Databases

TDA357/DIT622 – LP3 2024

Lecture 1

Ana Bove

(much of the material is based on material from both Thomas Hallgren and Jonas Duregård)

January 15th 2024

Overview of Today's Lecture

- Motivation;
- Organisation;
- Recap:
  - Sets;
  - Relations;
  - Functions.

What is a Database?

A *database* is a (probably huge) collection of data (efficiently) managed by a specialised software called a *Database Management System* (DBMS) that guarantees integrity constraints on data.

(contrary to using just file systems and a general purpose programming language)

Moreover the collection should be:

- *Structured* – data is stored in efficient structures;
- *Queryable* - one can query the data;
- *Generic* - data is accessible in different ways;
- *Persistent* – data is not lost without deliberate action;
- *Mutable* – data can be added/deleted/modified;
- *Concurrently accessed*.

Database Management Systems vs File Systems

File systems are structured, persistent and mutable ...
... but can be inefficient and 'bulky' to work with directly ...
... forces us to think about a lot of low-level issues that are already solved in a DBMS.

Some popular DBMSs:

- IBM DB2;
- Microsoft SQL Server;
- Oracle;
- PostgreSQL;
- MySQL;
- MariaDB;
- SQLite.

Why to Study Databases?

Because we encounter them everywhere/all the time!

- WWW - most web sites use a database under the hood;
- Finance - has driven the development of databases since the 60's;
- Industry - production control, test data, inventories, sales, ...
- Research - sensor data, biological data, ...
- Government - demographical data, SPAR, ...
- Public Transport - open data from Västtrafik.
- ...

Relational Databases

This is the main topic of this course.

Here, a database is basically a bunch of *tables* with columns and rows.

Name	**Abbr**	**Capital**	**Area**	**Population**	**Continent**	**Currency**
Denmark	DK	Copenhagen	43094	5484000	EU	DKK
Estonia	EE	Tallinn	45226	1291170	EU	EUR
Finland	FI	Helsinki	337030	5244000	EU	EUR
Norway	NO	Oslo	324220	5009150	EU	NOK
Sweden	SE	Stockholm	449964	9555893	EU	SEK
...	...	...	...	...	...	...

They might require significant design work in order to get it right!
(We will see some of the design issues during the course.)

Tables can be viewed as *mathematical relations*.
To combine different tables we use *Cartesean products*!

SQL: Structured Query Language

SQL is a standardized language for manipulating relational databases.

- Common language supported by lots of different DBMS;
- Use to create, manipulate and query databases;
- Arguably one of the most used computer languages in existence;
- Some people pronounce it "Sequel".

The Birth of the Relational Model

Unlike many discoveries, it is easy to pinpoint when databases started.

This is the first line from Edgar F. Codd's 1970 paper titled *A Relational Model of Data for Large Shared Data Banks*:

> *Future users of large data banks must be protected from having to know how the data is organized in the machine ...*

In a single paper, Codd outlines much of what we teach in this course today.

For his contributions he got the *Turing award* in 1981.

Other Database Models

The relational data model and **SQL** are so prevalent that other approaches are commonly referred to as NoSQL-databases:

- Semi-structured hierarchical models (XML, JSON, ...);

  We will see a bit of them in the course.

- Key-value stores (Oracle NoSQL, Riak, Cassandra, ...):
  - Easily distributed across multiple computers/data centers;
  - Very simplistic data model (maps and lists).

  The *Advanced Databases* course looks into some of these.

Usually NoSQL-databases are easier to design, sometimes more efficient, but also more limited when it comes to integrity constraints.

Course Content

Design of databases:

- Relational data model;
- Entity-relationship modelling;
- Functional dependencies and normalisation;
- Semi-structured data model.

Database programming:

- Data manipulation and querying in **SQL**;
- Relational algebra;
- Application programs in general purpose languages (like Java, Haskell or Python).

Database implementation: (just a bit)

- Indexes, transaction management, concurrency, data recovery, ...

Course Main Objectives

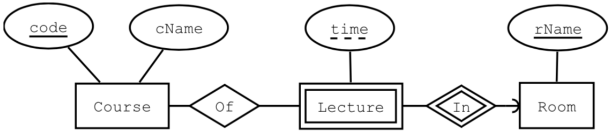
- *Design* a database that correctly models the domain and its constraints;
- *Construct* a database from a schema and related constraints;
- *Use* a database through queries and updates;
- Use a database from an external *application*.

We will go over these objectives throughout the assignment.

Course Main Objectives: Design a database that correctly models the domain and its constraints

Given a domain:

- Construct *Entity-Relationship diagrams* (ER);
- Determine *Functional Dependencies* (FD);
- Compute *Normal Forms* (NF).



code → cName
code, time → rName
rName, time → code

Courses (code, cName)
Rooms (rName)
Lectures (room, time, course)
room → Rooms.rName
course → Courses.code

Course Main Objectives: Construct

Given the database schema with its related constraints implement the database in a relational DBMS.

Courses (code, cName)
Rooms (rName)
Lectures (room, time, course)
  room → Rooms.rName
  course → Courses.code

```
CREATE TABLE Courses (
  code CHAR(6) PRIMARY KEY,
  cname TEXT NOT NULL);

CREATE TABLE Rooms (
  rName CHAR(20) PRIMARY KEY);

CREATE TABLE Lectures (
  room CHAR(20)
    REFERENCES Rooms,
  course CHAR(6) NOT NULL
    REFERENCES Courses,
  time TIMESTAMP,
  PRIMARY KEY (room, time) );
```

Course Main Objectives: Usage

- Query a database using **SQL**:

```
SELECT time, room
FROM Lectures
WHERE course = 'TDA357';
```

```
SELECT cName, time, room
FROM Courses, Lectures
WHERE Courses.code = Lectures.course;
```

- Modify the content of a database using **SQL**:

```
INSERT INTO Rooms VALUES ('HB2');
```

```
UPDATE Courses SET cname = 'Databases' WHERE code = 'TDA357';
```

```
DELETE FROM Lectures WHERE code = 'TDA357' AND room = 'HB3';
```

Course Main Objectives: Applications

Connect to and use a database from external applications:

```
Class.forName("org.postgresql.Driver");
Properties props = new Properties();
props.setProperty("user", user);
props.setProperty("password", pwd);
conn = DriverManager.getConnection(db, props);
```

```
import Databases.HDBC.Postgres
import Databases.HDBC

main =
  do conn <- connectPostreSQL "dbname=countries"
     run conn "UPDATE Currencies SET value=10.61 WHERE code='EUR' " []
     commit
```

Learning Objective

- Explain the semantic meaning of queries using relational algebra;
- Construct an entity-relationship (ER) diagram for a given domain;
- Translate an ER diagram into a relational database schema;
- Apply design theory concepts for relational databases such as functional dependencies and normalisation;
- Retrieve and modify data using a database language for respective task;
- Design a database interface using constraints, views, and triggers;
- Implement a relational database schema and related interface using a data definition language;
- Use a semi-structured data model;
- Communicate with a database, through a database interface, from a software application.
- Evaluate and create different models for a database domain using ER diagrams and relational schemas;
- Contrast the relational and the semi-structured data models.

Organisation and Examination

The course is *organised* as follows (give or take):

- Couple of lectures per week;
- An exercise class per week;
- Lab sessions: here is where you get help with the assignment!

Note: Check calendar page in Canvas or TimeEdit for updates.

The *examination* consists of two parts:

- A *programming assignment* (graded U/G) in 4 parts carried out in groups of 2 and which requires a *demonstration during the last week*;
- A *individual written exam* (graded U/3/4/5).

Both parts need to be passed to pass the course.
The grade on the exam will be your grade for the course.

Programming Assignment: Overview

Goal: Construct a *student portal* application in Java/Python+**SQL**.

- Part 0: Create an account and a group in Fire.
- Part 1: Starting with a domain description and a draft schema, implement the schema in a database and write queries (views) for common operations;
- Part 2: Use systematic design methods to add a few features, and find and eliminate flaws in the original schema;
- Part 3: Create triggers to further improve the database;
- Part 4: Connect to your database from a simple application, use JSON to send your data to a web client.

Programming Assignment: Deadlines

Note: All deadlines are *strict*!

	First Deadline	Last Deadline
Part 0		January 21st
Part 1	February 4th	February 14th
Part 2	February 11th	February 25th
Part 3	February 21st	March 3rd
Part 4	Demos March 4th-7th	Extra demo March 8th Files on March 10th

- You MUST submit a (possibly partial) solution by the first deadline!
- Make sure to read and correct the errors from the automatic tests!
- We will try to give feedback as soon as possible after each submission;
- Don't go for small incremental improvements before next submission;
- Make sure to get proper help so that submissions are as correct as possible from the start;
- All parts need to be passed before their final deadline to pass the lab!

The Boring Part ...

All this information and more can be found in the Canvas page of the course. Check it regularly!

https://chalmers.instructure.com/courses/28102

People: Ana (examiner), Jonas (guest lecturer), Lorenzo, ... (TAs).

Tools: Fire, Slack, PostgreSQL, Dia, ...

Literature: *Database Systems: The complete book*, 2nd edition, *Databases in 137 pages*, course material, the whole internet.

Syllabus: Chalmers, GU.

Cheating: Not allowed! We are obligued to report to disciplinary board.

Load: 7.5 hp $\sim$ 20-25 hours per week!

...: Check the Canvas page of the course for more info!

Communication Channels

Mail Ana: bove@chalmers.se

For general questions regarding the course or when a particular concept needs to be clarified.

If there are any administrative issues around assignments.

Ask during lectures/exercises: For general questions regarding the course or when a particular concept needs to be clarified.

Ask TAs during lab sessions: For help with programming/assignments.

Slack channels: (this link to join)

- **#general:** For important announcements from us;
- **#find-lab-partner:** To find a partner to do the assignment with;
- **#questions:** For general questions about the assignment or concepts; do not post part of your code in here!
- **#lab-help:** To queue in the lab room for help with the assignment during *scheduled* lab sessions;
- **#demo:** For the *online* demo during the last weeks.

No discussion on assignments outside the groups!

Course Evaluation

We need to have students representative, both from Chalmers and GU, ideally from different programmes.

Please mail me bove@chalmers.se before the end of the week if you are interested.

Otherwise I will randomly select students from several programs!

First meeting will take place in a couple of weeks.

Sets and Membership

Definition: A *set* is a collection of well defined and distinct objects or elements. (There is no repetition in sets!)

A set might be finite or infinite.

Definition: We denote that x is an *element* of set A by $x \in A$.

Sets can be described/defined in different ways:

Enumeration: mainly finite sets, infinite sets with help of ... (not formal though):

$$\text{WeekDays} = \{\text{Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday}\}.$$

Operations on Other Sets: $A \cup B$, $A \cap B$, $A \times B$, ... (see slide 23 and 25).

Characteristic Property: $\text{OddNat} = \{x \in \mathbb{N} \mid x \text{ is odd}\}$.

⋮

Some Operations and Properties on Sets

Union: $A \cup B = \{x \mid x \in A \text{ or } x \in B\}$.

Intersection: $A \cap B = \{x \mid x \in A \text{ and } x \in B\}$.

Difference: $S - A = \{x \mid x \in S \text{ and } x \notin A\}$.

Complement: When the set S is known, $S - A$ is written $\overline{A}$.

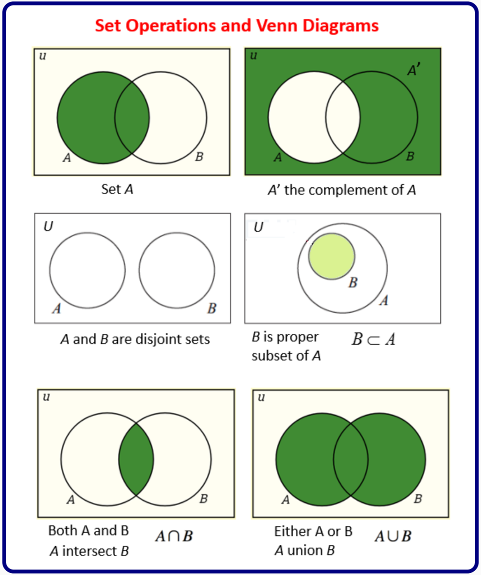
$S - A$ is sometimes denoted $S \backslash A$ and $\overline{A}$ is sometimes denoted A'

Subset: $A \subseteq B$ if for all $x \in A$ then $x \in B$.

Proper Subset: $A \subset B$ if $A \subseteq B$ and $A \neq B$.

Equality: $A = B$ if $A \subseteq B$ and $B \subseteq A$.

Venn Diagramas



(from https://www.onlinemathlearning.com)

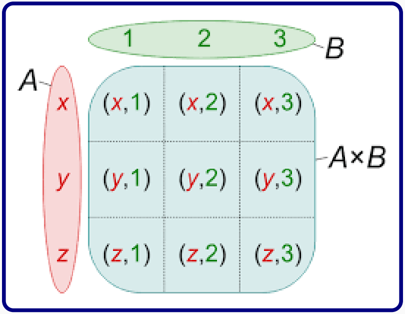
Cartesian Products

Definition: $A \times B = \{\langle x, y\rangle \mid x \in A \text{ and } y \in B\}$.

Observe this is a collection of ordered pairs: $\langle x, y\rangle \neq \langle y, x\rangle$!

Definition: The size of a cartesian product is $|A \times B| = |A| * |B|$.

Note: Can be generalised to the product of n sets: $A_1 \times A_2 \times \ldots \times A_n$.



(from https://en.wikipedia.org)

Projections

Definition: Given $\langle a, b \rangle \in A \times B$ we define the projections as

- $\pi_1 \langle a, b \rangle = a$
- $\pi_2 \langle a, b \rangle = b$

Example: A well-known cartesian product is that of points in 2D-space.

$$\mathbb{R}^2 = \mathbb{R} \times \mathbb{R}$$

If $p \in \mathbb{R}^2$ is a point then

- $\pi_1\ p$ gives the x-coordinate and
- $\pi_2\ p$ gives the y-coordinate.

Relations

Definition: A *relation* R is a subset of the cartesian product of two or more sets, $R \subseteq A_1 \times \ldots \times A_n$.

Notation: $\langle a_1, ..., a_n \rangle \in R$, $R(a_1, ..., a_n)$, $\langle a_1, ..., a_n \rangle$ satisfies R.

Definition: A binary relation $R \subseteq A \times B$ consists of a sets of pairs.

Notation: Here we can also write $a\,R\,b$.

Definition: A relation R over a set A, that is $R \subseteq A \times A$, is

Reflexive if $\forall\, a \in A.\, a\,R\,a$;

Symmetric: if $\forall\, a, b \in A.\, (a\,R\,b \to b\,R\,a)$;

Transitive: if $\forall\, a, b, c \in A.\, (a\,R\,b \wedge b\,R\,c \to a\,R\,c)$.

Example: $\leqslant$ over $\mathbb{R}$ is both reflexive and transitive but not symmetric.

Relations are Sets!

Then all operations on sets are also available on relations!

Union: $R \cup S = \{p \mid p \in R \text{ or } p \in S\}.$

Intersection: $R \cap S = \{p \mid p \in R \text{ and } p \in S\}.$

Difference: $S - R = \{p \mid p \in S \text{ and } p \notin R\}.$

Cartesian product: $$R \times S = \{\langle x_1, \dots, x_n, y_1, \dots, y_m \rangle \mid \langle x_1, \dots, x_n \rangle \in R, \langle y_1, \dots, y_m \rangle \in S\}.$$

Note: We have $\langle x_1, \dots, x_n, y_1, \dots, y_m \rangle$, no the nested tuple $\langle \langle x_1, \dots, x_n \rangle, \langle y_1, \dots, y_m \rangle \rangle$.

Reflexive-transitive Closure

Let R be a binary relation over A.

Definition: The *reflexive-transitive closure* of R is the relation R^* defined as follows:

- For all $a \in A$, aR^*a;
- For all $a, b, c \in A$, if aRb and bR^*c then aR^*c.

Example: Let $A = \{a, b, c, d\}$ and $R = \{\langle a, b\rangle, \langle b, c\rangle, \langle c, d\rangle\} \subseteq A \times A$.
Then $R^* = \{\langle a, a\rangle, \langle b, b\rangle, \langle c, c\rangle, \langle d, d\rangle, \langle a, b\rangle, \langle b, c\rangle, \langle c, d\rangle, \langle a, c\rangle, \langle a, d\rangle, \langle b, d\rangle\}$.

Example: Consider a directed graph as our relation.
The reflexive-transitive closure of this relation is the set of all paths in the graph, including those of length 0.

Transitive Closure

Let R be a binary relation over A.

Definition: The *transitive closure* of R is the relation R^+ defined as follows:

- For all $a, b \in A$, if aRb then aR^+b;
- For all $a, b, c \in A$, if aRb and bR^+c then aR^+c.

Example: Let $A = \{a, b, c, d\}$ and $R = \{\langle a, b\rangle, \langle b, c\rangle, \langle c, d\rangle\} \subseteq A \times A$.
Then $R^+ = \{\langle a, b\rangle, \langle b, c\rangle, \langle c, d\rangle, \langle a, c\rangle, \langle a, d\rangle, \langle b, d\rangle\}$.

Example: In our directed graph example, the transitive closure would be the set of all paths of at least length 1.

Functions

Definition: A *function* f from A to B is a relation $f \subseteq A \times B$ such that, given $x \in A$ and $y, z \in B$, if $x\, f\, y$ and $x\, f\, z$ then $y = z$.

Notation: That $x\, f\, y$ is usually written as $f(x) = y$.

Example: $\mathrm{sq} \subseteq \mathbb{Z} \times \mathbb{N}$ such that $\mathrm{sq}(n) = n^2$.

Observe that $\mathrm{sq}(2) = 4$ and $\mathrm{sq}(-2) = 4$.

Domain, Codomain, Range and Image

Let $f \subseteq A \times B$ be a function.

Definition: The *domain* $\mathrm{Dom}(f)$ or Dom_f is the set for which the *function is defined*.
More formally, $\mathrm{Dom}_f = \{x \in A \mid \exists\, y \in B.\, f(x) = y\} \subseteq A$.

Definition: The set B is called the *codomain* of the function.

Definition: The set $\{y \in B \mid \exists\, x \in A.\, f(x) = y\} \subseteq B$ is called the *range* or *image* of f and denoted $\mathrm{Im}(f)$ or Im_f.

Example: The image of sq is NOT all $\mathbb{N}$ but $\{0, 1, 4, 9, 16, 25, 36, \ldots\}$.

Overview of Next Lecture

- Basic **SQL** and PostgreSQL :
  - Working with PostgreSQL;
  - Create, delete and alter database tables;
  - Types and constraints;
  - Relational schemas;
  - Insert, update and delete data in tables;
  - Database queries involving one table;
  - Group-by and aggregations.

Reading:

Book: chapters 2, 6.1–6.5, 7.1–7.4, 8.1–8.2 and 8.5

Notes: chapter 2, 7.4.1–7.4.3 and 4.9