# Databases and information management

Lecture 1

Piotr Syga

2022-10-06

# Course Outline

Main goals:

Show different aspects of information management
Teach how to select the representation of the data
Teach how to aggregate and extract the information

Selected topics:

- SQL and relational data bases
- Designing database systems
- Database normalization
- NOSQL
- Security and privacy of the information
- Incorporating databases in the app

# Milestones

- Showing the difficulties of managing unorganized information
- Showing the value of information and gains of a proper information processing
- Showing the practical side of the storing data, data processing and analysis
- Pointing out the main differences between relational and non-relational databases
- Showing the means of optimizing the data aggregation process
- Presenting data-driven paradigm
- Allowing the students to produce a system with large amount of data operations – from the design to the implementation and testing

# Literature


- J.D. Ullman, J. Widom, A First Course in Database Systems.
- J.D. Ullman, Principles of Database Systems.
- G. Harrison, Next Generation Databases: NoSQL and Big Data.
- E. Redmont, Seven Databases in Seven Weeks: A Guide to Modern Databases and the NoSQL Movement.
- A. Petrov, Database Internals: A deep-dive into how distributed data systems work.
- M.J. Hernandez, Database Design For Mere Mortals A Hand-on Guide to Relational Database Design.
- H. Garcia-Molina, J.D. Ullman, J. Widom, Database Systems The Complete Book.
- D. Sullivan, NoSQL for Mere Mortals.
- T. Hills, NoSQL and SQL Data Modeling: Bringing Together Data, Semantics, and Software.
- S. Chellappan, D. Ganesan, MongoDB Security.
- C. Dwork, A. Roth, The Algorithmic Foundations of Differential Privacy.

# Grading

Option 1 – approx. 4 lists + designing and implementing small DB app (individually or in pairs; the design and the implementation is graded separately). All lists are graded, with a $-0.5$ penalty for each (started) fortnight after the deadline. To pass the class one must solve all the lists (even after the deadline).
The final grade is a (rounded) mean of all the grades.

# Grading

Option 2 – similarly (1 list less), but with a final quiz at the last lecture or 2 separate short quizzes. The final grade is equal to 30% of the quizzes, 40% of the project and 30% of the lists.

# Absence

Due to NeurIPS 2022, I'll be absent between 2022-11-26 and 2022-12-05, this means:

- Lecture scheduled to 2022-12-01 should be moved
- All laboratories scheduled to 2022-12-01 should be moved

Some decisions to make

**Lecture**

Should we make an additional lecture before the conference, in January or on February 3, 6 or 7? Which day of the week is the most appropriate? Depending on the possibilities should we make the stand in as a remote or on premise lecture?

**Laboratories**

Should we make the additional laboratories in January or on February 3 or if it is possible extend the classes to make up the 90 minutes? Which day of the week is the most appropriate?

# Operations

- the empty set, the power set and set of sets
- basic operations: complement, union, intersection, difference
- Cartesian product (and other products)
- de Morgan's laws
- deviation from the set structure: multisets and lists

# The motivation

- the importance of the information
- how to store the data
- how to (effectively) process the data

# Storing the data

We need

- data representation – e.g., data as a set, a document, a graph
- the operations for extracting specific piece of information both: selection of specific objects from the set and specific properties of the objects
- a manner of comparing and aggregating the knowledge
- rules describing the order and constraining the possibilities of the operations

# Storing the data



$\Rightarrow$

(white, 400.0, 48)

(yellow and black, 65.0, 100)

(brown, 1.5, 57)

(red, 23000.0, 160)

$\Rightarrow$

| color | weight | speed |
|---|---|---|
| string | float | int |

# Operations

Having a set $\mathcal{A}$ we want to define such operations $\mathtt{Op}_1(\cdot)$, $\mathtt{Op}_2(\cdot)$,... that applied to $\mathcal{A}$ produce some sets $\mathcal{A}'$, $\mathcal{A}''$, ...

- mathematical model – Relational algebra, e.g., $\sigma_{\mathsf{weight}>200}(\mathcal{A})$, $\pi_{\mathsf{color}}(\mathcal{A})$, $\pi_{\mathsf{speed}}(\sigma_{\mathsf{weight}<3 \vee \mathsf{weight}>150}(\mathcal{A}))$
- SQL:
  - a typed (all types are ordered), non-procedural programming language
  - included RA-based optimizer
  - memory management
  - additional procedural extensions and further functionalities in DBMS
- non-relational approach

# Base of a query

## Schema of a simple query

Select *columns*
From *table*
Where *condition*

# Base of a query

## Schema of a simple query

Select *columns*
From *table*
Where *condition*

## Example

Select Speed
From A
Where weight>150;

# Base of a query

## Schema of a simple query

Select *columns*
From *table*
Where *condition*

## Example

```
Select Distinct  speed
From             A
Where            weight>150;
```

# What we can SELECT?

## 'Displaying' the result

- entire row (Select *)
- a single column (Select speed)
- a group of columns (Select speed, color)
- a constant (Select 1)
- arithmetic operations on columns (Select 3*speed+weight-4)
- aggregations (Select MAX(speed))
- few more (string operations, 'custom' functions, . . . )

# FROM where do we select?

## Data sources

- a table or group of joined tables
- a derived table or a subquery
- a view

# FROM where do we select?

## JOINing tables

`SELECT` *columns* `FROM A JOIN B (ON` *condition*)

JOIN types:

- `CROSS JOIN` (,) — Cartesian product of all the tuples
- `INNER JOIN` — joining only the tuples that follow the condition (both in A and B)
- `LEFT` (outer) `JOIN` — all tuples from A are extended by B's as in `INNER JOIN` or by `NULL`
- `RIGHT` (outer) `JOIN` — all tuples from B are extended by A's as in `INNER JOIN` or by `NULL`
- `FULL JOIN` — records look like in `LEFT` and `RIGHT`, joining only the tuples that meet the condition

# Conditions

## Logical operators

- Basic comparison: `=`, `<>` (`!=`), `>`, `<`, `<=`, `>=`
- Conjunctions `AND`, `OR`, `NOT`
- `BETWEEN` a `AND` b
- `LIKE` '%*pattern*_'
- (`NOT`) `IN` *set*
- `IS` (`NOT`) `NULL`
- additional (dialect dependent) features (e.g., REGEX)

# Conditions

## Logical operators

- Basic comparison: `=`, `<>` (`!=`), `>`, `<`, `<=`, `>=`
- Conjunctions `AND`, `OR`, `NOT`
- `BETWEEN` a `AND` b
- `LIKE` '%*pattern*_'
- (`NOT`) `IN` *set*
- `IS` (`NOT`) `NULL`
- additional (dialect dependent) features (e.g., REGEX)

e.g., `SELECT` speed `FROM` A `WHERE` color `LIKE` '%and%';