COMP5338 – Advanced Data Models

Week 1: Big Data and NoSQL Database

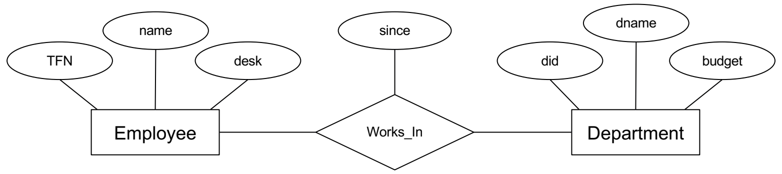
Dr. Ying Zhou
School of Computer Science

Outline

- **The Value and limitations of SQL**
- Typical Scale-Out Options
- NoSQL Database Systems

The Relational Model of Data

- Entity-Relationship (ER) Data Model describes data as
  - Entities – distinct objects in the domain
  - Relationships – between two or more entities
- Entities – described using a set of attributes
- Relationships – have attributes too
- Used for Conceptual Database Design
  - Translated to final database implementation



The Rational RDBMS

- Commercial vendors: Oracle, IBM, Microsoft, …
- Open source systems: MySQL, PostgreSQL, …
- Common features
  - Disk-oriented storage;
  - Table stored row-by-row on disk
  - B-trees as indexing mechanism
  - Dynamic locking as the concurrency-control mechanism
  - A write-ahead log, or WAL for crash recovery
  - SQL as the access language
  - A "row-oriented" query optimizer and executor

http://cacm.acm.org/magazines/2011/6/108651-10-rules-for-scalable-performance-in-simple-operation-datastores/fulltext

The Value of Relational Databases

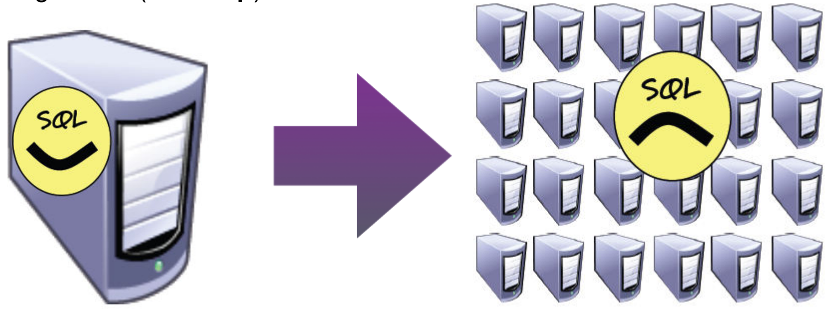
- Store persistent data
  - Storing large amounts of data on disks, while allowing applications to grab the bits they need through queries
- Application Integration
  - Many applications in an enterprise need to share information, which might happen at the database level
- Concurrency Control
  - Database provide transactions to ensure consistent interaction when many users access the same information at the same time
- Mostly Standard
  - Relational model is widely used and understood.
  - SQL is the standard language.
- Reporting
  - SQL's simple data model and standardization has made it a foundation for many reporting tools

http://martinfowler.com/articles/nosql-intro.pdf

The Scaling Problem of SQL

Relational databases are designed to run on a single machine, so to scale, you need buy a bigger machine or increase capacity of existing server (**scale up**)

But it's cheaper and more effective to **scale out** by buying lots of machines.



http://martinfowler.com/articles/nosql-intro.pdf

The Fixed Schema Problem of SQL

- In a relational database
  - Table structure are <u>predefined</u>
  - Tables are related with relationships, which are <u>predefined</u> as well
- Schema evolution in RDBMS has large impact on queries and applications
- Example
  - MediaWiki had been through 171 schema versions between April 2003 and November 2007.
    - MySQL backend
    - ~ 34 tables, ~242 columns, ~700GB in wikipedia (note: 2008 data)
  - Schema change has big impact on queries
    - Large number of queries could fail due to schema change.

http://yellowstone.cs.ucla.edu/schema-evolution/documents/curino-schema-evolution.pdf

World of Big Data

- Big Data are high-**volume**, high-**velocity**, and/or high-**variety** information assets that require new forms of processing to enable enhanced decision making, insight discovery and process optimization.
  [Gartner 2012 report]
- Leaders in database research community identified "big data as a defining challenging of our time" in a 2013 meeting.
- Three major trends behind big data
  - "It has become **<u>much cheaper to generate</u>** a wide variety of data, due to inexpensive storage, sensors, smart devices, social software, multiplayer games and the Internet of Things, …"
  - "It has become **<u>much cheaper to process</u>** large amount data, due to advances in multicore CPUs, solid state storage, inexpensive cloud computing, and open source software"
  - "**<u>data management has become democratized</u>**. The process of generating, processing, and consuming data is no longer just for database professionals. Decision makers, domain scientists, .. and everyday consumers now routinely do it"

D. Abadi, et al. "The Beckman report on database research". *Commun. ACM* 59, 2 (January 2016), 92-99

Schema Change is Unavoidable

- News paper site example
  - Early days, for each news article, we may only record the following information
    - Title, author, publishing date and time, actual content
  - Gradually we may want to record more about an article
    - Keywords, views, geotags, comments, who “favoured” it, who emailed it, who twittered it … etc
- Evolution of an application is inevitable
  - Accept it, incorporate it in the long-term plan for the system
  - Pick a system that allows schema evolution or have a strategy

Outline

- The Value and limitations of SQL
- **Typical Scale Out Options**
- NoSQL Storage Systems

When Scalability Becomes an Issue

- “***Scalability*** is the capability of a system, network, or process to handle a growing amount of work, or its potential to be enlarged in order to accommodate that growth.”

  [wikipedia: https://en.wikipedia.org/wiki/Scalability]
- In database context, the need for scaling occurs when the **size of the database** and/or the **traffic against it** grows to the point of crossing an optimal level of performance
  - Scale up
  - Scale out

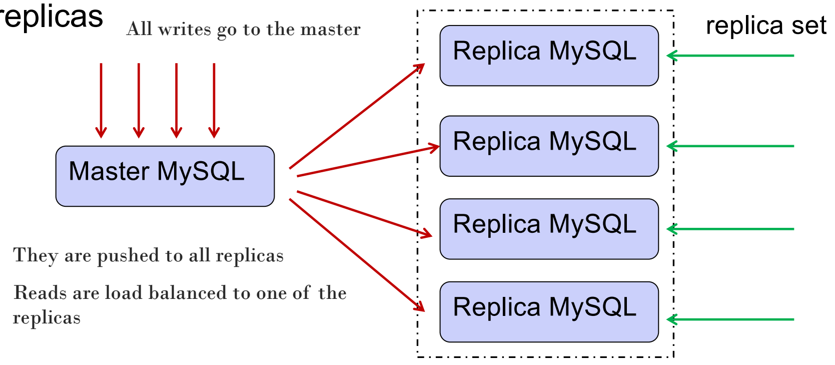
Scalability Scenario I

- Persistent Storage Requirements
  - Medium data size (can fit in one server)
  - Typical query workload consists of **large number of read request** and **relatively low number of write request**
- Example: wikipedia
  - Only article meta data such as article revision history, articles relations, user account and setting are stored in core relational database system (MySQL)
  - Article text and images are stored separately
- Key challenge
  - Scale to maintain reasonable **read latency**

http://www.nedworks.org/~mark/presentations/san/Wikimedia%20architecture.pdf

Scalability Scenario I -- Solution

- Master-Slave Replication
  - Adopted by many companies
  - Also a typical approach to ensure durability
- Example: Wikipedia has one Master database and many replicas



All writes go to the master
Master MySQL
Replica MySQL
Replica MySQL
Replica MySQL
Replica MySQL
replica set
They are pushed to all replicas
Reads are load balanced to one of the replicas

http://www.nedworks.org/~mark/presentations/san/Wikimedia%20architecture.pdf

Scalability Scenario I -- Implications

- When the master dies
  - One of the replica can be elected as the new master
- Some read may return old data if the latest value has not been pushed from the master
  - It is possible to let Master handle read request for data requiring strong consistency
- Relatively easy to setup in most RDBMS

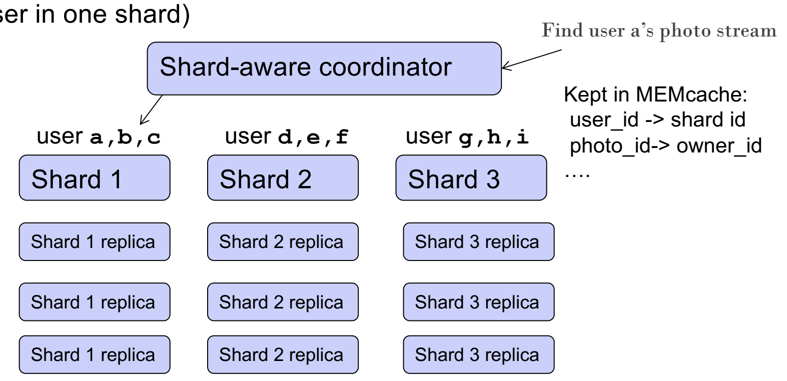
Scalability Scenario II

- Persistent Storage Requirements
  - Medium or large data size (cannot fit in one server)
  - Typical query workload consists of **large number of read request** and **large number of write request**
- Example: **flickr.com**
  - Heavy write traffic: upload photos, adding tags, adding favourite, …
  - Over 400,000 photos being added every day. (note: 2007 data)
  - More than 4 billion queries per day. (note: 2007 data)
  - Uses MySQL as backend storage
- Key challenge
  - Scale to maintain both **read and write latency**

http://highscalability.com/flickr-architecture
http://mysqldba.blogspot.com.au/2008/04/mysql-uc-2007-presentation-file.html

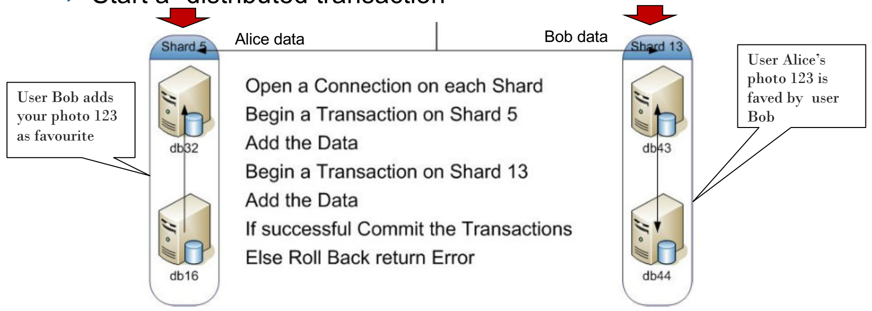
Scalability Scenario II -- Solution

- Database Sharding
  - The process of slicing a database across multiple machines
  - Most likely horizontally (e.g., store all data related with a particular user in one shard)



Scalability Scenario II – Flickr Example

- User Bob adds User Alice’s photo 123 as “favourite”
  - Pulls the photo (123) owner’s account from cache (“Alice”), to get the shard location
    - SHARD-5
  - Pulls Bob’s information from cache, to get Bob’s shard location
    - SHARD-13
  - Start a “distributed transaction”


Alice data
Bob data
Shard 5
Shard 13
db32
db16
db43
db44
User Bob adds your photo 123 as favourite
User Alice’s photo 123 is faved by user Bob

Open a Connection on each Shard
Begin a Transaction on Shard 5
Add the Data
Begin a Transaction on Shard 13
Add the Data
If successful Commit the Transactions
Else Roll Back return Error

http://mysqldba.blogspot.com.au/2008/04/mysql-uc-2007-presentation-file.html

Scalability Scenario II -- Implications

- Data have to be de-normalized
  - E.g. in the previous example, the “fav” relation is stored in both Bob and Alice’s record.
  - Join is too expensive when data are sharded
  - Sometimes join can’t be avoided, e.g. building friends network
- Re-balancing or Re-Sharding is hard
  - What to do when data do not fit in one shard
- Deciding on a partition factor/plan is hard
  - May generate hotspots
    - See the twitter example on next slides
- Sharding is largely managed outside RDBMS
  - Recent version of RDBMS may provide limited support for sharding

Scalability Scenario II – Twitter Example

- Twitter’s problem
  - To store 250 million tweets a day using MySQL
- Twitter’s original Tweet Store:
  - Sharding based on time
  - Range partition (timestamp range)
  - The benefits: shards are filled one by one sequentially
  - The downsides: very unbalanced traffic
    - Shards with old tweets do not get any traffic
- Twitter’s new Tweet Store:
  - Sharding based on random partition (id based)
  - A set of in-house systems to manage shards on top of MySQL

http://highscalability.com/blog/2011/12/19/how-twitter-stores-250-million-tweets-a-day-using-mysql.html

Outline

- The Value and limitations of SQL
- Handling Scalability
- **NoSQL Storage Systems**

The Coming of NoSQL Storage Systems

- There is no standard definition of NoSQL, the term came up during a workshop on 2009 with presentations from **`Voldemort`**, **`Cassandra`**, **`Dynomite`**, **`HBase`**, **`Hypertable`**, **`CouchDB`** and **`MongoDB`**
  - Means "Not Only SQL"
- Typical features
  - They don't use the relational data model and thus don't use the SQL language
  - They don't have fixed schema, allowing you to store any data in any record
  - Many of them are designed to run on a cluster
    - Manage "sharding", fault-tolerance, etc. efficiently
  - Many of them can be integrated with big data processing framework such as MapReduce

http://martinfowler.com/articles/nosql-intro.pdf

NoSQL Ecosystem -- Scalability

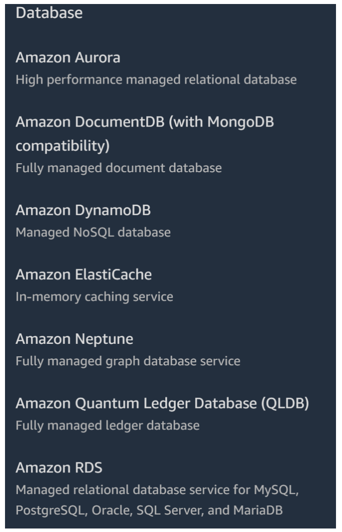
- Distributed NoSQL systems
  - Designed to run on a cluster
  - Support automatically partitioning data across multiple machines
  - Machines can add or leave a *running* cluster
  - Handles failover, fault-tolerance
- Example Distributed NoSQL systems
  - HBase, Cassandra,…
- Non-distributed NoSQL systems
  - Designed to run on a single machine
  - Some has limited support for replication and sharding
  - Schema-less and "object" support
- Example
  - MongoDB, Neo4j, etc..

http://www.rackspace.com/blog/nosql-ecosystem/

NoSQL Ecosystem – Data Models

- Document store
  - Has "table" like concept
  - Each "record" in a "table" is a semi-structured document
  - Examples: MongoDB, CouchDB
- Key Value Store
  - Inspired by Amazon's Dynamo storage
  - The overall storage is structured like a big hash table
  - May or may not have a "table" concept
  - Redis, Memcached, Voldemort, *S3*, *Cassandra*, *DynamnoDB*
- Graph model
  - Storage is organized as nodes and edges
  - Neo4j
- Column based store
  - Inspired by Google's Bigtable structure
  - Has "table" like concept
  - Storage is organized around column family instead of "row"
  - Examples: Hbase, Cassandra

AWS Database Services


Database

Amazon Aurora
High performance managed relational database

Amazon DocumentDB (with MongoDB compatibility)
Fully managed document database

Amazon DynamoDB
Managed NoSQL database

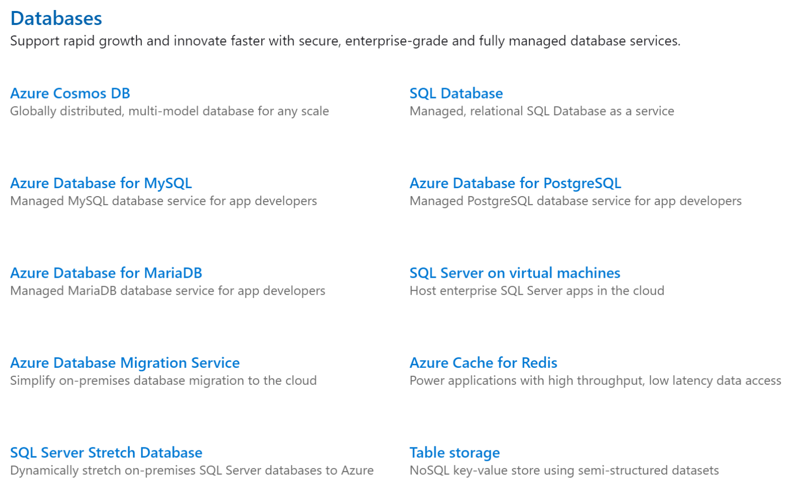
Amazon ElastiCache
In-memory caching service

Amazon Neptune
Fully managed graph database service

Amazon Quantum Ledger Database (QLDB)
Fully managed ledger database

Amazon RDS
Managed relational database service for MySQL, PostgreSQL, Oracle, SQL Server, and MariaDB

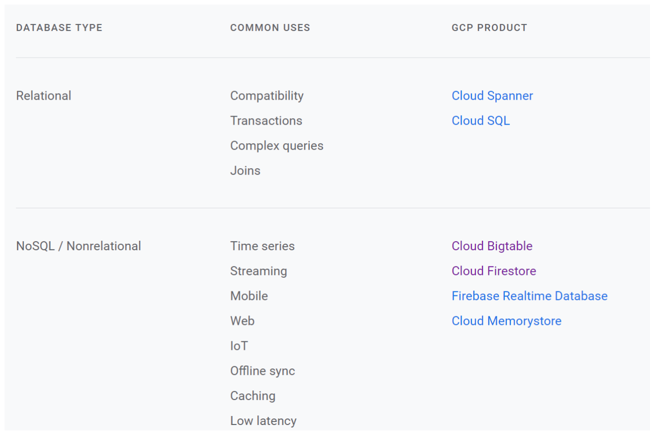
Microsoft Azure Database Service



Databases
Support rapid growth and innovate faster with secure, enterprise-grade and fully managed database services.

Azure Cosmos DB Globally distributed, multi-model database for any scale	**SQL Database** Managed, relational SQL Database as a service
Azure Database for MySQL Managed MySQL database service for app developers	**Azure Database for PostgreSQL** Managed PostgreSQL database service for app developers
Azure Database for MariaDB Managed MariaDB database service for app developers	**SQL Server on virtual machines** Host enterprise SQL Server apps in the cloud
Azure Database Migration Service Simplify on-premises database migration to the cloud	**Azure Cache for Redis** Power applications with high throughput, low latency data access
SQL Server Stretch Database Dynamically stretch on-premises SQL Server databases to Azure	**Table storage** NoSQL key-value store using semi-structured datasets

Google Cloud Platform Database Services

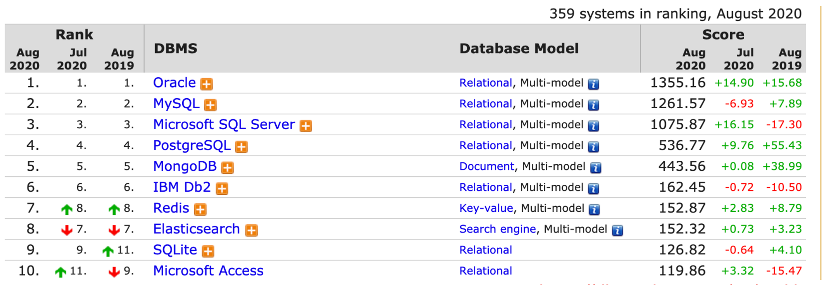


DATABASE TYPE	COMMON USES	GCP PRODUCT
Relational	Compatibility Transactions Complex queries Joins	Cloud Spanner Cloud SQL
NoSQL / Nonrelational	Time series Streaming Mobile Web IoT Offline sync Caching Low latency	Cloud Bigtable Cloud Firestore Firebase Realtime Database Cloud Memorystore

The reality

- **The rise of NoSQL databases marks the end of the era of relational database dominance**
- But NoSQL databases will not become the new dominators. Relational will still be popular, and used in the majority of situations. They, however, will no longer be the automatic choice.

http://martinfowler.com/nosql.html



359 systems in ranking, August 2020

Rank			DBMS	Database Model	Score		
Aug 2020	Jul 2020	Aug 2019			Aug 2020	Jul 2020	Aug 2019
1.	1.	1.	Oracle	Relational, Multi-model	1355.16	+14.90	+15.68
2.	2.	2.	MySQL	Relational, Multi-model	1261.57	-6.93	+7.89
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	1075.87	+16.15	-17.30
4.	4.	4.	PostgreSQL	Relational, Multi-model	536.77	+9.76	+55.43
5.	5.	5.	MongoDB	Document, Multi-model	443.56	+0.08	+38.99
6.	6.	6.	IBM Db2	Relational, Multi-model	162.45	-0.72	-10.50
7.	↑ 8.	↑ 8.	Redis	Key-value, Multi-model	152.87	+2.83	+8.79
8.	↓ 7.	↓ 7.	Elasticsearch	Search engine, Multi-model	152.32	+0.73	+3.23
9.	9.	↑ 11.	SQLite	Relational	126.82	-0.64	+4.10
10.	↑ 11.	↓ 9.	Microsoft Access	Relational	119.86	+3.32	-15.47

https://db-engines.com/en/ranking

Social Commerce Application Example

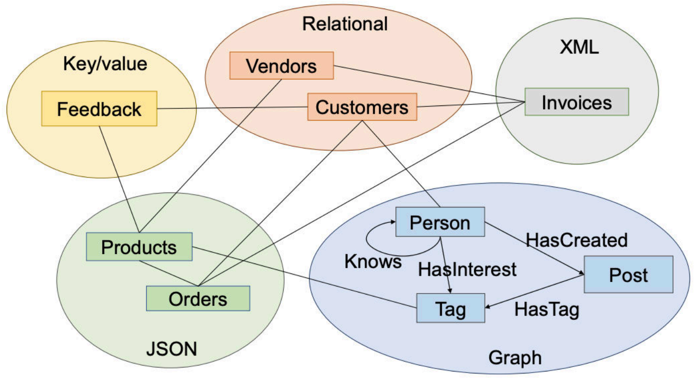


Fig. 1: Unibench Data Model

UniBench: A Benchmark for Multi-Model Database Management Systems"

Polyglot Persistence vs Multi-Model Database

- Polyglot Persistence
  - Use multiple data storage technologies, chosen based upon the way data is being used by individual applications.
- Multi-Model Database
  - A single database that supports multiple data model abstraction
  - It usually has a base or primary model with extensions to support other models
  - Object relational database management system is an early version of multi-model database
  - Typical models supported include spatial model, graph model, time series model, etc..
  - Separate abstract model (high level model) from physical model (low level representation)
  - Provide special supports for those extended models

References

- Paramod J. Sadalage and Martin Fowler, “NoSQL Distilled”, Addison-Wesley Professional; 1 edition (August 18, 2012)
  - Chapter 1
- Daniel Abadi, Rakesh Agrawal, et, al 2016. The Beckman report on database research. *Commun. ACM* 59, 2 (January 2016), 92-99. DOI=http://dx.doi.org/10.1145/2845915
- Curino, Carlo A., Hyun J. Moon, Letizia Tanca, and Carlo Zaniolo. "Schema Evolution In Wikipedia." ICEIS, 2008.
  - http://yellowstone.cs.ucla.edu/schema-evolution/documents/curino-schema-evolution.pdf
- High Scalability post: “How Twitter Stores 250 Million Tweets A Day Using MySQL”, 2011
  - http://highscalability.com/blog/2011/12/19/how-twitter-stores-250-million-tweets-a-day-using-mysql.html
- Dathan Pattishall, “Federation at Flickr: Doing Billions of Queries per Day”, 2007
  - http://www.scribd.com/doc/2592098/DVPmysqlucFederation-at-Flickr-Doing-Billions-of-Queries-Per-Day
- Chao Zhang, Jiaheng Lu, Pengfei Xu, Yuxing Chen. "UniBench: A Benchmark for Multi-Model Database Management Systems" (PDF). *TPCTC 2018*.