# IT360: Applied Database Systems

From Entity-Relational Model
To Relational Model

Chapter 6, 7 in Kroenke



## 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

3

## 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

4

# ER to Relational

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

5

# 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

6

# Specify Data Types

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


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

7

# Specify Null Status

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

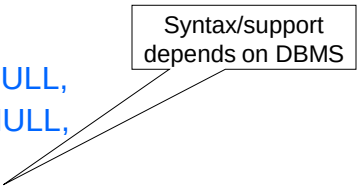
```
CREATE TABLE EMPLOYEE (
    EmployeeNumber integer NOT NULL,
    EmployeeName char (50) NOT NULL,
    Phone char (15) NULL,
    Email char(50) NULL,
    HireDate date NOT NULL,
    ReviewDate date NULL
)
```

8

# 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
)
```

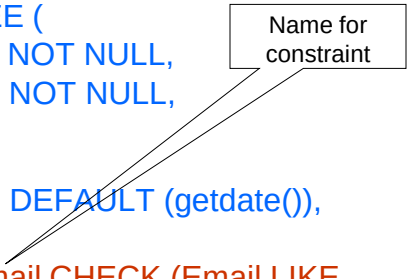



9

# 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')
)
```




10

# 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)
)
```

11

# 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)
)
```

12

## ER to Relational

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

13

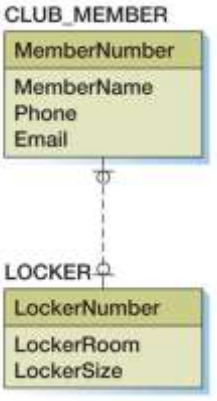
## Foreign Keys and Referential Integrity Constraints

- 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'

14

# Transform Relationships: 1:1 Strong Entity Relationships

1. Place the key of one entity (parent) in the other entity (child) as a foreign key:
   - Either design will work – both could be parent, or child
   - Minimum cardinality considerations:
     - If only one entity is mandatory, that one should be the "parent"
2. Choose NULL status for the FK column(s) based on the min cardinality (mandatory/optional)
3. Declare UNIQUE constraint for the FK column(s)






# Transform 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))
```

# Transform Relationships: 1:1 Strong Entity Relationships

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



# 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

# 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))
```

19

# Transform Relationships: 1:N Relationships

- "Place the key of the parent in the child"

20

# Transform Relationships: 1:N Strong Entity Relationships


(a) 1:N Strong Entity Relationship

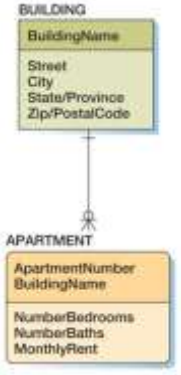
```
CREATE TABLE COMPANY(
CompanyName char(50)
PRIMARY KEY,
City char(50),
Country char(50),
Volume decimal)
```

```
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)
```

21

# Trasnform Relationships: 1:N Identifying Relationship



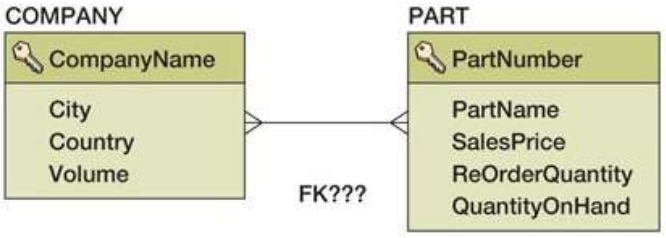

```
CREATE TABLE BUILDING(
  BuildingName char(50) PRIMARY KEY,
  Street varchar(50),
  City char(50),
  State char(30),
  Zip integer)

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)
```

22

## Transform 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



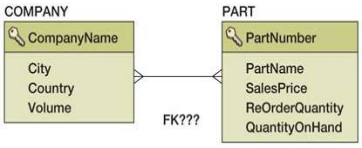

23

## Trasnform 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

24

# Trasnform Relationships:
# N:M Strong Entity Relationships

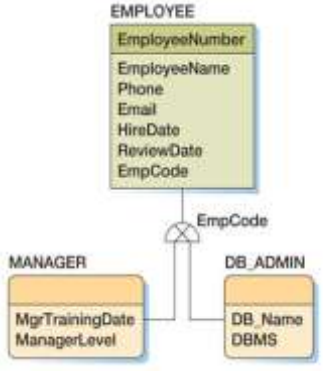



```
CREATE TABLE COMPANY(
    CompanyName char(50) PRIMARY KEY,
    City char(50),
    Country char(50),
    Volume decimal)
CREATE TABLE PART(
    PartNumber integer PRIMARY KEY,
    PartName char(50),
    SalesPrice decimal,
    ReOrderQuantity integer,
    QuantityOnHand integer)

CREATE TABLE 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)
```

25

# IS-A Relationship




```
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
)
```

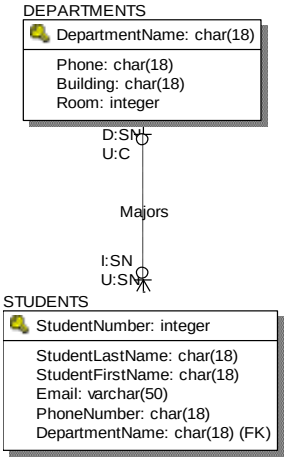
26

# ER to Relational

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

27

# FOREIGN KEY Constraints




| 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 | MajorDepartmentName |
|---|---|---|---|---|---|
| 190 | Smith | John | jsmith@usna.edu | 410-431-3456 | |
| 673 | Doe | Jane | jdoe@usna.edu | | Computer Science |
| 312 | Doe | Bob | bred@usna.edu | 443-451-7865 | Mathematics |

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

28

# 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
)
```

29

# 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

30

# 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



# SQL: Creating Tables

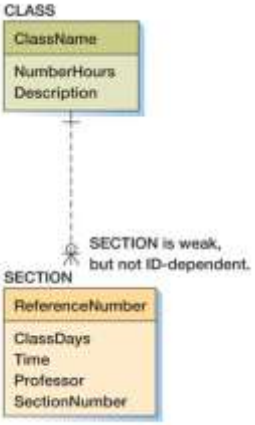
CREATE TABLE *table_name(*
*column_name1 column_type1 [constraints1],*
*…,*
*[[CONSTRAINT constraint_name] table_constraint]*
)

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)
Specify surrogate key in MySQL:
*column_name int_type AUTO_INCREMENT*

# Class Exercise

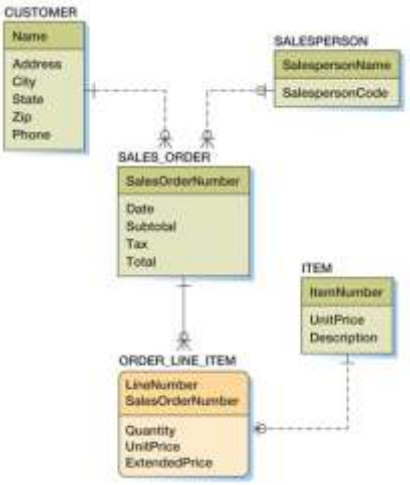



33

# Class Exercise

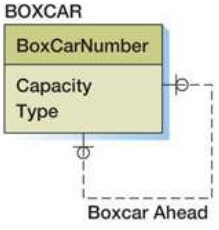

34

# Class Exercise



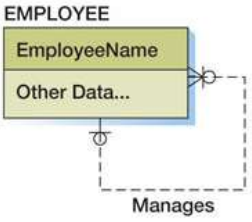

35

# Class Exercise



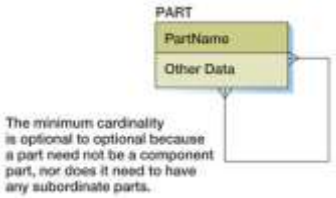

36

# Class Exercise




37

# Class Exercise




38

# Class Exercise:
# University ER Data Model