Chapter 10: the design of the persistence layer

Stefan L. Bund, MS
Cal Poly Pomona

What will cover in today's lecture

- We will compare and contrast different database and data storage technologies
- Map application strategies to persistence technologies
- Discuss data transfer, security and their relationship to different means of persistence, and provide examples for each
- help analysts strategize persistence on a per-application basis

First segment: the relational database model

- Most modern persistence models orient around traditional and historical examples in the relational database model

- The SQL database here reign supreme; attributes, primary keys and foreign key relationships between separate tables, which are wrapped by a server process

- The SQL server paradigm begun by Oracle

- further iterated by MS SQL Server, IBM and most recently, MySQL

The relevance of keys in relational databases

- Keys help in defining rules for dating uniqueness and referential integrity

- Primary keys create unique rows, and represent data that is not optional and mandatory, and must be unique

- Non-primary attributes represent a diverse array data types, and our maps to string, character, number, and text formats

Example forming tables in my SQL

- The syntax for creating tables in MySQL illuminates the richness data captured in a relational data table

- http://dev.mysql.com/doc/refman/5.1/en/create-table.html

Relational database fundamentally supports relational algebra

- Relational model, keys of like data provide means to relate the data sets, and support mathematical *set theory*
- Databases that are designed well enable deep business model analysis, where one piece of data in one data set, leads to deeper insight, from data in another
- business intelligence networks are now built upon the relationships between data in one source, and another
- interesting and valuable systems help mine data in one data set, with related data in another
- therein lies the strength of the relational model

Relational Model

Activity Code	Activity Name
23	Patching
24	Overlay
25	Crack Sealing

Key = 24

Activity Code	Date	Route No.
24	01/12/01	I-95
24	02/08/01	I-66

Date	Activity Code	Route No.
01/12/01	24	I-95
01/15/01	23	I-495
02/08/01	24	I-66

Entity relationship modeling

- Let's do an entity relationship diagram in support of a business model

- Here, we will craft the schema for a set of tables, comprised by database

- The schema should represent the domain of business considerations, but does not need to map directly to the requirements of any application

Designing relational schema

- any application, over the course of many decades can theoretically map to a rich data set, so in the design phase it's more important to describe the problem domain rather than the specificities of the application

Designing relational schema and implementing it in a relational database

- Here we will use mySQL's native Design and administration app, MySQL Workbench
- We will emphasize the idea of data normalization, which is a process that eliminates complexity, provides intermediary tables to resolve complex relationships, then provides small, agile tables that support concurrency, online

Our sample database schema

Part two:

- XML, JavaScript object notation, and Internet data transfer, REST ful apps, web apps

A Decade of Internet integration databases

- XML, or extensible markup language, initially became popular in year 2000, and it represented the ability to transfer a single file that contain a data set
- XML represents relational schema well, tying the structure of a database into a flat, hierarchical data set
- Since this snippet of rich data can be transferred over the Internet and the format of a single file full of XML, it gave rise to RESTful protocols, then web applications, which relied upon asynchronous JavaScript and XML, or Ajax

MSFT's contribution to internet data transfer

- originated at Microsoft's laboratories, XML helped push the rise of internet data transfer, given XML's outstanding capacity to capture and represent a structure of data, best evinced here, in this example --

- http://msdn.microsoft.com/en-us/library/ms762271(v=vs.85).aspx

Google's rise

- The massive data sets Google was attempting to synthesize gave rise to many good experiments in web applications, the most popular of which was Google mail, or Gmail, which relied upon asynchronous JavaScript to load personal email

- moved email completely online, with the fastest response to incoming mail -- we would camp beside our gmail, and communicate entirely over it

Gmail's rise

- If we can load all of your email messages as XML, with a fashionable, attractive web interface around it, then the popular inbox can be converted into a feed of XML
- the asynchronous nature of AJAX helped power the rise of xml feeds that arrived in real-time, from their server farm data centers
- gmail started a revolution for browser based applications that worked with the efficacy of desktop applications, at internet scale and real time
- XML engendered a whole new breed of web-based applications, dependent on representing the persistence layer in real time

XML leads to web services

- The rise of social networking and weblogs between 2000 and 2003 lead to the really simple syndication format, or RSS, which drastically popularized working in the N-tier

- The entire digests of the newspaper, personal blog, or email inbox could be expressed as an RSS feed, then transferred over the Internet, thus mapping complex data structures into a readable format that transferred well to RSS readers

- Digg leads the pack, in creating real time news digests

XML continues its rise

- Because of XML, massively online multiplayer games could also flourish, enriching the validity of the Internet
- Hence the Internet experience began to centralize around XML feeds, be it email, gaming, news, or other emerging services such as Twitter
- twitter put the RSS feed on steroids, delivering very small message packets in real time
- Barack Obama was the first major Twitter adopter, thereby popularizing the service

From XML to JSON

- But the tag heavy presentation of XML created a deafening signal-to-noise ratio
- Programmers admitted that the structure of XML was unwieldy, and data inside of markup was costly and inhibited by the volume data transfer associated with tags
- Thus the tags could be dropped by in-memory structures of data, expressed as an understood format
- Java Script Object Notation, or JSON, could capture the structure of XML, and map to the persistence layer, but do so without markup
- JSON freed developers up to make streams of database content, and transfer without preformatting data inside of XML schemas

JSON example

- here's your best example of this transition, on http://json.org/example.html

so let's review

- relational databases need schema to organize data

- xml captured schema: data types, rows, uniqueness

- JSON captures XML's structure without the ML, or markup, creating smaller internet packets, for faster transfer

Putting persistence on steroids

- If relational data could be encoded then transferred across the Internet efficiently, application content favors data that moves fast, and reflects real-world events

- RDBMS schema mirrors real-world things, so as its content updates, so should its representational state

- RDBMS change gets reflected in updates to its applications

- My SQL driven services such as Facebook loaded the user experience with imagery, and employed lead Google engineer, Brett Taylor, the creator Google maps, to spearhead the delivery of the real-time Internet to this fast-emerging platform

- BT's migration to FB emobodied the rush to Google engineering talent to facebook, in anticipation of its IPO, and the maturation of FB's backend

The Brett Taylor era

- Though any Internet aficionado could dispute this, Brett Taylor's appointment as CTO of Facebook let delivery of the real-time wall, which led Facebook to its remarkable position at the center of the Internet experience

- Here, massive indexing of posts behind the scenes at facebook led to many innovations in back and persistence models, as the front and experience could rapidly deliver json enabled data feeds

- mysql databases create the basis for massive new data services like facebook's

- http://blog.redfin.com/devblog/2010/06/evolving_a_new_analytical_platform_with_hadoop.html#.U2urs17gGJU

Move fast and break things

- In the agile development environment of Facebook, many new backend innovations were getting traction, especially as engineers were migrating out of Google and into the pre-IPO environment at Facebook

- The rise of the cloud meant that thousands of relational databases could be tied together, and back and persistence could move into memory, alone

Memory is your friend

- In memory databases move much faster than their traditional disk space, relational counterparts
- RDBMS rely on the disk, to read and write
- RDBMS also needs a master-slave relationship, to backup and create hot copies of saved data
- the redundancy model for RDBMS was insufficient to support large services like FB over the long term, and FB engineering hit scalability walls

Moving the relational model into memory

- The rise of massive scalability, with the onset of Twitter and Facebook, let the data center engineers to move the presentation of data into JSON, and the persistence of the data to in- memory databases that moves millions of times faster than their on-disk counterparts

- Once memory became the primary medium for persistence, the cloud could reign supreme and become the primary focus the backend IT

- roughly in 2009 FB began the transition toward less structured data, in MySQL, and move toward the looseness embodied in the No-SQL movement

The rise of the cloud

- Though relational databases in 2004 serve as the backbone of emerging services such as Facebook, by 2012 the transition to the cloud had become complete
- Relational systems, though vital to traditional businesses since 1968, had transition to massive scalability, in response to the Internet content revolution
- Google search becomes Google Now, and high responsiveness categorizes the search for value
- writing to disk is tens of thousands of times slower than writing to memory, and responding to precached data, rather than responding to disk reads

the evolution of the cache

- given the responsiveness of memory, effort was placed in scaling shared memory across the server farm

- grids were instituted which created 'markets' for memory, where incoming jobs were paired with memory capacity, in the data center

- where applications needed immediate response from the back-end, memory caches were given the responsibility to serve rich clients

early movements to move toward the cloud

- the globus toolkit captured the imagination of my generation: http://toolkit.globus.org/toolkit/

- the grid helped share the memory, CPU and storage of many machines, allowing their hardware to be controlled from a central piece of middleware

- good illustration: http://dev.globus.org/images/f/f5/GemIcaconcept.png

New databases for new platforms

- Mongo DB, and Amazon Web services became the de facto means for new initiatives in information technology

- The cheapness, security, and general liability of the cloud provider backbone for thousands of new products, and drove an employment boom in downtown San Francisco, where tech employment in that city is becoming equivalent to that of Silicon Valley

- AWS drove the most recent startup revolution, where teams of programmers could initiate games, apps, sites, and even hardware projects in the cloud, cheaply -- AWS moves massive productivity toward developers, who could spin up a data center for their projects for very little money

The departure of the relational model, for the big data model

- But a new strategy emerged, where the additional data would be replaced by unstructured data bins run on the backbone of the *hadoop* Apache engine
- hadoope encouraged developers to dump large quantities of data into a stored, cloud group of servers
- the relatively unstructured nature of the content could be mined, without relationships
- which is interesting: the real dream of the RDBMS was to provide an 'oracle' into large data sets, but it took the innovations at Google to power the rise of Hadoop

The elephant in the room and zookeeper

- Google's own Map Reduce algorithm provided the groundwork for strining many small, cheap servers together, to create large in-memory indices of searchable data
- Google helped fuel the rise of non-relational, cached, in-memory systems
- so when we adopt new cloud, NoSQL systems, we morph the large scale systems used at GOOG
- Hadoop's own HDFS file system lends itself to binning large data sets, then running scalable machine learning algorithms on top of it, to discover patterns

the emerging ecosystems of big data

- nosql db captures data
- memcache serves it to clients over JSON
- HTML5 presents marshalled JSON
- back in the server room, data is binned in Hadoop, where analytical processes help categorize the data

Search engines break down relational models

- With tools such as properties, clear and distinct relationships need not be developed at the beginning of the project
- Instead, with the rise of text indexing, the organization of data comes *after* the data is stored
- Hadoop's massive capacity to string many computers together in the cloud, then store massive amounts of unstructured data, then offer searchable indices
- this gives rise to whole new families of applications

A new use case

- Now, instead of predefined relationships in data, we need not have relationships, only the capacity to store it
- and mine it -- via ML
- Machine learning algorithms that emphasize bayesian rubrics, classification, characterization, and canopy clustering become the norm for businesses seeking to make order out of massive amounts of organizational data

The rise of the data scientists

- In this environment of massive in memory databases, unstructured search, and fast Internet presentation, companies envision that they can tackle much larger social problems, faster

followup

- hadoop's maintenance page on Apache: http://hadoop.apache.org

- AWS elastic map reduce, which rents hourly hadoop jobs: https://aws.amazon.com/elasticmapreduce/ (video)