Data Modeling  -  Part 2

Relational  Model

MBA 8473

Learning Objectives (Module 2 continued)

32. Examine the purpose of data modeling in information management
33. Explain the three-step process of data-driven information system (IS) development
34. Explain Data Model quality issues.
35. Provide examples of four common difficulties in a poorly designed data model: (1) redundancies, (2) update, (3) deletion anomalies, and (4) insertion anomalies.
36. Explain entity-relationship (ER) model and identify its building blocks: Entity, Identifier, Relationship, and Attribute.
37. Explain three properties of an entity-relationship: (1) Degree of a relationship, (2) Minimum and Maximum Cardinalities, and (3) Optional and Mandatory Associations.
38. Do an ER Model or Diagram.
39. Explain relational model, and identify its building blocks, and underlying core principles: (1) Tables and attributes, (2) Primary keys, and entity integrity, (3) Foreign keys and referential Integrity.
40. Apply rules for mapping E-R models to the relational model.
41. Apply normalization theory to assess the quality of a relational model up to 1st Normal Form.
42. Apply normalization theory to assess the quality of a relational model up to 3rd Normal Form.
Organization of Concepts

The Rationale
- Shared Understanding
- Uncover Relationships
- Prevent Anomalies
  - Insert
  - Update
  - Delete

Building Blocks
- Entity
- Attributes
- Relationships
- Identifiers

Types of Relationships
- Degree of a Relationship
- Min. & Max. Cardinalities
- 1:1, 1:M, M:N
- Optional & Mandatory
- Recursive
- Supertype - Subtype

Data Modeling

E-R Model

Building Blocks
- Tables
- Attributes
- Primary Key
- Foreign Key

Relational Model

Building Blocks
- Integrity
- Primary Key
- Referential
- Domain

Data Quality
- Normalization Theory

Types of Relationships

Entity

Attributes

Relationships

Identifiers

Data-Driven IS Development (c.o. 33)

Problem Domain

Conceptual Design

Conceptual Schema, e.g., ER Model

Logical Design

Logical Schema, e.g., Relational Model

Physical Design

Physical Schema, e.g., Access implementation
Building Blocks of the Relational Model (c.o. 39)

A relational model typically consists of several interrelated tables to logically capture the business concept expressed in an ER Model.

- Table (also called a ‘relation’)
  - A table mostly represents an entity of ER Model
  - A table has n columns and m rows
- Give each table (relation) a meaningful name
  - STUDENT, EMPLOYEE, CAR, COMPANY

Note the distinction between a relation in the relational model and a relationship in the E-R model

Representing Attributes, Instances (c.o. 39)

- Attributes of an Entity (is represented by the Columns of a Table)
  - For each attribute, the appropriate domain must be defined
  - Acceptable Majors, Range of acceptable values for GPA.
- Instances of an Entity are represented by the Tuples or rows of a Table
  - Table_Name(Attribute[1], Attribute[2], …. Attribute[n])
    - STUDENT(SocSecNO, Major, GPA, Work Experience)
    - CAR(VIN#, Model, Make, Year)
**Instances of an Entity represented by the Tuples or rows of a Table (c.o. 39)**

- **Tuples (rows)**
  - A row represents data about a specific instance
  - Each row should be uniquely identifiable
  - No two rows can be identical (ensured by using Primary Key – see next slide)
  - The attribute values reflect the characteristics of the specific instance

- **One Representation Approach**
  - Table_Name(Attribute[1], Attribute[2], …. Attribute[n])
  - STUDENT(SocSecNO, Major, GPA, Work Experience)
  - CAR(VIN#, Model, Make, Year)

---

**Entity Integrity improves data quality: Use of Primary Key (c.o. 39)**

- **Basis to ensure entity integrity**
  - A primary key uniquely identifies every possible instance (row) in a relational table
  - It can be defined by one or multiple attributes
  - No primary key (or any component thereof) of a relational table can be null

- Useful to have semantically meaningful primary keys
  - STUDENT(SocSecNO, Major, GPA, Work Experience)
  - Course(Course#, Name, Credit Hours)
  - REGISTRATION(Student#, Course#, Grade)

*The primary key must be clearly indicated for each table*
Referential Integrity improves data quality: Use of Foreign Keys (c.o. 39)

- Foreign key is defined as “an attribute of a relational table that is the primary key of another relational table.”
  - Becomes the basis to logically link tables or entities and to referential integrity.
  - Can be composite of two or more attributes
- Consider the all popular (by now!), student-course-registration scenario
  - STUDENT(SSNO, major, GPA, work experience)
  - COURSE(Course#, Name, Credit Hours)
  - REGISTRATION(SSNO, Course#, Quarter Taken, Grade)

Mapping E-R Model to the Relational Model (c.o. 40)

- Each entity is defined and implemented as a table. Use the entity name as the table name. Make the entity identifier as the primary key for the table. Examine domain constraints for each attribute.
- 1:1 relationship
  - Include the primary key of one of the tables in the second one. This becomes the foreign key and enables linkage between the two tables.
- 1:M relationship
  - Add the identifier of the first entity (one-side) as an attribute of the second table (many-side). This becomes the foreign key and enables linking the two tables.
  - Note here that you should not add the identifier of the second entity (many-side) as an attribute of the first table (one-side).
- M:N relationship
  - Create a table with the attributes at the intersection of the two entities, i.e. the relationship. Concatenate the primary keys of the entities participating in the relationship to create the primary key for the intersection table.
  - What are the foreign keys in this case?

- A real example from Procter and Gamble follows.
EXAMPLE: Procter and Gamble used to have the following relationship between SALES-AGENT and STORE, before and after they redesigned their outbound supply chain:

**ER Model – Before P&G’s business process redesign:** A many to many (M:N) relationship.

SALES-AGENT

\[\text{Assigned to}\]

STORE

Attributes: EMPLOYEE-NUMBER
EMPLOYEE-F-NAME
EMPLOYEE-L-NAME

... Attributes: STORE-NUMBER
STORE-NAME
STORE-STREET-ADDRESS

**ER Model – After P&G’s Business process redesign:** A one to one (1:1) relationship. It was based on a business decision.

SALES-AGENT

\[\text{Assigned to}\]

STORE

Mapping a 1:1 relationship into a Relational model – alternative 1:

**Tables:**

SALES-AGENT (EMPLOYEE-NUMBER, EMPLOYEE-F-NAME, EMPLOYEE-L-NAME, ...)
STORE (STORE-NUMBER, EMPLOYEE-NUMBER, STORE-NAME, ...)

Mapping a 1:1 relationship into a Relational model – alternative 2:

**Tables:**

SALES-AGENT (EMPLOYEE-NUMBER, STORE-NUMBER, EMPLOYEE-F-NAME, ...)
STORE (STORE-NUMBER, EMPLOYEE-NUMBER, STORE-NAME, ...)
EXAMPLE: Let’s say Procter and Gamble made it a 1:M (instead of a 1:1) relationship as a business decision.

ER Model – A one to many (1:M) relationship. No change in attributes.

Attributes: EMPLOYEE-NUMBER
EMPLOYEE-F-NAME
EMPLOYEE-L-NAME
...

Attributes: STORE-NUMBER
STORE-NAME
STORE-STREET-ADDRESS

Mapping into a Relational model (Similar to alternative 1 of 1:1 Relational Model shown earlier):

Tables:

SALES-AGENT (EMPLOYEE-NUMBER, EMPLOYEE-F-NAME, EMPLOYEE-L-NAME, …)
STORE (STORE-NUMBER, STORE-NAME, STORE-STREET-ADDRESS, …)

(Note: Alternative 2 of 1:1 Relational Model shown earlier NOT allowed here.)

Relational model Alternative 1 from last slide implemented in Access here:
To implement a 1:1 (or, 1:M) relationship in Access use Relationships menu and Add tables of the relationship to plate. Here we show a 1:1.

Then use Edit Relationship and match identifier according to the intended relationship.

A one-to-many implementation in Access.
Normalization (c.o. 41)

- A process to design good relations
- Not a requirement of the relational model or any other model
- Attempts to reflect the semantics of the data. This meaning is determined by the organization.
  - What does salary mean? Profits? Stock Price?
- Relations can be in 1 NF, 2 NF, 3 NF, Boyce-Codd NF, 4 NF, and 5 NF
- We will concern ourselves with 1 NF, 2 NF, and 3 NF.

First Normal Form (c.o. 41)

- All attributes must be single-valued
  - Each row can assume only one value for a given attribute
- Consider employees working for a given department at a given salary level and we design an EMPLOYEE relation as:
  - EMPLOYEE(EMPNO, department, salary)
- Now consider employees who works for a given department and receive a raise every year. We are interested in maintaining data about each employee’s yearly salary. Will there be any problem due to the design of above relation?
- EMPLOYEE(EMPNO, YEAR, department, salary)
Second Normal Form (c.o. 42)

- Consider the following:
  - REGISTRATION (SSNO, CNO, quarter taken, grade, student_major, work experience)
  - What’s wrong with this table? What problems can this create?
- Quarter taken and grade are fully functionally dependent on the primary key, which is SSNO+CNO
- Student-major and work experience are dependent on part of the primary key (SSNO), but not on CNO
- Decompose the table into two tables
  -- Student(SSNO, student_major, work experience)
  -- Registration(SSNO, CNO, quarter taken, grade)

2nd NF - The Rule

- First, ensure that you are in 1 NF.
- Check if there are any partial functional dependencies.
- Engage in lossless decomposition to multiple tables. We use the term “lossless” to emphasize that no meaning (semantic content) is lost as a result of such decomposition.
- Notice that these tables can be “joined” back using the linking information
Third Normal Form (c.o. 42)

- Consider the following: A large urban university offers four types of scholarship, each with a predefined code. The type of scholarship that a student is eligible for is based on their major. An analyst sets up the following database:
  - Scholarship (SSNO, name, major, GPA, scholarship type)
  - What problems are we likely to encounter? Why?
  - Note that scholarship type is not dependent on (determined by) SSNO. Rather, it is determined by major, which is determined by SSNO.
  - This represents a transitive dependency.

Third Normal Form

- Decompose the relation into two relations
  - Student(SSNO, name, major, GPA)
  - Scholarship_Eligibility(Major, Scholarship_Type)
- The 3rd NF rule
  - Once you are in 2 NF, examine if there are any transitive dependencies
  - Engage in loss-less decomposition so as to remove these transitive dependencies
  - You are now in 3 NF!