DATABASDESIGN FÖR INGENJÖRER - 1056F

Sommar 2005

En introduktionskurs i databassystem

http://user.it.uu.se/~udbl/dbt-sommar05/
alt. http://www.it.uu.se/edu/course/homepage/dbdesign/st05/

Kjell Orsborn (Ruslan Fomkin)
Uppsala Database Laboratory
Department of Information Technology, Uppsala University,
Uppsala, Sweden
Introduction to the Relational Model

Elmasri/Navathe ch 5, 7

Kjell Orsborn &
Ruslan Fomkin

Department of Information Technology
Uppsala University, Uppsala, Sweden
The Relational Model

- The relational model was introduced by Dr. Edgar (Ted) F. Codd (1924-2003) in 1970.
  - Dr. Codd, a mathematician from Oxford (UK), was at that time working as an IBM researcher in the IBM San Jose Research Laboratory (USA).
- Many DBMS’s are based on the relational data model.
- It supports simple declarative, but yet powerful, languages for describing operations on data.
- Operations in the relational model applies to relations (tables) and produce new relations.
  - This means that an operation can be applied to the result of another operation and that several different operations can be combined.
  - Operations are described in an algebraic notation that is based on relational algebra.
Relations as mathematical objects

• In set theory, a relation is defined as a subset of the product set (cartesian product) of a number of domains (value sets).

• The product set of the domains $D_1,D_2,...,D_n$ is written as $D_1 \times D_2 \times ... \times D_n$.

• $D_1 \times D_2 \times ... \times D_n$ constitute the set of all ordered sets $<v_1,v_2,...,v_n>$ such that $v_i$ belongs to $D_i$ for all $i$.
  • If $n=2$, $D_1=\{T, F\}$ and $D_2=\{P, Q, R\}$ one gets the product sets:
    $D_1 \times D_2 = \{<T,P>,<T,Q>,<T,R>,<F,P>,<F,Q>,<F,R>\}$
    $D_2 \times D_1 = \{<P,T>,<P,F>,<Q,T>,<Q,F>,<R,T>,<R,F>\}$
  • For example, we have the relations:
    $R_1 \subseteq D_2 \times D_1$  \hspace{1cm} $R_1= \{<P,T>,<Q,T>,<R,T>\}$
    $R_2 \subseteq D_2 \times D_1$  \hspace{1cm} $R_2= \{<P,T>,<P,F>\}$

• Members of a relation is called **tuples**. If the relation is of **degree** $n$, the tuples are called **$n$-tuples**.
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>
First Normal Form

• Only simple or atomic values are allowed in the relational model.
• Attributes is not allowed to have composite or multiple values.
• The theory for the relational model is based on these assumptions which is called:

_The first normal form assumption_
Null values

• 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.
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 $R$, 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 $R$. 
Keys - continued . . .

- A superkey k is *minimal* if there is no other superkey k' such that k' ⊂ 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.
Key examples

• Example superkey:
  – \{customer-name, customer-street\} and \{customer-name\} are both superkeys of \textit{Customer}, if no two customers can possibly have the same name.

• Example candidate key:
  – \{customer-name\} is a candidate key for \textit{Customer}, since it is a superkey (assuming no two customers can possibly have the same name), and no subset of it is a superkey.
Integrity constraints for a relational database schema

• 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”
Steps in translation from E-R model to relational model

• Translation of entity types and their attributes
  – Step 1) Entity types
  – Step 2) Weak entity types

• Translation of relationships
  – Step 3) 1-1 Relationship
  – Step 4) 1-N Relationship
  – Step 5) M-N Relationship

• Translation of multivalued attributes and relationships
  – Step 6) Multivalued attributes
  – Step 7) Multivalued relationships
Determining keys from E-R types

- **Strong entity type.** The primary key of the entity type becomes the primary key of the relation.

- **Weak entity type.** The primary key of the relation consists of the union of the primary key of the strong entity type and the discriminator of the weak entity type.

- **Relationship type.** The union of the primary keys of the related entity types becomes a super key of the relation.
  - For binary many-to-many relationship types, above super key is also the primary key.
  - For binary many-to-one relationship types, the primary key of the “many” entity type becomes the relation’s primary key.
  - For one-to-one relationship types, the relation’s primary key can be that of either entity type.
Translating entity types and their attributes

• 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:

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

<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>
Translating entity types cont. . .

- 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.

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

```
<table>
<thead>
<tr>
<th>pk</th>
<th>k</th>
<th>a2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
Translating entity types cont. . .

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

- Example of the payment schema and table:
  - The payment table already contains the information that would appear in the loan-payment table (i.e., the columns loan-number and payment-no).

\[ \text{Payment(loan-number, payment-no, pay-date, amount)} \]

<table>
<thead>
<tr>
<th>loan-number</th>
<th>payment-no</th>
<th>pay-date</th>
<th>amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>L-17</td>
<td>5</td>
<td>10 May 1996</td>
<td>50</td>
</tr>
<tr>
<td>L-23</td>
<td>11</td>
<td>17 May 1996</td>
<td>75</td>
</tr>
<tr>
<td>L-15</td>
<td>22</td>
<td>23 May 1996</td>
<td>300</td>
</tr>
</tbody>
</table>
Translating relationship types

- Step 3: 1-1 Relationship types
  - 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>
</table>

Alt 2:

<table>
<thead>
<tr>
<th>pk1</th>
<th>a1</th>
<th>fk2</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>pk2</th>
<th>a2</th>
<th></th>
</tr>
</thead>
</table>
Translating 1-1 relationship types cont. . .

Alt 3:

<table>
<thead>
<tr>
<th>E1</th>
<th>R</th>
<th>E2</th>
</tr>
</thead>
<tbody>
<tr>
<td>(pk_1)</td>
<td>(a_1)</td>
<td>(f_{k1})</td>
</tr>
</tbody>
</table>

Alt 4:

<table>
<thead>
<tr>
<th>E1</th>
<th>E2</th>
</tr>
</thead>
<tbody>
<tr>
<td>(pk_1)</td>
<td>(a_1)</td>
</tr>
</tbody>
</table>
Translating relationship . . . cont.

- **Step 4: 1-N Relationship types**
  - 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.

```
<table>
<thead>
<tr>
<th>1</th>
<th></th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Alt 1:

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

```
<table>
<thead>
<tr>
<th>pk2</th>
<th>a2</th>
<th>f k1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Alt 2:

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

```
<table>
<thead>
<tr>
<th>f k1</th>
<th>f k2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<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>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
Translating relationship . . . cont. . .

• Step 5: M-N Relationship types
  – Always a separate table with columns for the primary keys of the two participating entity types, and any descriptive attributes of the relationship type.
Translating relationship . . . cont. . .

• Step 6: 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.
Translating relationship . . . cont. . .

• Step 7: Multivalued relationship types
  – 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.
Translating relationship . . . cont. . .

- Step 7: Multivalued relationship types continued
  - 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.
Example E-R to relational model translation
Relational schemas for the example
Data for the example schema
From E-R to relational model

• The basic procedure defines a set of relational schemas that represent entity and relationship types in the E-R model. This model should further with integrity constraints.
  – Primary keys allow entity types and relationship types to be expressed uniformly as tables which represent the contents of the database.
  – A database which conforms to an E-R diagram can be represented by a collection of tables.
  – For each entity type and relationship type there is a unique table which is assigned the name of the corresponding entity type or relationship type.
  – Each table has a number of columns (generally corresponding to attributes), which have unique names.
  – Converting an E-R diagram to a table format is the basis for deriving a relational database design from an E-R diagram.
Summary

• Entity types and their attributes
  – Step 1) Entity types
    • Each entity gets a corresponding table, with the primary key column set to its key attribute.
  – Step 2) Weak entity types
    • The primary key of a weak entity type table has the primary key of the owner table as a component.

• Relationships
  – Step 3) 1-1 Relationship
    • 4 alternatives: fk in E1 or E2, separate R table, common table for E1 & E2
  – Step 4) 1-N Relationship
    • fk in entity on the N-side, separate R table
  – Step 5) M-N Relationship
    • separate R table
Summary cont. . .

• Multivalued attributes and relationships
  – Step 6) Multivalued attributes
    • Separate table for the attribute with its pk composed of the owner pk and the value column.
  – Step 7) Multivalued relationships
    • Separate R table. N-N-N: pk composed of all fk’s. 1-N-N: pk is fk to the E1-table.
## Short summary E-R -> R

<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 anf 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>