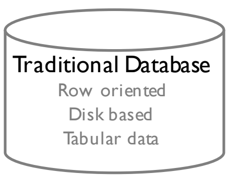
Statistics

Median: 93
Mean: 90
Stddev: 15.9

Everything else about Databases

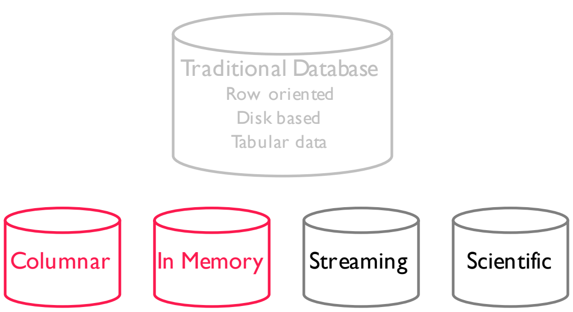
One Size Fits All



Consistent tools to access data
Widely understood
Rich, sophisticated queries, tools and features
Reliable

One Size Does Not Fits All

One Size Does Not Fits All



DBMSes in the Wild

- Classic Relational
  - $$: Oracle, IBM, Microsoft, Teradata, EMC, etc
  - Free: MySQL, PostgreSQL
- New Relational
  - In-Memory, Column-store, Streaming
- Non-traditional
  - Search (Google, Bing, Lucene), Scientific, Geographic
- NoSQL
  - Big Data: Hadoop, Spark, …
  - Key-value: Mongo, Cassandra, Memcache, Redis, …
- DBMS-as-a-Service
  - Microsoft Azure, Amazon Redshift/RDS, etc…

DBMSes in the Wild

Classic Relational
- $$: Oracle, IBM, Microsoft, Teradata, EMC, etc
- Free: MySQL, PostgreSQL

New Relational
- In-Memory, Column-store, Streaming

Non-traditional
- Search (Google, Bing, Lucene), Scientific, Geographic

NoSQL
- Big Data: Hadoop, Spark, etc
- Key-value: Mongo, Cassandra, Memcache, Redis, …

DBMS-as-a-Service
- Microsoft Azure, Amazon Redshift/RDS, etc…

Modern Database Systems

90s: The Internet:

Every application is 24x7x365

Some applications have huge numbers of users

Traditional solutions fall over

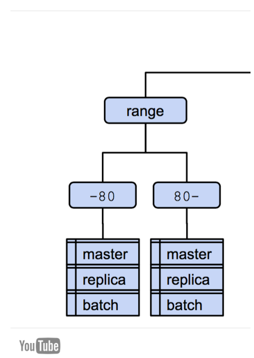
Slashdot effect, Twitter fail whale, etc etc

Solution? Sharding (partitioning)

Split one database into many
Don't access different "shards" at once

Ebay: Shard per auction
Gmail: Shard per email

Many successes: EBay, Facebook, YouTube, Salesforce



Sharding challenges

Limits supported operations
- e.g. No joins/transactions between shards

Hot/large/imbalanced shards
- e.g. Huge customers, popular pages

Manage many database instances

Transfers many challenges back to application

Google Bigtable; 2006

Sharding means you can't use relational model

Use simpler model: Key/value

Store unique keys, associated with values

key: "evan"
Value: "adjunct:w4111:ej@evanjones.ca"

Google Bigtable; 2006

Simple key/value model can be easily scaled

Split tables into tablets
Distribute tablets to multiple servers
Split tablets that have grown too big

Created the "NoSQL" movement

Other NoSQL systems

MongoDB, Cassandra: Distributed systems

Memcached (2003): Simple key/value in memory
Redis (2009): Key/value with lots of features!

Google Spanner, 2012

Globally distributed transactions
SQL interface

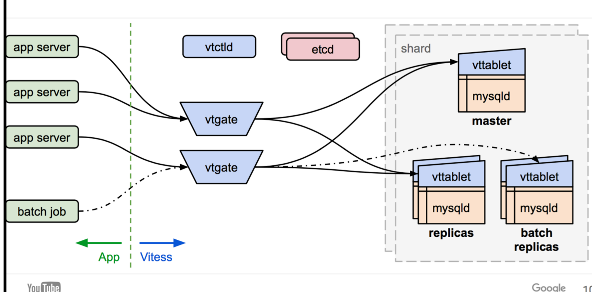
… basically a global relational database

Distributed databases today

Surprisingly few that are widely used

Many analytic database product (next slides)
NoSQL: Cassandra, MongoDB, HBase
Many startups, no clear winners

Lots of sharded MySQL/Postgres/etc with custom tools: e.g. YouTube Vitess



Why no commercial products? Hypothesis:

A "commodity" server today is big:

32 CPUs, 208 GB RAM, $1000/month cloud

Big enough for many apps
Extremely large apps: have own dev team
Sharding is painful, but does work

OLTP vs OLAP

OnLine Transaction Processing
- Interactive queries, low latency
- Small amount of data per transaction
- Modifies data

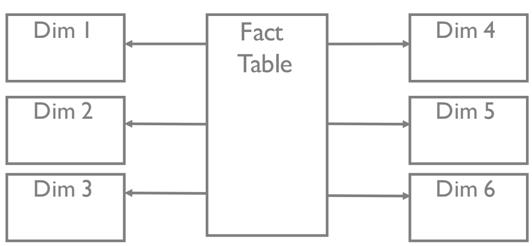
OnLine Analytical Processing
- Batch queries; "high" latency
- Aggregates, summaries
- Mostly read only

Data Warehouses

Store all historical data for future analysis
- Sales by month over past 20 years
- Clicks by youth in Texas
- Cost by product component

Most companies have something that serves this purpose

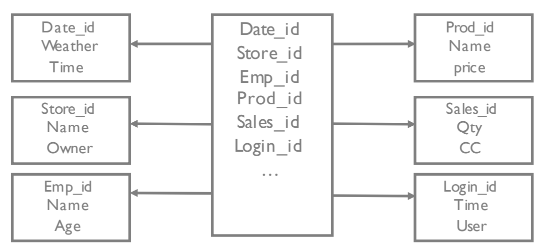
Star Schema

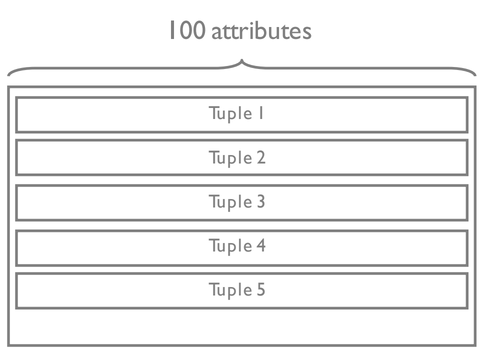


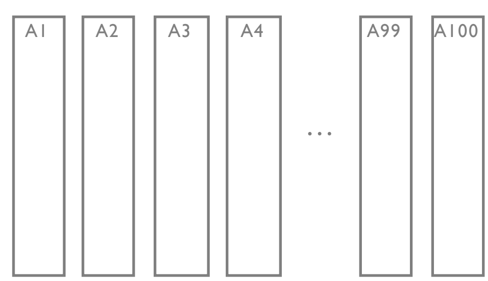
Star Schema

Fact table is "fat"; Dimensions are denormalized

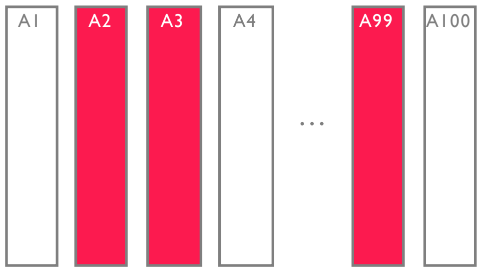
Queries access ~6 attrs







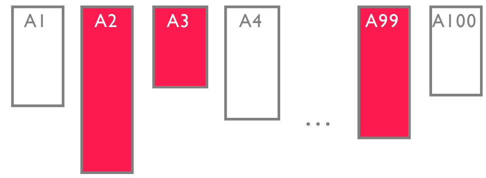
16x less data read. Unfair advantage.



16x less data read. Unfair advantage.

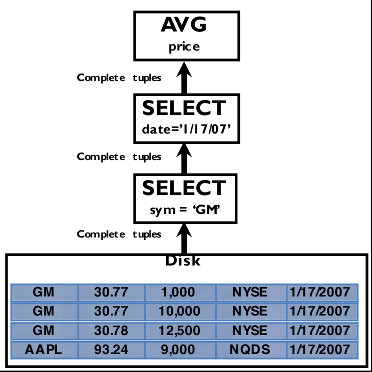
Compression better on single column

Execute on compressed data



Traditional DBMS

```
SELECT avg(price)
FROM tickstore
WHERE symbol = 'GM'
AND date = '1/17/2007'
```



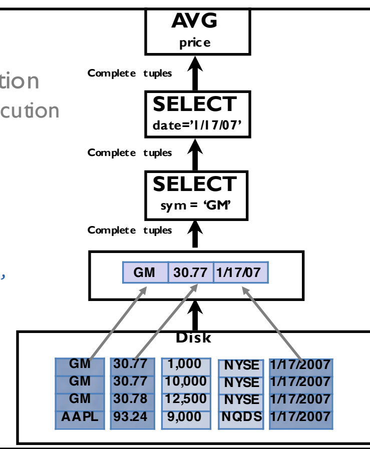
27

Naïve:

Early Materialization

Row oriented execution

```
SELECT avg(price)
FROM tickstore
WHERE symbol = 'GM'
AND date = '1/17/2007'
```



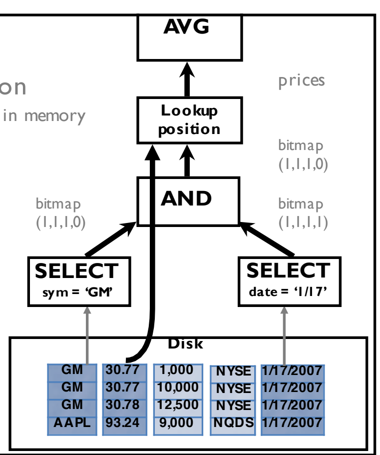
28

C-Store

Late Materialization

Much less data moving in memory

```
SELECT avg(price)
FROM tickstore
WHERE symbol = 'GM'
AND date = '1/17/2007'
```



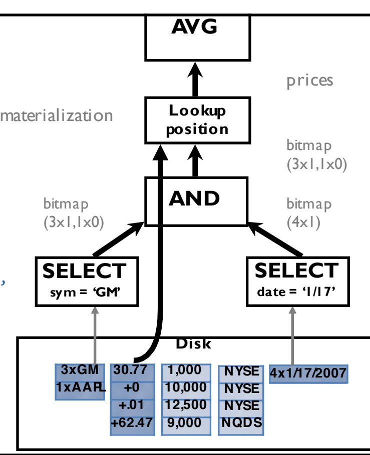
29

C-Store

Compression

Only possible w/ late materialization

```
SELECT avg(price)
FROM tickstore
WHERE symbol = 'GM'
AND date = '1/17/2007'
```



30

Column Stores

Optimized for data warehouses
Store data by attribute/column rather than row
Compression
Compressed *query plan execution*

50-100x faster than row store

Column Stores

Optimized for data warehouses
Store data by attribute/column rather than row
Compression
Compressed *query plan execution*

50-100x faster than row store
(for OLAP queries)

Many successful products

HP Vertica
Teradata
Netezza

… many others

Meanwhile at the Internet companies

Record everything!
On hundreds or thousands of computers!

… how do we do anything with it?

Google MapReduce 2004

map(records) → (key, value)
sort all keys
reduce(key, [value]) → (key2, value2)

distributed to thousands of machines

Example: requests per day

map(request log record) → (day, 1)
sort all keys
reduce(day, [1, 1, …]) → (day, sum(counts))

Apache Hadoop, January 2006

Open source clone started by Doug Cutting
Cutting was at Yahoo!, working on search

Gained significant adoption within ~2 years
Cloudera started to commercialize
Hortonworks spun out of Yahoo much later

Hadoop versus Databases "debate"

Database industry: MapReduce is a bad implementation of distributed databases

MapReduce crowd: databases can't scale

More rational perspective

MapReduce:
Relatively easy to scale
Amazing for unstructured data
Manual work for every "query"

Databases:
Setup and "loading" takes work/time
Rich queries without much effort

Today:

Google Powerdrill/BigQuery: SQL interface
Cloudera Impala: SQL interface
Facebook Presto: SQL interface

Reason? Not perfect but well understood
Queries better than implementing joins by hand!
Optimizers frequently faster than humans

In-Memory DBMSes

Transaction-oriented apps
- remove 1 unit from product
- move 5 units from org 1 to org 2
- (shopping carts, inventory)

Data stored in memory
- Disk only used for recovery

Active-active replication for fault-tolerance

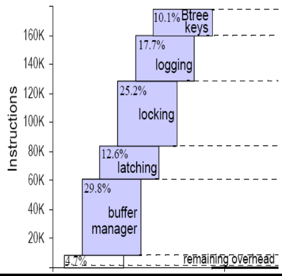
Traditional Database

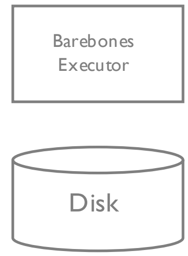
Indexes	queries go faster
Concurrency	queries go faster
Locking	serializability (go slower)
Logging	recovery (go slower)
Buffer Manager	manage pages in memory (go slower)

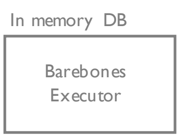
make up your mind!

Results after removing the components (in # instruction)

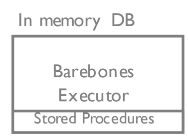
Instruction of useful work is only <2% of a memory resident DB







What about
- Parsing
- Concurrency?
- Recovery?



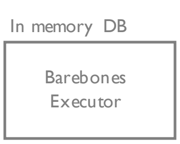
What about
- Parsing
- Concurrency?
- Recovery?

Procedure:

```
p1 = SELECT cost
     FROM fact_table
     WHERE id = ?
```

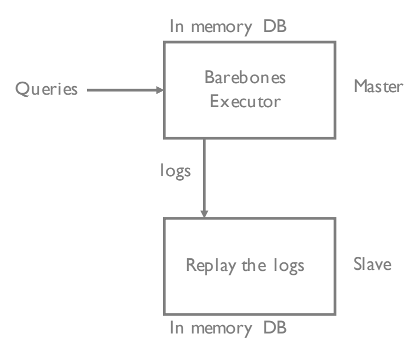
Query:

```
p1(10)
```

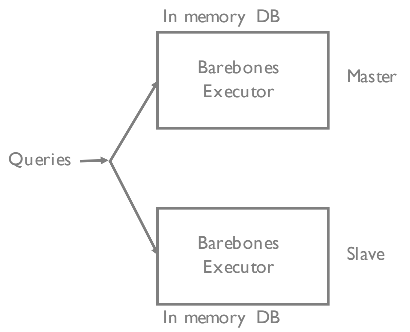


What about
- Parsing
- ~~Concurrency?~~
  - no buffer manager, no concurrency, no locks
- Recovery?

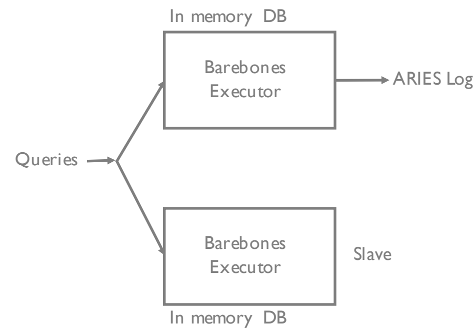
Log Shipping



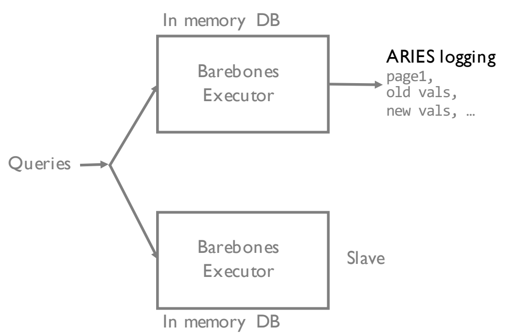
Active-Active



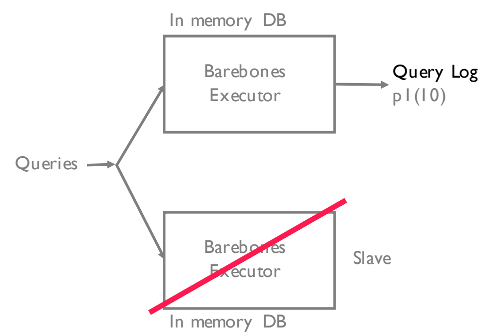
Recovery in Active-Active



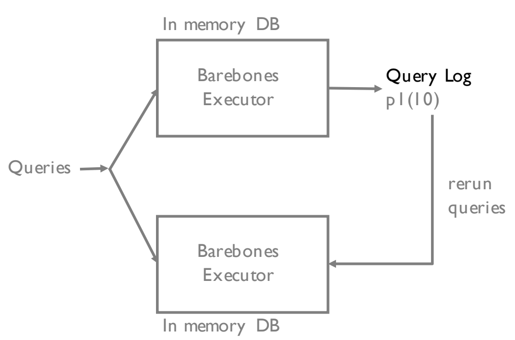
Recovery in Active-Active



Recovery in Active-Active



Recovery in Active-Active



Databases as a Service

Amazon:
- Redshift (columnar)
- RDS (traditional)
- DynamoDB (key/value)

Google:
- BigTable (key/value)
- Cloud SQL (traditional)
- BigQuery (columnar)

Microsoft:
- SQL Azure
- DocumentDB (key/val)

Database Service Challenges

Legal rules:

Data residency	(EU, others)
Privacy/Compliance	(e.g. HIPAA for health)
Security/Compliance	(e.g. PCI for credit cards)

Data gravity:

- Large data is hard to move
- Latency across the Internet can be bad

Dave Patterson interviews Jim Gray

DP "Sneaker net" was when you used your sneakers to transport data?

JG In the old days, sneaker net was the notion that you would pull out floppy disks, run across the room in your sneakers, and plug the floppy into another machine. This is just TeraScale SneakerNet. You write your terabytes onto this thing and ship it out to your pals. Some of our pals are extremely well connected—they are part of Internet 2, Virtual Business Networks (VBNs), and the Next Generation Internet (NGI). Even so, it takes them a long time to copy a gigabyte. Copy a terabyte? It takes them a very, very long time across the networks they have.

http://queue.acm.org/detail.cfm?id=864078

home › tech

Amazon.com

Amazon's Snowmobile will let you upload stuff by the truckload – literally

The truck carries a shipping container holding a mobile data centre which can store up to 100 petabytes

Alex Hern

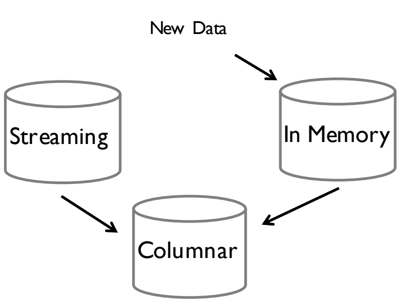
@alexhern

Monday 5 December 2016 04.52 EST

The AWS snowmobile in action. Photograph: AWS

One Size Does Not Fits All

"Modern" Systems: Connected!



Other Aspects of Database Research

- Domain specific data management
- Data quality/cleaning
- Database usability
- Crowdsourced Databases
- Information extraction and mining
- Query compilation
- Data visualization and exploration

Societal Data

Health	Fake data
Investigative Journalism	Biased data
Recommendations	Incorrect data
Politics	Mixed data
Surveillance	
Identity	

Data will be crucial to
how we live
as individuals and as a society

Go forth and make a
(positive) difference!