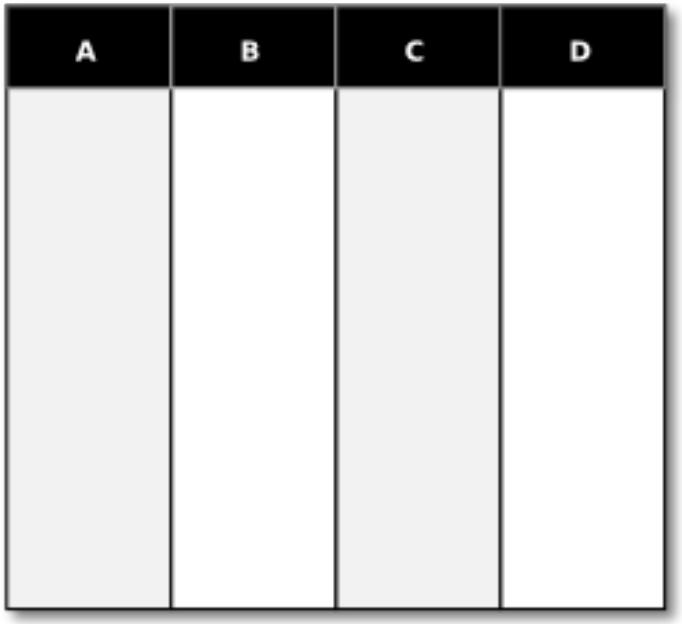
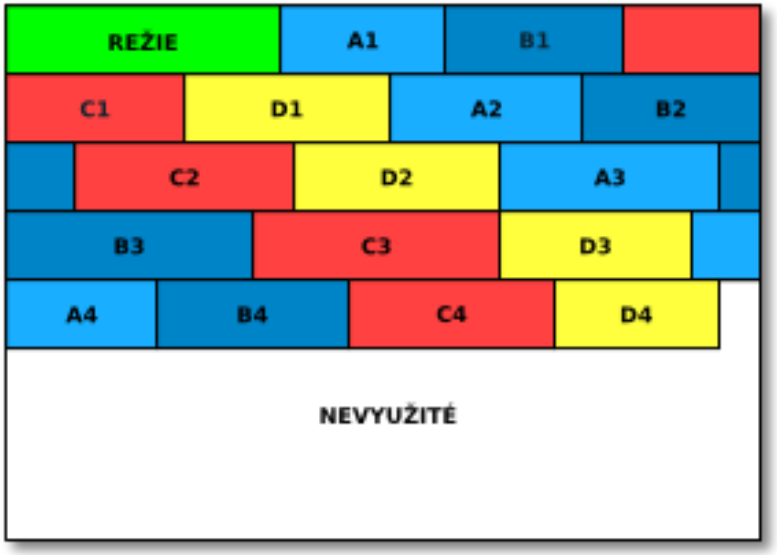
History

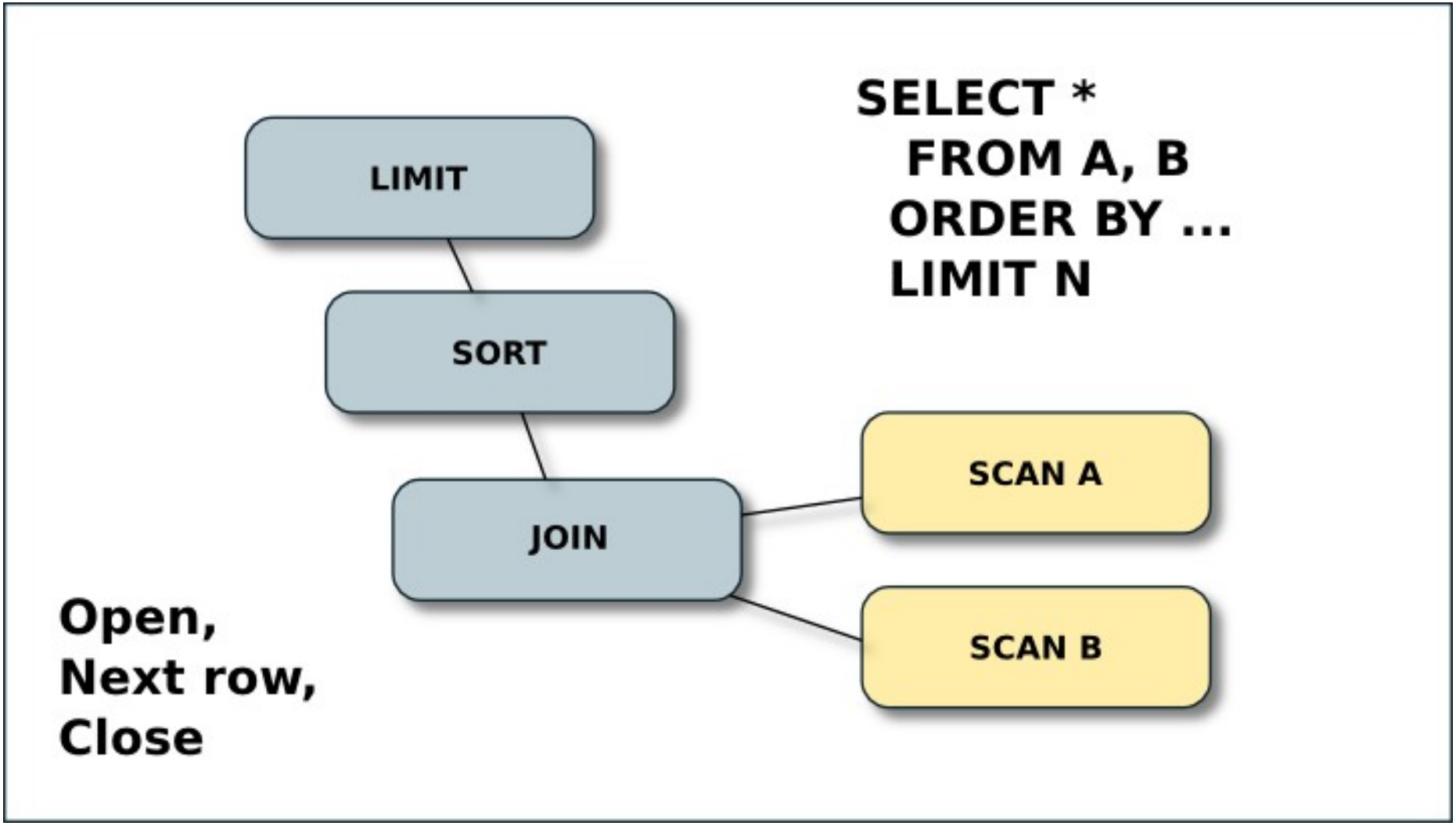
- 1970, 1980, **slow** CPU, **slow** disk, **low** memory
  - start of current SQL databases era
  - row storage (optimized for OLTP)
  - rule based, statistic based query planner
  - "open, get next row, close" executor (pipeline)
- 1990 faster CPU, more memory, **slow** disc
  - same architecture (more memory is used as cache)
  - replications
- now, ***fast CPU, GPU, fast SSD, lot of memory***
  - same architecture ???
  - OLAP, Big Data ???

Row/Page storage





Pipeline Executor



2000-...

- New concepts (**NO*** ACID, **NO*** SQL, **NO*** relational)
  - Stream databases - online views
  - Memory databases - memcache
  - NoSQL databases - KEY/VALUE concepts
  - Distributed "cloud" databases - Hadoop
  - Array, Raster databases - SciDB, RasDaMan
  - **Column store databases**

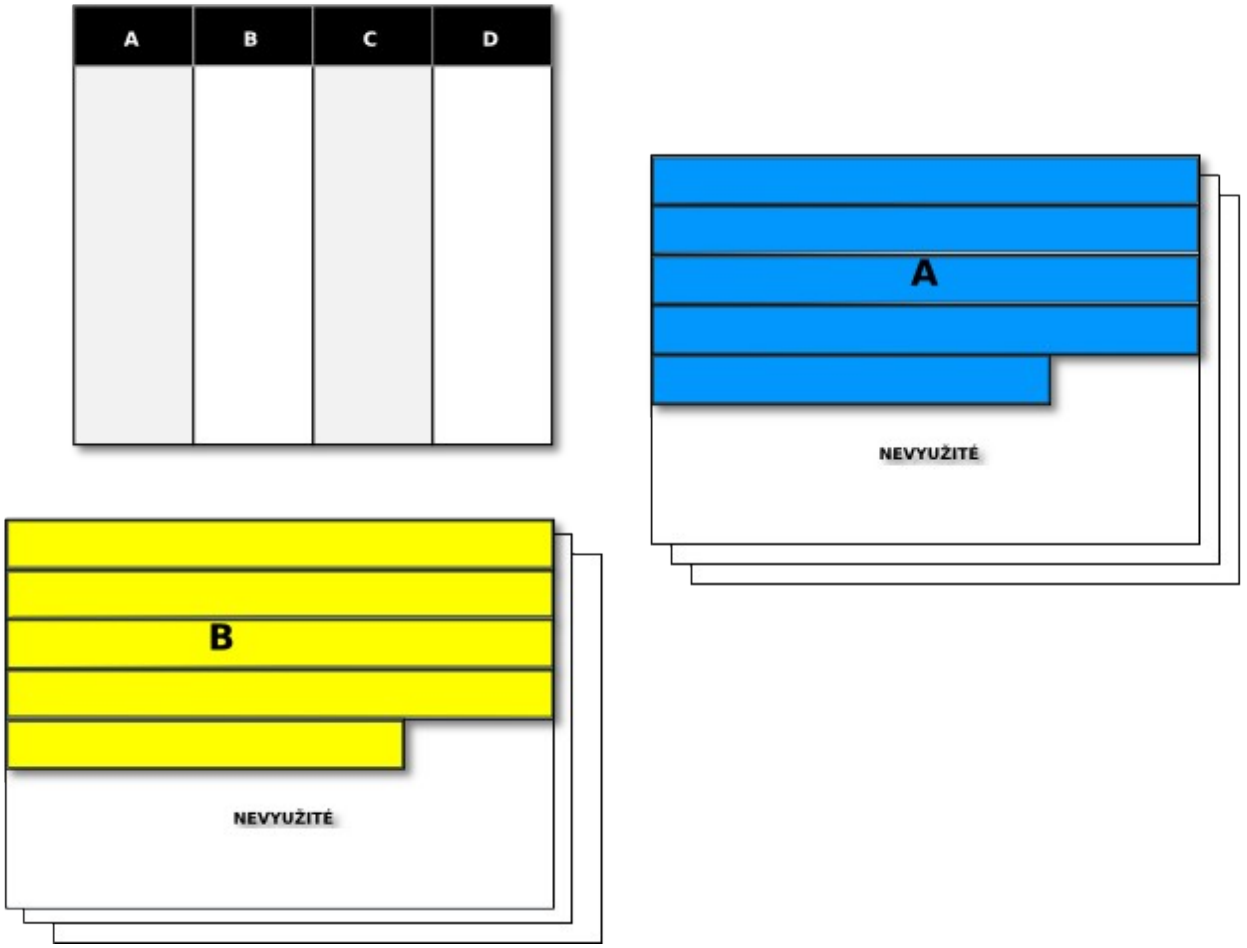
* **NOT ONLY**

Column storage databases

- optimized for OLAP
- optimized for Read Only usage
- optimized for CPU
- optimized for star schema - wide tables are ok
- data should be well compressed

- **10-100x faster** on OLAP queries
- Fast bulk load, slow individual update

Column/Page storage



MonetDB, LucidDB

- Column storage databases
- ACID databases
- SQL databases
- OLAP databases
- optimization for star schema aggregate queries

LucidDB

- **Eigenbase** framework - generic database framework
  - parser, optimizer, catalog, client in Java
  - storage, executor in C++
- Columns are effective compressed on disk
- Page level MGA - best for bulk load (32KB page)
- ETL via JDBC or SQL/MED
- UPSERT is supported - SQL 2003 MERGE
- Cluster pages (more columns)
- Hash JOIN, Hash AGG
- GPL (from 2007)

LucidDB

- Missing consistent documentation
  - some parts are undocumented
- Minimal, but friendly community
  - zero traffic mailing list, but fast response
- Not completed security
- Without RI, time zones, intervals
- Only JDBC API
- Not well integrated to Linux
- unfriendly tools (console, explain, monitoring)

MonetDB

- Mature column storage memory OLAP database
- Strings are compressed
- Fast simple column operations (CPU optimized)
- Implemented in C
- Mature code base
- Syntax of bulk import is similar to pg COPY

- Depends on system memory management
  - not optimized for databases - **needs lot of memory**

MonetDB
Fast Simple Column Operations

- **SELECT** (age - 30) * 4
  FROM data
  WHERE age > 30

- for(i =j= 0; i < n; i++)
  if (age[i] > 30)
  x1[j++] = age[i];
- for(i=j=0; i < n; i++)
  x2[j++] = x1[i] - 30;
- for(i=j=0; i < n; i++)
  x3[j++] = x2[i] * 4;

MonetDB

- Needs lot of memory
- MonetDB/X100 commercial fork (VectorWise)
  - Integrated to **Ingres**
  - **Optimized on CPU cache**
  - mix vector and pipelined executor
  - better usage of memory, faster (similar to native C++)
- Documentation is not complete, some parts are not detailed, new orientation to scientific databases (arrays)
- Integration to Linux is not complete
- Minimal, but friendly community
  - minimal traffic mailing list, but fast response
- unfriendly tools (console, explain, monitoring)

Test - tables

- Customer (30K rows, 6MB)
- Items (45K rows, 7.8MB)
- **Orderdetails** (16.7M rows, 2.9GB)
  - 17 numerics
  - 1 PK integer
  - 6 FK integers
  - 4 integers
  - 1 dates
  - ALL: 29 columns

Test - query

```
SELECT items.grouping_id as a1,
       SUM(orderdetails.val1) as m1,
       SUM(orderdetails.val2) as m2
FROM orderdetails , items, customers
WHERE
      ( customers.lid IN
(112896,112831,112878,112924,15069,98727,79888,112943,112854,3015860,
112810,112833, 112903,113012,112934,112790,112844,112941,112927,112846,
113004,112892,112858,112841,112774,112989,2886024,71,112771, ....))
   AND items.id = orderdetails.items_id
   AND customer.id = orderdetails.customer_id
GROUP BY items.grouping_id
```

Test - results

database	label	time	note
MySQL	default, MyISAM	4 min	slow nested loop over MyISAM
MySQL	memory engines	35 sec	fast nested loop memory engines copy to mem indexes on PK
PostgreSQL		26 sec	HASH JOIN, HASH AGG,
LucidDB		23 sec	
MonetDB		14 sec	depends on predicates complexity smaller list of custid -> query is faster