CS383

Databases in Practice

Topics

- Databases
  - SQL — PostgreSQL
  - NoSQL — Mongo
  - Text-only databases -- Information retrieval
  - Spreadsheets
- Database access
  - CRUD (create, read, update, and delete),
  - ODBC (open database connectivity)
  - Interface creation
    - HTML & javascript
    - Node.js
- Other DB topics
  - transactions, concurrency, indexing
  - (maybe) Big Data, High Availability

Course Details

- https://cs.brynmawr.edu/cs383gt
  - Class website
    - Lab times / Office Hours

- 165.106.10.133 -- loin.cs.brynmawr.edu
  - Class server
    - All software used in class is on this server
    - **All** work will be done on this server
      - Homework 1
      - Lab 1

Presentation Topics

Basic SQL Queries (UML diagrams?) ch 3.3-3.4	Jan 30	
More SQL ch 3.5-3.8	Jan 30	
SQL Joins ch 4.1	Feb 1	
JDBC / ODBC (Java, Python, …) ch 5.1	Feb 6	
URI (Also get/post)	Feb 8	
HTML	Feb 8	
Intro to Javascript	Feb 13	
Intro to the DOM — updating DOM via javascript	Feb 13	
Javascript closures and first class functions	Feb 15	
Javascript, passing in functions for later execution / JSON	Feb 15	
Javascript and server communication,	Feb 20	
Javascript short polling, long polling and web sockets	Feb 22	
Linking Node.js to PostgreSQL	Feb 27	
Basic DDL (in PostgreSQL) ER-Modeling ch 6	Feb 29	
DB Normalization ch 7.3, 7.7	Mar 5	
MongoDB Queries	Mar 21	
MongoDB Embedded Documents	Mar 21	
indexing mongo	Mar 26	
Linking Mongo to Node	Mar 28	
Using Postgres as a document store (mongo equivalent)	April 2	
Concurrency (ch 18) in Postgres (ch 32))	April 2	
B-Trees and Other data models underlying DBMS (ch 14.3,4,5)	April 2	
Transactions (ch 17)	April 2	
Sharding (ch 10.2.2 — will require considerable outside information	April 4	
Map/Reduce (ch 10.3)	April 4	
XML and queries on XML xpath, xquery (online 30)		
Alternate DB models — Excel — the DB functions	April 9	
Alternate DB Models — Google BigTable, Apache Cassandra (ch 10.6?)	April 11	

There are lots of ways to store data

- Flat text files
  - fixed width fields
  - Delimited, e.g., CSV, ...
- Structured Text FIles
  - JSON
  - XML
- Binary formats
  - Java Objects, C bits
- Image-based formats:
  - QR
  - GeoTiff, ...

```
27.08 50.15 98.72 57.62
70.31 43.93 66.87 86.02
72.18  7.41 87.81 92.04
26.47 27.28 98.89 44.58
14.66 37.28 93.62 61.84
68.10 11.17 91.69 41.61
```

```
54.4,90.40,4.746,37.169050714
72.9,37.6,96.7205,51.5027943330
70.0,84.6,53.617,7.7
34.7,50.132,92.9177,17.07535291
79.9,11.06,38.20521,47.5
82.9,69.0,87.1345433,17.8180
```

```
try (ObjectOutputStream oos = new ObjectOutputStream(
       new FileOutputStream("AA.out"))) {
            oos.writeObject(als);
        } catch (Exception ee) {
            System.err.println(ee);
        }
```

```
FILE* f = fopen("wb.bin", “r");
char c[20];
r = fwrite(c, 20, sizeof(char), f);
```

Database Systems

- Contain information [about a particular enterprise]
  - Collection of interrelated data
  - Set of programs to access the data
  - An environment that is both *convenient* and *efficient* to use
- Manage collections of data that are:
  - Highly valuable
  - Relatively large -- what is "large"
  - Accessed by multiple users and applications, often at the same time.
- are complex software systems whose task is to manage a [large] collections of data.

	Bytes
Megabyte	1,000,000
Gigabyte	1,000,000,000
Terabyte	1,000,000,000,000
Petabyte	1,000,000,000,000,000
Exabyte	1,000,000,000,000,000,000
Zettabyte	1,000,000,000,000,000,000,000
Yottabyte	1,000,000,000,000,000,000,000,000

My Mac -- 16GB RAM

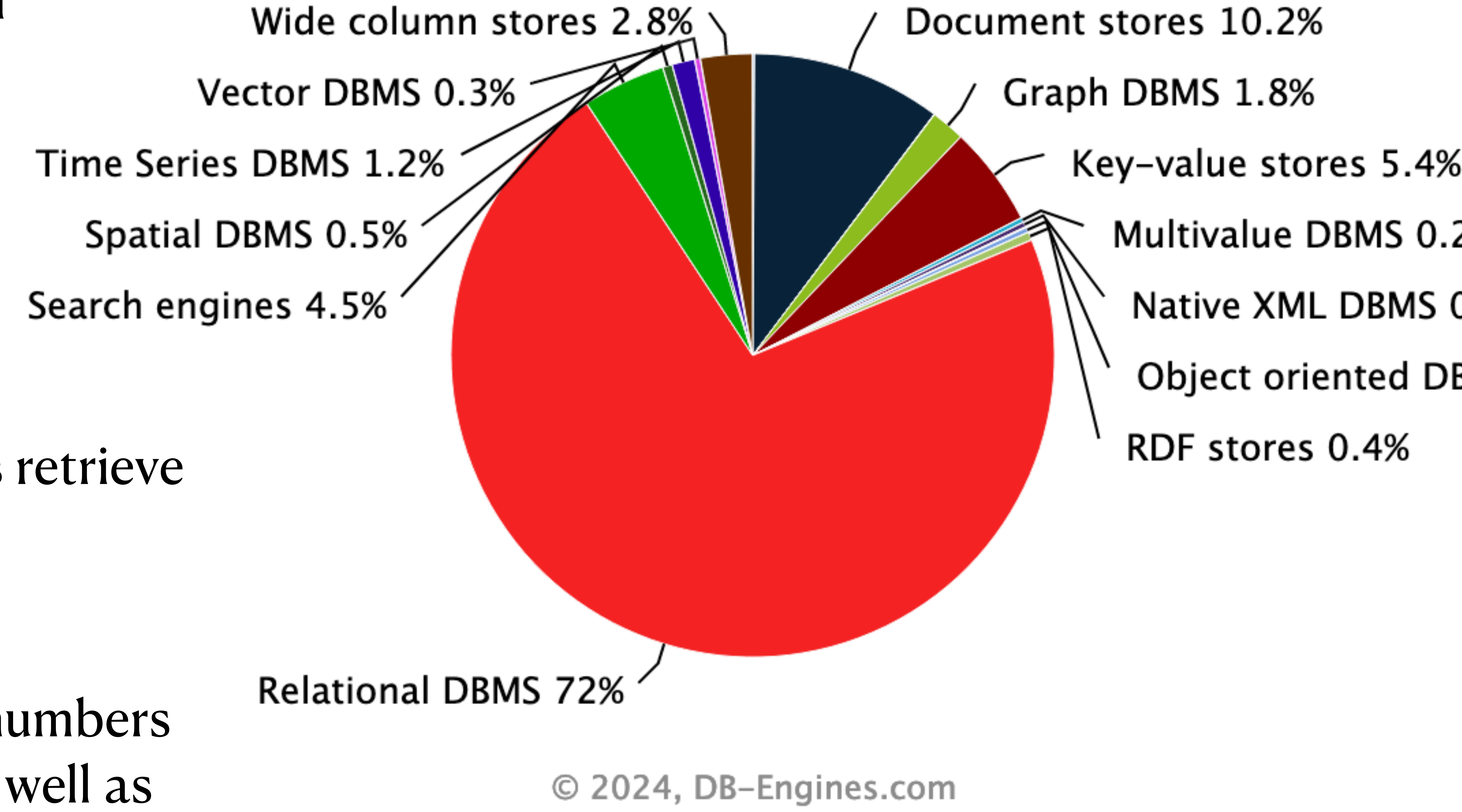
My Mac -- 1TB disk

DataBases -- Why

- With databases you are not reading/writing files, rather you are asking another process for information
  - Other examples of doing this
- Advantages/Disadvantages

DataBases Types

- Relational
  - relational --table-oriented -- data model
- Document store
  - schema-free organization of data
    - usually JSON
- Key Value stores
  - store pairs of keys and values, as well as retrieve values when a key is known
- Wide Column stores
  - data records have ability to hold large numbers of dynamic columns. Column names as well as the record keys are not fixed (a 2d key-value store)



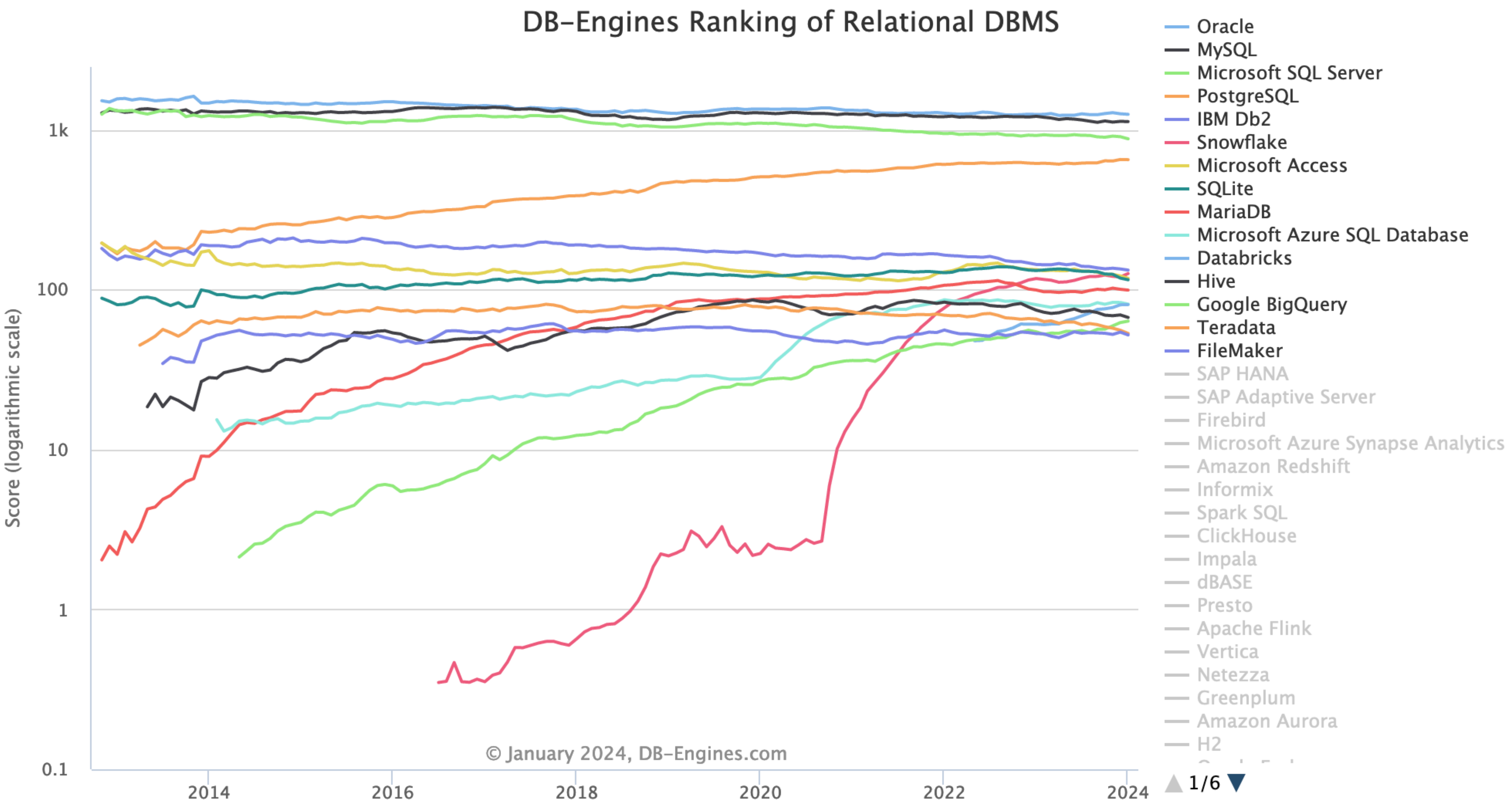
Ranking various DBMS

417 systems in ranking, January 2024

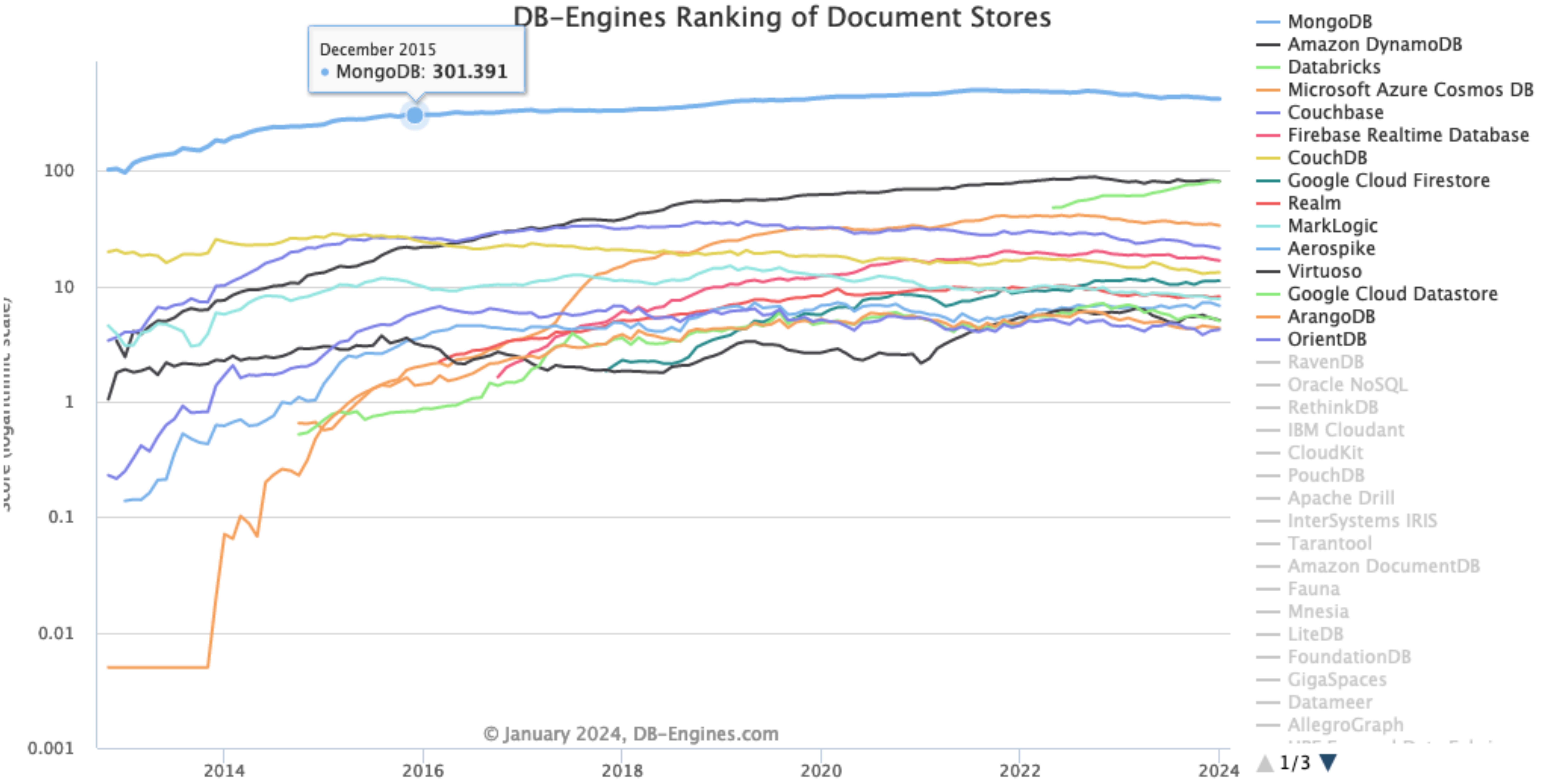
Rank			DBMS	Database Model	Score		
Jan 2024	Dec 2023	Jan 2023			Jan 2024	Dec 2023	Jan 2023
1.	1.	1.	Oracle	Relational, Multi-model	1247.49	-9.92	+2.33
2.	2.	2.	MySQL	Relational, Multi-model	1123.46	-3.18	-88.50
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	876.60	-27.23	-42.79
4.	4.	4.	PostgreSQL	Relational, Multi-model	648.96	-1.94	+34.11
5.	5.	5.	MongoDB	Document, Multi-model	417.48	-1.67	-37.70
6.	6.	6.	Redis	Key-value, Multi-model	159.38	+1.03	-18.17
7.	7.	↑ 8.	Elasticsearch	Search engine, Multi-model	136.07	-1.68	-5.09
8.	8.	↓ 7.	IBM Db2	Relational, Multi-model	132.41	-2.19	-11.16
9.	↑ 10.	↑ 11.	Snowflake	Relational	125.92	+6.04	+8.66
10.	↓ 9.	↓ 9.	Microsoft Access	Relational	117.67	-4.08	-15.69
11.	11.	↓ 10.	SQLite	Relational	115.20	-2.75	-16.29
12.	12.	12.	Cassandra	Wide column, Multi-model	111.04	-1.16	-5.27

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

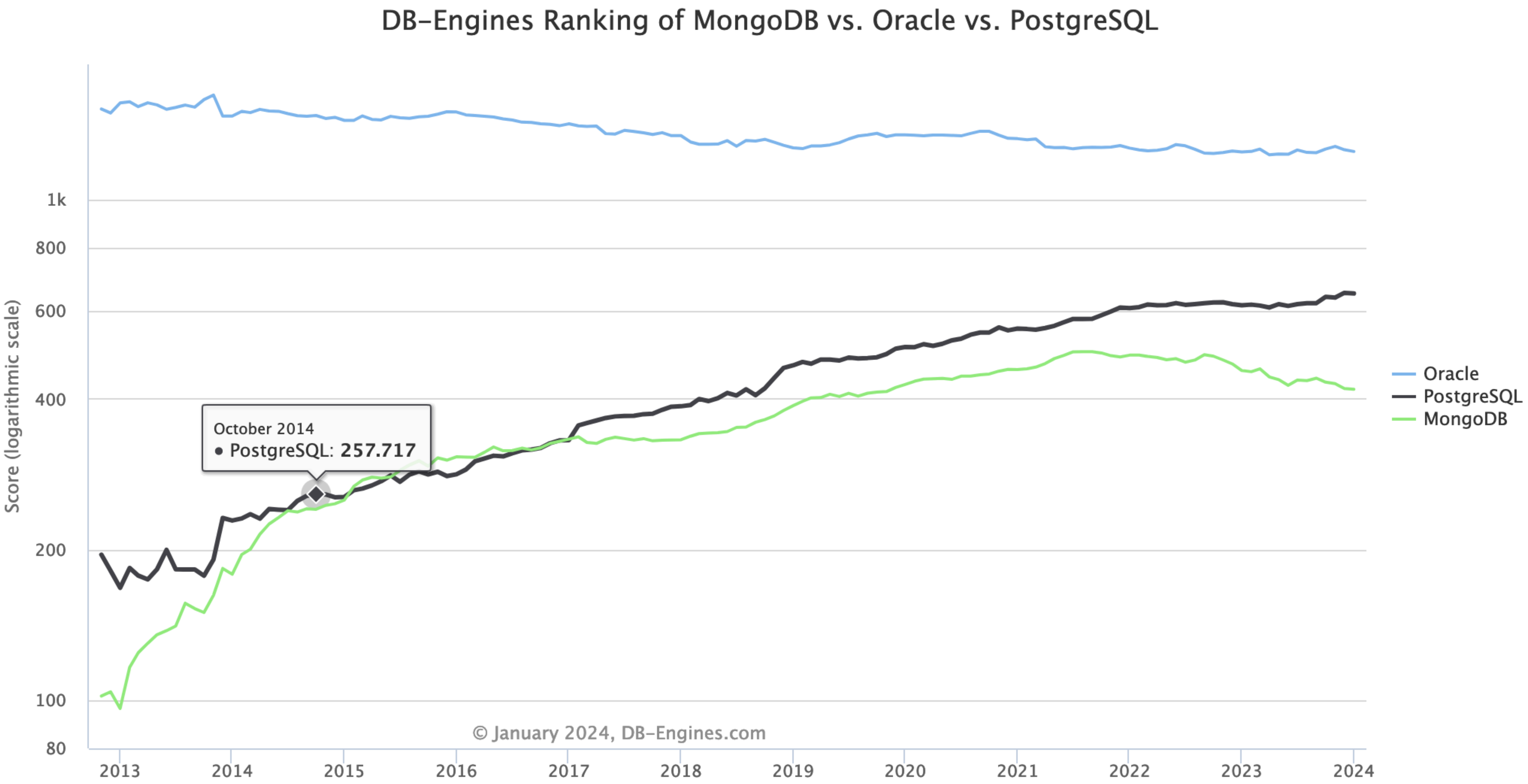
Relational Databases ranked by usage



DocumentStore Databases ranked by usage



DBMS -- in this class*



Database Applications Examples

- Enterprise Information
  - Sales: customers, products, purchases
  - Accounting: payments, receipts, assets
  - Human Resources: Information about employees, salaries, payroll taxes.
- Manufacturing: management of production, inventory, orders, supply chain.
- Banking and finance
  - customer information, accounts, loans, and banking transactions.
  - Credit card transactions
  - Finance: sales and purchases of financial instruments (e.g., stocks and bonds; storing real-time market data
- Universities: registration, grades

Issues addressed by Database Systems

- Data redundancy and inconsistency: data is stored in multiple file formats resulting in duplication of information in different files
- Difficulty in accessing data
  - Need to write a new program to carry out each new task
- Data isolation
  - Every file is considered independently from every other file
- Integrity problems
  - Integrity constraints (e.g., account balance > 0) become “buried” in program code rather than being stated explicitly
  - Hard to add new constraints or change existing ones

More Issues

- Atomicity of updates
  - Failures may leave database in an inconsistent state with partial updates carried out
  - Example: Transfer of funds from one account to another should either complete or not happen at all
- Concurrent access by multiple users
  - Concurrent access needed for performance
  - Uncontrolled concurrent accesses can lead to inconsistencies
    - Ex: Two people reading a balance (say 100) and updating it by withdrawing money (say 50 each) at the same time
- Security problems
  - Hard to provide user access to some, but not all, data

Central Features of DBMS

- **Feature 1: Physical Data Independence** – the ability to modify the physical schema without changing the logical schema
  - Applications depend on the logical schema
  - In general, the interfaces between the various levels and components should be well defined so that changes in some parts do not seriously influence others.
  - “physical schema”
  - “logical schema”
  - ~ Java / OO Encapsulation
    - physical == instance variables
    - logical == accessor methods

DBMS interface

- Feature 2: Interface

- 2 parts
  - Data Defintion Language
    - what the data looks like
      - NoSQL DBMS may not have this at all
  - Data Manipulation Language
    - most of CRUD

Data Definition Language (DDL) Applies to only RDMS

- Specification notation for defining the database schema

Example:

```
create table instructor (
    ID          char(5),
    name        varchar(20),
    dept_name   varchar(20),
    salary      numeric(8,2))
```

- DDL compiler generates a set of table templates stored in a ***data dictionary***
- Data dictionary contains metadata (i.e., data about data)
  - Database schema
  - Integrity constraints
    - Primary key (ID uniquely identifies instructors)
  - Authorization
    - Who can access what

Data Models

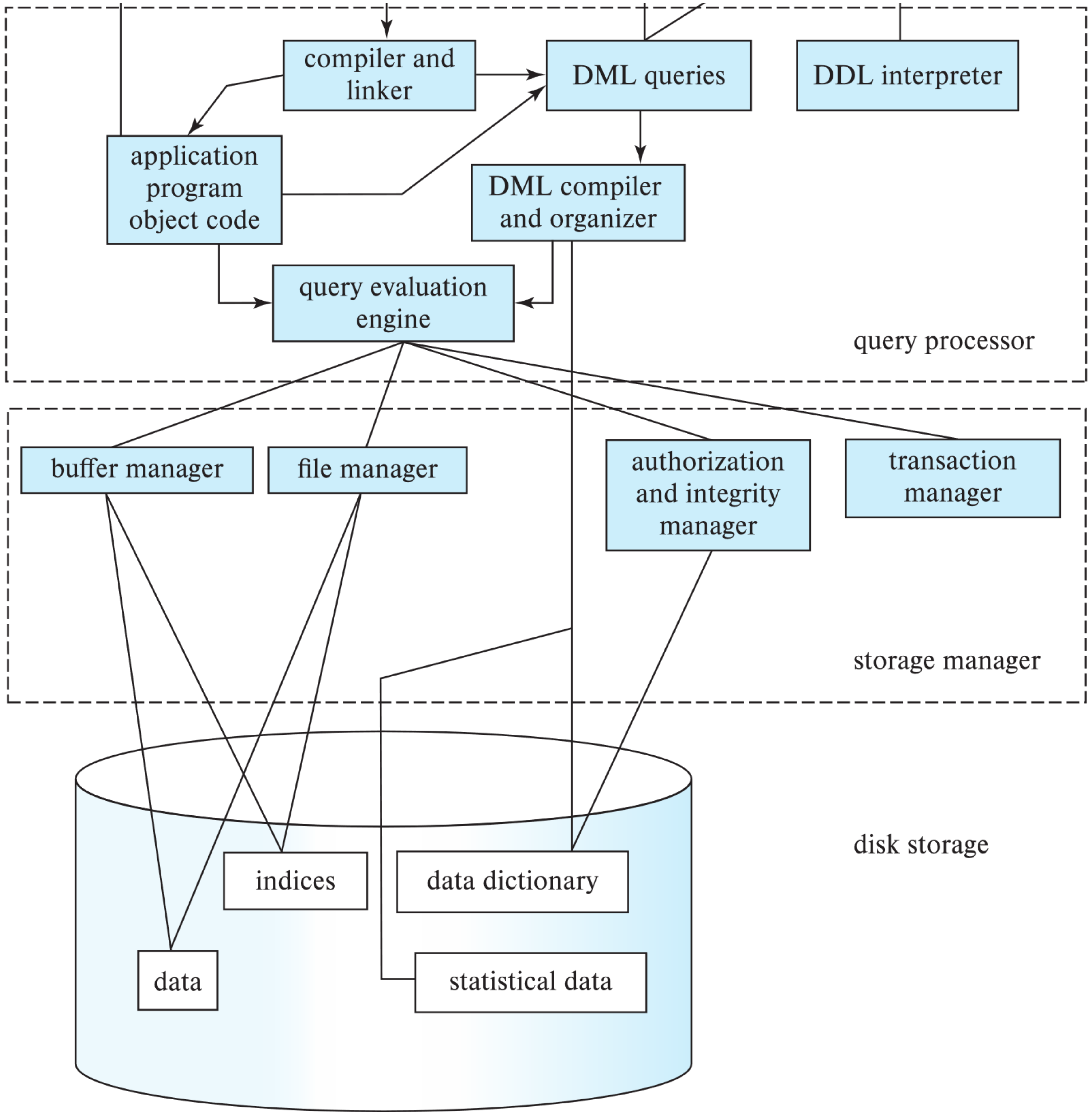
- A collection of tools for describing
  - Data
  - Data relationships
  - Data semantics
  - Data constraints
- Relational model
- Object-based data models (Object-oriented and Object-relational)
- Semi-structured data model (XML, JSON)

- Data models are expressed by the Data Definition Language

Data Manipulation Language (DML)

- Language for accessing and updating the data organized by the appropriate data model
  - DML also known as query language
- There are basically two types of data-manipulation language
  - **Procedural DML** -- require a user to specify what data are needed and how to get those data.
  - **Declarative DML** -- require a user to specify what data are needed without specifying how to get those data.
- Declarative DMLs are usually easier to learn and use than are procedural DMLs.
  - both Postgres and Mongo have declarative DMLs

Database Architecture (Centralized/Shared-Memory)



Storage Manager

- A program module that provides the interface between the low-level data stored in the database and the application programs and queries submitted to the system.
- The storage manager is responsible to the following tasks:
  - Interaction with the OS file manager
  - Efficient storing, retrieving and updating of data
- The storage manager components include:
  - Authorization and integrity manager
  - Transaction manager
  - File manager
  - Buffer manager

Storage Manager (Cont.)

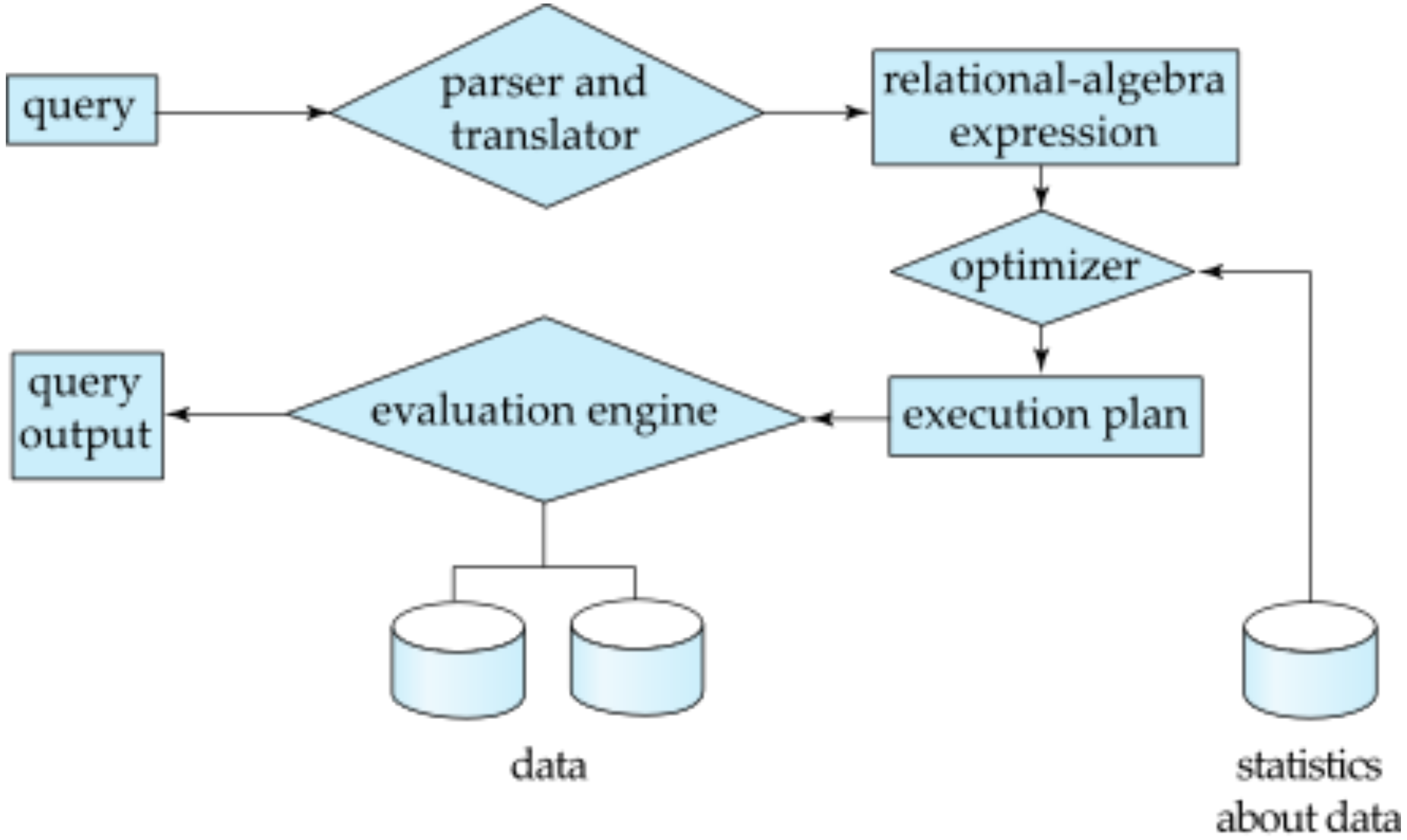
- The storage manager implements several data structures as part of the physical system implementation:
  - Data files -- store the database itself
  - Data dictionary -- stores metadata about the structure of the database, in particular the schema of the database.
  - Indices -- can provide fast access to data items. A database index provides pointers to those data items that hold a particular value.

Query Processor

- The query processor components include:
  - DDL interpreter -- interprets DDL statements and records the definitions in the data dictionary.
  - DML compiler -- translates DML statements in a query language into an evaluation plan consisting of low-level instructions that the query evaluation engine understands.
    - The DML compiler performs query optimization; that is, it picks the lowest cost evaluation plan from among the various alternatives.
  - Query evaluation engine -- executes low-level instructions generated by the DML compiler.

Query Processing

1. Parsing and translation
2. Optimization
3. Evaluation



Transaction Management

- A **transaction** is a collection of operations that performs a single logical function in a database application
- **Transaction-management component** ensures that the database remains in a consistent (correct) state despite system failures (e.g., power failures and operating system crashes) and transaction failures.
- **Concurrency-control manager** controls the interaction among the concurrent transactions, to ensure the consistency of the database.

- **The transaction management system** can be thought of as surrounding the Query Processor

Database Applications

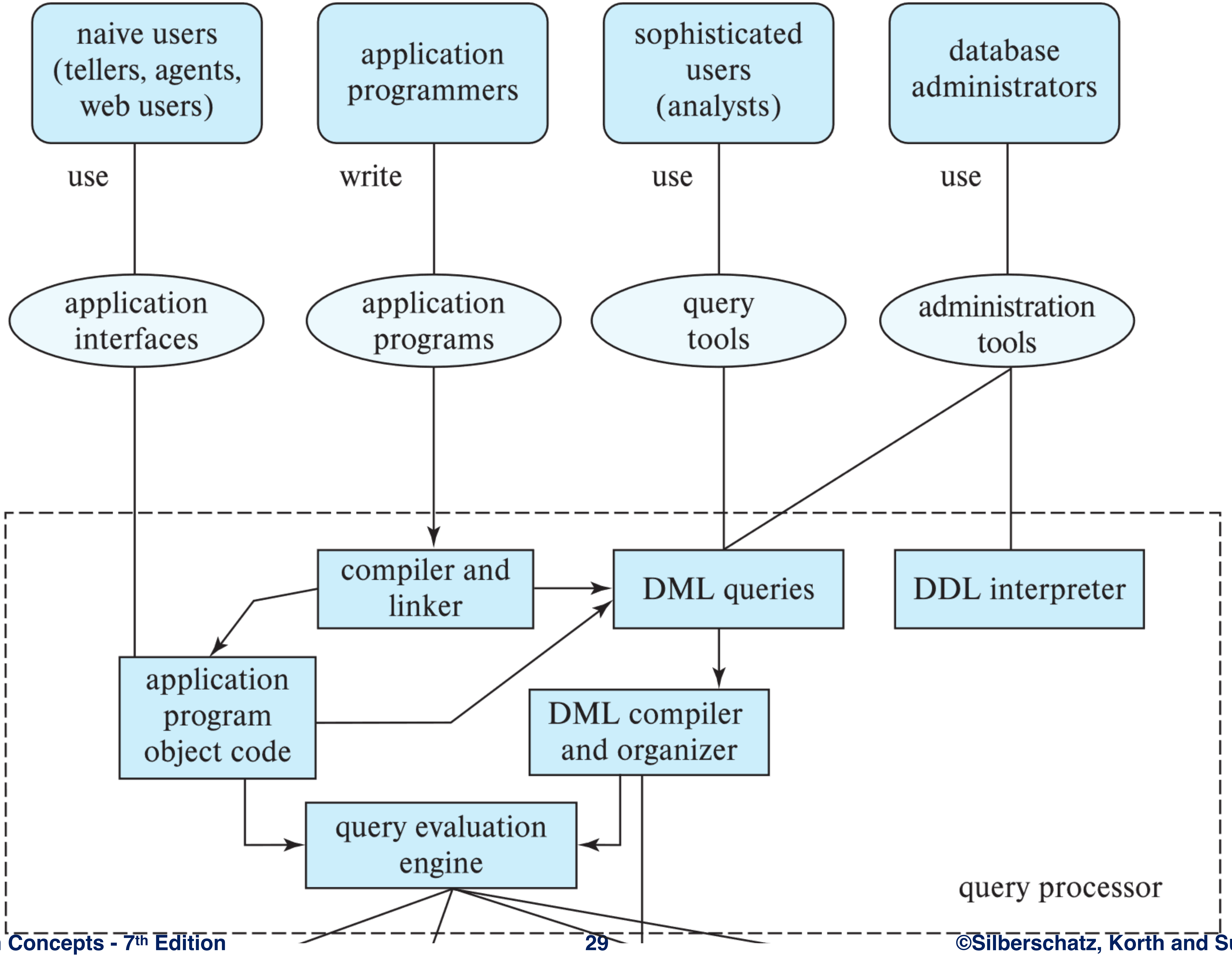
Database applications are usually partitioned into two or three parts

- Two-tier architecture -- the application resides at the client machine, where it invokes database system functionality at the server machine
- Three-tier architecture -- the client machine acts as a front end and does not contain any direct database calls.
  - The client end communicates with an application server, usually through a forms interface.
  - The application server in turn communicates with a database system to access data.

Rare, now

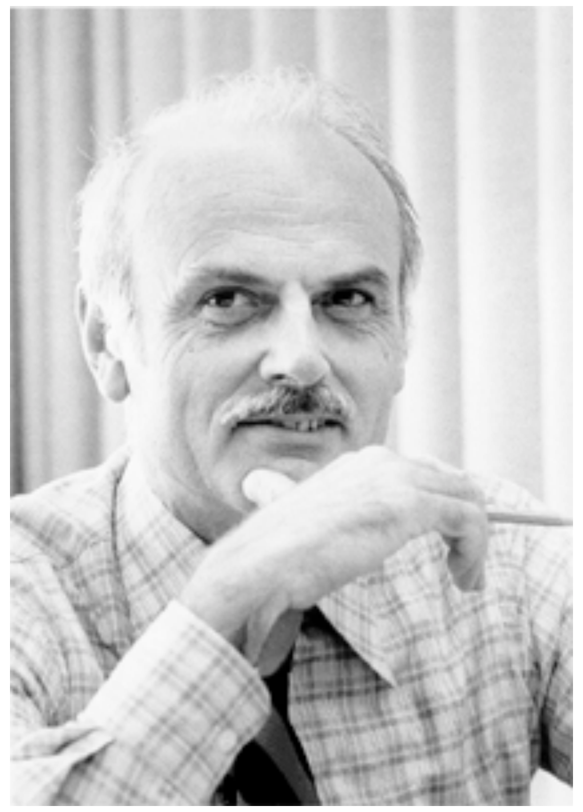
You will be building one of these

Database Users



History of Database Systems

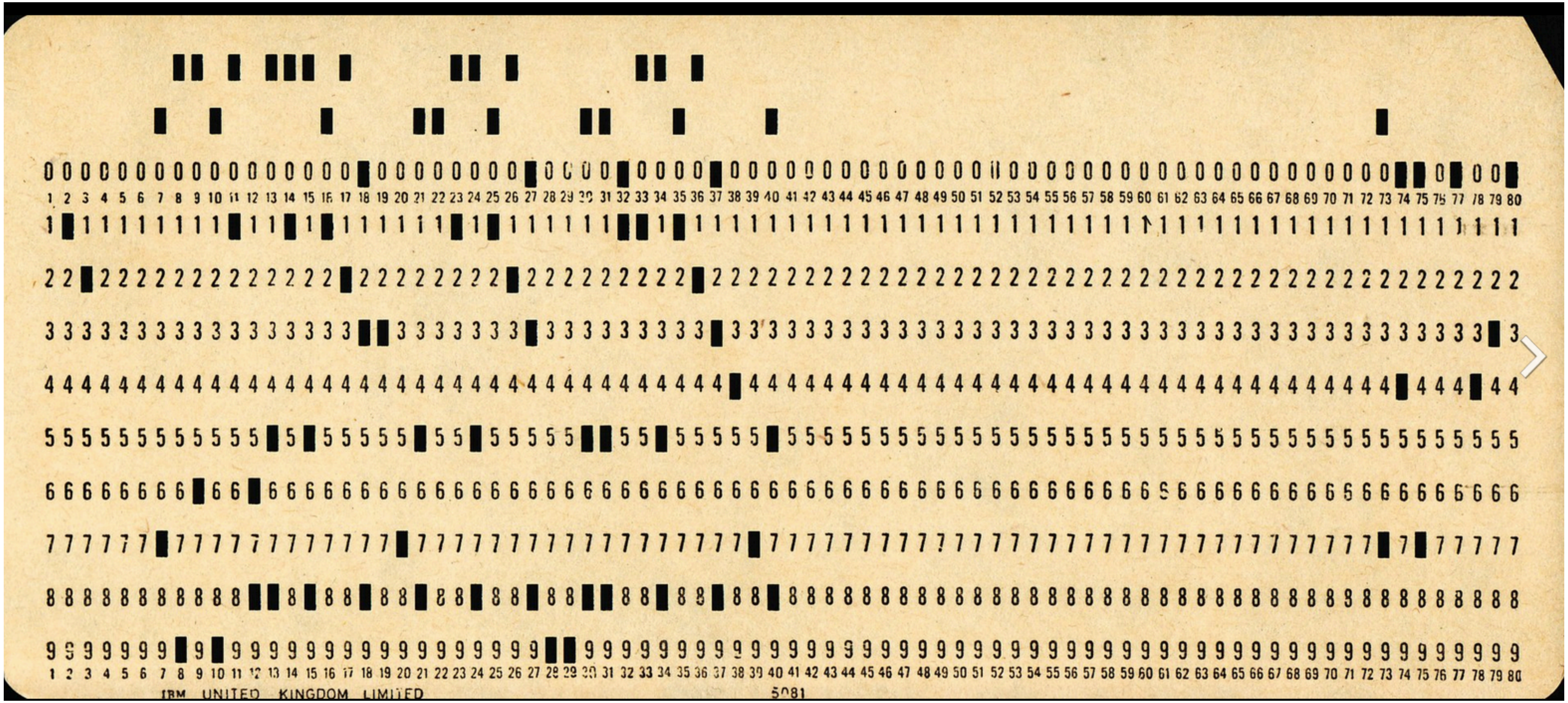
- Pre 1950
  - Punch cards and Hollerith Machines
- 1950s and early 1960s:
  - Data processing using magnetic tapes for storage
    - Tapes provided only sequential access
  - Punched cards for input
- Late 1960s and 1970s:
  - Hard disks allowed direct access to data
  - Network and hierarchical data models in widespread use
  - Ted Codd defines the relational data model
    - Would win the ACM Turing Award for this work
    - IBM Research begins System R prototype
    - UC Berkeley (Michael Stonebraker) begins Ingres prototype
    - Oracle releases first commercial relational database
  - High-performance (for the era) transaction processing
  - Punched cards for input



1923--2003

1943--

Punch Card



History of Database Systems (Cont.)

- 1980s:
  - Research relational prototypes evolve into commercial systems
    - SQL becomes industrial standard
  - Parallel and distributed database systems
    - Wisconsin, IBM, Teradata
  - Object-oriented database systems
  - Punch cards finally die!!!!
- 1990s:
  - Large decision support and data-mining applications
  - Large multi-terabyte data warehouses
  - Emergence of Web commerce

History of Database Systems (Cont.)

- 2000s
  - Big data storage systems
    - Google BigTable, Yahoo PNuts,
    - “NoSQL” systems.
  - Big data analysis: beyond SQL
    - Map reduce and friends
- 2010s
  - SQL reloaded
    - SQL front end to Map Reduce systems
    - Massively parallel database systems
    - Multi-core main-memory databases