

Relational Data Model

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

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

- 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.

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**.

- A set of **domains** $\mathcal{D}_1, \mathcal{D}_2$ (sets of values);
- A set of corresponding **domain names** d_1, d_2, \dots
- A set of **attributes** a_1, a_2, \dots

Tuple: any finite sequence (v_1, \dots, v_n) .
 n is the **arity** of this tuple.

Relation schema:

$$r(a_1:d_1, \dots, a_n:d_n)$$

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

Relation instance:

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

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.

- A tuple is a set of pairs $\{ (a_1, v_1), \dots, (a_n, v_n) \}$ denoted by $\{ a_1=v_1, \dots, a_n=v_n \}$,
- Let d_1, \dots, d_n be domain names ar $\mathcal{D}_1, \dots, \mathcal{D}_n$ be the corresponding domains.
- The tuple **conforms** to a relation schema $r(a_1:d_1, \dots, a_n:d_n)$ if $v_i \in D_i$ 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.

- Consider the relation admire:

admirer	admired
wim	andy
mike	wim
liam	andy
liam	arsenal

- 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 $r(a_1:d_1, \dots, a_n:d_n)$ in S the relation instance $I(r)$ conforms to this relation schema.

- **No attributes**
- a tuple is simply a **sequence** (v_1, \dots, v_n) of values.
- The components of tuples can therefore be identified by their **position** in the tuple.

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

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

$$\begin{aligned} & \{ (v_{11}, \dots, v_{1n}), \\ & \dots \\ & (v_{k1}, \dots, v_{kn}) \} \end{aligned}$$

$$R_1 \cup R_2 =$$

$$\{(c_1, \dots, c_k) \mid (c_1, \dots, c_k) \in R_1 \text{ or } (c_1, \dots, c_k) \in R_2\}$$

$$R_1 - R_2 =$$

$$\{(c_1, \dots, c_k) \mid (c_1, \dots, c_k) \in R_1 \text{ and } (c_1, \dots, c_k) \notin R_2\}$$

$$R_1 \times R_2 =$$

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

$$(c_1, \dots, c_k) \in R_1 \text{ and}$$

$$(d_1, \dots, d_m) \in R_2 \}.$$

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

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

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

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

For example, #1 = #2.

$$\sigma_F(R) = \{(c_1, \dots, c_k) \mid \\ (c_1, \dots, c_k) \in R \text{ and} \\ F \text{ holds on } (c_1, \dots, c_k)\}.$$

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

$$\begin{aligned} & \{ \{ a_1 = v_{11}, \dots, a_n = v_{1n} \}, \\ & \quad \dots \quad \dots \quad \dots \\ & \{ a_1 = v_{k1}, \dots, a_n = v_{kn} \} \} \end{aligned}$$

Let R_1, R_2 be relations with the same attributes.

$$R_1 \cup R_2 = \{ t \mid t \in R_1 \text{ or } t \in R_2 \}$$

R_1

A	B
α	1
α	2
β	1

R_2

A	B
α	2
β	3

$R_1 \cup R_2$

A	B
α	1
α	2
β	1
β	3

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

Let b_1, \dots, b_n be distinct attributes such that

$$\{b_1, \dots, b_n\} \cap \{c_1, \dots, c_m\} = \emptyset$$

Then

$$\rho_{a_1 \rightarrow b_1, \dots, a_n \rightarrow b_n}(R) = \{ \{b_1 = v_1, \dots, b_n = v_n, c_1 = w_1, \dots, c_m = w_m\} \mid \{a_1 = v_1, \dots, a_n = v_n, c_1 = w_1, \dots, c_m = w_m\} \in R \}$$

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

A typical SQL query has the form:

select a_1, \dots, a_n
from R_1, \dots, R_m
where P

This query is equivalent to relational algebra expression:

$$\pi_{a_1, \dots, a_n} (\sigma_P (R_1 \times \dots \times R_m))$$

The result of an SQL query is a relation.

Exceptions?

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

- More general, defined constraints.

- How to translate them?

Allow one to define:

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