Logical Schema Design:
The Relational Data Model

Basics of the Relational Model
From Conceptual to Logical Schema
Logical Schema Design

- Select data model
  - Hierarchical data model: hierarchies of record types, mainframe oldie, still in use, outdated
  - Network data model: graph like data structures, still in use, outdated
  - Relational data model: the most important one today
  - Object-Oriented data model: more flexible data structures, less powerful data manipulation language
  - Object-Relational: the best of two worlds?

- Transform conceptual model into logical schema of data model
  - easy for Relational Data Model (RDM)
  - can be performed automatically (e.g. by Oracle Designer)
Logical Schema Design: Relational Data Model

- The Relational Data Model
  - 1970 introduced by E.F. Codd, honoured by Turing award

- Basic ideas:
  - Database is collection of relations
  - Relation \( R = \text{set of } n\)-tuples
  - Relation schema \( R(A_1, A_2, \ldots, A_n) \)
  - Attributes \( A_i = \text{atomic values of domain } D_i = \text{dom}(A_i) \)

### Important terms
- SID

<table>
<thead>
<tr>
<th></th>
<th>FName</th>
<th>Name</th>
<th>Email</th>
<th>SID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tina</td>
<td>Müller</td>
<td>mueller@...</td>
<td>13555</td>
<td></td>
</tr>
<tr>
<td>Anna</td>
<td>Katz</td>
<td>katz@...</td>
<td>12555</td>
<td></td>
</tr>
<tr>
<td>Carla</td>
<td>Maus</td>
<td>piep@...</td>
<td>11222</td>
<td></td>
</tr>
</tbody>
</table>

Relation schema:
- Student(Fname, Name, Email, SID) or
- Student(Fname:string, Name:string, Email:string, SID:number)
Logical Schema Design: Relational Data Model

- Relation Schema $R(A_1, A_2, \ldots, A_n)$
  notation sometimes $R:\{[A_1, A_2, \ldots, A_n]\}$

- Relation $R \subseteq \text{dom}(A_1) \times \text{dom}(A_2) \times \ldots \times \text{dom}(A_n)$

- Attribute set $\mathcal{R} = \{A_1, A_2, \ldots, A_n\}$

- Degree of a relation: number of attributes

- Example:
  Student($Fname, Name, Email, SID$)
  $\text{Student} \subseteq \text{string} \times \text{string} \times \text{string} \times \text{number}$
  $\mathcal{R} \ (\text{Student}) = \{Fname, Name, Email, SID\}$

- Database Schema is set of relation schemas
Logical Schema Design: Relational Data Model

- Time-variant relations
  - Relations have state at each point in time.
  - Integrity constraints on state part of DB schema

- Tuples (rows, records)
  - Not ordered
  - No duplicate tuples (Relations are sets)
  - Null-values for some attributes possible
  - Distinguishable based on tuple values (key-concept)

Different to object identification in o-o languages:
there, each object has implicit identity,
usually completely unrelated to its fields
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(Fname, Name, Email, SID)

Superkeys:
- \{Fname, Name, Email, SID\}, \{Name, Email, SID\},
- \{Fname, Email, SID\}, \{Fname, Name, SID\},
- \{Fname, SID\}, \{Name, SID\}, \{Fname, Name, Email\},
- \{Fname, Email\}, \{Name, Email\}, \{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 R1 that references relation R2 if:

- Attributes of FK have the same domain as the attributes of primary key K2 of R2
- A value of FK in tuple t1 in R1 either occurs as a value of K2 for some t2 in R2 or is null.

Example:

R1: Exercise(EID, Tutor, ExcerciseHours, Lecture)  
R2: Student(Fname, Name, Email, SID)

K1={EID}, K2={SID}  
Tutor is foreign key of Student (from Exercise).
Logical Schema Design: Transformation

1. Select data model $\rightarrow$ 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 1: Transform entities
- For each entity E create relation R with all attributes
- Key attributes of E transform to keys of the relation

Relational Schema:
Movie(id, title, category, year, director, Price_per_day, length)
Logical Schema Design: Weak Entities

- **Step 2: Transform weak entities**
  - For each weak entity WE create relation R
  - Include all attributes of WE
  - Add key of identifying entity to weak entities key
  - Part of key is foreign key

Relational Schema:
- Employee(eid)
- Child(cid, eid)

Note: weak entity and defining relationship transformed together!
Step 2: Transform weak entities

Example:

Relational Schema:
Delivery(did, date)
Order(oid, did, date, contact)
Item(iid, oid, did)
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)
Logical Schema Design: Relationships

- Step 3: Transform Relationships
  - For each N:M-relationship create relation R
  - Include all key attributes
  - Define as key: keys from both entities

```
User_account
<table>
<thead>
<tr>
<th>has</th>
</tr>
</thead>
<tbody>
<tr>
<td>username</td>
</tr>
</tbody>
</table>

emailMessage
  |     (0,*)   |
  |     (1,*)   |
  | id          |
  | from        |

Relational Schema:
  User_account(username)
  emailMessage(id, from)
  has(username, messageid)
```
Step 3: Transform Relationships

- Include all relationship attributes in relation R

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

- Rename keys in recursive relationships

  Relational Schema:
  - require(preLNR, succLNR)
Step 3: Transform Relationships

- For each n-ary relationship (n>2) create relation R
- Include all key attributes
- Define as key: keys from all numerous involved entities

Relational Schema:
- Lecture(lnr)
- Lecturer(lid, name)
- Textbook(ISBN, title, author)
- recommend(lid, lnr, ISBN, rating)
Step 4: Generalization

- 3 Variants for Transformation

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

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

Person(pid, name, fname, email, sid, contract)
Logical Schema Design: Generalization

1. Separate relation for each entity
   - Key in specialized relations: foreign key to general relation

   \[ A(aid, a) \]
   \[ B(aid, b) \]
   \[ C(aid, c) \]

- 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

- Removes generalization!
- Relation may contain many NULL-values

ABC(aid, a, b, c)
Logical Schema Design: Example

- **Format**
  - charge
  - name
  - id
  - hold (1, 1)
  - is_in (0, *)

- **Tape**
  - id
  - belong_to (0, *)
  - is_in (1, 1)

- **Rental**
  - from
  - until (1, 1)
  - have (0, *)

- **Movie**
  - title
  - category
  - director
  - id
  - length (0, *)
  - price_per_day

- **Actor**
  - first_name
  - address
  - telephone
  - last_name
  - mem_no
  - stage_name
  - birthday
  - real_name

- **Customer**
  - mem_no
  - price_per_day (0, *)
  - rental (1, 1)

- **Other Relationships**
  - charge
  - play (1, *)
  - have (0, *)
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)

Reduce relation number by simplification!
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!

Diagram:

- Lecture(lid, title, hours)
- Person(pid, fname, name, email)
- Student(pid, sid)
- Lecturer(pid, contract)
- Attend(lid, pid)
- Hold(lid, pid)
- Lecture(lid, title, hours, lecturer)
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

Consequence:
- 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 entities, relationships
  - Simplification

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