These lecture notes include some material from Professors Bertossi, Kolaitis, Guagliardo, Libkin and Baghban Karimi

Database History and an Introduction to Relational Data Model

Lecture Handout

Dr Eugenia Ternovska

Simon Fraser University

Data

The **most important asset** of any enterprise

- Companies, universities, websites use it in their internal organization
- Companies gather and use it to understand their customers
- "Free" apps gather it to profile you ...

Data and AI

- For training AI systems, you need lots of (high quality) data
- Cleaning and managing data is a crucial part of ML training
- These days: ML directly in databases

Data

Data enables and supports decision making

So, it must be effectively, efficiently and reliably

- collected and stored
- maintained and updated
- processed and analyzed

to be turned into meaningful information

Dealing with lots of data efficiently is called **Data Management**

What is a database?

> *A collection of data items, related to a specific enterprise, which is structured and organized so as to be more easily accessed, managed, and updated*

What is a Database Management System (DBMS)?

- system software for creating and managing databases
  - mediates interaction between end-users (or applications) and the database
  - ensures that data is consistently organized and remains easily accessible

Database Management Systems

A database management system (DBMS) provides **support** for:

- At least one data model (a mathematical abstraction for representing data);
- At least one high level data language (language for defining, updating, manipulating, and retrieving data);
- Access control (limit access of certain data to certain users);
- Resiliency (ability to recover from crashes);
- Integrity and consistency of information;
- Transaction management;
- Concurrent access to information: so that different transactions can simultaneously access and modify the database
- Possibility of building applications on top:
  Writing and running programs that interact with the database

Why use a DBMS?

A DBMS allows one to take advantage of all those features, and, at the same time, to provide

- Uniform data administration
- Efficient access to resources
- Data independence
- Reduced application development time

Different kinds of databases

Relational databases ⇐ **main focus of this course**
Data organized in tables (relations) with typed attributes

Document stores (XML)
Text documents structured using tags (or other markers)

Graph databases
Data organized in graph structures with nodes and edges

Key-value stores
Data organized in associative arrays (a.k.a. dictionaries or maps)

Graph databases ⇐ **increasingly important**
Data organized in graph structures with nodes and edges

...

Data Models and Data Languages

- A data model is a mathematical formalism for describing and representing data.
- A data model is accompanied by a data language that has two parts:
- A **data definition language (DDL)** has a syntax for describing database templates in terms of the underlying data model.
- A **data manipulation language (DML)** supports the following operations on data:
  - Insertion
  - Deletion
  - Update
  - Retrieval and extraction of data (query the data).
- The first three operations are fairly standard. However, there is much variety on data retrieval and extraction (query languages).

Logical Separation between Data and Programs

1950s, early 60s (Pre-history): Data tapes and punch cards

1960s, early 70s

The structure of data was **embedded** in the data manipulation programs, in a file-based system (or data models before the relational)

Any change in the structure of data implied change of the programs and vice versa

It all changed with the invention of **Relational Databases**

Relational Databases: How it Started

The history of relational databases is the history of a **scientific and technological revolution**.

The scientific revolution started in 1970 by Edgar (Ted) F. Codd at the IBM San Jose Research Laboratory (now the IBM Almaden Research Center)

Codd introduced the relational data model and two database query languages: **relational algebra** and **relational calculus**.

"A relational model for data for large shared data banks", CACM, 1970.

"Relational completeness of data base sublanguages", in: Database Systems, ed. by R. Rustin, 1972.

Brief History:

1961: First DBMS: *Integrated Data Store* of GE

1962: IBM and AA develop SABRE

1966-1969: IBM develops *Information Management System* (IMS). Uses hierarchical model.

1970: Edgar Codd (IBM) proposes the relational model of data, and specifies how a relational DBMS could be built accordingly

1975: First international conferences ACM SIGMOD and VLDB

1976: Peter Chen introduces the ER model

70s: Development of first RDBMS (relational DBMS): more on it on the next slide

80s: DBMSs for PCs (DBASE, Paradox, etc.). Relational model has become competitive

Relational Databases: the Early Years

In the 1970th, researchers at the IBM San Jose Laboratory embark on the System R project, the first implementation of a relational DBMS. Showed viability of the model.

In 1974-1975, they develop SEQUEL, a query language that eventually became the industry standard **SQL**.

System R evolved to DB2 – released first in 1983.

M. Stonebraker and E. Wong embark on the development of the Ingres RDBMS at UC Berkeley in 1973.

Ingres is commercialized in 1983; later, it became **PostgreSQL**, a free software OODBMS (object-oriented DBMS).

L. Ellison founds a company in 1979 that eventually becomes Oracle Corporation; Oracle V2 is released in 1979 and Oracle V3 in 1983.

1981: Edgar F. Codd receives the ACM Turing Award

For his fundamental and continuing contributions to the theory and practice of database management systems.
He ***originated the relational approach to database management*** *in a series of research papers published commencing in 1970.*
His paper "A Relational Model of Data for Large Shared Data Banks" was a seminal paper, in a continuing and carefully developed series of papers.

The contribution had impact on numerous related areas, including database languages, query subsystems, database semantics, locking and recovery, and inferential subsystems.

About ACM Turing Award

A.M. Turing Award given by the Association for Computing Machinery

ACM's **most prestigious technical award** is accompanied by a prize of $250,000.

It is given to an individual selected for contributions of a technical nature made to the computing community.

The contributions should be of lasting and major technical importance to the computer field.

Financial support of the Turing Award is provided by the Intel Corporation and Google Inc.

http://awards.acm.org/homepage.cfm?srt=all&awd=140

Brief History (Cont.)

1985: Preliminary publication of standard for SQL. Object Oriented DBMSs. Client/server architectures. Distributed DBs.

90s: New functionalities: Spatial DBs, Temporal DBs, active rules (or triggers), Deductive DBs, OO DBs

Multimedia DBs, Data Mining, Data Warehouses Decision support for querying re-emerged

1998: **Jim Gray receives the ACM Turing Award**

For seminal contributions to database and transaction processing research and technical leadership in system implementation.

Brief History (Cont.)

2000s: Internet boom, larger data volume and faster updates that could not be handled by single machine

Unstructured & semi-structured data has become increasingly important:

- XML – data-exchange standard,
- JSON – more compact data-exchange standard, suitable for storing objects from JavaScript and other programming languages
- spacial data – support for geographic systems

Open-source systems PostgreSQL, mySQL saw an increasing use

Brief History (Cont.)

2000s: Not Only SQL (NoSQL)

Non-Relational data models (document, key-value), due to the need for rapid development (startups)

- Document stores (Data Model: JSON) – Example Systems: SimpleDB, CouchBase, MongoDB
- Column stores (Data Model: Big Table) – Example Systems: Hbase, Cassandra, HyperTable
- Key-value stores (Data Model: Hash) - Example Systems: DynamoDB, Riak, Redis, Membase

Custom APIs instead of SQL, usually open source

Lack of Standardization: 100+ NoSQL systems

Light form of data management, "eventual data consistency" only

Brief History (Cont.)

2000s: MapReduce – to deal with huge volumes of data

facilitated the use of parallelism by programmers

a Program model rather than a database system

With time, support of its features **migrated into traditional DB systems**

Graph DBs (Data Model: Graph) – in response to the rapid development of social platforms

Example Systems: Neo4J, InfoGrid, GraphBase

Can be viewed as a part of NoSQL movement

Brief History (Cont.)

2010–2020s: Outsorcing data storage and management

- Cloud services
- Data delivered to users via web-based services

Significant savings in cost

Data breach and security issues

Unresolved data ownership & individual data privacy concerns

Rapid development of the graph data model

ISO now works on stardartization of Graph Query Languages

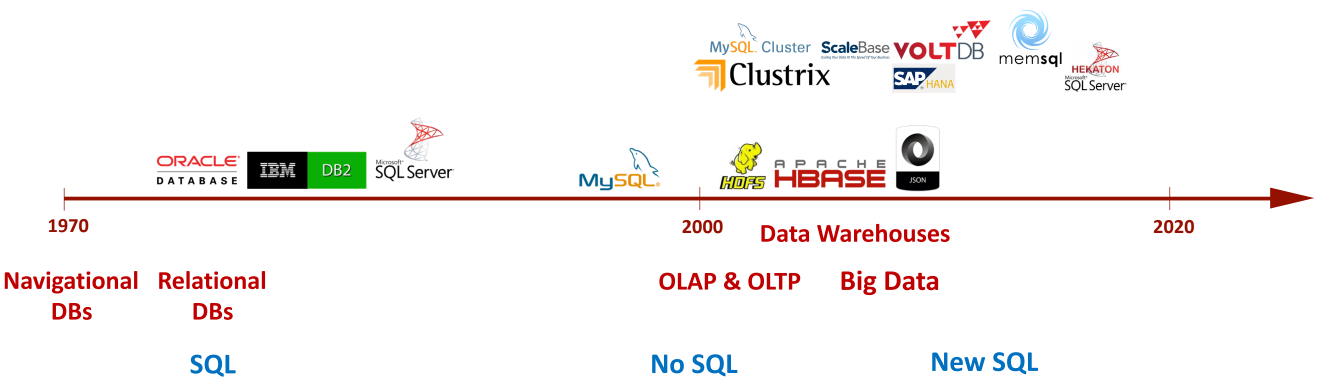


Figure by Ouldooz Baghban Karimi

Relational Database Industry Today

According to Gartner, Inc., June 2007:

"Worldwide relational database management systems (RDBMS) total software revenue totalled \$15.2 billion in 2006, a 14.2 percent increase from 2005 revenue

In 2007, the total RDBMS software revenue increased to \$17.1 billion

In 2015, \$35.9 billion, more than doubled compared to 2007

In 2023, **\$60 billion** industry

The Relational Data Model (E.F. Codd – 1970)

The Relational Data Model uses the mathematical concept of a relation as the formalism for describing and representing data.

Question: What is a relation?

Answer:

- Formally, a relation is a subset of a cartesian product of sets.
- Informally, a relation is a "table" with rows and columns.

Customer

CustID	Name	City	Address
cust1	Renton	Edinburgh	2 Wellington Pl
cust2	Watson	London	221B Baker St
cust3	Holmes	London	221B Baker St

Relations and Attributes

Note: a relation (in the mathematical sense) can be viewed as a table with k columns

Example

Table Sales:

Customer$_{25}$	0	3672	45	28	3672	3	67
Customer$_{2}$	9	8392	88	72	7292	8	23

In the relational data model, we want to have names for the columns; these are the **attributes** of the relation.

Relation Schemas and Relational Database Schemas

A k-ary **relation schema** $R(A_1, A_2, \ldots, A_K)$ is a set $A_1, A_2, \ldots, A_k$ of k attributes.

Example

COURSE(course-no, course-name, term, instructor, room, time)
CITY-INFO(name, state, population)

Thus, a k-ary relation schema is a "blueprint", a "template" for some k-ary relation.

An **instance** of a relation schema is a **relation** conforming to the schema (arities match; also, in DBMS, data types of attributes match).

A **relational database schema** is a set of relation schemas for all relations

A **relational database instance** of a relational schema is a set of relations each of which is an instance of the relation schema

Relational Database Schemas - Examples

BANKING relational database schema with relation schemas

CHECKING-ACCOUNT(branch, acc-no, cust-id, balance)
SAVINGS-ACCOUNT(branch, acc-no, cust-id, balance)
CUSTOMER(cust-id, name, address, phone, email)
. . . .

UNIVERSITY relational database schema with relation schemas

STUDENT(student-id, student-name, major, status)
FACULTY(faculty-id, faculty-name, dpt, title, salary)
COURSE(course-no, course-name, term, instructor)
ENROLLS(student-id, course-no, term)
. . . .

Schemas vs. Instances

Keep in mind that there is a **clear distinction** between

- relation schemas and instances of relation schemas

and between

- relational database schemas and relational database instances.

Important Difference:

Syntactic Notion	**Semantic Notion (discrete mathematics notion)**
Relation Schema	Instance of a relation schema (i.e., a relation)
Relational Database Schema	Relational database instance (i.e., a database)

Query Languages

Used to ask questions (queries) to a database

Procedural
Specify a **sequence of steps** to obtain the expected result

Declarative
Specify **what** you want, not **how** to get it

- Queries are typically asked in a **declarative** way
- DBMSs figure out internally how to translate a query into procedures that are suitable for getting the results

Query Languages for the Relational Data Model

Codd introduced two different query languages for the relational data model:

Relational Algebra, which is a procedural language.
It is an *algebraic* formalism in which queries are expressed by applying a sequence of operations to relations.

Relational Calculus, which is a declarative language.

It is a *logical* formalism in which queries are expressed as formulas of classical first-order logic

Codd's first contribution, stated informally: Relational Algebra and Relational Calculus are **essentially equivalent** in terms of expressive power

Desiderata for a Database Query Language

The language should be sufficiently high-level to secure physical data independence, i.e., the separation between the physical level and the conceptual level of databases.

The language should have **high enough expressive power** to be able to pose useful and interesting queries against the database.

The language should be **efficiently implementable** to allow for the fast retrieval of information from the database.

Important:

There is a **tension** between the last two desiderata.
Increase in expressive power comes at the expense of efficiency.

Relational Algebra (& Calculus)

strike a **good balance** between expressive power and efficiency.

Codd's key contribution was to identify a small set of basic algebraic operations on relations and to demonstrate that useful and interesting queries can be expressed by combining these operations.

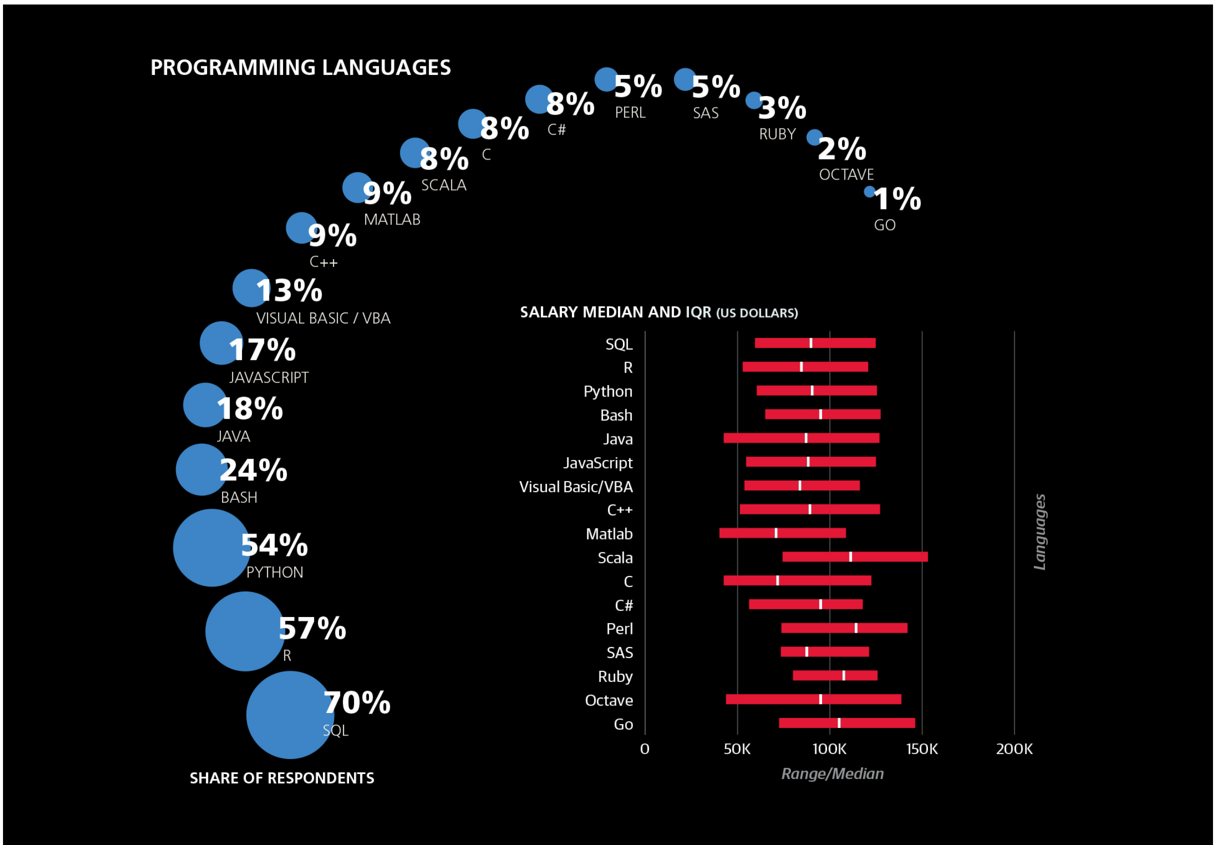
- He provided a **rich enough language**, even though, as we will see later on, it suffers from certain limitations in terms of expressive power.

The first RDBMS prototype implementations (System R and Ingres) demonstrated that the relational algebra operations **can be implemented efficiently**

SQL

This is how we interact with relational databases in practice

- **S**tructured **Q**uery **L**anguage
- Declarative language for querying relational databases
- A "programmer version" of logic (Relational Calculus)
- Implemented in all major (free and commercial) RDBMSs
- SQL was first **standardized** in 1986
  (ISO = International Organization for Standardization);
  Standards: SQL-86, SQL-89, SQL-92, SQL:1999, SQL:2003, SQL:2008, SQL:2011, SQL:2016, next SQL:2023 (instead of 2021) Standards are **not** publicly available
- Most common tool used by **data scientists**



Source: O'Reilly Data Science Salary Survey 2016

Conclusion

Relational Model provides

- Simple, limited approach to **structuring data**
- Yet **reasonably versatile**, so anything can be modelled

- A limited collection of **operations on data**
- Yet **useful** collection of operations

(good balance between expressiveness and efficiency of queries)

Tables **do not** prescribe how they are implemented or stored on disk

- This is called physical data independence

Acknowledgements

[1] Database Systems: The Complete Book, 2nd EditionHector Garcia-Molina, Jeffrey D. Ullman, Jennifer WidomPrentice Hall, 2009

[2] Database System Concepts, Seventh EditionAvi Silberschatz, Henry F. Korth, S. SudarshanMcGraw-Hill, March 2019www.db-book.com

Additional references and resources used in preparation of this course are listed on
https://canvas.sfu.ca/courses/77505/pages/references-and-resources
or mentioned in slides.