Handout 4

Logical Database Modeling, Part 1:
Relational Data Model.
Transforming EER model to Relational.

1. Relational Data Model.

Introduced in a journal article written in 1970 by Dr. Edgar F. Codd, from IBM. Relational DBMS Dominate the database market till the present time - strong theoretical foundation enabled development of fast and well-defined algorithms for data querying and manipulation.

Early research prototypes of relational systems were developed throughout the 1970s Commercial RDBMS emerged in the 1980s;

Main Components:

Relation.

Data (entity and relationship instances) are stored in named, two-dimensional tables, called relations.

Click to see an example.

Relational model vs. ER model

- Relations (tables) correspond to entity types and to many-to-many relationship types
- Rows (a.k.a. records, tuples) correspond to entity instances and to many-to-many relationship instances
- Columns (a.k.a. fields) correspond to attributes

Note the difference between the terms RELATION in the relational model and RELATIONSHIP in the ER model.

Not all tables qualify as relations. Requirements on relations:
1. Every relation has a unique name.
2. Attributes in have unique names within each table.
3. Every attribute value is atomic (not multivalued, not composite)
4. Every row is unique (can’t have two rows with exactly the same values for all their fields)
5. The order of the columns is irrelevant
6. The order of the rows is irrelevant

A field in a record may have a value or in some cases be NULL, i.e. contain no data.
Keys:

Candidate key – a collection of attributes whose value uniquely identifies (differentiates) each row in a relation.

One Candidate Key must be designated as a Primary Key

Every relation must have a Primary Key:

Designated Attribute (or a collection of attributes) whose value uniquely identifies (differentiates) each row in a relation (e.g., Employee_ID). This is how we can guarantee that all rows are unique.

Foreign Key:

Attribute (or set of attributes) in one relation that serves as the primary key of another relation in the same database, thus linking the records of two tables.

Composite Key:

a key made up of more than one attribute (e.g., F_Name + M_Name + L_Name)

Practice problem:

A database is used to keep records of all company employees, departments, and customers. Each customer is assigned a customer representative – an employee of the company, and belongs to one of several customer groups.

Given the following DB schema underline the primary key of each relation. Use a dashed line to identify foreign keys.

**EMPLOYEE** (Emp_nbr, Emp_lname, Emp_fname, Emp_phone, Emp_dateofbirth, Emp_date_hired, Emp_nbr_of_dependents, Emp_dept)

**DEPT** (Dept_nbr, Dept_name, Dept_phone, Dept_building, Dept_mgr)

**CUSTOMER** (Cust_nbr, Cust_name, Cust_street_address, Cust_city, Cust_state, Cust_zip, Cust_phone, Cust_rep, Cust_contact_date, Cust_group)

**CUSTGROUP** (Custg_nbr, Custg_name)
Integrity Constraints

Built-in mechanism that enables enforcement of certain rules facilitating data accuracy and integrity (validity, consistency)

**Domain constraints:** the values in each of the columns of a relation have to come from a certain domain (domain - set of possible values of an attribute).

**Entity integrity:** no primary key value can be NULL. Primary key value should be unique.

**Referential integrity:** a foreign key value has to be either NULL or a valid primary key value in the relation to which the foreign key refers.

**Action assertions:** enforce conditions on actions (adding/deleting/modification of records) that implement some business rules, e.g. “a student may not register for a course unless they have passed all prerequisites”

All part of SQL (data definition and manipulation language for relational model) – we will study this later.

2. Transforming EER diagrams into Relations.

a. Transforming **Entities:**
   (create relation, identify primary key, identify foreign key)

Create a relation.
Include a field for each simple (non-composite) single-valued attribute
**Composite** Attributes - include only the component attributes.

**Multivalued Attribute** – create a separate table. Include all primary key attributes from the main table – they become the **foreign key** in the new table. **Primary Key** of this table is a composite key that includes all fields.

**Example:**

<table>
<thead>
<tr>
<th>EMPLOYEE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Employee_ID</td>
</tr>
<tr>
<td>Employee_Name</td>
</tr>
<tr>
<td>Skill (MV)</td>
</tr>
</tbody>
</table>

is represented as follows: note the composite primary key in SKILL

EMPLOYEE (Employee_ID, Employee_Name )

SKILL (Skill_emp_ID, Skill)
If it is a **Weak Entity** (always associated with the owner-entity through an identifying relationship; only has a partial identifier)

Include the primary key field of the owner-entity table – it becomes a foreign key to the owner-entity’s table.

**Primary key of the weak-entity’s table** = attributes of partial identifier + foreign key to owner-entity’s table.

b. Transforming **Relationships:**

*Assuming the tables for participating Entities have already been created. Only need to create field to link appropriate records from each of the two tables, i.e. add foreign keys.*

**one-to-one** relationship between entities A and B -

- include the primary key fields of A into B’s table as a foreign key,
- or include the primary key fields of B into A’s table as a foreign key

(Note that it is better to place the foreign key in the table whose relationship to the other is mandatory, not optional. This prevents from storing NULL values in the table.)

include all attributes of the relationship into the same table.

**one-to-many** relationship between entities A and B –

- include the primary key fields of A into B’s table as a foreign key
- include attributes of the relationship into B.

**many-to-many** relationship between entities A and B

create a new table C

- include into C all primary key fields of A and B as foreign keys to each table,
- and all attributes of the relationship.

**Primary key** of C is a composite key consisting of both foreign keys.

c. Transforming **Associative Entities:**

C is an associative entity between entities A and B.

Create a new table for C. Include all attributes of the associative entity.

- **Associative Entity without an Identifier** (like many-to-many relationship)
  - Include primary keys from A and B into the table for C. These fields become foreign keys to A and B respectively.
  - **Default primary key** for the associative relation (also called the intersection table) is a composite key composed of the primary keys of the two entities.
- **Associative Entity with an Identifier**
  
  In some situations, the combination of the primary keys of A and B does not uniquely identify each record in C.

  **Example:** A Patient receives a Treatment from a Physician.

  Then, a different identifier is used for a primary key of the associative table C.
  
  Often, this other identifier consists of a combination of the primary keys of A and B plus some other attribute(s).

**d. Transforming Ternary Relationships:**

Transform the ternary relationship into an associative entity in the ER model (as described in Chapter 3).

Apply the steps of the associative entity as described in part c. above, noting that the associative entity connects three other relationships, instead of two.

**Practice problem:** Consultant works on a Problem of a Client.

**e. Transforming Supertype/Subtype relationships:**

Create a table for the supertype. Include all attributes of the supertype (i.e. those that are common to all subtypes).

Create a table for each subtype. In each such table include a field for the primary key of the supertype, and all attributes unique to the subtype.