1 Relational Model

The Relational Model is a data model. The basic elements are **attributes**, **relations**, and **operations** on relations.

An **attribute** is a **domain** of atomic values. '**Atomic**' means that the values are not sets.

A **relation schema** is a set of attributes, e.g. Person = (name, age).

A **relation** is a mathematical relation between attributes in a relation schema, i.e. a subset of the cartesian product of the attributes. Therefore it is a set of **tuples**, the rows of the tables, e.g. members p of the schema: Person = (name, age), should be of the type:

$$p \subseteq \text{name (string)} \times \text{age (positive integer)},$$

2 Relational Algebra

Relational algebra is a **procedural query language** ('how', not 'what', sequence of operations), and consists of a set of operations.

2.1 Basic Operations

- The **select** operation $\sigma_{\text{predicate}}(\text{relation})$ selects tuples from 'relation' that satisfies 'predicate'
- The **projection** operation $\Pi_{\text{attributes}}(\text{relation})$ discards the attributes from 'relation' not listed in 'attributes'.
- The **rename** operation $\rho_{\text{newname}}(\text{attribute})$ renames an attribute.
- The **union** $r \cup s$ and the **difference** $r - s$ operations work like standard set theory. r and s must of course have the same schema.
- The **cartesian product** $r \times s$ combines all tuples from r with all tuples from s.

2.2 Derived Operations

Derived operations can be expressed by combining the basic operations.

- The **intersection** operation is defined as $r \cap s = r - (r - s)$.

- The **join** operation have different flavours. The **natural join** $r \bowtie s$ are all combinations of tuples from r and s (cartesian product), where attributes with the **same name** are equal. The **theta-join** $r \bowtie_{a\theta b} s, \theta \in \{<, \leq, =, \geq, >\}$ is more general, it selects all tuples from the cartesian product where $a\theta b$. The **equijoin** $r \bowtie_{a=b} s$ is just a special case of the theta-join. The **semijoin** $r \ltimes s$ are like natural join, but only keeps the attributes from r.
- The **division** operator $r \div s$ projects attributes unique to r, where all their combinations with tuples in s are in r.

3 Extended Relational Algebra

3.1 Generalized Projection

The **Generalized Projection** allows arithmetic expressions in parameters, e.g. $\Pi_{\text{name},(\text{age}+1)}(person)$.

3.2 Aggregate Functions

An **aggregate function** is a function that compute a single result value from a set of input values. $\mathcal{G}_{\text{average}}, \mathcal{G}_{\text{count}}, \mathcal{G}_{\text{max}}, \mathcal{G}_{\text{min}}, \mathcal{G}_{\text{sum}}$ are common aggregates.

As an example, ${}_{city}\mathcal{G}_{\textbf{average}(salary)}(\text{person})$ form groups according to the value of 'city' and return a table with the schema (city, average salary).

3.3 Outer Joins

Outer joins $r \mathbin{⟕} s$ (left outer join) are like natural joins, except that every tuple from the left table will appear, and if same name attributes match the right table, the right tuple will appear, otherwise the attributes of the right table will be padded with NULLs.

Similarly, there are the **right outer join** and **full outer join** operations. The latter takes all tuples (cartesian product) and pad with NULLs, where the same name attributes don't match.

3.4 Database Modifications

Extended relational algebra allows modifications of relations.

The **insertion** operation $r \leftarrow r \cup T$ inserts tuples into a relation. The **deletion** operation $r \leftarrow r - T$ deletes tuples. The **update** operation changes the values of tuples using **generalized projection**, e.g.

$$\text{person} \leftarrow \Pi_{name,age+1}(\sigma_{\text{name="Jakob"}}(\text{person})) \cup (person - \sigma_{\text{name="Jakob"}}(\text{person}))$$

.

Views are used for hiding information in situations where it may be relevant. This is done by creating a new pseudo-relation that is actually a **projection** of an existing relation.

4 NULL Values

NULL values are interpreted as missing information, so:

- **arithmetics** with NULL yields NULL
- **comparison** with NULL yields UNKNOWN (neither TRUE nor FALSE)
- **logical connectives** behave as expected, FALSE AND UNKNOWN = FALSE, and TRUE OR UNKNOWN = TRUE, etc.
- **aggregates** ignore NULL values