Data Modelling and Databases - Week 1 (Lecture)

- Author: Ruben Schenk
- Date: 27.04.2021
- Contact: ruben.schenk@inf.ethz.ch

Data in Real World

A `database` is a collection of data, for example information about bank accounts or data on facebook, Amazon, etc.

A `database management software (DBMS)` is software designed to assist in maintaining and utilizing large collections of data.

First Database Example

We have several "whishes" for DBMS:

- *Data Independence:* application should not know how data is stored
- *Declarative Efficient Data Access*: the system should be able to store and retrieve data efficiently, without users worrying about it
- *Transactional Access:* as if there is only a single users using a system that does not fail
- *Generic Abstraction:* Users do not need to worry about all the above issues for each new query

What's the **potential downside** of using a DBMS?

- *Worklaod mismatch:* maybe your specialized application is not what a certain DBMS is designed for
- *Data model mismatch:* maybe your application cannot be naturally modeled by a given DBMS

Data Model: Relational Model

In this course, we focus on a specific combination - we represent knowledge as a `collection of facts`, and do interference using `mathematical logic`.

Relational Model - Schema

A `database schema` is a set of relation schema, where a `relation schema` is defined by a name and a set of attributes/fields/columns. A `field` or `attribute` is defined by a name and a domain, e.g. Integer, String, etc.

For example:

```
Students(sid:string, name:string, login:string, age:int, gpa:float)
```

The above code defines the relation schema "Students" by the attributes "sid, name, login, age", and "gpa".

Relational Model - Instance

For a relation $R(f_1 : D_1, \dots, f_n : D_n)$, an `instance` I_R is a set of tuples: $I_R \subseteq D_1 \times \dots \times D_n$. Inutitively, an instance is the "content" of a relation if you think about it as a "table". It is important to remember that a relation instance is a **set**, this means we cannot have duplicated tuples and that the order of tuples doesn't matter.

Relational Model - More Concepts

A `candiate key` is the minimal set of fields that identify each tuple inquely. A `primary key` is one candidate key, marked in a schema by underlining.

Query Language 1: Relational calculus

Union: $\cup$

$$x \in R_1 \cup R_2 \Leftrightarrow x \in R_1 \vee x \in R_2$$

Difference: $-$

$$x \in R_1 - R_2 \Leftrightarrow x \in R_1 \wedge \neg(x \in R_2)$$

Intersection: $\cap$

$$R_1 \cap R_2 = R_1 - (R_1 - R_2)$$

Selection: σ

Return tuples which satisfy a given condition c.

$$x \in \sigma_c(R) \Leftrightarrow x \in R \wedge c(x) = True$$

Projection: $\Pi_{A_1, \dots, A_n}(R)$

Only keep a subset of columns.

Cartesion Product: $\times$

$$(x, y) \in R_1 \times R_2 \Leftrightarrow x \in R_1 \wedge y \in R_2$$

Renaming: $\rho_{B_1, \ldots, B_n}(R)$

Change the name of the attributes of R to $B_1, \ldots, B_n$.

Natural join: $\bowtie$

$$R_1(A, B) \bowtie R_2(B, C) = \Pi_{A,B,C}(\sigma_{R_1.B=R_2.B}(R_1 \times R_2))$$

If there are `no shared attributes` in a natural join, e.g. $R(A, B, C)$ and $S(D, E)$, then $R \bowtie S = R \times S$. If two relations `share all attributes`, then $R \bowtie S = R \cap S$.

Theta Join: $\bowtie_\theta$

$$R_1 \bowtie_\theta R_2 = \sigma_\theta(R_1 \times R_2)$$

Equi-Join: $\bowtie_{A=B}$

$$R_1 \bowtie_{A=B} R_2 = \sigma_{A=B}(R_1 \times R_2)$$

It is important to note that relational algebra uses `Bag semantics` instead of set semantics:

- Each relation is a bag of tuples
- You can have duplicated tuples in the same relation
- i.e. set: $\{1,\ 2,\ 3\}$, bag: $\{1,\ 2,\ 3,\ 1,\ 2,\ 1\}$

It is furthermore important to remember that `bag operator semantics` are different to set operator semantics:

- *Bag Union:* $\{1,\ 2,\ 1\} \cup \{1,\ 2,\ 3\} = \{1.\ 1,\ 1,\ 2,\ 2,\ 3\}$
- *Bag Difference:* $\{1,\ 2,\ 1\} - \{1,\ 2,\ 3,\ 3\} = \{1\}$