specific data, e.g. graphs

CAP Theorem:

- Consistency
- Availability
- Partition Tolerance

Only two of the three goals can be achieved

NoSQL cont.

Short explanation:
http://www.youtube.com/watch?v=pHAItWE7QMU&list=PLB9uLawXQoggpG9MGz5v9wDodr9f_p4lg

‚Debate' RDBMS – NoSQL (Mongo DB):
http://www.youtube.com/watch?v=b2F-DItXtZs