The Relational model (RM) and translation from ER model to RM

Conceptual design/schema

Logical design/schema

But first, what is the Relational Model?

RM

ER

ER to RM mapping

So, this is about translating from ER to tables.

• Conceptual model, e.g. expressed in ER
• Good to describe a mini-world
• On an abstract level, not good for any specific DBMS

• Implementation model
• Describes database design
• Data model for a DBMS, here Relational DBMS.
The Relational model (RM) and translation from ER model to RM

Elmasri/Navathe ch 3, 8
Padron-McCarthy/Risch ch 5-6

DATABASE DESIGN I - 1DL300 Spring 2013
Sobhan Badiazamany
Department of Information Technology, Uppsala University

Conceptual design/schema

But first, what is the Relational Model?

Logical design/schema

ER to RM mapping

So, this is about translating from ER to tables

- Conceptual model, e.g. expressed in ER
- Good to describe a mini-world
- On an abstract level, not good for any specific DBMS

- Implementation model
- Describes database design
- Data model for a DBMS, here Relational DBMS
Conceptual design/schema

- Conceptual model, e.g. expressed in ER
- Good to describe a mini-world
- On an abstract level, not good for any specific DBMS

But first, what is the Relational Model?

Logical design/schema

ER to RM mapping

So, this is about translating from ER to tables

- Implementation model
- Describes database design
- Data model for a DBMS, here Relational DBMS.
But first, what is the Relational Model?
The relational model

• Introduced by Dr. Edgar (Ted) F. Codd (1924-2003) in 1970
  • Codd was a mathematician from Oxford (UK) invented the relational model while at IBM San José Research Lab (USA)
• Many (in practice, all) DBMSs are based on the relational data model
  • Very simple model
  • A multi billion $ industry
• Query using high-level languages
  • Simple but very expressive.
• Operations are based on
  • the relational algebra (covered in another lecture)
• Closedness: Operations in the relational model applies to relations (tables) and produce new relations.
- An operation can be applied to the result of another operation.
- Several operations can be combined.
The relational model

- Introduced by Dr. Edgar (Ted) F. Codd (1924-2003) in 1970
  - Codd was a mathematician from Oxford (UK) who invented the relational model while at IBM San José Research Lab (USA)
  - Many (in practice, all) DBMSs are based on the relational data model
    - Very simple model
    - A multi-billion $ industry
  - Query using high-level languages
    - Simple but very expressive.
  - Operations are based on
    - the relational algebra (covered in another lecture)
  - Closedness: Operations in the relational model apply to relations (tables) and produce new relations.
Relational model definitions

Informal

- A table contains set of rows
- Each row represents certain facts that correspond to a real-world entity or relationship.
- Each column has a column header that gives an indication of the meaning of the data items in that column.

Formal

<table>
<thead>
<tr>
<th>Table</th>
<th>Relation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Column header</td>
<td>Attribute</td>
</tr>
<tr>
<td>Integrity constraints</td>
<td>Integrity constraints</td>
</tr>
</tbody>
</table>

Table definition

- Populated table
- Row
- All possible column values

Informal: A table is a mathematical concept based on the idea of sets.
- A relation contains set of tuples.
A table contains set of rows.
Each row represents certain facts that correspond to a real-world entity or relationship.
Each column has a column header that gives an indication of the meaning of the data items in that column.

Table........................................................................................................Relation
Column header............................................................................................Attribute
Integrity constraints.......................................................................................Integrity constraints
Table definition..............................................................................................Schema of a relation
Populated table...................................................................................................State of a relation
Row.....................................................................................................................Tuple
All possible column values..............................................................................Domain

- A Relation is a mathematical concept based on the ideas of sets.
- A relation contains set of tuples.
A **domain** has a logical definition:
- Example: “USA_phone_numbers” are the set of 10 digit phone numbers valid in the U.S.

A domain also has a data-type or a format defined for it.
- The USA_phone_numbers may have a format: (ddd)ddd-dddd where each d is a decimal digit.
- Dates have various formats such as year, month, date formatted as yyyy-mm-dd, or as dd mm,yyyy etc.
Relation schema and instance

- $A_1, A_2, \ldots, A_n$ are attributes
- $R = (A_1, A_2, \ldots, A_n)$ is a relation schema
  - Customer-schema(customer-name, customer-street, customer-city)
- $r(R)$ is a relation on the relation schema $R$
  - customer (Customer-schema)
- The current values (relation instance) of a relation are specified by a table.
- An element $t$ of $r$ is a tuple - represented by a row in a table customer

<table>
<thead>
<tr>
<th>customer-name</th>
<th>customer-street</th>
<th>customer-city</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jones</td>
<td>Main</td>
<td>Harrison</td>
</tr>
<tr>
<td>Smith</td>
<td>North</td>
<td>Rye</td>
</tr>
<tr>
<td>Curry</td>
<td>North</td>
<td>Rye</td>
</tr>
<tr>
<td>Lindsay</td>
<td>Park</td>
<td>Pittsfield</td>
</tr>
</tbody>
</table>
Keys

- Because relations are sets, all tuples in the relation are different.
- There is usually a subset $k$ of the attributes in a relation schema $R$, i.e. $k \subseteq R$, that has the characteristic that if the tuples $t_1, t_2 \in r(R)$ and $t_1 \neq t_2$, the following holds: $t_1[k] \neq t_2[k]$ (i.e. the value of $k$ in $t_1 \neq$ the value of $k$ in $t_2$)
- Every such subset $k$ is called a superkey for $R$. 
A superkey $k$ is *minimal* if there is no other superkey $k'$ such that $k' \subseteq k$.

Every minimal superkey (NOTE! there can be more than one) is called a *candidate key* for $R$.

The candidate key *chosen* by the database designer as the key for $R$ is called $R$'s *primary key* or just *key*.

In addition, term *foreign key* is used when a tuple is referenced, from another relation, with its key.
What is the "Null"

- A special value, **null** or \( \perp \), can sometimes be used as an attribute value.
- Every occurrence of null is unique. Thus, two occurrences of null is not considered to be equal even if they are represented by the same symbol.
- null is used:
  - when one does not know the actual value of an attribute.
  - when a certain attribute does not have a value.
  - when an attribute is not applicable.
- Examples of the use of null are showed later.
**Integrity constraints**

- **1. Domain constraint**
  - attribute values for attribute A shall be atomic values from dom(A)
- **2. Key constraint**
  - candidate keys for a relation must be unique
- **3. Entity integrity constraint**
  - no primary key is allowed to have a null value
- **4. Referential integrity constraint**
  - a tuple that refers to another tuple in another relation must refer to an existing tuple
- **5. Semantic integrity constraint**
  - e.g. “an employee’s total work time per week can not exceed 40 hours for all projects taken all together”
Relations as mathematical objects

• Let $A_1, A_2, \ldots, A_n$ be attributes with Domains $D_1, D_2, \ldots, D_n$

• **Relation**: subset of the Cartesian product of domains $D_1, \ldots, D_n$ (set of values), i.e. $R \subseteq D_1 \times D_2 \times \ldots \times D_n$

• $D_1 \times D_2 \times \ldots \times D_n$: set of all ordered n-tuples $(v_1, v_2, \ldots, v_n)$, $v_i \in D_i$ for all $i$.

  • Example: $n=2$, $D_1 = \{1, 2\}$ and $D_2 = \{P, Q, R\}$ one gets the product sets:
    
    $D_1 \times D_2 = \{(1,P), (1,Q), (1,R), (2,P), (2,Q), (2,R)\}$
    
    $D_2 \times D_1 = \{(P,1), (P,2), (Q,1), (Q,2), (R,1), (R,2)\}$

  • For example, we have the relations:
    
    $R_1 \subseteq (D_1 \times D_2)$  \hspace{1cm} $R_1 = \{(1,Q), (1,R), (2,P)\}$
    
    $R_2 \subseteq (D_1 \times D_2)$  \hspace{1cm} $R_2 = \{(2,P), (2,Q), (2,R)\}$
    
    $R_3 \subseteq (D_2 \times D_1)$  \hspace{1cm} $R_3 = \{(P,2), (Q,1)\}$

• Members of a relation are called **tuples**

  Members of relations with **degree** $n$: $n$-tuples.
But first, what is the Relational Model?
Conceptual design/schema

- Conceptual model, e.g. expressed in ER
- Good to describe a mini-world
- On an abstract level, not good for any specific DBMS

Logical design/schema

- Implementation model
- Describes database design
- Data model for a DBMS, here Relational DBMS.

But first, what is the Relational Model?

ER to RM mapping

So, this is about translating from ER to tables.
ER to RM mapping

So, this is about translating from ER to tables
Translating entity types and their attributes

**Strong entity types**

Step 1: Entity types - a strong entity type reduces to a table with the same attributes.
- Key attributes (primary key - pk) is made the primary key column(s) for the table.
- Each attribute gets their own column.
- Composite attributes are normally represented by their simple components.
- Example customer schema and table:

  ```
  Customer(social-security, customer-name, c-street, c-city)
  ```

<table>
<thead>
<tr>
<th>social-security</th>
<th>customer-name</th>
<th>c-street</th>
<th>c-city</th>
</tr>
</thead>
<tbody>
<tr>
<td>321-12-3123</td>
<td>Jones</td>
<td>Main</td>
<td>Harrison</td>
</tr>
<tr>
<td>019-28-3746</td>
<td>Smith</td>
<td>North</td>
<td>Rye</td>
</tr>
<tr>
<td>677-89-9011</td>
<td>Hayes</td>
<td>Main</td>
<td>Harrison</td>
</tr>
</tbody>
</table>

**Weak entity types**

Step 2: **Weak entity types** - a weak entity type becomes a table that includes a column for the primary key of the identifying strong entity type.

The table corresponding to a relationship type linking a weak entity type to its identifying strong entity type is redundant.
Strong entity types

Step 1: Entity types - a strong entity type reduces to a table with the same attributes.

- Key attributes (primary key - pk) is made the primary key column(s) for the table. Each attribute gets their own column.
- Composite attributes are normally represented by their simple components.
- Example customer schema and table:

  \[
  \text{Customer}(\text{social-security, customer-name, c-street, c-city})
  \]

<table>
<thead>
<tr>
<th>social-security</th>
<th>customer-name</th>
<th>c-street</th>
<th>c-city</th>
</tr>
</thead>
<tbody>
<tr>
<td>321-12-3123</td>
<td>Jones</td>
<td>Main</td>
<td>Harrison</td>
</tr>
<tr>
<td>019-28-3746</td>
<td>Smith</td>
<td>North</td>
<td>Rye</td>
</tr>
<tr>
<td>677-89-9011</td>
<td>Hayes</td>
<td>Main</td>
<td>Harrison</td>
</tr>
</tbody>
</table>
Step 2: **Weak entity types** - a weak entity type becomes a table that includes a column for the primary key of the identifying strong entity type.

The table corresponding to a relationship type linking a weak entity type to its identifying strong entity type is redundant.
Translation of relationships

1-1 Relationship

1-N Relationship
- Include the primary key of the "1-side" as a foreign key on the "N-side", else the foreign key column is placed on the entity on the N-side.
- Alternatively, an extra table (R) is created whose primary key is a foreign key composed by the primary key from the N-side.

Alt 1:

<table>
<thead>
<tr>
<th>R</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>a1</td>
<td>a</td>
<td>b</td>
</tr>
<tr>
<td>a2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Alt 2:

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>a1</td>
<td>a</td>
<td>b</td>
<td>c</td>
</tr>
<tr>
<td>a2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

M-N Relationship
- Always a separate table with columns for the primary keys of the two participating entity types, and any descriptive attributes of the relationship type.

<table>
<thead>
<tr>
<th>M</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>a1</td>
<td>a2</td>
</tr>
<tr>
<td>a3</td>
<td></td>
</tr>
<tr>
<td>b1</td>
<td>b2</td>
</tr>
<tr>
<td>b3</td>
<td>b4</td>
</tr>
</tbody>
</table>
The foreign key column (fk) is a copy of the other entity’s primary key column (pk). The values in a fk-column point to a unique row in the other table, and thus implement the relationship.
- The foreign key column (fk) is a copy of the other entity’s primary key column (pk). The values in a fk-column point to unique row in the other table, and thus implement the relationship.

Alt 1:

```
<table>
<thead>
<tr>
<th>pk1</th>
<th>a1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Alt 2:

```
<table>
<thead>
<tr>
<th>pk1</th>
<th>a1</th>
<th>fk2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>pk2</th>
<th>a2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
1-N Relationship

- Include the primary key of the “1-side” as a foreign key on the “N-side”, (i.e. the foreign key column is placed on the entity on the N-side).
- Alternatively, an extra table (R) is created whose primary key is a foreign key composed by the primary key from the N-side.

Alt 1:

Alt 2:
M-N Relationship

- Always a separate table with columns for the primary keys of the two participating entity types, and any descriptive attributes of the relationship type.
Translation of relationships

1-1 Relationship

1-N Relationship
- Include the primary key of the "1-side" as a foreign key on the "N-side", else the foreign key column is placed on the entity on the N-side.
- Alternatively, an extra table (R) is created whose primary key is a foreign key composed by the primary key from the N-side.

M-N Relationship
- Always a separate table with columns for the primary keys of the two participating entity types, and any descriptive attributes of the relationship type.
Translation of multivalued attributes and relationships
Multivalued attributes

A separate table is created for the multivalued attribute. Its primary key is composed of the owning entity’s primary key, and the attribute value itself.
Multivalued relationships

- First try to remove multivalued relationships on the E-R model level by model transformation.
- A separate table is created, with foreign keys to all tables that are included in the relationship. Its primary key is composed of all foreign keys.

- In the case where R is 1-N-N, the primary key on R shall not include the fk for the table with cardinality 1.
- First try to remove multivalued relationships on the E-R model level by model transformation.
- A separate table is created, with foreign keys to all tables that are included in the relationship. Its primary key is composed of all foreign keys.
– In the case where R is 1-N-N, the primary key on R shall not include the fk for the table with cardinality 1.
<table>
<thead>
<tr>
<th>E-R concept</th>
<th>Relational concept</th>
</tr>
</thead>
<tbody>
<tr>
<td>entity type</td>
<td>relation</td>
</tr>
<tr>
<td>1:1 relationship type</td>
<td>include one of the primary keys as a foreign key of the other &quot;entity relation&quot;</td>
</tr>
<tr>
<td>1:N relationship type</td>
<td>include the &quot;1-side&quot; primary key as a foreign key at the &quot;n-side&quot;</td>
</tr>
<tr>
<td>M:N relationship type</td>
<td>relation with two foreign keys</td>
</tr>
<tr>
<td>n-ary relationship type (degree &gt; 2)</td>
<td>relation with n foreign keys</td>
</tr>
<tr>
<td>simple attribute</td>
<td>attribute</td>
</tr>
<tr>
<td>composite attribute</td>
<td>simple attribute components</td>
</tr>
<tr>
<td>multivalued attribute</td>
<td>relation and foreign key</td>
</tr>
<tr>
<td>value set</td>
<td>domain</td>
</tr>
<tr>
<td>key attribute</td>
<td>primary (or secondary key)</td>
</tr>
</tbody>
</table>
ER to RM mapping

So, this is about translating from ER to tables
Figure 8.1
The ER conceptual diagram for the Company database
Figure 8.1
The ER conceptual diagram for the Company database
Result of mapping the COMPANY ER schema into a relational database schema.
ConCEPTUAL DESIGN/SCHeMA

Conceptual model, e.g. expressed in ER
Good to describe a mini-world
On an abstract level, not good for any specific DBMS

ER

ER to RM MAPPING
So, this is about translating from ER to tables

LOGICAL DESIGN/SCHeMA

Implementation model
Describes database design
Data model for a DBMS, here Relational DBMS.

But first, what is the Relational Model?