CSI T5300: Advanced Database Systems

L02: Relational Data Model

Dr. Kenneth LEUNG

Department of Computer Science and Engineering
The Hong Kong University of Science and Technology
Hong Kong SAR, China
• Entities and relationships of the E-R model are stored in **tables** also called **relations** (not to be confused with **relationships** in the E-R model)

• **Benefits:**
  - There can be well-defined semantics and languages for manipulating the tables
  - Ease of implementation – write queries on tables without caring about the physical level and optimization issues

• Most popular DBMSs today are based on the relational data model (or an extension of it, e.g., object-relational data model)
Terminology

• Relation ↔ table: denoted by R(A₁, A₂, ..., Aₙ) where R is a relation name and (A₁, A₂, ..., Aₙ) is the relation schema of R
• Attribute (column) ↔ denoted by Aᵢ
• Tuple (Record) ↔ row
• Attribute value ↔ value stored in a table cell
• Domain ↔ legal type and range of values of an attribute denoted by dom(Aᵢ)
  - Attribute: Age             Domain: [0-100]
  - Attribute: EmpName        Domain: 50 alphabetic chars
  - Attribute: Salary         Domain: non-negative integer
<table>
<thead>
<tr>
<th>Name</th>
<th>Student-id</th>
<th>Age</th>
<th>CGA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chan Kin Ho</td>
<td>99223367</td>
<td>23</td>
<td>11.19</td>
</tr>
<tr>
<td>Lam Wai Kin</td>
<td>96882145</td>
<td>17</td>
<td>10.89</td>
</tr>
<tr>
<td>Man Ko Yee</td>
<td>96452165</td>
<td>22</td>
<td>8.75</td>
</tr>
<tr>
<td>Lee Chin Cheung</td>
<td>96154292</td>
<td>16</td>
<td>10.98</td>
</tr>
<tr>
<td>Alvin Lam</td>
<td>96520934</td>
<td>15</td>
<td>9.65</td>
</tr>
</tbody>
</table>
• A relation schema is denoted by: \( R(A_1, A_2, \ldots, A_n) \)
  - \( \text{STUDENT}(\text{Name}, \text{Student-id}, \text{Age}, \text{CGA}) \)
• Degree of a relation: the number of attributes \( n \) in the relation.
• Tuple \( t \) of \( R(A_1, A_2, \ldots, A_n) \): An ordered set of values \(<v_1, v_2, \ldots, v_n>\) where each \( v_i \) is an element of \( \text{dom}(A_i) \).
• Relation instance \( r(R) \): A set of tuples in \( R \).
  - \( r(R) = \{t_1, t_2, \ldots, t_m\} \), or alternatively

\[
r(R) \subseteq \text{dom}(A_1) \times \text{dom}(A_2) \times \ldots \times \text{dom}(A_n)
\]
• A relation is any subset of the Cartesian product of the domains of values

• Example:

Let

\[
\text{Dom}(\text{Name}) = \{ \text{Lee, Cheung} \} \\
\text{Dom}(\text{Grade}) = \{ A, B, C \}
\]

Then the Cartesian product of the domains is

\[
\text{Dom}(\text{Name}) \times \text{Dom}(\text{Grade}) = \{ <\text{Lee, A}>, <\text{Lee, B}>, <\text{Lee, C}>, \\
<\text{Cheung, A}>, <\text{Cheung, B}>, <\text{Cheung, C}> \}
\]

A relation StudentGrade(Name, Grade) can be defined as any subset of the Cartesian product Dom(Name) × Dom(Grade), e.g.,

\[
r(\text{StudentGrade}) = \{ <\text{Lee, A}>, <\text{Cheung, C}> \} \subseteq \text{Dom}(\text{Name}) \times \text{Dom}(\text{Grade})
\]
• Tuples in a relation are not considered to be ordered, even though they appear to be in a tabular form (recall that a relation is a set of tuples)
• All attribute values are considered atomic. Multivalued and composite attribute values are not allowed in tables, although they are permitted by the E-R diagrams
• A special null value is used to represent values that are:
  - Not applicable (phone number for a client that has no phone)
  - Missing values (there is a phone number, but we do not know it yet)
  - Not known (we do not know whether there is a phone number or not)
Let $K \subseteq R$ (i.e., $K$ is a set of attributes which is a subset of the schema of $R$)

$K$ is a superkey of $R$, if $K$ can identify a unique tuple in a given instance $r(R)$

**Example:**

$Student(SID, HKID, Name, Address, \ldots)$

where $SID$ and $HKID$ are unique

Possible superkeys:

- $\{SID\}$
- $\{HKID\}$
- $\{SID, Name\}$
- $\{HKID, Name, Address\}$
- plus many others

$K$ is a candidate key if $K$ is minimal

- In the above example there are two candidate keys: $SID$ and $HKID$

Every relation must have at least one candidate key

If there are multiple, one is chosen as the primary key
Storing all information as a single relation such as 
\[\text{bank}(\text{account-number}, \text{balance}, \text{customer-name}, \text{customer-addr}, \ldots)\]
results in
- repetition of information (e.g. repeat the customer info for each of his/her accounts)
- the need for null values (e.g. represent a customer without an account)

That is why we need the E-R diagrams (and some additional normalization techniques discussed later) to break up information into parts, with each relation storing one part.

**Example:**
- **Account**: stores information about accounts
- **Customer**: stores information about customers
- **Depositor**: stores information about which customer owns which account
A database which conforms to an E-R diagram can be represented by a collection of tables.

Converting an E-R diagram to a table format is “automatic”.

For each entity set there is a unique table which is assigned the name of the corresponding entity set.

Each table has a number of columns (generally corresponding to attributes), which have unique names.
Composite attributes are flattened out by creating a separate attribute for each component attribute
- E.g., given entity set customer with composite attribute name comprised of first-name and last-name, the customer table has two attributes: name.first-name and name.last-name

A multivalued attribute \( M \) of an entity \( E \) is represented by a separate table \( EM \)
- Table \( EM \) has attributes corresponding to the primary key of \( E \) and an attribute corresponding to multivalued attribute \( M \)
- **Example:**
  - Multivalued attribute phone-number of employee is represented by a table employee-phone(employee-id, phone-number)
  - Each value of the multivalued attribute maps to a separate row of the table EM
  - E.g., an employee with primary key 19444 and phones 23580000, 95555555 maps to two rows: (19444, 23580000) and (19444, 95555555)
Representing Weak Entities

- A weak entity set becomes a table that includes a column for the primary key of the identifying strong entity set.
A many-to-many relationship set is represented as a table with columns for the primary keys of the two participating entity sets, and any descriptive attributes of the relationship set.

**Example:**
- Table for relationship set *borrower*

```plaintext
<table>
<thead>
<tr>
<th>customer-id</th>
<th>loan-number</th>
</tr>
</thead>
<tbody>
<tr>
<td>019-28-3746</td>
<td>L-11</td>
</tr>
<tr>
<td>019-28-3746</td>
<td>L-23</td>
</tr>
<tr>
<td>244-66-8800</td>
<td>L-93</td>
</tr>
<tr>
<td>321-12-3123</td>
<td>L-17</td>
</tr>
<tr>
<td>335-57-7991</td>
<td>L-16</td>
</tr>
<tr>
<td>555-55-5555</td>
<td>L-14</td>
</tr>
<tr>
<td>677-89-9011</td>
<td>L-15</td>
</tr>
<tr>
<td>963-96-3963</td>
<td>L-17</td>
</tr>
</tbody>
</table>
```
Many-to-one and one-to-many relationship sets that are total on the many-side can be represented by adding an extra attribute to the many side, containing the primary key of the one side.

**Example:**
- Instead of creating a table for relationship `account-branch`, add the key of `branch` (`branch-name`) to the entity set `account`.
- `branch-name` in `account` is a foreign key.
For one-to-one relationship sets, either side can be chosen to act as the “many” side in the previous slide.
- That is, extra an attribute can be added to either of the tables corresponding to the two entity sets.

If participation is partial on the many side, adding an extra attribute in the relation corresponding to the “many” side could result in null values (redundancy).
- Maybe it is better to represent the relationship set by a separate table as in the many-to-many case.

The table corresponding to a relationship set linking a weak entity set to its identifying strong entity set is redundant.
- Example: The payment table already contains the information that would appear in the loan-payment table (i.e., the columns loan-number and payment-number).
• **Method 1:**
  - Form a table for the higher level entity
  - Form a table for each lower level entity set, and include the primary key of the higher level entity set and local attributes
  - **Example:**

<table>
<thead>
<tr>
<th>table</th>
<th>table attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>person</td>
<td>id, name, street, city</td>
</tr>
<tr>
<td>customer</td>
<td>id, credit-rating</td>
</tr>
<tr>
<td>employee</td>
<td>id, salary</td>
</tr>
</tbody>
</table>

- **Drawback:** getting information about, e.g., *employee*, requires accessing two tables
Method 2:
- Form a table for each entity set with all local and inherited attributes
- Example:

<table>
<thead>
<tr>
<th>table</th>
<th>table attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>person</td>
<td>id, name, street, city</td>
</tr>
<tr>
<td>customer</td>
<td>id, name, street, city, credit-rating</td>
</tr>
<tr>
<td>employee</td>
<td>id, name, street, city, salary</td>
</tr>
</tbody>
</table>

- If the specialization is total, no need to create a table for the generalized entity (person)
- Drawback: street and city may be stored redundantly for persons who are both customers and employees