Transformation of an \textit{m:n}-relationship set

- For each binary \textit{m:n}-relationship set $R$ a new relation schema $S$ is created. Add to $S$ as foreign keys the primary keys of the relation schemas that correspond to the two entity sets participating in $R$. Their combination forms the primary key of $S$. Furthermore, all simple attributes and all simple components of composite attributes of $R$ are taken as attributes of $S$.

- Example university database:
  
  $\text{attends}(\text{reg-id, id})$ \hspace{2cm} $\text{is\_precondition\_of}(\text{predecessor, successor})$

Transformation of multivalued attributes

- For each multivalued attribute $A$ a new relation schema $R$ is created. $R$ comprises an attribute corresponding to $A$ and as foreign key the primary key $K$ of the relation schema which corresponds to the entity set or relationship set containing $A$ as attribute. The primary key of $R$ is the combination of $A$ and $K$. If the multivalued attribute is composite, its simple components are added to $R$.

- Example:

  $\text{department}$

  $\text{name}$

  $\text{location}$

  $\text{dept-no}$

  $\text{department(}\text{dept-no, name})$

  $\text{dept-loc(}\text{location, dept-no})$
Transformation of an \( n \)-ary relationship set

- For each \( n \)-ary relationship set \( R \) with \( n > 2 \) a new relation schema \( S \) is created. Add to \( S \) as foreign keys the primary keys of the relation schemas corresponding to entity sets participating in \( R \). Furthermore, all simple attributes and all simple components of composite attributes of \( R \) are taken as attributes of \( S \). The primary key of \( S \) is the combination of all foreign keys.

- example university database:

  tests(reg-id, id, pers-id, grade)

Complete schema of the university database

students(reg-id : integer, name : string, sem : integer)
lectures(id : integer, credits : integer, title : string, held_by : integer)
professors(pers-id : integer, name : string, room : integer, rank : string)
assistants(pers-id : integer, name : string, room : integer, boss : integer)
attends(reg-id : integer, id : integer)
is_precondition_of(predecessor : integer, successor : integer)

tests(reg-id : integer, id : integer, pers-id : integer, grade : integer)
Transformation of generalizations

- Generalizations are not represented by an own relation. The relationship is already expressed by the fact that the key of the common superclass is also used as key of the specialized subclasses.

- example:

```
employees(pers-id, name, room)
professors(pers-id, rank)
assistants(pers-id)
```

- information about a professor distributed to two tuples of two relations, namely to a tuple of the relation *employees* and to a tuple of the relation *professors*

- To obtain the complete information requires a connection of both relations and tuples, respectively (join). There is no inheritance in the relational data model.
4.4 Relational Algebra

Introduction

- so far: structural description by means of a database schema
- required: language for extracting information from the database

  (likewise necessary, but later dealt with: **data manipulation language** (DML) with operations for inserting, changing and deleting information)

- two formal languages
  - **relational calculus** (tuple relational calculus, domain relational calculus): **declarative** language which allows to specify *which* data one would like to retrieve or which criteria these data have to fulfil, but not *how* a query has to be evaluated
  - **relational algebra: procedural** language which allows to specify *how* a query has to be evaluated (execution plan)
  - both languages are closed, i.e., the results of queries, which operate on relations, are again relations.

- universal algebra
  - given a set $T$ (“anchor of the algebra”)
  - given a set of operations $\{\sigma_1, \ldots, \sigma_n\}$ of the form $\sigma_i : T^k \rightarrow T$
relational algebra is a universal algebra
- anchor is the set of all relations
- 5 (+ 1) basic operations: union, difference, Cartesian product, project, select, (rename)
- other algebra operations are derived, i.e., they can be expressed by the basic operations

given: two relations $R(A_1 : C_1, A_2 : C_2, ..., A_r : C_r)$ and $S(B_1 : D_1, B_2 : D_2, ..., B_s : D_s)$ with arity $r$ and $s$

$R$ and $S$ are schema compliant (identical except for renaming), if $r = s$ holds and if there exists a permutation $\varphi$ of the indices $\{1, ..., r\}$, so that $\forall 1 \leq i \leq r : C_i = D_{\varphi(i)}$

The Union operation

- $R$ and $S$ are schema compliant
- $R \cup S = \{t | t \in R \lor t \in S\}$

The Difference operation

- $R$ and $S$ are schema compliant
- $R - S = \{t | t \in R \land t \notin S\}$
The Cartesian product operation

- Result relation $T$ has the attributes \{ $A_1 : C_1, A_2 : C_2, ... , A_r : C_r, B_1 : D_1, B_2 : D_2, ... , B_s : D_s$ \} as the union of the attributes of $R$ and $S$, $T$ has arity $r + s$.

- Tuple concatenation of two tuples $t = (v_1, ..., v_r)$ and $u = (w_1, ..., w_s)$ is defined as the tuple $t \circ u = (v_1, ..., v_r, w_1, ..., w_s)$ of arity $r + s$.

- $R \times S = \{ t \circ u \mid t \in R, u \in S \}$

The Project operation

- Restriction of a relation schema to specified attributes.

- Correspondingly restriction of the tuples of the respective relation to these attributes.

- Elimination of possibly emerging duplicates (set property!)

- Let $R$ be a relation over the attribute set $A$, and let $B \subseteq A$. We obtain for the projection $\pi$ a restriction of each tuple to the value of the attributes in $B$:

  $\pi_B(R) = \{ t|_B \mid t \in R \}$

  ['$t|_B$' means 'tuple $t$ reduced to the values of the attribute set $B \subseteq A$']