Big Data and Security

Jeffrey Borowitz, PhD

Lecturer

Sam Nunn School of International Affairs

Big Data or Just Data?

Big Data vs Data?

- As we've seen, how fast computers can access the data they need depends on how, when, and in what order they access it
- Databases are programs that store data in a particular way
  - They have a language to access specific pieces of data
  - The program contains algorithms for building indexes, using them, and returning results.

Two Database Queries

- Get information (e.g. average age) on everyone in zip code XXXXX
  - Find where info is stored on people
  - Check if their zip code is XXXXX
  - Return information if it is
- How many unique people live in zip code XXXXX?
  - Can go faster!
  - Maintain a list of unique addresses
  - For each zip code, store information about each person there:
    - Rest of address
    - Age
  - Now, think about how much work needs to be done

Current Database World

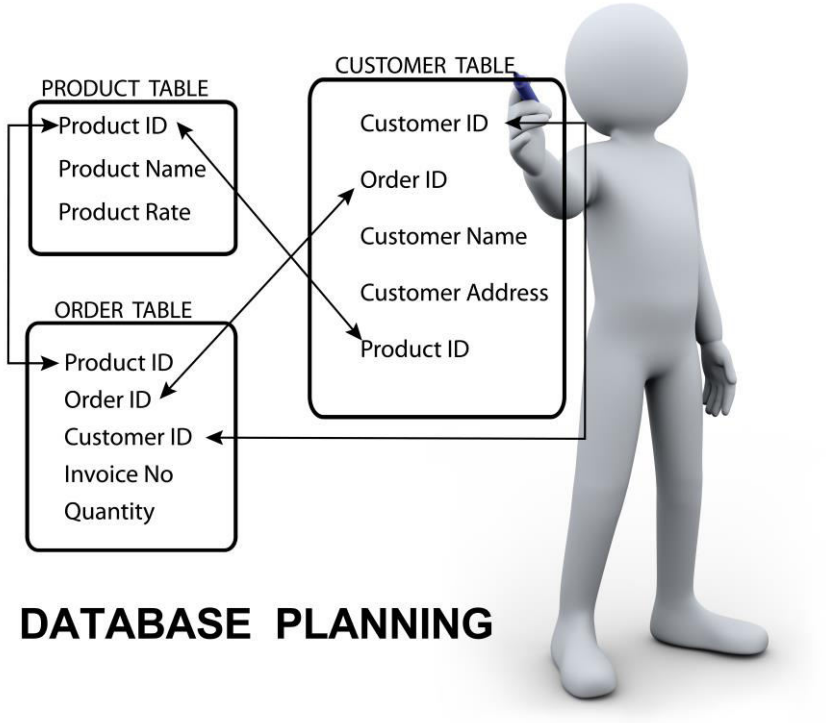
- Currently, most database functionality is designed around e.g. commerce: customers, transactions, products
  - Need high availability
  - Can read and write and the same time
  - Usually you look up a little info, not a lot (e.g. look up your account info
- But analyzing data is really different:
  - How many transactions occurred each of the last few months?
  - Which customer looks the most like this customer
  - This requires looking at all the data

Databases

- Relational Database
- Row vs column stores
  - Row stores make sense for adding new data/looking data up
  - Column stores make sense for analyzing the data set
  - SQL: Structured Query Language
- “NoSQL” database
  - Databases with less structure
  - NoSQL is a marketing buzzword, which people say means “not only SQL”
  - As far as I know, NoSQL stuff always does something which is a subset of SQL functionality

Relational Database

- Data is stored in tables
- Items in tables can be references to other tables



Georgia Tech

Structured Query Language

- SQL is a language for creating views of tables or statistics which are useful
- Benefits:
  - Structure of the tables can represent rich relationships between the data
  - You can look at very specific combinations of data
    - “Show me the things that people who bought yellow umbrellas in June 2012 bought in December 2013”
- Costs:
  - You have to create a very detailed **schema** for your data
  - It’s hard to scale all these very detailed interactions

What Examples of Relational Databases?

PostgreSQL
Open source

MySQL
Open source, owned by Oracle

Oracle Database
Expensive database, many features

Microsoft SQL Server
Cheaper than Oracle

Georgia Tech

Key-Value Stores

- The opposite end of data storage
  - Each key has a “value” which is the data associated with it
  - In its purest form, the only question you can ask the database is “What is the information on SSN XXXX”
- Benefits:
  - It’s relatively easy to scale very large (you just have to store different keys on different computers according to some rules)
  - It is very fast
  - You don’t have to worry about a schema
- Costs:
  - You can’t do complicated queries without looking at all the data (looking up every key)

Examples of Key-Value Stores

- Key value stores are used for temporary access in memory
  - Redis
  - Memcached
  - Amazon Elasticache
- Or longer term storage on disk:
  - Amazon S3 - a large key value store which scales well

Document Oriented Database

- Instead of rows and tables, there are a bunch of “documents” which might have different combinations of fields
- Benefits:
  - These typically support some querying: get all emails from XXX
  - Different documents can have different fields: if each document pertains to a family, not all families have children
  - These scale better than relational databases: different documents just go on different computers
- Costs:
  - You still can’t query arbitrary relationships
- The bottom line is these are somewhere between simple key-value stores and full relational databases
- Examples include MongoDB and ElasticSearch
  - We will use ElasticSearch to store the Enron emails

Georgia Tech

Tabular Databases

- Relational databases have rich relationships between tables
- “Tabular” databases allow arbitrary querying on just a single table
- Benefits
  - Full querying ability for the single table
  - Can scale fairly easily: put different rows on different computers
- Costs:
  - You only get a single table, so you can’t model complex relationships
- In some ways, tabular databases are kind of like Key-Key-Value stores, with some extra functionality

Examples of Tabular Databases

- BigTable - Google's proprietary implementation - they published a paper which everyone implemented
- HBase/Cassandra/Hypertable - open source implementations of Google's BigTable paper
- Accumulo - another open source implementation, developed at NSA, with "cell level security"

Other Big Data Databases

- SQL on Hadoop
  - Most relational databases are built for availability and transaction processing
  - So you can just build relational databases on top of Hadoop
  - This gives you very good scalability AND full relationships
  - The only problem is some queries can require lots of looking at references across different computers, etc, so this can be super super slow.
  - But if you are a data analyst, you can sometimes just wait
- Examples
  - Hive - creates relational databases in Hadoop
  - Impala - like Hive but faster
  - Spark SQL - creates relational databases in Spark (like Hadoop, but stores data in memory as much as possible)

Lesson Summary

- Databases are programs that store data in a particular way, and use languages to access that data
- 3 Types of databases:
  - Relational – items in one table reference other tables
  - Key-Value Stores – a very fast way to access a value for a particular key
  - Tabular – arbitrary querying on just a single table
- SQL is the language used to create views of tables and statistics