WOW
CPSC350

- "relational schemas"
- "table normalization"
- "practical use of relational algebraic operators"
- "tuple relational calculus"
- "and their expression in a declarative query language"
relational schemas
CPSC350

- "relational schemas"
- "table normalization"
- "practical use of relational algebraic operators"
- "tuple relational calculus"
- "and their expression in a declarative query language"
<table>
<thead>
<tr>
<th>Name</th>
<th>EmpId</th>
<th>DeptName</th>
</tr>
</thead>
<tbody>
<tr>
<td>Harry</td>
<td>3415</td>
<td>Finance</td>
</tr>
<tr>
<td>Sally</td>
<td>2241</td>
<td>Sales</td>
</tr>
<tr>
<td>George</td>
<td>3401</td>
<td>Finance</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Production</td>
</tr>
<tr>
<td>Harriet</td>
<td>2202</td>
<td>Sales</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DeptName</th>
<th>Manager</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finance</td>
<td>George</td>
</tr>
<tr>
<td>Sales</td>
<td>Harriet</td>
</tr>
<tr>
<td>Production</td>
<td>Charles</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>EmpId</th>
<th>DeptName</th>
<th>Manager</th>
</tr>
</thead>
<tbody>
<tr>
<td>Harry</td>
<td>3415</td>
<td>Finance</td>
<td>George</td>
</tr>
<tr>
<td>Sally</td>
<td>2241</td>
<td>Sales</td>
<td>Harriet</td>
</tr>
<tr>
<td>George</td>
<td>3401</td>
<td>Finance</td>
<td>George</td>
</tr>
<tr>
<td>Harriet</td>
<td>2202</td>
<td>Sales</td>
<td>Harriet</td>
</tr>
</tbody>
</table>
practical knowledge
the stuff you need to know to be a better software developer
An introduction to the non-SQL part of the course
Cloud Storage in a Post SQL World

Ian Wilkes ars technica
Cloud storage in a post-SQL world
Since the rise of the Web, SQL-based relational databases have been the ...
“Since the rise of the Web, SQL-based relational databases have been the dominant structured storage technology behind online applications.”
"The past few years have seen the emergence of the cloud as a compelling environment for online application development, bringing true utility computing into the infrastructure pantheon"
“But the cloud and SQL do not mix well, and multiple efforts are now underway to offer viable alternatives to the venerable database.”
“While relational databases are by no means doomed, they will soon be joined in the cloud, and possibly out-shined by, new non-relational database technologies.”
<dissect>
more and more computing is in the cloud

• google docs
• dropbox
more and more computing is in the cloud

• google docs
• dropbox
• chromeOS
• Google Cr-48
more and more computing is in the cloud

• google docs
• dropbox
• chromeOS
• Google Cr-48
• iPad / Android / mobile computing
more and more computing is in the cloud

- Even cloud/browser based IDEs.
Take a moment
do back of envelope
calculation
what % of your computer time
on cloud?
results
some believe the cloud and SQL do not mix well
<on one side>
SQL databases work fine

• setup straightforward
• programming not that hard
• wide range of support
• for small projects XAMPP stack near plug and play.
• SQL on a Amazon ec2 instance easy to set up and work with
SQL can scale vertically
big iron
SQL can grow vertically
SQL cannot grow horizontally easily.
Brewer’s Conjecture and the Feasibility of Consistent, Available, Partition-Tolerant Web Services

Seth Gilbert*  
Nancy Lynch*  

Abstract

When designing distributed web services, there are three properties that are commonly desired: consistency, availability, and partition tolerance. It is impossible to achieve all three. In this note, we prove this conjecture in the asynchronous network model, and then discuss solutions to this dilemma in the partially synchronous model.
Brewer’s Theorem  
a.k.a. CAP Theorem

A system cannot have high consistency, availability, and partition tolerance simultaneously.
SQL focuses on consistency
SQL focuses on consistency constraints on foreign keys
ACID semantics (ch11)
Wouldn't it be dreamy if a series of SQL statements could be executed as a group, all at once, and if something goes wrong be rolled back as if they'd never been executed? But it's only a dream...
Wouldn't it be dreamy if a series of SQL statements could be executed as a group, all at once, and if something goes wrong be rolled back as if they'd never been executed? But it's only a dream...

During a transaction, if all the steps can't be completed without interference, none of them should be completed.
ACID: ATOMICITY

All of the pieces of the transaction must be completed, or none of them will be completed. You can’t execute part of a transaction. Mrs. Humphries’ samoleons were blinked into non-existence by the power outage because only part of the transaction took place.
**ACID: ATOMICITY**
All of the pieces of the transaction must be completed, or none of them will be completed. You can’t execute part of a transaction. Mrs. Humphries’ samoleons were blinked into non-existence by the power outage because only part of the transaction took place.

**ACID: CONSISTENCY**
A complete transaction leaves the database in a consistent state at the end of the transaction. At the end of both of the samoleon transactions, the money is in balance again. In the first case it’s been transferred to savings; in the second it’s been translated into cash. But no samoleons go missing.
**ACID: ATOMICITY**

All of the pieces of the transaction must be completed, or none of them will be completed. You can’t execute part of a transaction. Mrs. Humphries’ samoleons were blinked into non-existence by the power outage because only part of the transaction took place.

**ACID: CONSISTENCY**

A complete transaction leaves the database in a consistent state at the end of the transaction. At the end of both of the samoleon transactions, the money is in balance again. In the first case it’s been transferred to savings; in the second it’s been translated into cash. But no samoleons go missing.

**ACID: ISOLATION**

Isolation means that every transaction has a consistent view of the database regardless of other transactions taking place at the same time. This is what went wrong with John and Mary: Mary’s ATM could see the balance while John’s ATM was completing the transaction. She shouldn’t have been able to see the balance, or should have seen some sort of “transaction in progress” message.
**ACID: ATOMICITY**

All of the pieces of the transaction must be completed, or none of them will be completed. You can’t execute part of a transaction. Mrs. Humphries’ samoleons were blinked into non-existence by the power outage because only part of the transaction took place.

**ACID: CONSISTENCY**

A complete transaction leaves the database in a consistent state at the end of the transaction. At the end of both of the samoleon transactions, the money is in balance again. In the first case it’s been transferred to savings; in the second it’s been translated into cash. But no samoleons go missing.

**ACID: ISOLATION**

Isolation means that every transaction has a consistent view of the database regardless of other transactions taking place at the same time. This is what went wrong with John and Mary: Mary’s ATM could see the balance while John’s ATM was completing the transaction. She shouldn’t have been able to see the balance, or should have seen some sort of “transaction in progress” message.

**ACID: DURABILITY**

After the transaction, the database needs to save the data correctly and protect it from power outages or other threats. This is generally handled through records of transactions saved to a different location than the main database. If a record of Mrs. Humphries’ transaction had been kept somewhere, then she might have gotten her 1,000 samoleons back.
Brewer’s Theorem
a.k.a. CAP Theorem

A system cannot have high consistency, availability, and partition tolerance simultaneously.
availability

online web apps -
availability is a must
Brewer’s Theorem
a.k.a. CAP Theorem

A system cannot have high consistency, availability, and partition tolerance simultaneously.
mismatch
Clustered machines -- SQL semantics
BASE
instead of ACID semantics
BASE

Basically Available, Soft state, Eventual consistency
Basically Available:

This constraint states that the system does guarantee the availability of the data as regards CAP Theorem; there will be a response to any request. But, that response could still be ‘failure’ to obtain the requested data or the data may be in an inconsistent or changing state, much like waiting for a check to clear in your bank account.
Soft state:

The state of the system could change over time, so even during times without input there may be changes going on due to 'eventual consistency,' thus the state of the system is always 'soft.'
Eventual consistency:

The system will eventually become consistent once it stops receiving input. The data will propagate to everywhere it should sooner or later, but the system will continue to receive input and is not checking the consistency of every transaction before it moves onto the next one.
Wait, there's more
Our model

PythonFlask → SQL Server

JOIN

ORDER BY
Extending our model
Our model
Extending our model

Apache S.

Apache S.

Apache S.

Apache S.

Apache S.

Apache S.

Join

Order by

Join

Order by

Join

Order by

Join

Order by

Join

Order by

Join

Order by

SQL Server
All data processing on database server adds load to CPU further inhibits scaling
Some SQL platform providers offer cloud relational db

• Amazon's mySQL based RDS
• Heroku's PostgreSQL
• but these more like managed hosting than true cloud (distributed) computing
push to NoSQL alternatives

sacrifice consistency in favor of scalability and availability
first implementations
proprietary

• Google Big Table
• Amazon Dynamo
NoSQL features

- **No schemas** -- no fixed table structure, no fixed columns
- *little to no protection against invalid data*
NoSQL features

• **No or limited joins** - no built in methods for connecting/joining entries.
• must be done at application level
NoSQL features

• Restricted query interface - SQL has rich query interface. NoSQL somewhat limited.

• although some NoSQL systems (e.g., CouchDB) offer different views.
NoSQL features

• Limited transactions or locks
  - ACID transactions not distributed system friendly.
NoSQL open source projects
Cassandra

• developed by Facebook and used in-house

• now an Apache project

• used in a large number of production environments

  • Digg, Rackspace, Twitter (internally)

  • Netflix

  • Reddit

  • CERN

  • not Facebook
CouchDB

• an Apache project
• operates on JSON documents
• the read state of the db is not always consistent with the latest writes.
• reasonably stable
• used by:
  • wego, BBC, World Wildlife Fund,
  • http://www.couchbase.com/customers/case-studies
Orbitz Takes Off with Couchbase

CouchConf San Francisco 2012

Orbitz’s leading travel website books hotels, flights, and car rentals. It is one of the most active sites in its market, logging millions of flight and hotel searches on an average day. Facing scalability and performance challenges with its relational database, as well as soaring costs and reliability issues with its caching layer, Orbitz turned to open source NoSQL technology. In this session, Orbitz will share how they replaced their caching tier with Couchbase, using Couchbase Server extensively across their site to achieve astronomical results in terms of scalability, reliability, performance, and cost savings.
mongoDB

• operates on JSON documents
• considered faster than CouchDB.
• good stability
• used by
  • foursquare, bit.ly, intuit, shutterfly, nyTimes, Etsy, Cern (Large Hadron Collider) justin.tv
Voldemort

- developed by LinkedIn
- now open source
- bare-bones key-value pair db.
- all data management work done in the applications.
HBase

- A Big Table clone
- uses map-reduce
Amazon’s simpleDB

• not open source
• disappointing performance
• not mature
The top ones

- Cassandra
- CouchDB
- mongoDB
- Redis
Failure of NoSQL

DIGG
JargonSpy
How Digg’s Cassandra Debacle Could Have Been Avoided
Dan Woods, 09.21.10, 11:00 AM EDT

Less traumatic paths to scalability than re-engineering your infrastructure.

If you are an ambitious CTO, it is likely you’ve found yourself in the same predicament as John Quinn, former vice president of engineering at Digg, at some point in your career. You looked at the current state of your technology. You had a vision for how things could be so much better if the infrastructure were re-engineered to take advantage of breakthrough technology. In the new world you were going to create, so much more would be possible. The technology would really start to serve the business. You could say yes to meeting business needs much more often at lower cost and risk.

As Digg Struggles, VP Of Engineering Is Shown The Door
Erick Schonfeld
Sep 7, 2010

Ever since Digg launched its new site design, it’s been plagued with all kinds of trouble, not least of which is that it keeps going down. The problems with the new architecture are so bad that VP of Engineering John Quinn is now gone, we’ve confirmed with sources close to Digg.

In a DiggNation video today, CEO Kevin Rose explained some of the technical issues the site is dealing with and why it can’t simply roll back to the previous architecture. The new version of Digg, v4, is based on a distributed database called Cassandra, which replaced the MySQL database the site ran on before. Cassandra is very advanced—it is supposed to be faster and scale better—but perhaps it is still too experimental. Or
How This Kid Made $60 Million In 18 Months

The Hottest Small Company in America
How Kevin Rose, founder of Digg, captured the hearts and minds of the most desirable customers
Digg for sale -- Google and Microsoft may be buying
Digg is for sale again, and could go for something over $200 million
Social sharing site Digg is sold for $500,000 - after turning down an offer from Google for $200million just four years ago

By EDDIE WRENN

PUBLISHED: 04:07 EST, 13 July 2012 | UPDATED: 04:07 EST, 13 July 2012
Road Map for remainder of course

• MongoDB
• CouchDB
• Redis