CS150: Database & Datamining

Lecture 1: Course overview

Xuming He

Spring 2019

Acknowledgement: Slides are adopted from the Berkeley course CS186 by Joey Gonzalez and Joe Hellerstein, Stanford CS145 by Peter Bailis.

The world is increasingly **driven by data…**

This class teaches **the basics** of how to use & manage data.

Key questions we will answer

- How can we **collect and store** large amounts of data?
  - By building tools and data structures to efficiently index and serve data
- How can we **efficiently query** data?
  - By compiling high-level declarative queries into efficient low-level plans
- How can we **safely update** data?
  - By managing concurrent access to state as it is read and written
- How do different database systems manage **design trade-offs**?
  - e.g., at scale, in a distributed environment?

When you'll use this material

- Building almost any software application
  - e.g., mobile, cloud, consumer, enterprise, analytics, machine learning
  - Corollary: every application you use uses a database
  - Bonus: every program consumes data (even if only the program text!)
- Performing data analytics
  - Business intelligence, data science, predictive modeling
  - (Even if you're using Pandas, you're using relational algebra!)

- Building data-intensive tools and applications
  - Many core concepts power deep learning frameworks to self-driving cars

Today’s Lecture

1. Introduction, admin & setup
2. Overview of the relational data model
3. Introduction to SQL

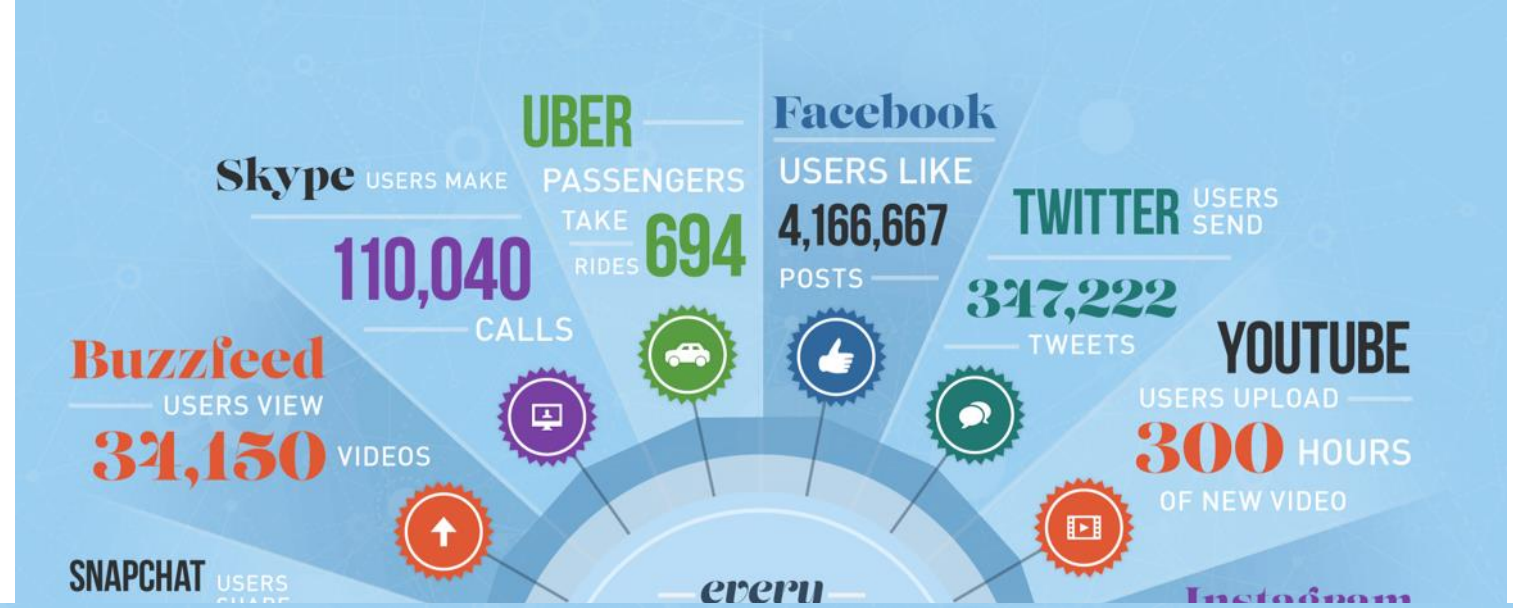
1. Introduction, admin & setup

Data is Growing

https://www.domo.com/blog/2015/08/data-never-sleeps-3-0/

https://www.domo.com/blog/2015/08/data-never-sleeps-3-0/

https://www.domo.com/blog/2015/08/data-never-sleeps-3-0/

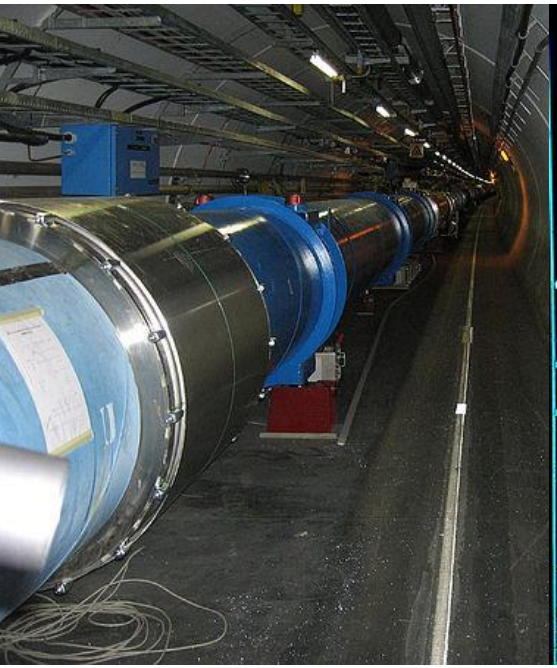
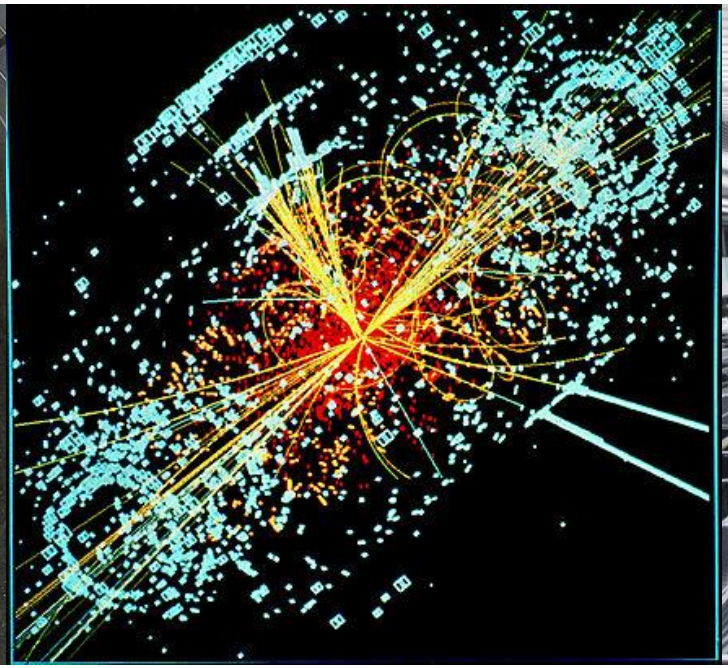
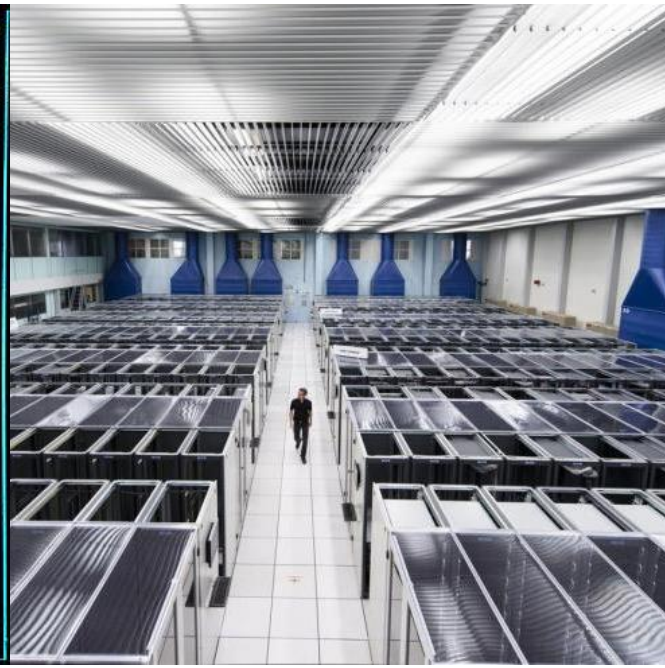


THE GLOBAL INTERNET POPULATION GREW **18.5%** FROM 2013–2015 AND NOW REPRESENTS **3.2 BILLION PEOPLE.**

https://www.domo.com/blog/2015/08/data-never-sleeps-3-0/

Scale of scientific data

Large Hadron Collider (LHC)

- 150M Sensors @ 40M times per second: 6 Peta-events per second
- Massive filtering (data loss) → 700 MB per second → 15 PB per year
- Data collection is **limited by compute and storage**

https://lhcb-public.web.cern.ch/lhcb-public/en/Data%20Collection/Triggers2-en.html

Forces driving data growth

- Ubiquitous sensors and reporting:
  - Cameras, mobile computing, blogging, ...
- Large collaborative science projects
- Philosophy: *More Data → More Value*?

Save *all* the *data*!

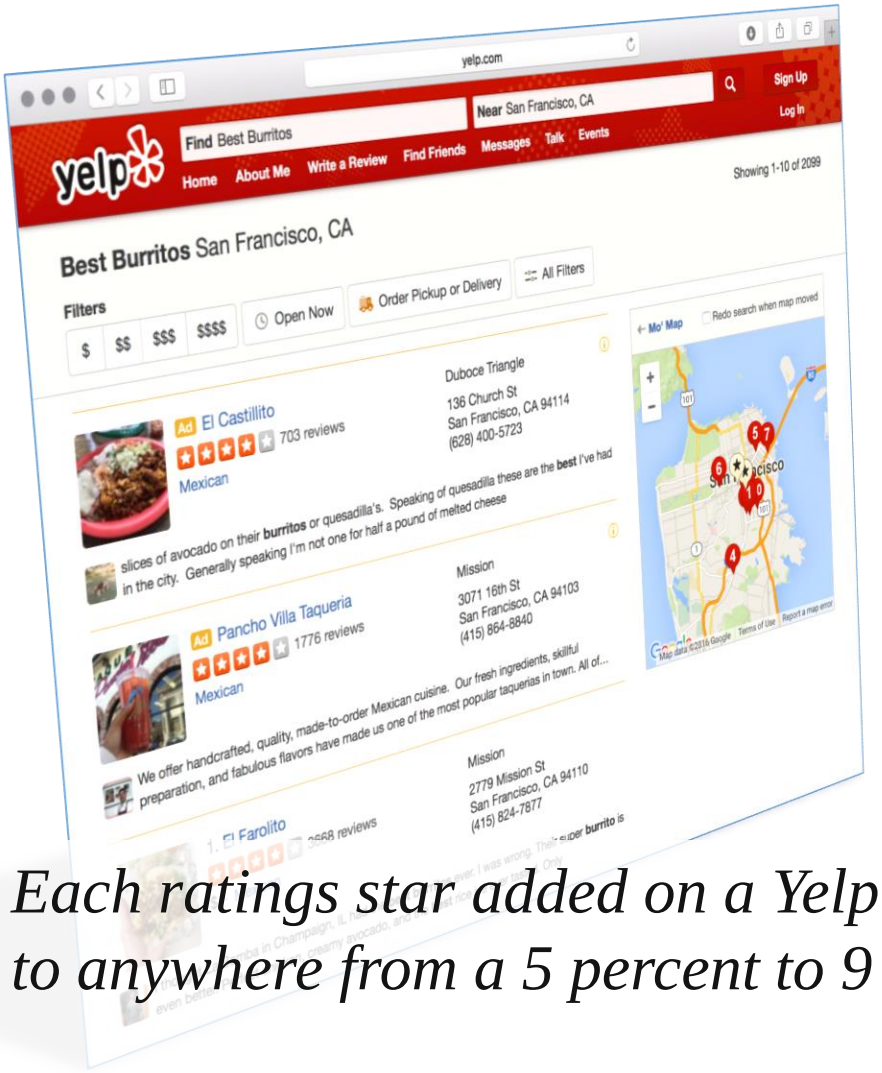
http://hyperboleandahalf.blogspot.com

Enabling Technology

- **Cheap, Scalable** Data Managment Systems

Data is Changing the World

Where should I eat?



Where can I get the best burrito in SF?

Each ratings star added on a Yelp restaurant review translated to anywhere from a 5 percent to 9 percent effect on revenues.

-- Harvard Business School

http://hbswk.hbs.edu/item/the-yelp-factor-are-consumer-reviews-good-for-business

Making connections?

Friends

What are they doing?
What have they done?

Love

Finding love through
Data + Algorithms

Daily Life

Social + News +
Financial service…

Berkeley cs186
is
not
yet
involved …

GLOBAL PARTNERSHIP
FOR SUSTAINABLE DEVELOPMENT DATA

Berkeley cs186
is
not
yet
involved …

Data can help farmers:

- predict when to plant and harvest
- help them get a fair price

*In 2013, a computer model powered by **data** on crop growth and weather cycles **saved $3.8 million dollars** on seeds during a drought in Columbia.*

http://www.data4sdgs.org

Why should **you** study databases?

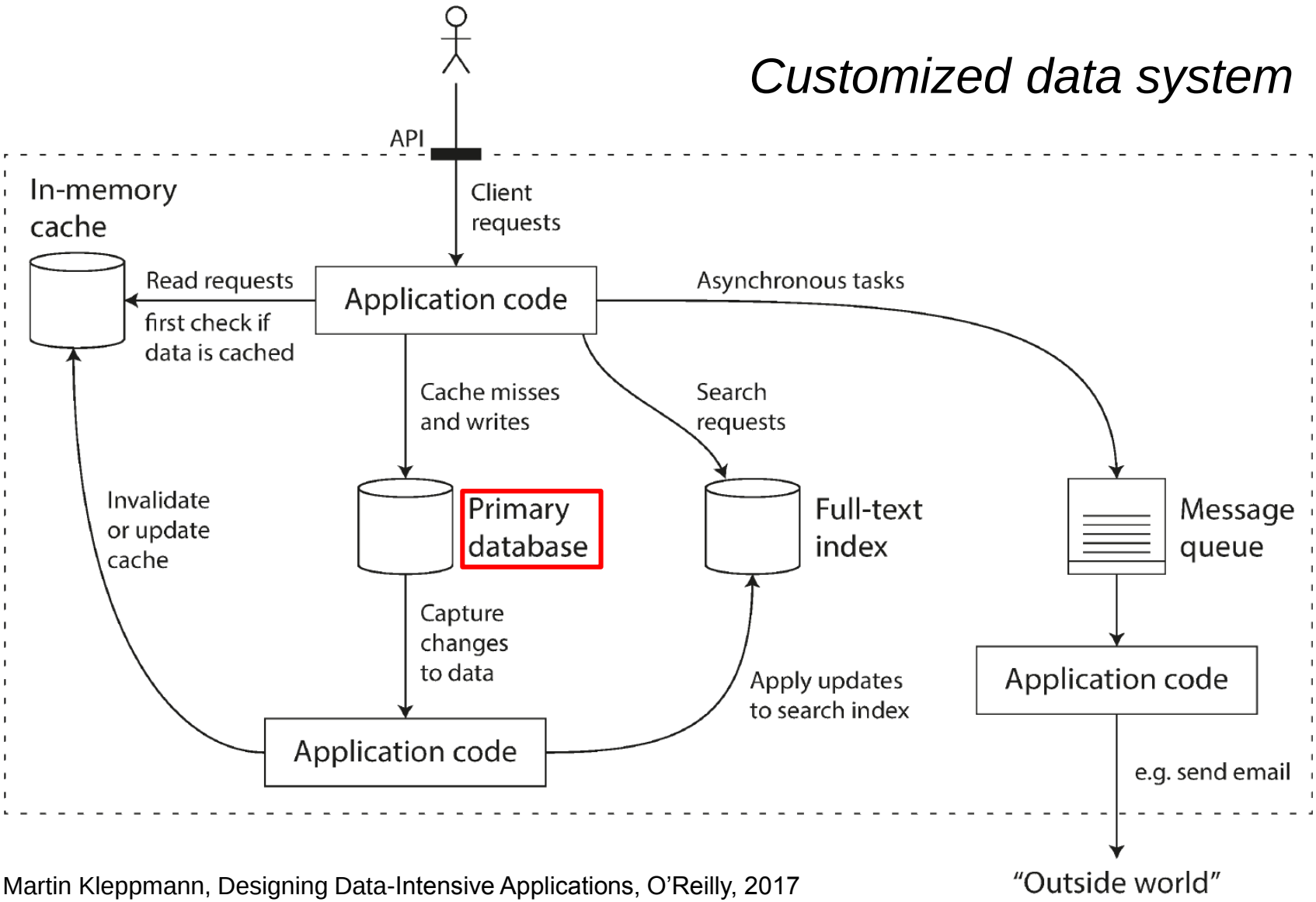
- **Practical**: **Lots of real-world apps:**
  - Startups need DB talent right away = low employee #
  - Massive industry…

- **Intellectual**:
  - Science: data poor to data rich
    - No idea how to handle the data!
  - Fundamental ideas to/from all of CS:
    - Systems, theory, AI, logic, stats, analysis….

Many great computer systems ideas started in DB.

Why should **you** study databases?

Customized data system



Martin Kleppmann, Designing Data-Intensive Applications, O'Reilly, 2017

WHAT IS A DATABASE?

Is this a database?

Rolodex contains

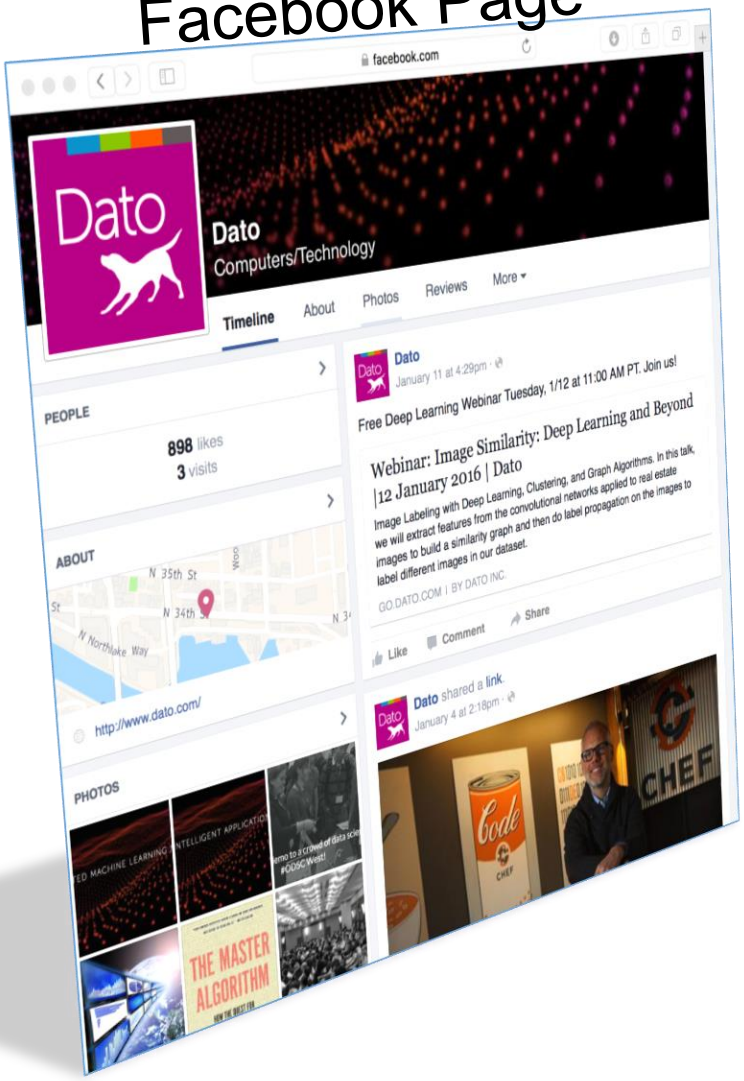
- Contacts

Organized alphabetically

≈

Is this a database?

Facebook Page



Facebook contains

- Contacts
- Events
- Posts

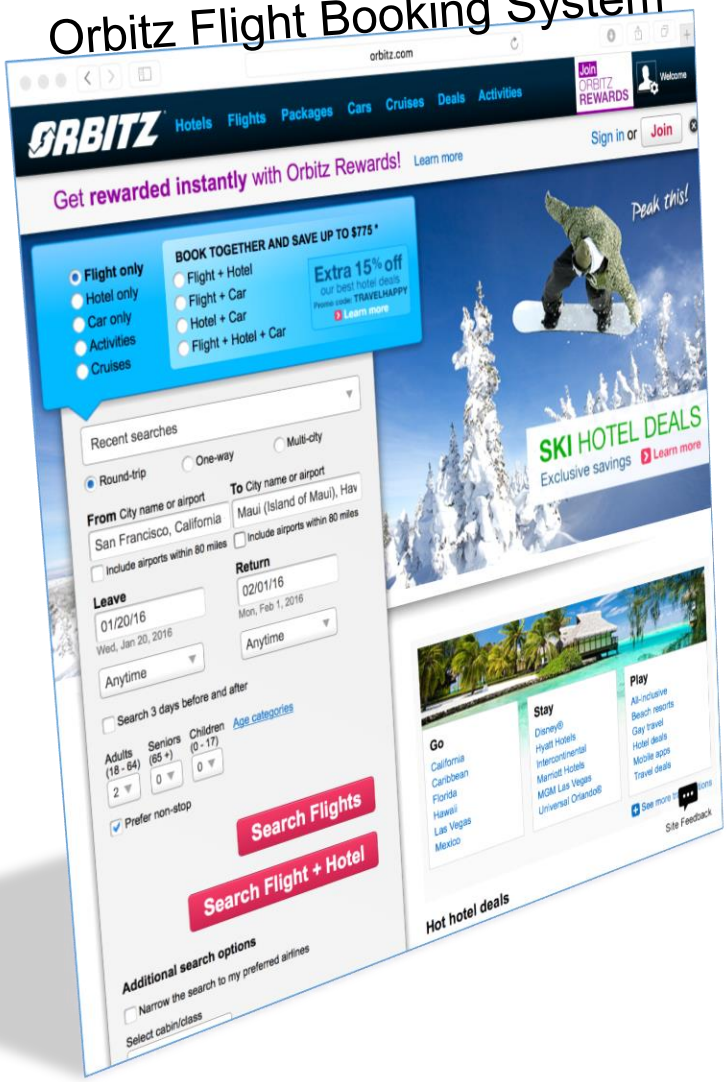
Organized …

Facebook

Collection of DBs and “business logic”

Is this a database?

Orbitz Flight Booking System



Flight Booking:

- Early application of of database systems

Orbitz

Collection of DBs and “business logic”

What is a database?

A *database* is a large, organized collection of data.

Sometimes confused with a Database Management Systems (DBMS)

A *Database Management System (DBMS)* is software that **stores**, **manages** and **facilitates** access to databases.

Relational DBMS

- Traditionally DBMS referred to relational databases

- **RDBMS** is a more appropriate term
- **SQL** data description and manipulation language
- **ACID** transaction consistency
- **Durable** writes (prevent data loss)
- **Mature** technologies …

Ranking of DBMS technologies

DB Engines Ranking

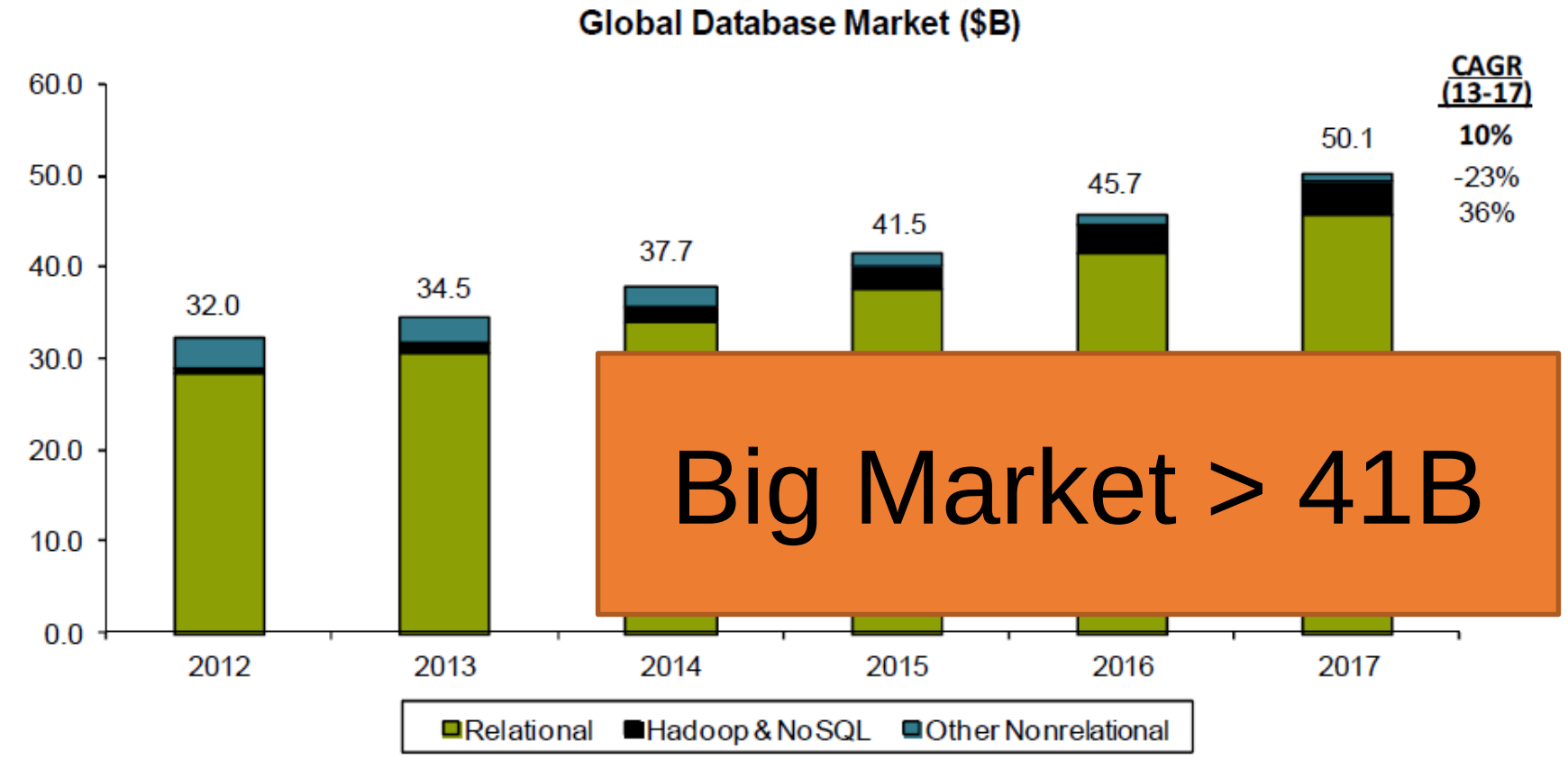
292 systems in ranking, January 2016

Rank Jan 2016	Rank Dec 2015	Rank Jan 2015	DBMS	Database Model	Score Jan 2016	Score Dec 2015	Score Jan 2015
1.	1.	1.	Oracle	Relational DBMS	1496.08	-1.47	+56.92
2.	2.	2.	MySQL	Relational DBMS	1299.26	+0.72	+21.75
3.	3.	3.	Microsoft SQL Server	Relational DBMS	1144.06	+20.90	-54.55
4.	4.	↑ 5.	MongoDB	Document store	306.03	+4.64	+55.13
5.	5.	↓ 4.	PostgreSQL	Relational DBMS	282.40	+2.31	+27.91
6.	6.	6.	DB2	Relational DBMS	196.37	+0.24	-3.76
7.	7.	7.	Microsoft Access	Relational DBMS	134.04	-6.17	-5.10
8.	8.	8.	Cassandra	Wide column store	130.95	+0.11	+32.20
9.	9.	9.	SQLite	Relational DBMS	103.74	+2.89	+7.54
10.	10.	10.	Redis	Key-value store	101.16	+0.62	+6.92

Based on #mentions (e.g., stack overflow), google trends, job postings, profile data on LinkedIn, tweets …

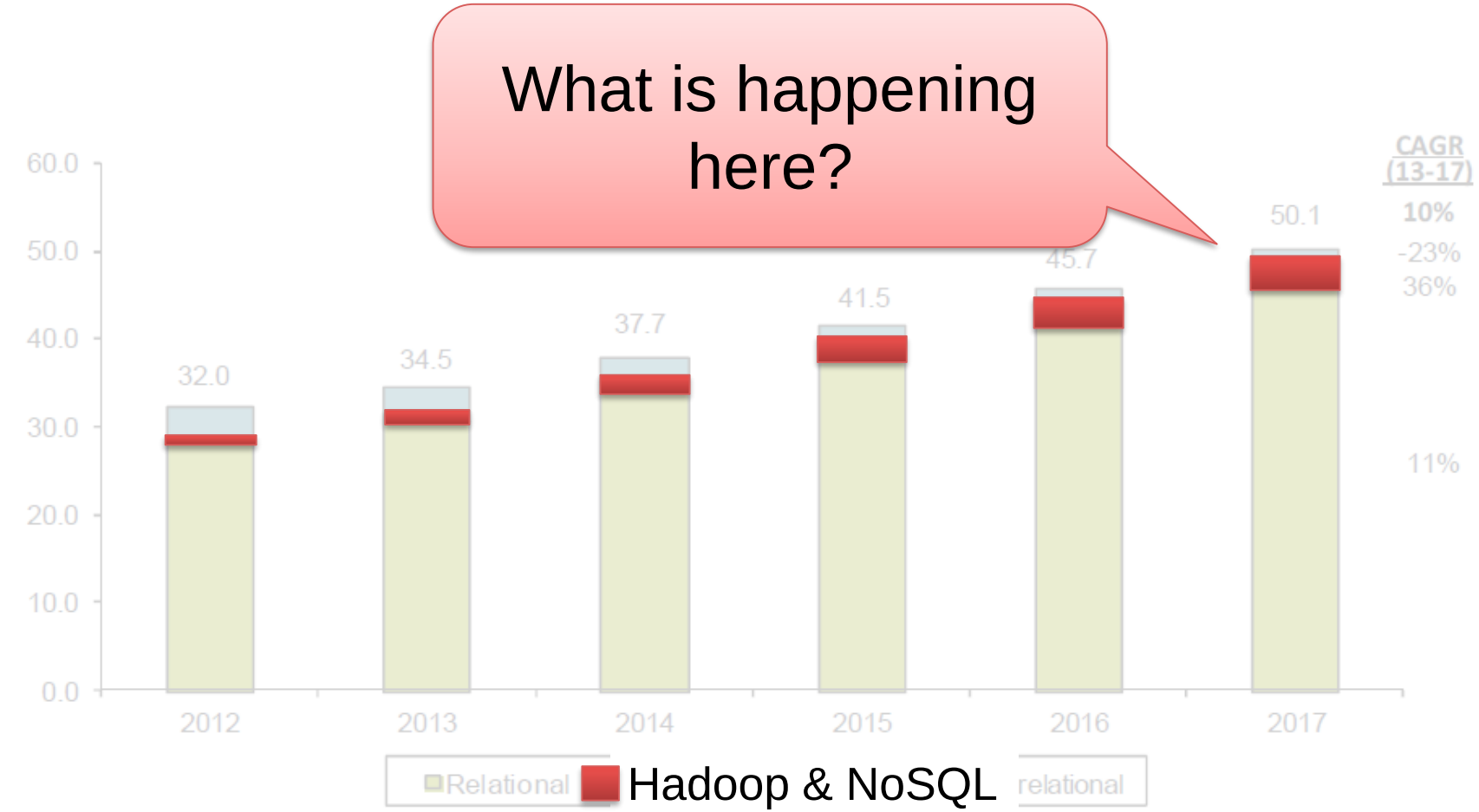
http://db-engines.com/en/ranking

Relational Database market



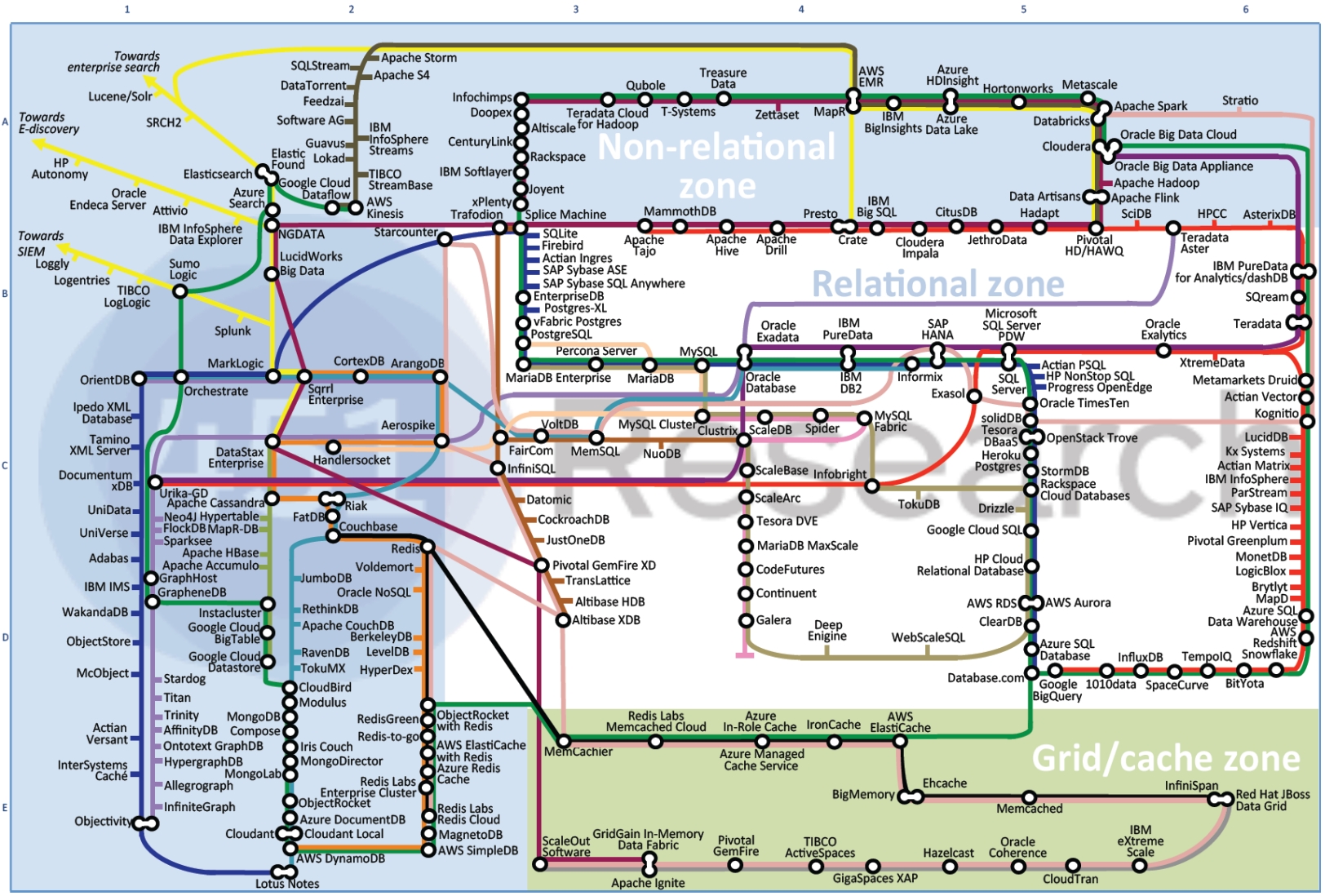
Source: IDC, Bernstein analysis

Relational Database market



Source: IDC, Bernstein analysis

Map of data platforms today



451 Research

Data Platforms Map

June 2015

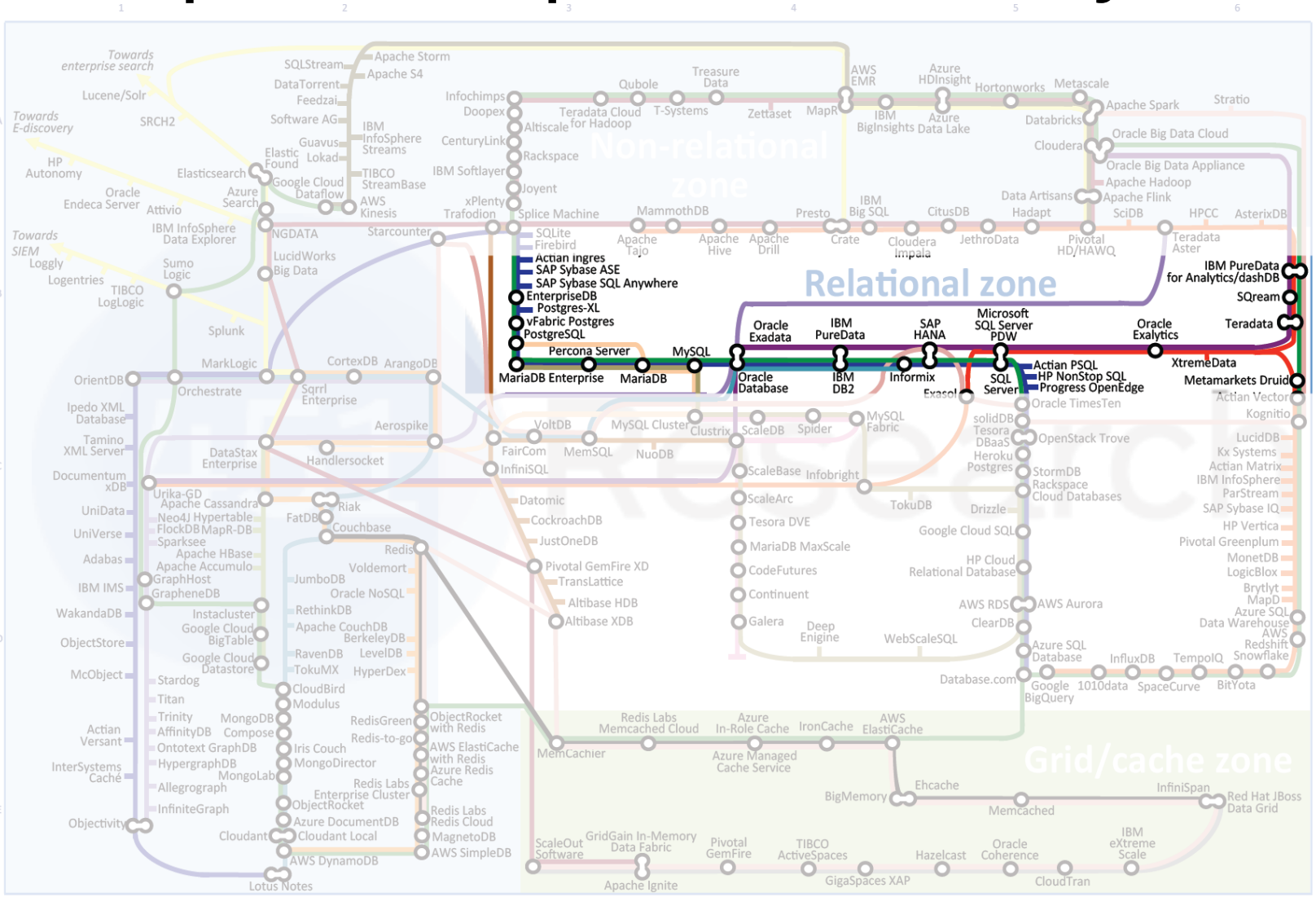
Key:

- General purpose
- Specialist analytic
- -as-a-Service
- BigTables
- Graph
- Document
- Key value stores
- Key value direct access
- Hadoop
- MySQL ecosystem
- Advanced clustering/sharding
- New SQL databases
- Data caching
- Data grid
- Search
- Appliances
- In-memory
- Stream processing

https://451research.com/dashboard/dpa

Over 200 Platforms

Map of data platforms today



451 Research

Data Platforms Map

June 2015

Key:

- General purpose
- Specialist analytic
- -as-a-Service
- BigTables
- Graph
- Document
- Key value stores
- Key value direct access
- Hadoop
- MySQL ecosystem
- Advanced clustering/sharding
- New SQL databases
- Data caching
- Data grid
- Search
- Appliances
- In-memory
- Stream processing

https://451research.com/dashboard/dpa

Over 200 Platforms

DBMS technology is changing

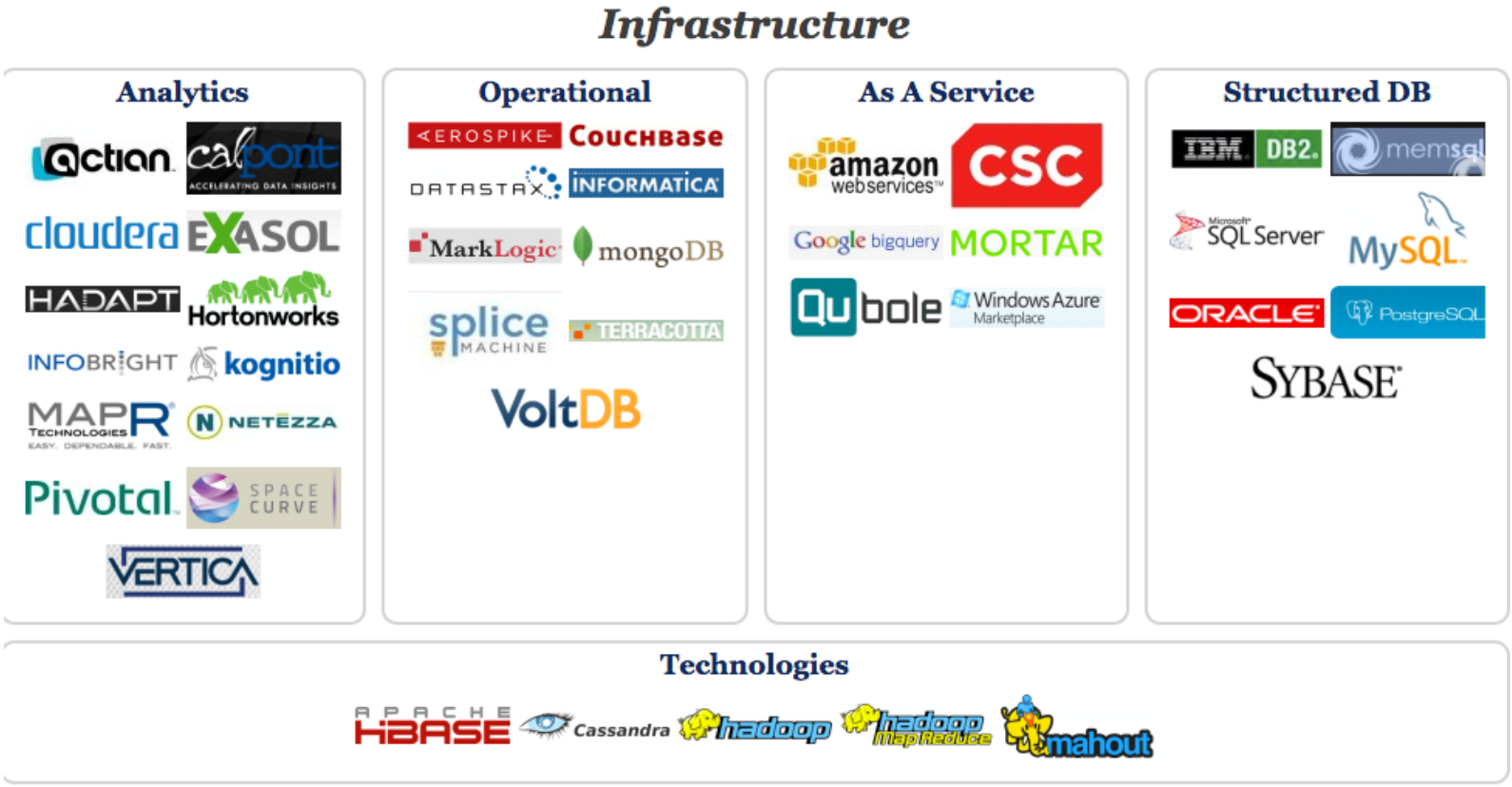
Driven by:

- **Hardware** trends: cheap *RAM & SSDs*
- Need to **scale**: *storage* and *transactions*
- New **data-types**: *text*, *json*, image, …
- New **workloads**: *machine learning & analytics*

New DBMS technologies are emerging:

- **NoSQL**: *abandon relational model*
- **Map-Reduce**: *distributed batch analytics*
- *…*

Big Data landscape… infrastructure is changing



http://www.bigdatalandscape.com/

***New* tech. *Same* Principles.**

What this course is (and is not)

- Discuss **fundamentals of data management**
  - How to design databases, query databases, build applications with them.
  - How to debug them when they go wrong!
  - Not how to be a DBA or how to tune Oracle 12g.

- We'll cover **how database management systems work**

- And some (but not all of) **the principles of how to build** them

What this course is (and is not)

- Also introduce **basics of data mining**
  - Basic data analytics
  - Basic machine learning in a distributed environment
- We'll cover **how large-scale data analysis work**
- Not how to use a **specific** machine learning toolkit

You will learn …

- Data Oriented Programming with SQL
- Foundations of Data System Design
  - Storage, indexing, and query processing
- Transactions
  - Concurrency, Consistency, and Recovery
- Applications
  - Analytics & Machine Learning
  - Data modeling

Course objective

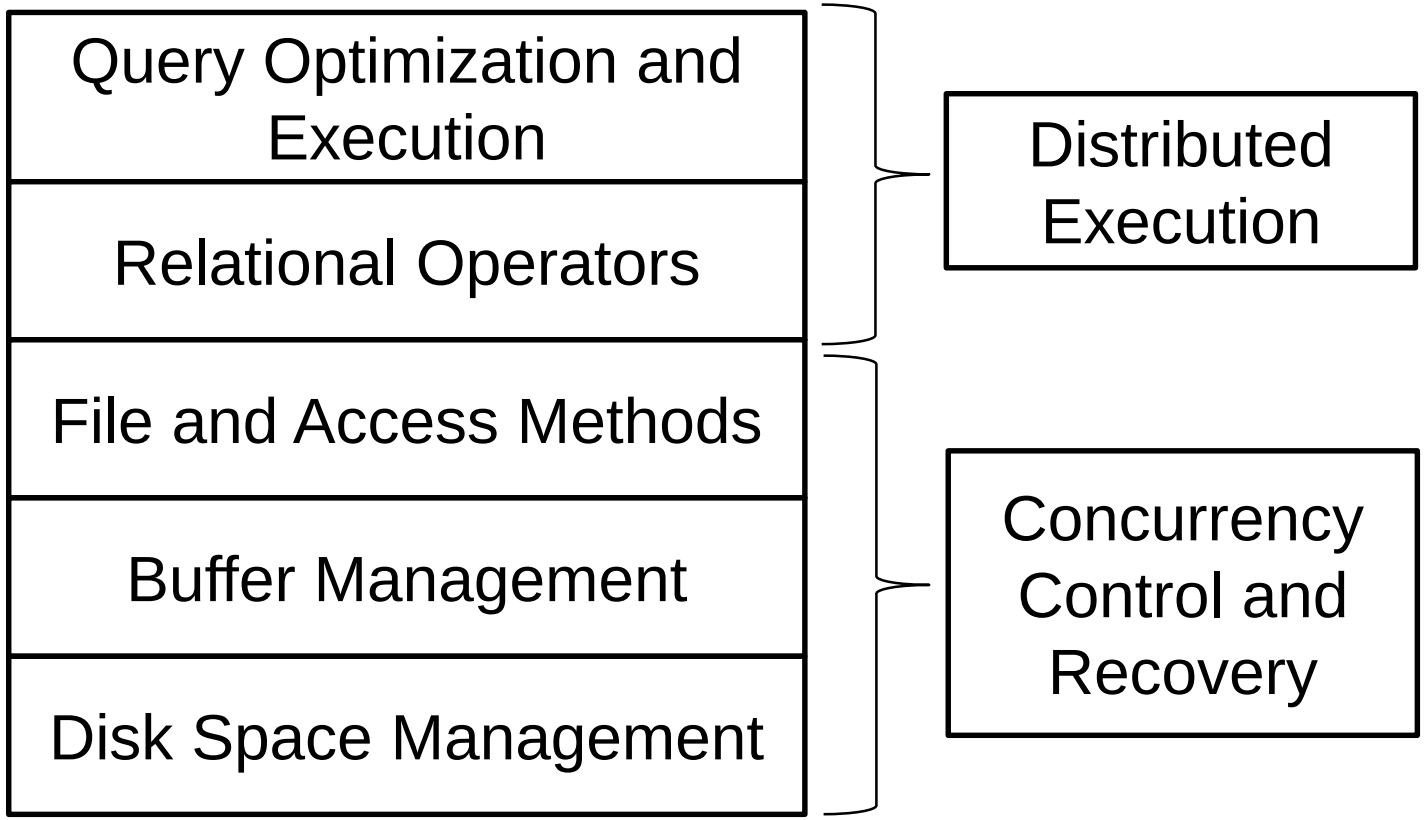
Focus: **Foundational System Principles**

- Reusable Ideas and Components
- Compositional Approach

You will be able to **use** existing & **build new** DBMS technologies!

Systems

Examine various levels of a database system



Plan: I. from a user's perspective

1. **Foundations:** Relational data models & SQL
   - How to manipulate data with SQL, a declarative language
     - *reduced expressive power but the system can do more for you*
2. **Database Design**: Design theory and constraints
   - Designing relational schema to keep your data from getting corrupted
3. **Transactions:** Syntax & supporting systems
   - A programmer's abstraction for data consistency

Plan: II. understanding how it works

4. Introduction to database systems

- Indexing
- External Memory Algorithms (IO model) for sorting, joins, etc.
- Basics of query optimization (Cost Estimates)
- Relational algebra

5. Specialized and New Data Processing Systems

- Key-Value Stores
- Hadoop and map-reduce
- SparkSQL. The re-rise of SQL
- Next-gen analytics systems & current intersections with ML & AI

Plan: III. data analytics and mining

6. Introduction to data mining

- Data clustering
- Dimensionality reduction
- Classification
- Regression
- Deep learning
- Next-gen analytics systems & current intersections with ML & AI

Subject to Change …

Who are we

- Xuming He, Associate Professor in SIST
- Research Interests
  - Computer Vision
  - Machine Learning
  - Homepage: https://xmhe.bitbucket.io
  - Lab website: http://plus.sist.shanghaitech.edu.cn

TAs

- Jiangwei Xie
- Jiale Zhou
- TBA

How? Administrivia, cont.

- All class communication via Piazza
  - https://piazza.com/shanghaitech.edu.cn/spring2019/cs150 announcements and discussion
  - **Slides/Reference post after/before lectures**
  - ***Homework/Project also announced on Piazza***

  - post all questions/comments there
  - *direct email is not a good idea*

How? Administrivia, cont.

- Reference
  - *Database Management Systems, 3rd Edition*
    - Ramakrishnan and Gehrke
  - Suggested
    - I wouldn't buy any more textbooks
    - Website/HWs will have links to programming resources
- Cheating policy: zero tolerance
  - We have the technology…

How? Workload

- Homework with real world focus:
  - SQL Programming
  - Distributed Dataflow with PySpark
  - Building Transactions
- Short weekly quizzes in discussion section
  - Mandatory attendance
- Course project on data mining
- Exams – 1 Midterms & 1 Final

Grading Policy

- Problem Sets (20%) + Quizs (20%)
- Course project (25%)
- Midterm (15%)
- Final exam (20%)

Assignments are typically due Tuesday before class, typically 2 weeks to complete

Jupyter Notebook “Hello World”

- Jupyter notebooks are interactive shells which **save output in a nice notebook format**
  - They also can display markdown, LaTeX, HTML, js…

FYI: “Jupyter Notebook” are also called iPython notebooks but they handle other languages too.

- You’ll use these for
  - in-class activities
  - interactive lecture supplements/recaps
  - homeworks, projects, etc.- if helpful!

Note: you **do need to know or learn python** for this course!

Jupyter Notebook setup

1. **HIGHLY RECOMMENDED.** Install **on your laptop** via the instructions on Piazza

2. Other options running via one of the alternative methods:
    1. **Ubuntu VM.**

Please help out your peers by posting issues / solutions on Piazza!

As a general policy in upper-level CS courses, **Windows is not officially supported**. However we are making a best-effort attempt to provide some solutions here!

2. Overview of the relational data model

What you will learn about in this section

1. Definition of DBMS
2. Data models & the relational data model
3. Schemas & data independence

What is a DBMS?

- A large, integrated collection of data
- Models a real-world *enterprise*
  - *Entities* (e.g., Students, Courses)
  - *Relationships* (e.g., Alice is enrolled in 145)

A **Database Management System (DBMS)** is a piece of software designed to store and manage databases

A motivating, running example

- Consider building a course management system (**CMS**):
  - Students
  - Courses
  - Professors

  } *Entities*

  - Who takes what
  - Who teaches what

  } *Relationships*

Data models

- A **data model** is a collection of concepts for describing data
  - The relational model of data is the most widely used model today
    - Main Concept: the *relation*- essentially, a table
- A **schema** is a description of a particular collection of data, **using the given data model**
  - E.g. every *relation* in a relational data model has a *schema* describing types, etc.

Modeling the CMS

- *Logical Schema*
  - Students(sid: *string*, name: *string*, gpa: *float*)
  - Courses(cid: *string*, cname: *string*, credits: *int*)
  - Enrolled(sid: *string,* cid*: string,* grade*: string*)

Relations

sid	Name	Gpa
101	Bob	3.2
123	Mary	3.8

Students

sid	cid	Grade
123	564	A

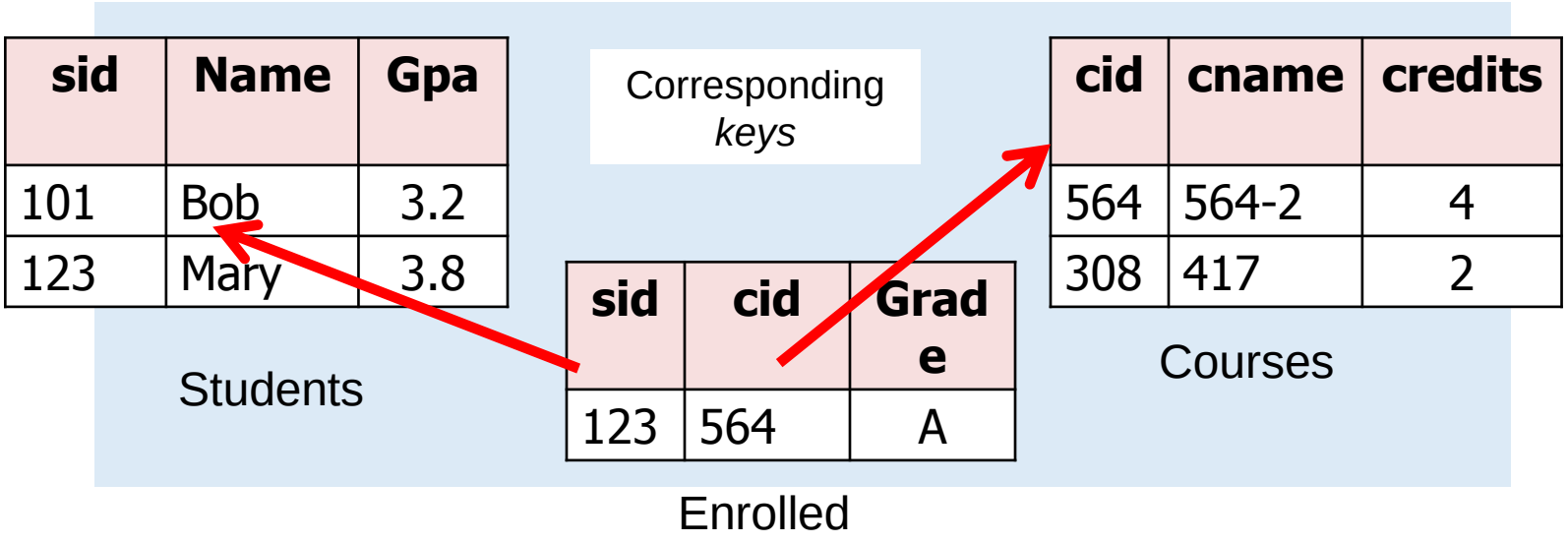
Enrolled

cid	cname	credits
564	564-2	4
308	417	2

Courses

Modeling the CMS

- *Logical Schema*
  - Students(sid: *string*, name: *string*, gpa: *float*)
  - Courses(cid: *string*, cname: *string*, credits: *int*)
  - Enrolled(sid: *string,* cid*: string,* grade*: string*)



Other Schemata…

- *Physical Schema*: describes data layout
  - Relations as unordered files
  - Some data in sorted order (index)

Administrators

- *Logical Schema:* Previous slide

Applications

- *External Schema*: (Views)
  - Course_info(cid: *string*, enrollment: *integer*)
  - Derived from other tables

Data independence

Concept: Applications do not need to worry about *how the data is structured and stored*

Logical data independence: protection from changes in the *logical structure of the data*

I.e. should not need to ask: can we add a new entity or attribute without rewriting the application?

Physical data independence: protection from *physical layout changes*

I.e. should not need to ask: which disks are the data stored on? Is the data indexed?

One of the most important reasons to use a DBMS

Summary

- Motivation: DBMS
- Overview of the relational data model
- Next: Key problems in DBMS and SQL

3. SQL Introduction & Definitions

What you will learn about in this section

1. What is SQL?
2. Basic schema definitions
3. Keys & constraints intro
4. ACTIVITY: CREATE TABLE statements

SQL: “Intergalactic Dataspeak”

- Developed @IBM Research in the 1970s
  - System R project
  - Vs. Berkeley’s Quel language (Ingres project)
- Commercialized/Popularized in the 1980s
  - IBM beaten to market by a startup called Oracle
- Questioned repeatedly
  - 90’s: OO-DBMS (OQL, etc.)
  - 2000’s: XML (Xquery, Xpath, XSLT)
  - 2010’s: NoSQL & MapReduce
- SQL keeps re-emerging as the standard
  - Even Hadoop, Spark etc. see lots of SQL
  - May not be perfect, but it is useful

SQL Pros and Cons

- Declarative!
  - Say *what* you want, not *how* to get it
- Implemented widely
  - With varying levels of efficiency, completeness
- Constrained
  - Not a Turing-complete language
- General-purpose and feature-rich
  - many years of added features
  - extensible: callouts to other languages, data sources

SQL Introduction

SQL stands for
Structured Query Language

- SQL is a standard language for querying and manipulating data
- SQL is a **very high-level** programming language
  - This works because it is optimized well!
- Many standards out there:
  - ANSI SQL, SQL92 (a.k.a. SQL2), SQL99 (a.k.a. SQL3), ….
  - Vendors support various subsets

NB: Probably the world's most successful **parallel** programming language (multicore?)

SQL is a…

- Data Definition Language (DDL)
  - Define relational *schemata*
  - Create/alter/delete tables and their attributes

- Data Manipulation Language (DML)
  - Insert/delete/modify tuples in tables
  - Query one or more tables – discussed next!

Tables in SQL

Product

PName	Price	Manufacturer
Gizmo	$19.99	GizmoWorks
Powergizmo	$29.99	GizmoWorks
SingleTouch	$149.99	Canon
MultiTouch	$203.99	Hitachi

A **relation** or **table** is a multiset of tuples having the attributes specified by the schema

Let's break this definition down

Tables in SQL

Product

PName	Price	Manufacturer
Gizmo	$19.99	GizmoWorks
Powergizmo	$29.99	GizmoWorks
SingleTouch	$149.99	Canon
MultiTouch	$203.99	Hitachi

A **multiset** is an unordered list (or: a set with multiple duplicate instances allowed)

List: [1, 1, 2, 3]
Set: {1, 2, 3}
Multiset: {1, 1, 2, 3}

i.e. no *next()*, etc. methods!

Tables in SQL

Product

PName	Price	Manufacturer
Gizmo	$19.99	GizmoWorks
Powergizmo	$29.99	GizmoWorks
SingleTouch	$149.99	Canon
MultiTouch	$203.99	Hitachi

An **attribute** (or **column**) is a typed data entry present in each tuple in the relation

*NB: Attributes must have an **atomic** type in standard SQL, i.e. not a list, set, etc.*

Tables in SQL

Product

PName	Price	Manufacturer
Gizmo	$19.99	GizmoWorks
Powergizmo	$29.99	GizmoWorks
SingleTouch	$149.99	Canon
MultiTouch	$203.99	Hitachi

*Also referred to sometimes as a **record***

A **tuple** or **row** is a single entry in the table having the attributes specified by the schema

Tables in SQL

Product

PName	Price	Manufacturer
Gizmo	$19.99	GizmoWorks
Powergizmo	$29.99	GizmoWorks
SingleTouch	$149.99	Canon
MultiTouch	$203.99	Hitachi

The number of tuples is the **cardinality** of the relation

The number of attributes is the **arity** of the relation

Data Types in SQL

- Atomic types:
  - Characters: CHAR(20), VARCHAR(50)
  - Numbers: INT, BIGINT, SMALLINT, FLOAT
  - Others: MONEY, DATETIME, …

- Every attribute must have an atomic type
  - Hence tables are flat

Table Schemas

- The **schema** of a table is the table name, its attributes, and their types:

Product(Pname: *string*, Price: *float*, Category: *string*, Manufacturer: *string*)

- A **key** is an attribute whose values are unique; we underline a key

Product(Pname: *string*, Price: *float*, Category: *string*, Manufacturer: *string*)

Key constraints

A **key** is a **minimal subset of attributes** that acts as a unique identifier for tuples in a relation

- A key is an implicit constraint on which tuples can be in the relation

Students(sid:string, name:string, gpa: float)

  - i.e. if two tuples agree on the values of the key, then they must be the same tuple!

1. Which would you select as a key?
2. Is a key always guaranteed to exist?
3. Can we have more than one key?

NULL and NOT NULL

- To say “don’t know the value” we use NULL
  - NULL has (sometimes painful) semantics, more detail later

Students(sid:string, name:string, gpa: float)

sid	name	gpa
123	Bob	3.9
143	Jim	NULL

Say, Jim just enrolled in his first class.

In SQL, we may constrain a column to be NOT NULL, e.g., “name” in this table

General Constraints

- We can actually specify arbitrary assertions
  - E.g. "*There cannot be 25 people in the DB class*"

- In practice, we don't specify many such constraints. Why?
  - Performance!

Whenever we do something ugly (or avoid doing something convenient) it's for the sake of performance

Summary of Schema Information

- Schema and Constraints are how databases understand the semantics (meaning) of data
- They are also useful for optimization
- SQL supports general constraints:
  - Keys and foreign keys are most important
  - We'll give you a chance to write the others