Part 5

Entity-Relationship Modelling
Logical Database Design

- Constructive approach
- Considers *semantics*
- Documents
  - data dependencies
  - identifiers
  - entities
  - needed relations
  - “rules”
Entity-Relationship (E-R) Modelling

Graphical means of
  • naming and
  • depicting
the types of data in a database

Simple, yet precise

Useful to
  • technically-oriented analysts
  • application-oriented users

Easy to read

Supports the design task
  • logical structure design is hard
  • tool aids the design task
  • notation does not get in the way
Basic E-R Components

Entity
• any type of thing about which information is maintained

Attribute
• a characteristic of exactly one entity (fully functionally dependent on the entity)

Relationships
• an association between a pair of entities (or “roles”), one-to-one, one-to-many only

EXAMPLE: Student attributes

student

student_name
student_id#
soс_sec#
Example Relationships

1 - 1 Example: Monogamous marriage

Can label relationship

1-M Example: Students of a college

- Need not label a relationship if it can be stated as:
  college of student / students of college
  or
  student has college / college has students
Handling an M-M Relationship

M-M Example: Brother - Sister

Problem: how do you represent the presence of sibling rivalry?

THIS WON'T WORK

SOLUTION
Identifier Representation

Identifier: a set of attributes or relationships that uniquely identify an instance of an entity

Example:

(college_name

(college#

student_name

(student_id#

(soc_sec#
Primary Key / Candidate Key

- state_name
- state_abbrev
- city_name
- city#

- primary key
- candidate key
Sample Database

Employee: (emp)
  attributes: Ename, Job, Mgr, Hired, Rate, Bonus

Department: (dept)
  attributes: DeptNo, Dname, Loc, Dbudget

Task: (task)
  attributes: Tname, Hours

Project: (proj)
  attributes: Project_id, Description, Pbudget, Due_date

Relationships
  • employees are members of a department
  • employees have a manager who is an employee
  • employees are assigned to tasks on projects
LDS for Sample Database

dept
  DeptNo
  Dname
  Loc
  Dbudget

emp
  Ename
  Job
  Hired
  Rate
  Bonus
  Mgr

proj
  Project_id
  Description
  Pbudget
  Due_date

task
  Tname
  Hours

part 5
Functional Dependency Revisited

DeptNo identifies dept instances

- DeptNo -> Dbudget
  - Dbudget is fully functionally dependent on DeptNo
- DeptNo -> Loc
  - Loc is fully functionally dependent on DeptNo

Dname is an alternate key

- Dname -> Dbudget
  - Dbudget is fully functionally dependent on Dname

Ename identifies emp instances

- Ename -> Job
  - Job is fully functionally dependent on Ename
- Ename -> Rate
  - Rate is fully functionally dependent on Ename

an employee instance determines exactly one department

- Ename -> DeptNo
  - DeptNo is fully functionally dependent on Ename
- Ename -> Loc
  - Loc is fully functionally dependent on Ename, but this is a transitive full functional dependence
A supplier supplies many parts, and a part can be supplied by many suppliers
LDS for Example 2 - Inventory

A product can be stored in many warehouses and a warehouse can contain many products
LDS for Example 3 - Departments

A department can have many employees, and employee can only be in one department
LDS for Example 4 - Locations

Departments have one number, one name, and one location

Department

department

dep_num

department

dep_department

dep_name

dep_location
LDS for Example 5 - Stock

An inventory is comprised of combinations of various parts from various suppliers - a supplier can supply many parts, and a part can be supplied by many suppliers.
LDS for Example 6 - Enrollment

A student can take many subjects, a subject can be taken by many students. A subject can be taught by many teachers, a teacher can teach only one subject. A student can be taught by many teachers, a teacher can teach many students.
LDS for EXAMPLE 7 - SKILLS

Employees can have many skills, and a skill can be had by many employees; an employee can know many languages and a language can be known by many employees.

This diagram is correct if all 3 are interdependent

This diagram is almost never correct
(It implies that a skill can be held by only one employee)
CORRECT LDS for INDEPENDENCE

Assuming job skills and language skills are independent, they represent two separate many-to-many relationships.
LDS for EXAMPLE 8 - DEALERSHIPS

In the general case, a contract involves one dealer, one manufacturer, and one product. A dealer can have many contracts, a manufacturer can have many contracts, and a product can be mentioned in many contracts.

This diagram is correct in the general case.
DEALERSHIPS with CONSTRAINTS

Dealers can deal with many manufacturers, and manufacturers with many dealers. Dealers can sell many vehicle types and vehicle types can be sold by many dealers. Manufacturers can make many vehicles and vehicles can be made by many manufacturers. The combinations of who sells what is determined by symmetry.
LDS for EXAMPLE 9 - CUSTOMERS

A branch has many customers, a customer is in only one branch. There are only a limited number of legal branch names.
Modelling Concepts

Entities:
“it” must have
• identifier
• attributes
• relationships

“it” must be the focus of the system

need to develop for “it”:
• name
• description
• membership criteria

must examine roles within subsets of “it”

Attributes:
must be non-transitively fully functionally dependent on the entity it describes

must develop for each attribute:
• name
• description
• domain definition
Concept of Roles

when 2 entities share a set of attributes OR
when 2 entities have more than one relationship between
them OR
when subsets of an entity-instance have different
attributes OR
when subsets of an entity-instance participate in different
relationships

THEN MULTIPLE ROLES EXIST

Examples of roles in the sample database:
  In emp, an employee plays 2 roles:
  • works in a department
  • (some) manages department
  In emp
  • regular employees report to managers in the same
department
  • managers report to managers in a different
department
  In emp, certain employees eligible for bonus (even if
0)

ROLES ARE DOCUMENTED
WHEN THEY ARE SIGNIFICANT
This has changed the rule about the group of employees for whom the bonus is applicable. Previously, analysts were technically eligible, even if none of them actually received a bonus.
Modelling Concepts (Continued)

Identifiers:
- determine which attributes are part of it
- verify uniqueness
- establish “not null” requirements

Relationships:
- establish degree 1-1 or 1-M
- entity on 1 side must be functionally dependent on entity on M side
- develop:
  - name
  - definition
- incorporate constraints, rules
- note referential integrity
  - (values of foreign key must exist in key field of another relation)
  - (e.g. in the emp relation, if an employee is listed as being in department 402, then in the dept relation there must contain a row with a key value of 402)
Other Modelling Methods

Entity Analysis

- Oneness
- Sameness
- Categorization
- Identification

Object Abstraction (Smith & Smith)

Objective: “Intellectual Manageability”

- Create hierarchies of abstraction along 2 dimensions:
  - aggregation (has / part of)
  - generalization (is / subtype)
Object Extraction Examples

Aggregation (has / part of)

- Department
- Task
- Employee
- Project

- Tname
- Hours
- Ename
- Job
- Project_id
- Pbudget

Generalization (is / subtype)

- Employee
- Programmer
- Clerk
- Analyst
- Supervisor
- Language_skills
- Type_speed
- Bonus
- Other_employee

- Ename
- Rate
## Map LDS to Well-Formed Relations

<table>
<thead>
<tr>
<th>LDS</th>
<th>Relational Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>entity</td>
<td>relation name</td>
</tr>
<tr>
<td>attribute descriptor</td>
<td>attribute</td>
</tr>
<tr>
<td>single-valued relationship descriptor</td>
<td>attribute (foreign key)</td>
</tr>
<tr>
<td>multi-valued relationship descriptor</td>
<td>nothing</td>
</tr>
<tr>
<td>1-1 relationship</td>
<td>either or both relationship descriptors are attributes</td>
</tr>
<tr>
<td>1-M relationship</td>
<td>relationship descriptor with degree 1 (on the M side) is an attribute</td>
</tr>
</tbody>
</table>
Example: College students

college
  (college#, college_name)

student
  F.K.
  (student#, college#, student_name, soc_sec#)
Sample Database Relations

(See page 10 for the Sample Database LDS)

depth
(D deptNo, Dname, Loc, Dbudget)

temp
(En name, Job, Mgr, Hired, Rate, Bonus, DeptNo)

proj
(Project_id, Description, Pbudget, Due_date)

task
(T taskName, Ename, Project_id, Hours
Relations for Example 1 - Suppliers

supp
  (supplier)

part_type
  (part)

availability
  F.K.        F.K.
  (supplier, part)
Relations for Example 2 - Inventory

product
  (part#)

warehouse
  (warehouse#, wh_address)

inventory
  F.K.   F.K.
  (part#, warehouse#, quantity)
Relations for Example 3 - Departments

department
  (dept, dept_loc)

employee
  F.K.
  (name, dept)
Relations for Example 4 - Locations

department

(dept#, dept_name, dept_loc)
Relations for Example 5 - Stock

part
  (p#)

supplier
  (s#, sname)

inventory
  F.K.  F.K.
  (p#, s#, qty)
Relations for Example 6 - Enrollment

student
(student)

subj
(subject)

teach
F.K.
(teacher, subject)

registration
F.K. F.K.
(student, teacher)
Relations for Example 7 - Skills

employee

emp

job_skill

emp/job_skill

skill

lang

language

emp-lang

F.K.         F.K.
(employee, skill)

emp-lang

F.K.         F.K.
(employee, language)
Relations for Example 8 - Dealerships

(See page 21 for Dealership LDS with symmetry restrictions)

dealer
  (agent)

manufacturer
  (company)

vehicle
  (product)

dealer-mfgr
  F.K.  F.K.
  (agent, company)

dealer-vehicle
  F.K.  F.K.
  (agent, product)

mfgr-vehicle
  F.K.  F.K.
  (company, product)