Logical Schema Design: The Relational Data Model

Basics of the Relational Model
From Conceptual to Logical Schema

The Relational Data Model

- 1970 introduced by E.F. Codd, honoured by Turing award (Codd: *A Relational Model of Data for Large Shared Data Banks. ACM Comm. 13(6): 377-387, 1970*)
- Basic ideas:
  - Database is collection of relations
  - Relation R = set of n-tuples
  - Relation schema R(A1, A2, ..., An)
  - Attributes Ai = atomic values of domain Di=dom(Ai)

<table>
<thead>
<tr>
<th>Student</th>
<th>Name</th>
<th>Email</th>
<th>SID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tina</td>
<td>Anna</td>
<td>Carla</td>
<td></td>
</tr>
<tr>
<td>mueller@...</td>
<td>katz@...</td>
<td>piep@...</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Relation Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>Student</td>
</tr>
<tr>
<td>---------</td>
</tr>
<tr>
<td>Tina</td>
</tr>
<tr>
<td>mueller@...</td>
</tr>
</tbody>
</table>

| Student(Fname: string, Name: string, Email: string, SID: number) or Student(Frame, Name, Email, SID) |

- Superkey:
  - subset of attributes of a relation schema R for which no two tuples have the same value.
    - Every relation at least 1 superkey.
    - Example: Student(Frame, Name, Email, SID)
      - Superkeys: (Frame, Name, Email, SID), (Name, Email, SID), (Frame, Email, SID), (Name, SID), (Frame, Email), (Name, Email, SID), (Email, SID), (Email, SID)
Logical Schema Design: Relational Data Model

- **Candidate Key:**
  - superkey $K$ of relation $R$ such that if any attribute $A \in K$ is removed, the set of attributes $K \setminus A$ is not a superkey of $R$.
  - Example: `Student(Fname, Name, Email, SID)`
    - Candidate Keys: (Email), (SID)

- **Primary Key:**
  - arbitrarily designated candidate key
  - Example: `Student(Fname, Name, Email, SID)`
    - Primary Key: (SID)

- **Foreign Key:**
  - set of attributes $FK$ in relation schema $R_1$ that references relation $R_2$ if:
    - Attributes of $FK$ have the same domain as the attributes of primary key $K_2$ of $R_2$
    - A value of $FK$ in tuple $t_1$ in $R_1$ either occurs as a value of $K_2$ for some $t_2$ in $R_2$ or is null.
  - Example: `R_1: Exercise(EID, Tutor, ExerciseHours, Lecture)`
    - `R_2: Student(Fname, Name, Email, SID)`
    - $K_1 = \{EID\}$, $K_2 = \{SID\}$
    - Tutor is foreign key of `Student` (from `Exercise`).

Logical Schema Design: Transformation

1. Select data model → relational data model
2. Transform conceptual model into logical schema of relational data model

- Define relational schema, table names, attributes and types, invariants
- Design steps:
  - Translate entities into relations
  - Translate relationships into relations
  - Simplify the design
  - Define tables in SQL
  - Define additional invariants
  - (Formal analysis of the schema)
Step 3: Transform Relationships

- For each 1:1-relationship between E1, E2 create relation R
- Include all key attributes
- Define as key: key of entity E1 or entity E2
- Choose entity with total participation for key (driving licence)

Relational Schema:
- Country(CName)
- President(PName)
- has(CName, PName) or has(CName, PName)

Step 3: Transform Relationships

- For each 1:N-relationship between E1, E2 create relation R
- Include all key attributes
- Define as key: key of N-side of relationship

Relational Schema:
- Lecture(lnr)
- Lecturer(lid, name)
- hold(lnr, lid)

Lecture hold Lecturer (1,1) (0,*)
lnr lid name

Step 3: Transform Relationships

- For each N:M-relationship create relation R
- Include all key attributes
- Define as key: keys from both entities

Relational Schema:
- User_account(username)
- emailMessage(id, from)
- has(username, messageid)

User_account has emailMessage (1,*) (1,*)
username messageid

Step 3: Transform Relationships

- Include all relationship attributes in relation R
- Rename keys in recursive relationships

Relational Schema:
- Person(pid, name, fname, email)
- Student(pid, sid)
- Lecturer(pid, contract)

Person(pid, name, email)
Student(pid, sid)
Lecturer(pid, contract)

Step 4: Generalization

- 3 Variants for Transformation
### Logical Schema Design: Generalization

1. **Separate relation for each entity**
   - Key in specialized relations: foreign key to general relation
   - Gathering data from different tables time consuming
   - Appropriate specialization prevents unnecessary data access

2. **Relations for specialization**
   - Separate tables which include A’s attributes
   - Separate keys for relations
   - Only valid for exhaustive specialization (no data in A only)
   - Time consuming data retrieval for non-distinct specializations

3. **One relation for all entities**
   - Remains generalization!
   - Relation may contain many NULL-values

### Logical Schema Design: Example

<table>
<thead>
<tr>
<th>Format(name, charge)</th>
<th>Tape(id)</th>
<th>Movie(id, title, category, year, director, price_per_day, length)</th>
<th>Customer(mem_no, last_name, first_name, address, telephone)</th>
<th>Rental(tape_id, member, from, until)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Belong_to(formatname, tape_id)</td>
<td>Hold(tape_id, movie_id)</td>
<td>Play(movie_id, actor_name)</td>
<td>Reduce relation number by simplification</td>
<td></td>
</tr>
</tbody>
</table>

### Logical Schema Design: Simplification

- Collect those relation schemas with the same key attributes into one relation schema
- Semantics of attributes must match, not literal name
- Do not destroy generalization!
Logical Schema Design: Example

Format(name, charge)
Tape(id)
Movie(id, title, category, year, director, price_per_day, length)
Actor(stage_name, real_name, birthday)
Customer(mem_no, last_name, first_name, address, telephone)
Rental(tape_id, member, from, until)
Belong_to(formatname, tape_id)
Hold(tape_id, movie_id)
Play(movie_id, actor_name)

Simplified relation from Tape/Belong_to/Hold:
Tape(id, format, movie_id)
No simplification based on partial keys!

Logical Schema Design: Simplification

Restrictions on schema simplification
- Simplification (folding) of relations may result in NULL values in some tables
- optional relationships should not be transformed if many NULLS to be expected
- May lead to time consuming retrieval

Logical Schema Design: Short summary

Relational data model
- Representation data in relations (tables)
- the most important data model today

Important terms & concepts
- Relation: set of n-tuples
- Relation schema defines structure
- Attribute: property of relation, atomic values
- Superkey, candidate key, primary key identify tuple
- Transformation rules for entries, relationships
- Simplification

Open aspects of logical schema design
- DDL for defining and changing relation schemas
- Definition of constraints (Min-cardinalities, Value restrictions for attributes,..)