Database Systems

RDBMSs

The NoSQL movement

(R)DBMSs ”today”

DMBS	*Approx. market share*
(Microsoft Access)	
Oracle	40-50%
IBM DB2	20-30%
Microsoft SQL Server	~15%
Sybase	3%
(…)	
MySQL	1%
PostgreSQL	0.5%

Oracle DB

- First commercially sold database engine
  - V2 release 1979.
- HUGE market share historically (~80%)
  - Still clear market leader.
- Standard incompliant
  - Might makes right?

IBM DB2

- First (in-house) SQL database
  - 1974, DB2 released commercially 1982.

> *"At the time IBM didn't believe in the potential of Codd's ideas, leaving the implementation to a group of programmers not under Codd's supervision, who violated several fundamentals of Codd's relational model; the result was Structured English QUEry Language or SEQUEL."*
>
> - Wikipedia article on DB2

- Near-monopoly on "mainframes"

I believe this explains a lot…

- Towards Oracle-compliant(!)
  - IBM and Oracle cooperate on e.g. tools.

Microsoft SQL Server

- First release 1989.
- Market leading on Windows application platforms.
- Code base originally by Sybase, aquired by Microsoft 1991.

MySQL

- Open Source – but owned by Oracle since 2010
- Historically: fast but feature-poor
  - Subqueries(!) added in ”recent” release
  - FK constraints only with Oracle’s (InnoDB) backend
  - Not optimized for joins
- Still missing features (but getting closer)
  - No CHECK constraints (including assertions)
  - No sequencing (WITH)
  - No INSTEAD OF triggers on views.

- Big on the web: used by Wikipedia, Facebook, …
  - Early support in PHP helped boost

PostgreSQL

- Open Source – community development
- Historically: full-featured but (relatively) slow
- Much faster today – and optimized for complex tasks
  - Efficient support for joins
- Almost standard-compliant
  - Full constraint support – except assertions!
  - Full ACID transactions
  - Sequencing (WITH)

- Prominent users: Yahoo, MySpace, Skype, …

SQLite

- Small-ish C library
  - Embedded into application means no communication overhead
- Stores all data as a single file on host platform.
  - Not as fast as dedicated systems.
- Not full-featured.

- Prominent users: Firefox, Chrome, Opera, Skype

Feature Comparison - Queries

	OUTER JOIN	INTERSECT/EXCEPT	WITH
Oracle	Yes	Yes (MINUS)	Yes
IBM DB2	Yes	Yes	Yes
MS SQL Server	Yes	Yes	Yes
MySQL	Yes	No	No
PostgreSQL	Yes	Yes	Yes
SQLite	LEFT	Yes	No

Feature Comparison - DDL

	FK REF	CHECK	ASSERTION	TRIGGER	PROC/FUN
Oracle	Yes	Yes	No	Yes	Yes
IBM DB2	Yes	Yes	No	Yes	Yes
MS SQL Server	Yes	Yes	No	Yes	Yes
MySQL	InnoDB	No	No	Yes	Yes
PostgreSQL	Yes	Yes	No	Yes	Yes
SQLite	Yes	No	No	Yes	No

Assertions (schema-level constraints) are incompatible with most existing research on database optimization. No existing implementation supports them (except by using triggers to mimic).

Feature Comparison - Misc

	OS Support	Programming support
Oracle	Win, Mac, Unix, z/OS	PL/SQL
IBM DB2	Win, Mac, Unix, z/OS, iOS	SQL/PSM
MS SQL Server	Windows	T-SQL
MySQL	Win, Mac, Unix, z/OS, Symbian	SQL/PSM
PostgreSQL	Win, Mac, Unix, Android	PL/pgSQL, SQL/PSM, lots more!
SQLite	Any	N/A

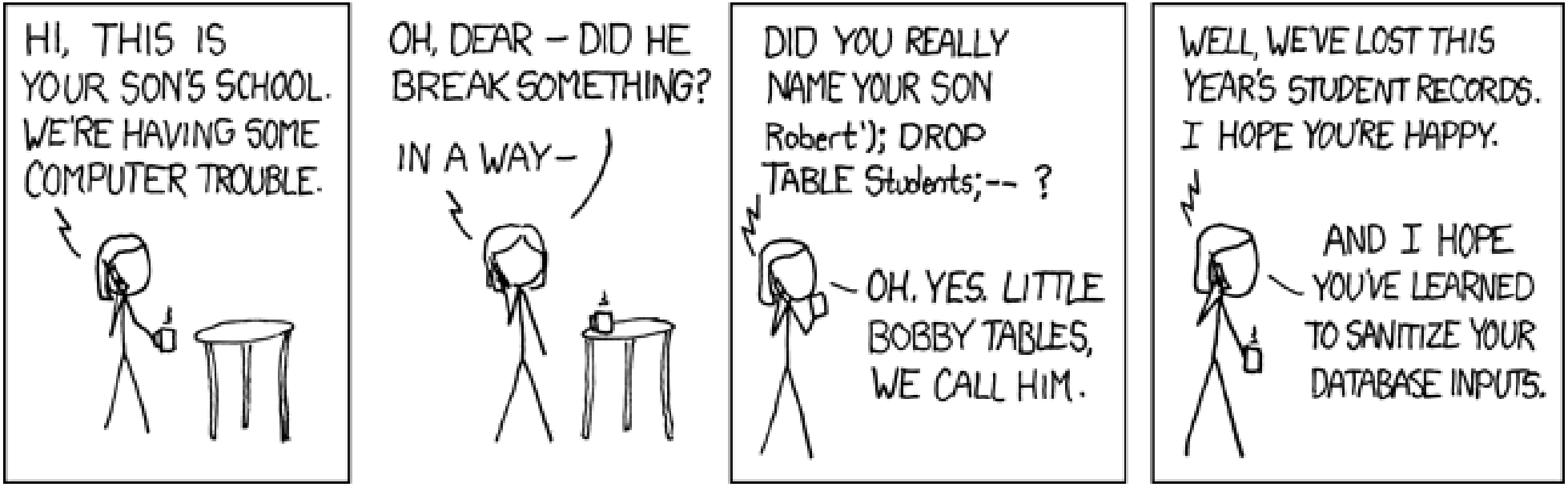
PostgreSQL allows procedures to be written in a wide variety of languages, including Java, Python, Tcl, Perl, PHP…

SQLite is a C library, so can be used with any language that has C FFI support.

Beyond SQL?

- SQL was first developed around 1970
  - Contemporaries: Forth, PL/I, Pascal, C, SmallTalk, ML, …
- With one prominent exception – C – these have all been succeded by newer, cooler languages.
- Has the time finally come for SQL as well?

SQL injection attacks



http://xkcd.com/327/

The possibility for SQL injection attacks has lead development away from literal SQL, towards higher-level interfaces, tools and libraries.

The world is a-changin’

- ”The Claremont Report”, 2008
  - Big Data is coming big-time
    - Social networks, e-science, Web 2.0, streaming media, …
  - Data IS the business
    - Efficient data management is no longer just a cost reduction – it’s a selling point!
  - Rapidly expanding user base
    - … means rapidly expanding user needs
  - Architectural shifts in computing
    - Storage hardware is still improving exponentially
    - Data networks, cloud computing, …

Examples of database sizes

- Digg: 3 TB – just to store the up/down votes
- Facebook: 50 TB – for the private messaging feature
- eBay: 2 PB data overall (2 000 000 GB)

RDBMS weakness

- RDBMSs typically handle ”massive” amounts of data in complex domains, with frequent small read/writes.
  - The archetypical RDBMS serves a bank.

- Data-intensive applications don’t fit this pattern:
  - MASSIVE+++ amounts of data (e.g. eBay)
  - Super-fast indexing of documents (e.g. Google)
  - Serving pages on high-traffic websites (e.g. Facebook)
  - Streaming media (e.g. Spotify)

Column-oriented databases

- Typical RDBMSs are row-oriented
  - All data associated with a particular key is stored close together.
  - Allows *horizontal partitioning (sharding)*
    - Different rows stored on different distributed nodes.
- Column-oriented (R)DBMSs
  - All data in a given *column* is stored close together.
  - Allows *vertical partitioning (normalization)*
    - Different columns stored on different nodes.
  - Fast computation of aggregations, e.g. *data mining.*

The ”NoSQL” movement

```
SELECT fun, profit
FROM Real_World
WHERE relational = FALSE;
```

- NoSQL (”Not only SQL”)
  - Originally the name of an RDBMS with a different interface language.
  - Nowadays a term that encompasses a wide variety of non-relational DBMSs (”NoRel”?).

Non-relational databases

- MapReduce framework
  - Google originally; Hadoop (Apache), …
- Key-Value stores
  - BigTable (Google), Cassandra (Apache), …
- Document stores
  - CouchDB, MongoDB, SimpleDB, …
- Graph databases
  - Neo4j, FlockDB, …
- Semi-structured databases
  - (Native) XML databases, …

MapReduce

- No data model – all data stored in files
- Operations supplied by user:
  - Reader :: file → [input record]
  - Map :: input record → <key, value>
  - Reduce :: <key, [value]> → [output record]
  - Writer :: [output record] → file
- Everything else done behind the scenes:
  - Consistency, atomicity, distribution and parallelism, ”glue”
- Optimized for broad data analytics
  - Running simple queries over all data at once

MapReduce implementations

- The ”secret” behind Google’s success
  - Still going strong.
- Hadoop (Apache)
  - Open Source implementation of the MapReduce framework
  - Used by Ebay, Amazon, Last.fm, LinkedIn, Twitter, Yahoo, Facebook internal logs (~15PB), …

Key-Value Stores

- Key-Value stores is a fancy name for persistant maps (associative arrays):

```
void Put(string key, byte[] data);
byte[] Get(string key);
void Remove(string key);
```

- Extremely simple interface – extremely complex implementations.

Key-Value Stores

- Built for extreme efficiency, scalability and fault tolerance
  - Records distributed to different nodes based on key
  - Replication of data to several nodes
- Sacrifice consistency and querying power
  - Single-record transactions
  - Not good for ”joins”
  - ”Eventual consistency” between nodes
    - AKA: ”why does Facebook tell me I have a notification when there isn’t anything new when I click the icon??!?”

Key-Value store implementations

- BigTable (Google)
  - Sparse, distributed, multi-dimensional sorted map
  - Proprietary – used in Google's internals: Google Reader, Google Maps, YouTube, Blogger, …
- Cassandra (Apache)
  - Originally Facebook's PM database – now Open Source (Apache top-level project)
  - Used by Netflix, Digg, Reddit, Spotify, …

Semi-structured data (SSD)

- More flexible than the relational model.
  - The type of each ”entity” is its own business.
  - Labels indicate meanings of substructures.
- Semi-structured: it is structured, but not everything is structured the same way!

- Support for XML and XQuery in e.g. Oracle, DB2, SQL Server.

- Special case: Document databases

Document stores

- Roughly: Key-Value stores where the values are ”documents”
  - XML, JSON, mixed semistructured data sets
- Typically incorporate a query language for the document type.
  - See previous lecture for discussion on XML querying.

Document store implementations

- MongoDB
  - Name short for ”Hu**mongo**us”
  - Open source – owned by 10gen
  - JSON(-like) semi-structured storage
  - JavaScript query language
  - Supports MapReduce for aggregations

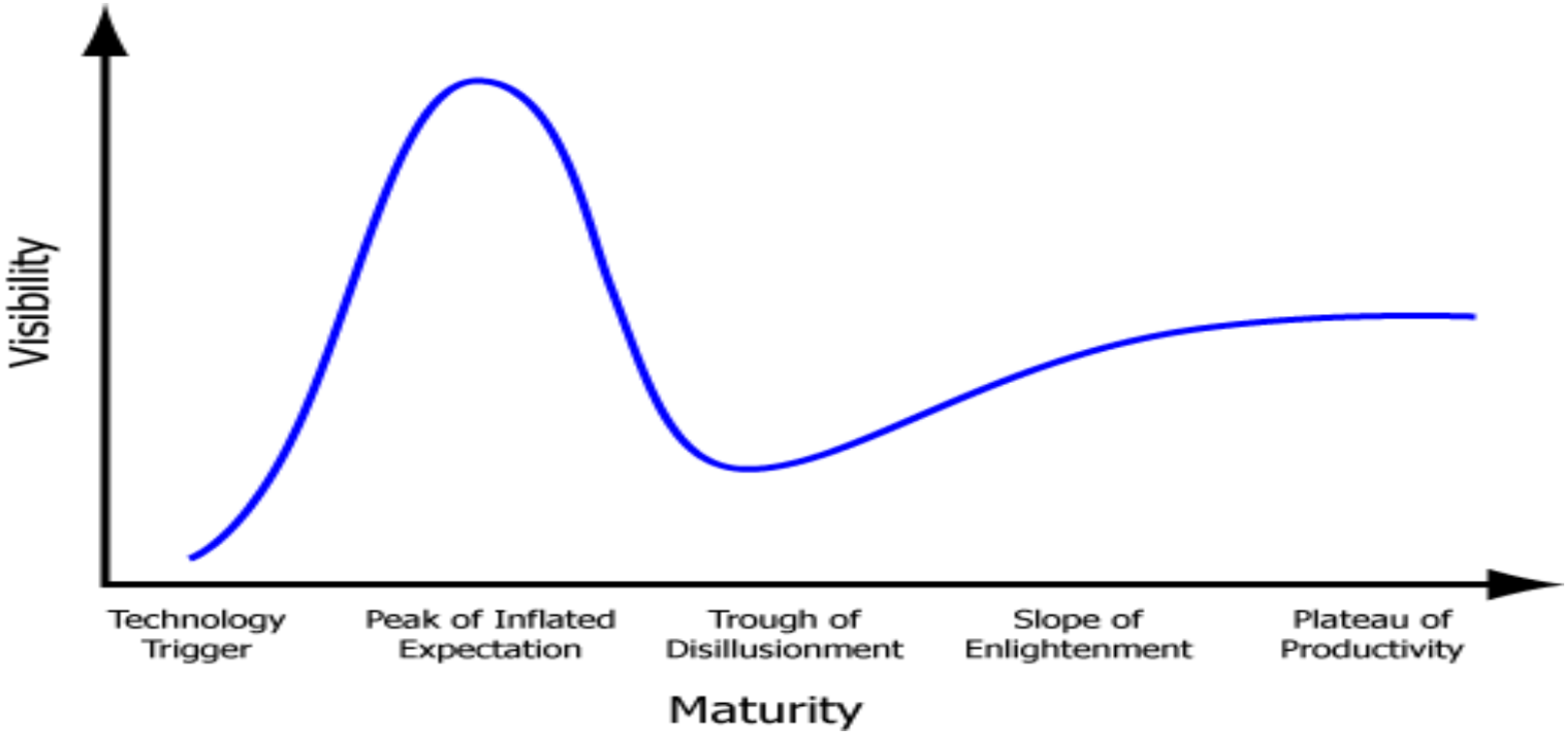
- Apache CouchDB

Graph Databases

- Data modeled in a graph structure
  - Nodes = ”entities”
  - Properties = ”tags”, attribute values
  - Edges connect
    - Nodes to nodes (relationships)
    - Nodes to properties (attributes)
- Fast access to associative data sets
  - All entities that share a common property
  - Computing association paths

- Used by Twitter (FlockDB) to store user relations, …

NoSQL – a hype?



- NoSQL is not ”the right choice” just because it’s new!
- Relational DBMSs still rule at what they were first designed for: efficient access to large amounts of data in complex domains. That’s still the vast majority!

NoSQL summary

- NoSQL = ”Not only SQL”
- Different data models optimized for different tasks
  - MapReduce, Key-Value stores, Document stores, Graph databases, …
- Typically:

  \+ efficiency, scalability, flexibility, fault tolerance

  \- (no) query language, (less) consistency

Next Lecture

Course summary

Assignment retrospective

Exam information