IT420: Database Management and Organization

Entity-Relationship Model to Relational Model
(Chapter 6)

Database Design Process

- Requirements analysis
- Conceptual design: Entity-Relationship Model
- Logical design: transform ER model into relational schema
- Schema refinement: Normalization
- Physical tuning

Database Design Process

- Requirements analysis
- Conceptual design: Entity-Relationship Model
- Logical design: transform ER model into relational schema
- Schema refinement: Normalization
- Physical tuning

Goals

- Transform ER model to relational model
- Write SQL statements to create tables

Goals

- Transform ER model to relational model
- Write SQL statements to create tables

ER Model

- Entities, attributes, identifiers
- Weak entities
  - ID-dependent entities - identifying relationships
- HAS-A Relationships
  - Degree: binary, ternary
  - Maximum cardinality: 1:1, 1:N, N:M
  - Minimum cardinality: O-O, O-M, M-O, M-M
- IS-A Relationships (Super-type / sub-type)
  - Inclusive, Exclusive

ER Model

- Entities, attributes, identifiers
- Weak entities
  - ID-dependent entities - identifying relationships
- HAS-A Relationships
  - Degree: binary, ternary
  - Maximum cardinality: 1:1, 1:N, N:M
  - Minimum cardinality: O-O, O-M, M-O, M-M
- IS-A Relationships (Super-type / sub-type)
  - Inclusive, Exclusive

VERY IMPORTANT to design a good model!

Relational Model

- Tables
- Integrity constraints
  - Primary key
  - Candidate key
  - Foreign key

Relational Model

- Tables
- Integrity constraints
  - Primary key
  - Candidate key
  - Foreign key

SQL - The Language of Databases

- Developed by IBM in the 1970s
- Create and process database data
- SQL programming is a critical skill !!!

SQL - The Language of Databases

- Developed by IBM in the 1970s
- Create and process database data
- SQL programming is a critical skill !!!
ER to Relational

- Transform entities in tables
- Transform relationships using foreign keys
- Specify logic for enforcing minimum cardinalities

Create a Table for Each Entity

- `CREATE TABLE` statement is used for creating relations/tables
- Each column is described with three parts:
  - column name
  - data type
  - optional constraints

Specify Data Types

- Choose the most specific data type possible!!!
- Generic Data Types:
  - `CHAR(n)`
  - `VARCHAR(n)`
  - `DATE`
  - `TIME`
  - `MONEY`
  - `INTEGER`
  - `DECIMAL`

Specify Null Status

- **Null status:** whether or not the value of the column can be NULL

```
CREATE TABLE EMPLOYEE (  
  EmployeeNumber integer NOT NULL,  
  EmployeeName char(50) NOT NULL,  
  Phone char(15) NULL,  
  Email char(50) NULL,  
  HireDate date NOT NULL DEFAULT (getdate()),  
  ReviewDate date NULL  
)
```

Specify Default Values

- **Default value** - value supplied by the DBMS, if no value is specified when a row is inserted

```
CREATE TABLE EMPLOYEE (  
  EmployeeNumber integer NOT NULL,  
  EmployeeName char(50) NOT NULL,  
  Phone char(15) NULL,  
  Email char(50) NULL,  
  HireDate date NOT NULL DEFAULT (getdate()),  
  ReviewDate date NULL  
)
```

Specify Other Data Constraints

- **Data constraints** are limitations on data values

```
CREATE TABLE EMPLOYEE (  
  EmployeeNumber integer NOT NULL,  
  EmployeeName char(50) NOT NULL,  
  Phone char(15) NULL,  
  Email char(50) NULL,  
  HireDate date NOT NULL DEFAULT (getdate()),  
  ReviewDate date NULL,  
  CONSTRAINT Check_Email CHECK (Email LIKE '%@gmail.com')  
)
```
Specify Primary Key
- Entity identifier → primary key (usually)

CREATE TABLE EMPLOYEE (  
  EmployeeNumber integer NOT NULL,  
  EmployeeName char (50) NOT NULL,  
  Phone char (15) NULL,  
  Email char(50) NULL,  
  HireDate date NOT NULL DEFAULT (getdate()),  
  ReviewDate date NULL,  
  CONSTRAINT Check_Email CHECK (Email LIKE '%@gmail.com'),  
  CONSTRAINT PK_Employee PRIMARY KEY (EmployeeNumber)
)

Specify Alternate Keys
- Alternate keys: alternate identifiers of unique rows in a table

CREATE TABLE EMPLOYEE (  
  EmployeeNumber integer NOT NULL,  
  EmployeeName char (50) NOT NULL,  
  Phone char (15) NULL,  
  Email char(50) NULL,  
  HireDate date NOT NULL DEFAULT (getdate()),  
  ReviewDate date NULL,  
  CONSTRAINT Check_Email CHECK (Email LIKE '%@gmail.com'),  
  CONSTRAINT PK_Employee PRIMARY KEY (EmployeeNumber),  
  CONSTRAINT AK_Email UNIQUE (Email),  
  CONSTRAINT AK_ENamePhone UNIQUE (EmployeeName, Phone)
)

ER to Relational
- Transform entities in tables
- Transform relationships using foreign keys
- Specify logic for enforcing minimum cardinalities

Create Relationships: 1:1 Strong Entity Relationships
- Place the key of one entity in the other entity as a foreign key:
  - Either design will work – no parent, no child
  - Minimum cardinality considerations may be important:
    - O-M will require a different design that M-O

Create Relationships:
CREATE TABLE CLUB_MEMBER(  
  MemberNumber integer PRIMARY KEY,  
  MemberName char(50),  
  Phone char(15),  
  Email char(50)
)
CREATE TABLE LOCKER(  
  LockerNumber integer PRIMARY KEY,  
  LockerRoom integer,  
  LockerSize integer,  
  MemberNumber integer NULL,  
  CONSTRAINT FK_Member FOREIGN KEY (MemberNumber) REFERENCES CLUB_MEMBER(MemberNumber),  
  CONSTRAINT Unique_Member UNIQUE(MemberNumber)
)

Create Relationships:
CREATE TABLE LOCKER(  
  LockerNumber integer PRIMARY KEY,  
  LockerRoom integer PRIMARY KEY,  
  LockerSize integer)

CREATE TABLE CLUB_MEMBER(  
  MemberNumber integer PRIMARY KEY,  
  MemberName char(50),  
  Phone char(15),  
  Email char(50),  
  LockerNumber integer NULL,  
  CONSTRAINT FK_Locker FOREIGN KEY (LockerNumber) REFERENCES LOCKER(LockerNumber),  
  CONSTRAINT Unique_Locker UNIQUE(LockerNumber)
)
Create Relationships:
1:N Relationships

- “Place the key of the parent in the child”

Create Relationships:
1:N Strong Entity Relationships

- CREATE TABLE COMPANY
  - CompanyName char(50) PRIMARY KEY
  - City char(50)
  - Country char(50)
  - Volume decimal

- CREATE TABLE DEPARTMENT
  - DepartmentName char(50) PRIMARY KEY
  - BudgetCode char(5)
  - MailStop integer
  - CompanyName char(50) NOT NULL

- CONSTRAINT FK_Company
  - FOREIGN KEY (CompanyName) REFERENCES COMPANY (CompanyName)

Create Relationships:
1:N Identifying Relationship

- CREATE TABLE BUILDING
  - BuildingName char(50) PRIMARY KEY
  - Street varchar(50)
  - City char(50)
  - State char(30)
  - Zip integer

- CREATE TABLE APARTMENT
  - ApartmentNumber integer NOT NULL
  - BuildingName char(50) NOT NULL
  - NumberBedrooms integer
  - NumberBaths integer
  - MonthlyRent decimal

- CONSTRAINT FK_Building
  - FOREIGN KEY (BuildingName) REFERENCES BUILDING (BuildingName)

Create Relationships:
N:M Strong Entity Relationships

- In an N:M relationship there is no place for the foreign key in either table:
  - A COMPANY may supply many PARTs
  - A PART may be supplied by many COMPANYs

- CREATE TABLE COMPANY
  - CompanyName char(50) PRIMARY KEY
  - City char(50)
  - Country char(50)
  - Volume decimal

- PART
  - PartNumber integer PRIMARY KEY
  - PartName char(50)
  - SalesPrice decimal
  - ReOrderQuantity integer
  - QuantityOnHand integer

- CONSTRAINT FK_Company
  - FOREIGN KEY (CompanyName) REFERENCES COMPANY (CompanyName)

- CONSTRAINT FK_Part
  - FOREIGN KEY (PartNumber) REFERENCES PART (PartNumber)

Create Relationships:
N:M Strong Entity Relationships

- Create an intersection table:
  - The primary keys of each table → composite primary key for intersection table
  - Each table’s primary key becomes a foreign key linking back to that table

- CREATE TABLE COMPANY_PART
  - CompanyName char(50) PRIMARY KEY
  - PartNumber integer PRIMARY KEY

- CONSTRAINT FK_Company
  - FOREIGN KEY (CompanyName) REFERENCES COMPANY (CompanyName)

- CONSTRAINT FK_Part
  - FOREIGN KEY (PartNumber) REFERENCES PART (PartNumber)
Subtype Relationships

CREATE TABLE EMPLOYEE(
    EmployeeNumber integer PRIMARY KEY,
    ...)

CREATE TABLE MANAGER(
    EmployeeNumber integer PRIMARY KEY,
    MgrTrainingDate date,
    ManagerLevel integer,
    CONSTRAINT FK_Emp FOREIGN KEY (EmployeeNumber) REFERENCES EMPLOYEE (EmployeeNumber)
)

CREATE TABLE DB_ADMIN(
    EmployeeNumber integer PRIMARY KEY,
    DB_Name char(50),
    DBMS char(50),
    CONSTRAINT FK_Emp FOREIGN KEY (EmployeeNumber) REFERENCES EMPLOYEE (EmployeeNumber)
)

ER to Relational

- Transform entities in tables
- Transform relationships using foreign keys
- Specify logic for enforcing minimum cardinalities

FOREIGN KEY Constraints

CREATE TABLE Departments
    (DepartmentName char(18),
    Phone char(18) NOT NULL,
    Building char(18),
    Room integer,
    PRIMARY KEY (DepartmentName)
)

CREATE TABLE Students
    (StudentNumber integer,
    StudentLastName char(18) NOT NULL,
    StudentFirstName char(18) NOT NULL,
    Email char(50),
    PhoneNumber char(18),
    MajorDepartmentName char(18),
    PRIMARY KEY (StudentNumber),
    UNIQUE(Email),
    CONSTRAINT FK_Dept FOREIGN KEY (MajorDepartmentName)
        REFERENCES Departments (DepartmentName)
        ON DELETE SET NULL
        ON UPDATE CASCADE
)

Enforcing Mandatory Parent

DEPARTMENT (DepartmentName, BudgetCode, ManagerName)

CREATE TABLE EMPLOYEE (EmployeeNumber integer PRIMARY KEY, EmployeeName char(50),
DepartmentName char(50) NOT NULL, CONSTRAINT FK_Dep FOREIGN KEY(DepartmentName) REFERENCES DEPARTMENT(DepartmentName)
ON DELETE NO ACTION
ON UPDATE CASCADE)

Enforcing Mandatory Child

- More difficult to enforce (write code – “triggers”)
- Tricky:
  - A department must have some employee
  - EMPLOYEE has DepartmentName as FK, NOT NULL
ER to Relational - Summary

- Transform entities in tables
  - Specify primary and alternate keys
  - Specify column types, null status, default values, constraints
- Transform relationships using foreign keys
  - Place the key of the parent in the child
  - Create intersection tables
- Specify logic for enforcing minimum cardinalities
  - Actions for insert, delete, update

Class Exercise: University ER Data Model

Class Exercise

Class Exercise

Class Exercise
Class Exercise

Representing Ternary and Higher-Order Relationships
- Transform them in binary relationship
- Apply the known procedure to transform in relational model

ER to Relational - Summary
- Transform entities in tables
  - Specify primary and alternate keys
  - Specify column types, null status, default values, constraints
- Transform relationships using foreign keys
  - Place the key of the parent in the child
  - Create intersection tables
- Specify logic for enforcing minimum cardinalities
  - Actions for insert, delete, update

SQL: Creating Tables
```
CREATE TABLE table_name(  
column_name1 column_type1 [constraints1],  
...  
[CONSTRAINT constraint_name] table_constraint  
)
```