

IT360: Applied Database Systems

Slide Set: #3

Relational Model
(Kroenke: Chapter 3, pg 71-81)
ER To Relational
(Kroenke: Chapter 6)

1

Database Design Process

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

2

Goals

- Understand:
 - The relational model
 - Relational model terminology
- Transform ER model to relational model
- Write SQL statements to create tables

3

Why Study the Relational Model?

- Most widely used model.
 - Vendors: IBM, Microsoft, Oracle, Sybase, etc.
- Recent competitors:
 - Object-Oriented model
 - ObjectStore, Versant, Ontos
 - A synthesis: *object-relational model*
 - Informix Universal Server, Oracle, DB2
 - XML

4

SQL - The Language of Databases

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

5

Facebook and Databases

- Relational databases are accessed in much the same way across the board: SQL. **Learning how SQL works is crucial** to getting anything done in databases, and any GUI is largely a wrapper around the SQL statements one uses to make those actions happen.
- Knowing a little about **database design** (layout, B-trees, file storage, normalization) is good, mostly for helping you understand good queries.
- We run the LAMP stack here, so we primarily use MySQL databases across the site.
- I hope this helps a little. Another good motivation may be found in the requirements for most engineering positions here on <http://www.facebook.com/jobs.php#Opportunities> ;)

Thanks!

Nick from Facebook

6

Relational Database

- A **relation** is a two-dimensional table
- **Relation schema** describes the structure for the table
 - Relation name
 - Column names
 - Column types
- A **relational database** is a set of relations

7

Relation Example

```
EMPLOYEE(EmployeeNumber:integer,  
          FirstName:string,  
          LastName:string,  
          Department:string,  
          Email:string,  
          Phone:integer)
```

EmployeeNumber	FirstName	LastName	Department	Email	Phone
100	Jerry	Johnson	Accounting	JJ@somewhere.com	236-9987
200	Mary	Abernathy	Finance	MA@somewhere.com	444-8898
300	Liz	Smathers	Finance	LS@somewhere.com	777-0098
400	Tom	Caruthers	Accounting	TC@somewhere.com	236-9987
500	Tom	Jackson	Production	TJ@somewhere.com	444-9980
600	Eleanore	Caldera	Legal	EC@somewhere.com	767-0900
700	Richard	Bandalone	Legal	RB@somewhere.com	767-0900

8

Relation

- All entries in a column are of the same kind
 - Each column has a unique name
 - Cells of the table hold a single value
 - The order of the columns is not important
 - The order of the rows is not important
 - No two rows may be identical
-
- Rows contain data about entity instances
 - Columns contain data about attributes of the entity

9

Tables That Are Not Relations

EmployeeNumber	FirstName	LastName	Department	Email	Phone
100	Jerry	Johnson	Accounting	JJ@somewhere.com	236-0000
200	Mary	Abernathy	Finance	MA@somewhere.com	444-8998
300	Liz	Smathers	Finance	LS@somewhere.com	777-0098
400	Tom	Caruthers	Accounting	TC@somewhere.com	236-0000, 236-0991, 236-0991
500	Tom	Jackson	Production	TJ@somewhere.com	444-9980
600	Eleanore	Caldera	Legal	EC@somewhere.com	767-0900
700	Richard	Bandalone	Legal	RB@somewhere.com	767-0900, 767-0911

EmployeeNumber	FirstName	LastName	Department	Email	Phone
100	Jerry	Johnson	Accounting	JJ@somewhere.com	236-9967
200	Mary	Abernathy	Finance	MA@somewhere.com	444-8998
300	Liz	Smathers	Finance	LS@somewhere.com	777-0098
400	Tom	Caruthers	Accounting	TC@somewhere.com	236-9987, Fax: 236-9987, Home: 556-7171
500	Tom	Jackson	Production	TJ@somewhere.com	444-9980
600	Eleanore	Caldera	Legal	EC@somewhere.com	767-0900, Fax: 236-9987, Home: 556-7171
700	Richard	Bandalone	Legal	RB@somewhere.com	767-0900

10

Alternative Terminology

- Although not all tables are relations, the terms table and relation are normally used interchangeably
- The following sets of terms are equivalent:

Table	Column	Row
Relation	Attribute	Tuple
File	Field	Record

11

ER to Relational

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

12

Create a Table for Each Entity

EMPLOYEE
EmployeeNumber
EmployeeName
Phone
Email
HireDate
ReviewDate

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

13

Specify Data Types

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

EMPLOYEE
EmployeeNumber
EmployeeName
Phone
Email
HireDate
ReviewDate

```
CREATE TABLE EMPLOYEE (  
EmployeeNumber integer,  
EmployeeName char(50),  
Phone char(15),  
Email char(50),  
HireDate date,  
ReviewDate date  
)
```

14

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,  
ReviewDate date NULL  
)
```

15

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  
)
```

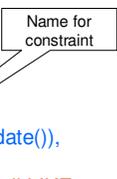
Syntax/support depends on DBMS

16

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')  
)
```



17

Integrity Constraints (IC)

- IC: condition that must be true for any instance of the database
 - Domain constraints
 - Key constraints
 - Foreign Key constraints
- ICs are **specified** when schema is **defined**
- ICs are **checked** when relations are **modified**
- A **legal instance** of a relation is one that satisfies all specified ICs
 - DBMS should not allow illegal instances

18

Keys

- A **key** is a combination of one or more columns that is used to identify rows in a relation
- A **composite key** is a key that consists of two or more columns
- A set of columns is a **key** for a relation if :
 1. No two distinct rows can have same values in all key columns, and
 2. This is not true for any subset of the key
- Part 2 false? A **superkey**

19

Candidate and Primary Keys

- A **candidate key** is a key
- A **primary key** is a candidate key selected as the primary means of identifying rows in a relation:
 - There is one and only one primary key per relation
 - The primary key may be a composite key
 - **The ideal primary key is short, numeric and never changes**

20

Surrogate Keys

- A **surrogate key** is an artificial column added to a relation to serve as a primary key:
 - DBMS supplied
 - Short, numeric and never changes – an ideal primary key!
 - Has artificial values that are meaningless to users
- Remember Access (ID – auto number)

21

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)
)
```

22

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)
)
```

23

ICE: Is This a Relation? Why?

A	X	C	A
John	Ryan	MD	jr@gmail.com
Bob	Smith	MD, VA, NY	bsm@gmail.com
Alice	Brown	CA	
Jane	Doe	WA	jd@yahoo.com
John	Ryan	MD	jr@gmail.com
5	4	5	4

24

ICE: Find possible PK, AK

X	Y	Z	W
John	Ryan	MD	jr@gmail.com
Bob	Smith	MD	bsm@gmail.com
Alice	Brown	CA	
John	Doe	WA	jd@yahoo.com

25

Foreign Keys and Referential Integrity Constraints

- A **foreign key** is the primary key of one relation that is placed in another relation to form a link between the relations
- A **referential integrity constraint**: the values of the foreign key must exist as primary key values in the corresponding relation → No 'dangling references'

26

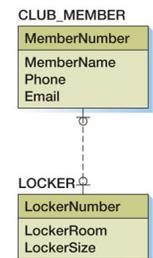
ER to Relational

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

27

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 than M-O



28

Create Relationships: 1:1 Strong Entity 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))
```

29

Create Relationships: 1:1 Strong Entity Relationships

```
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 TABLE LOCKER(  
  LockerNumber integer PRIMARY KEY,  
  LockerRoom integer,  
  LockerSize integer)
```

30

Enforcing Referential Integrity

- What if a new “Member” row is added that references a non-existent locker?
 - Reject it!
- What if a Locker row is deleted?
 - Also delete all Member rows that refer to it.
 - Disallow deletion of Locker row that is referred.
 - Set *LockerNumber* in **Member** to default value
 - Set *LockerNumber* in Member to *null*
- Similar if primary key of Locker row is updated

31

Referential Integrity in SQL/92

- SQL/92 supports all 4 options on deletes and updates.
 - Default is NO ACTION (*delete/update is rejected*)
 - CASCADE (*delete/update all rows that refer to deleted/updated row*)
 - SET NULL / SET DEFAULT

```
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) ON DELETE SET NULL ON UPDATE  
  CASCADE,  
  CONSTRAINT Unique_Locker UNIQUE(LockerNumber))
```

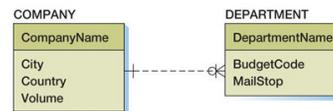
32

Create Relationships: 1:N Relationships

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

33

Create Relationships: 1:N Strong Entity Relationships



(a) 1:N Strong Entity Relationship

```
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) ON DELETE NO ACTION)
```

34

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)
```

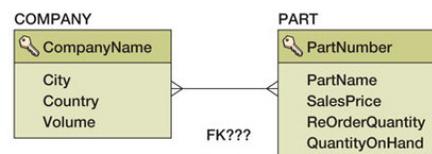
```
CREATE TABLE APARTMENT(  
  ApartmentNumber integer NOT NULL,  
  BuildingName char(50) NOT NULL,  
  NumberBedrooms integer,  
  NumberBaths integer,  
  MonthlyRent decimal,  
  CONSTRAINT PK_Apartment PRIMARY KEY (BuildingName,  
  ApartmentNumber),  
  CONSTRAINT FK_Building FOREIGN KEY (BuildingName)  
  REFERENCES BUILDING (BuildingName) ON DELETE  
  CASCADE ON UPDATE CASCADE)
```

35

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



36

Create Relationships: N:M Strong Entity Relationships

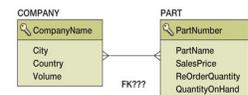
- Create an **intersection table**:
 - The primary keys of each table → **composite primary key** for intersection table
- Each table's primary key becomes a foreign key linking back to that table

37

Create Relationships: N:M Strong Entity Relationships

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

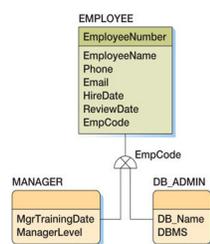
PART(
  PartNumber integer PRIMARY KEY,
  PartName char(50),
  SalesPrice decimal,
  ReOrderQuantity integer,
  QuantityOnHand integer)
```



```
COMPANY_PART(
  CompanyName char(50) NOT NULL,
  PartNumber integer NOT NULL,
  CONSTRAINT PK_CompPart PRIMARY KEY (CompanyName, PartNumber),
  CONSTRAINT FK_Company FOREIGN KEY (CompanyName) REFERENCES
  COMPANY (CompanyName) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT FK_Part FOREIGN KEY (PartNumber) REFERENCES PART
  (PartNumber) ON DELETE NO ACTION ON CASCADE UPDATE)
```

38

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) ON DELETE
  CASCADE

)

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

)
```

39

ER to Relational

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

40

FOREIGN KEY Constraints

DEPARTMENTS

DepartmentName: char(18)
Phone: char(18)
Building: char(18)
Room: integer

U:SN
U:C

Majors

I:SN
U:SN

STUDENTS

StudentNumber: integer
StudentLastName: char(18)
StudentFirstName: char(18)
Email: varchar(50)
PhoneNumber: char(18)
DepartmentName: char(18) (FK)

DepartmentName	Phone	Building	Room
Mathematics	410-293-4573	Michelson Hall	308
History	410-293-2255	Sampson Hall	120
Computer Science	410-293-6800	Michelson Hall	340

Student Number	Student LastName	Student FirstName	Email	PhoneNumber	Major/DepartmentName
190	Smith	John	jsmith@uconn.edu	410-431-3456	Computer Science
673	Doe	Jane	jdoe@uconn.edu		Mathematics
312	Doe	Bob	bdoe@uconn.edu	443-451-7885	Mathematics

```

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

41

Enforcing Mandatory Parent

DEPARTMENT (DepartmentName, BudgetCode, ManagerName)

```

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

42

Enforcing Mandatory Child

- More difficult to enforce (write code – “triggers”)

DEPARTMENT (DepartmentName, BudgetCode, ManagerName)
 EMPLOYEE (EmployeeNumber, EmployeeName, DepartmentName)

- Tricky:
 - A department must have some employee
 - EMPLOYEE has DepartmentName as FK, NOT NULL

43

Summary – Relational Model

- 2-D tables
- Relational schema: structure of table
- Constraints
 - Domain
 - Key
 - Candidate, Primary, Alternate, Surrogate
 - Foreign key – Referential integrity constraint

44

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, if needed
- Specify logic for enforcing minimum cardinalities
 - Actions for insert, delete, update

45

SQL: Creating Tables

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

Table constraints:

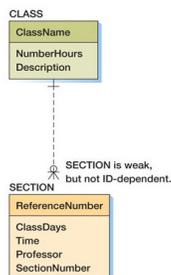
- NULL/NOT NULL
- PRIMARY KEY (*columns*)
- UNIQUE (*columns*)
- CHECK (*conditions*)
- FOREIGN KEY (*local_columns*) REFERENCES *foreign_table* (*foreign_columns*) [ON DELETE *action_d* ON UPDATE *action_u*]

Specify surrogate key in SQL Server:

```
column_name int_type IDENTITY (seed, increment)
```

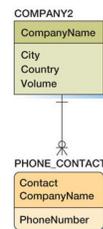
46

Class Exercise



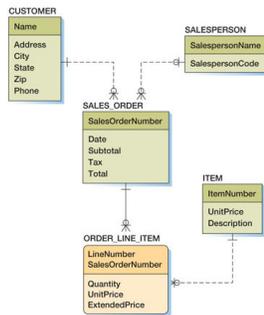
47

Class Exercise



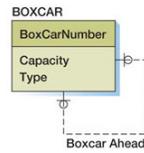
48

Class Exercise



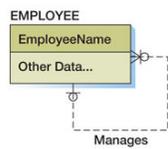
49

Class Exercise



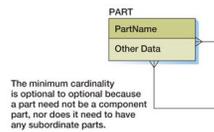
50

Class Exercise



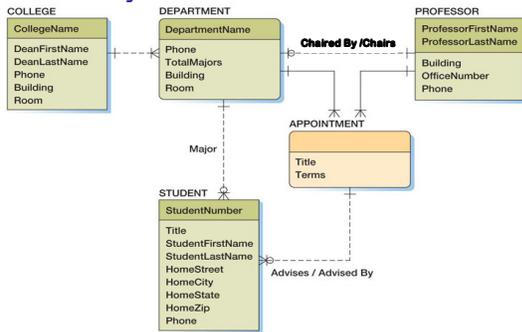
51

Class Exercise



52

Class Exercise: University ER Data Model



53