Relational Data Model

WeST
People and Knowledge Networks

Overview

- Relational data model;
- Tuples and relations;
- Schemas and instances;
- Named vs. unnamed perspective;
- Relational algebra;

Table

Player	Birth Year
Andy	1980
Wim	1975
Liam	1985
Mike	1988
Bert	1971

Observations

- The **rows** of the table contain pairs occurring in the relation **player**.
- There are two **columns**, labeled respectively by “name” and “birth year”.
- The **values** in each column belong to different **domains** of possible values.

How to specify a relation

1. specifying the names of the columns (also called **fields** or **attributes**);

2. specifying a **domain** of possible values for each column;

3. enumerate all tuples in the relation.

(1)–(2) refer to the **schema** of this relation, while (3) to an **instance**.

Domains and attributes

- A set of **domains** $D_1; D_2$ (sets of values);
- A set of corresponding **domain names** d_1, d_2, …
- A set of **attributes** a_1, a_2, …

Relation Schema and Instances

Tuple: any finite sequence ($v_1, \ldots , v_n$).

n is the **arity** of this tuple.

Relation schema:

$$r(a_1\text{:d}_1, \ldots, \text{a}_\text{n}\text{:d}_\text{n})$$

where n ¿ 0, r is a relation name, $a_1, \ldots, a_n$ are distinct attributes, $d_1, \ldots, d_n$ are domain names.

Relation instance:

finite set of tuples ($v_1, \ldots, v_n$) of arity n such that v_i 5 D_i for all i.

Observation

1. The attributes in each column must be unique.

2. A relation is a **set**. Therefore, when we represent a relation by a table, the order of rows in the table does not matter.

Let us add to this:

3. The order of attributes does not matter.

New notation for tuples

- A tuple is a set of pairs { (a_1, v_1), …, (a_n, v_n) } denoted by { a_1=v_1, …, a_n=v_n } ,

- Let d_1, …, d_n be domain names ar $D_1; : : : ; D_n$ be the corresponding domains.

- The tuple **conforms** to a relation schema r(a_1:d1, …, a_n:d$_n$) if v_i 5 D_i for all i.

Note that in the relational data model tuples stored in a table are **structured**:

- all tuples conform to the same relation schema;
- the values in the same column belong to the same domain.

Untyped perspective: there is a single domain, so the second condition can be dropped.

Typed or untyped?

- Consider the relation admire:

admirer	admired
wim	andy
mike	wim
liam	andy
liam	**arsenal**

Database schema and instance

- Relational database schema: a collection of relation schemas with distinct relation names.

- Relational database instance conforming to a relational database schema *S*:
  - a mapping *I* from the relation names of *S* to relation instances such that
    - for every relation schema r(a_1:d_1, … , a_n:d_n) in *S* the relation instance *I*(*r*) conforms to this relation schema.

- **No attributes**

- a tuple is simply a **sequence** (v_1, … , v_n) of values.

- The components of tuples can therefore be identified by their **position** in the tuple.

From Unnamed to Named Perspective

- Introduce a collection of attributes #1,#2, …,
- identify tuple (v_1, … , v_n) with the tuple { #1 = v_1, … ,#n = v_n }.
- Likewise, identify relation schema $r(d_1, \ldots, d_n)$ with $r(\#1{:}d_1, \ldots, \#n{:}d_n)$.

1. Can **define** new relations from existing ones;
2. Uses a collection of **operations** on relations to do so.

Constant

$$\{\ (v_{11},\ \ldots\ ,\ v_{1n}),$$

$$\ldots\ldots$$

$$(v_{k1},\ \ldots\ ,\ v_{kn})\ \}$$

Union

R_1 ^ R_2 =

$\{(c_1, \ldots , c_k) \mid \quad (c_1, \ldots , c_k)$ 5 R_1 or

$\quad (c_1, \ldots , c_k)$ 5 $R_2 \}$

Set difference

*R*_1 - *R*2 =

$$\{(c_1, \ldots , c_k) \mid \quad (c_1, \ldots , c_k) \; 5 \; R_1 \text{ and } (c_1, \ldots , c_k) \; \not{5} \; R_2 \}$$

Cartesian product

$R1 \ \S \ R2 =$

$\{(c_1, \ldots , c_k, d_1, \ldots , d_m) \mid$

$(c1, \ldots , c_k) \ 5 \ R_1$ and

$(d_1, \ldots , d_m) \ 5 \ R_2 \}.$

Let now R be a relation of arity k and $i_1, \ldots, i_m$ be numbers in $\{1, \ldots, k\}$.

$$\pi_{i_1, \ldots, i_m}(R) = \{(c_{i_1}, \ldots, c_{i_m}) \mid (c_1, \ldots, c_k) \in R\}.$$

We say that $\pi_{i_1, \ldots, i_m}(R)$ is obtained from R by **projection** (on arguments $i_1, \ldots, i_m$).

Selection

Assume **formulas on domains** with “variables” #1, #2, ….

For example, #1 = #2.

$\hat{}_{I}\,(R) = \{(c_1, \ldots, c_k) \mid$

$\quad (c_1, \ldots, c_k)\ 5\ R$ and

$\quad F$ holds on $(c_1, \ldots, c_k)\}$.

Overview

- Relational algebra, named perspective
- SQL
- Integrity constraints
- (Aggregates and grouping)

Constant

$$\{\,\{\, a_1 = v_{11},\ \ldots\ ,\ a_n = v_{1n}\,\},$$

$$\ldots \qquad \ldots \qquad \ldots$$

$$\{\, a_1 = v_{k1},\ \ldots\ ,\ a_n = v_{kn}\,\}\,\}$$

Union

Let *R*1, *R*2 be relations with the same attributes.

$$R1 \ \hat{}\ \ R2 = \{\, t \mid t\ 5\ R_1 \text{ or } t\ 5\ R_2 \,\}$$

Union, example

R_1

A	B
–	1
–	2
†	1

R_2

A	B
–	2
†	3

R_1 ^ R_2

A	B
–	1
–	2
†	1
†	3

Renaming

Let *R* be a relation whose set of attributes is $a_1,\ldots,a_n$, $c_1,\ldots,c_m$

Let $b_1, \ldots , b_n$ be distinct attributes such that

$\{b_1, \ldots , b_n\}$ ˇ $\{ c_1, \ldots , c_m\} =$ >

Then

´ $_{d_4\$\ e_4,\ \ldots\ ,\ d_q\$\ e_q}(R) =$

$\{\{b_1 = v_1, \ldots , b_n=v_n , c_1=w_1 , \ldots , c_m=w_m\} \mid$

$\{a_1 = v_1, \ldots , a_n=v_n , c_1=w_1 , \ldots , c_m=w_m\}$ 5 $R\}$

SQL is based on set and relational operations with certain modifications and enhancements

A typical SQL query has the form:

select $a_1, \ldots, a_n$
from $R_1, \ldots, R_m$
where P

This query is equivalent to relational algebra expression:

$$\grave{}_{d_4,\ldots,d_q}\,(\hat{}_{S}\,(R_1 \S\ \ldots\ \S\ R_m))$$

The result of an SQL query is a relation.

Exceptions?

Integrity constraints

- Domain constraints.
- Key constraints.
- Foreign key constraints.

- More general, defined constraints.

- How to translate them?

Query language

Allow one to define:

- Relation and database schemas;
- Relations through our relations;
- Integrity constraints;
- Updates.