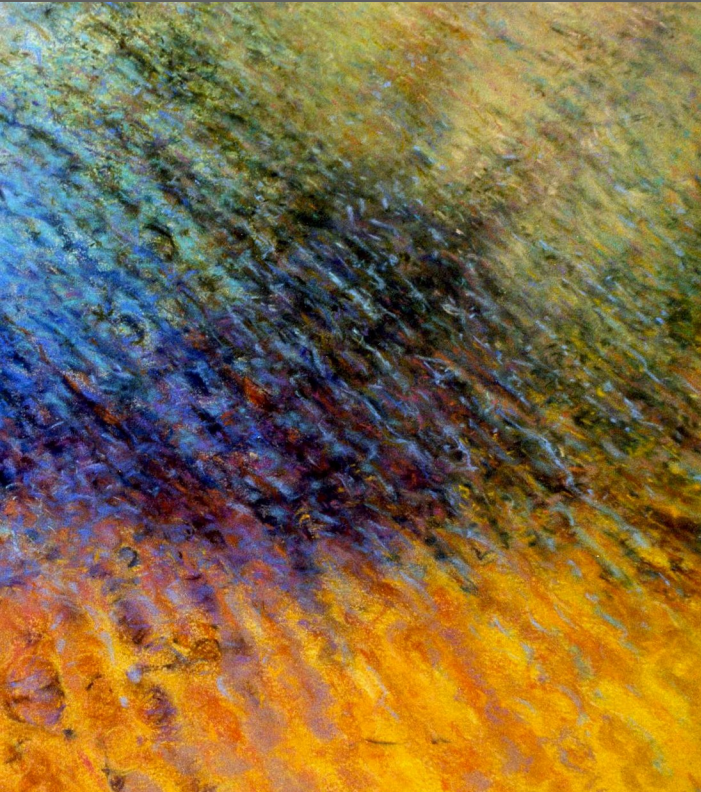


David M. Kroenke and David J. Auer

Database Processing:

Fundamentals, Design, and Implementation



Chapter Six:

Transforming Data Models into Database Designs

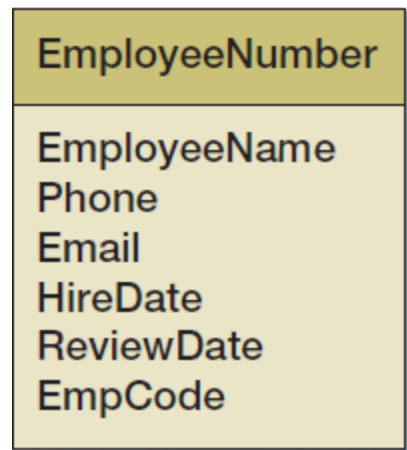
Chapter Objectives

- Understand how to transform conceptual model into logical model
 - AKA: Map conceptual model to logical model
- Understand how to represent 1:1, 1:N, and N:M relationships as tables (relations)
- Understand purpose and use of referential integrity constraints

Create a Table for Each Entity

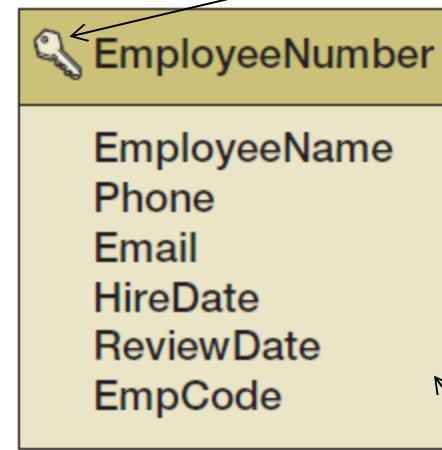
EMPLOYEE (EmployeeNumber, EmployeeName, Phone, Email, HireDate, ReviewDate, EmpCode)

EMPLOYEE



(a) EMPLOYEE Entity

EMPLOYEE



(b) EMPLOYEE Table


Primary key is designated by the key symbol

Note shadow-less table. Do not need to make shadows for course.

Select the Primary Key

- The **ideal primary key** is short, numeric, and fixed.
- **Surrogate keys** meet the ideal, but have no meaning to users.

EMPLOYEE


 EmployeeNumber
EmployeeName
Phone
Email
HireDate
ReviewDate
EmpCode

Specify Candidate (Alternate) Keys


- The terms **candidate key** and **alternate key** are synonymous.
- **Candidate keys** are alternate identifiers of unique rows in a table.
- Will use **AK n . m** notation, where **n** is the number of the alternate key, and **m** is the column number in that alternate key.

Specify Candidate (Alternate) Keys

EMPLOYEE

 EmployeeNumber
EmployeeName Phone Email (AK1.1) HireDate ReviewDate EmpCode

CUSTOMER

 CustomerNumber
Name (AK1.1) City (AK1.2) Phone Email (AK2.1)

Indicates that composite key (name, city)
can be used as a key.

Email is in the first (and only column of
the second alternative key.

Specify Column Properties: Null Status

Null status indicates whether or not the value of the column can be NULL.

What does null mean exactly? (Important concept)

EMPLOYEE



EmployeeNumber: NOT NULL

EmployeeName: NOT NULL

Phone: NULL

Email: NULL (AK1.1)

HireDate: NOT NULL


ReviewDate: NULL

EmpCode: NULL

Specify Column Properties: Data Type

- Generic data types:
 - Char(n), Nchar(n)
 - Varchar(n), Nvarchar(n)
 - Date
 - Time
 - Integer
 - Decimal(m,n)
 - Numeric(m,n)
 - Money(m,n)

EMPLOYEE

 EmployeeNumber: Int
EmployeeName: Varchar(50) Phone: Char(15) Email: Nvarchar(100) (AK1.1) HireDate: Date ReviewDate: Date EmpCode: Char(18)

Specify Column Properties: Data Type + Null Status

EMPLOYEE



EmployeeNumber: int NOT NULL

EmployeeName: char(50) NOT NULL

Phone: char(15) NULL

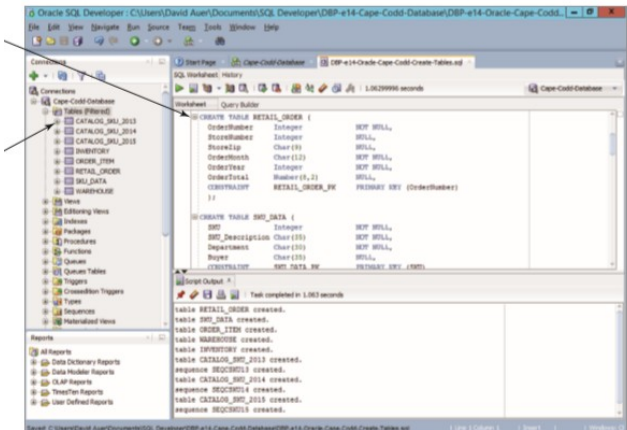
Email: char(50) NULL (AK1.1)

HireDate: datetime NOT NULL

ReviewDate: datetime NULL

EmpCode: char(18) NULL

Check Oracle Syntax: VarChar2(50)
See appendix posted under Oracle.



Specify Column Properties: Data Constraints

- **Data constraints** are limitations on data values:
 - **Domain constraint**—column values must be in a given set of specific values.
 - **Range constraint**—column values must be within a given range of values.
 - **Intrarelation constraint**—column values are limited by comparison to values in other columns in the *same* table.
 - **Interrelation constraint**—column values are limited by comparison to values in other columns in *other* tables (referential integrity constraints on foreign keys).

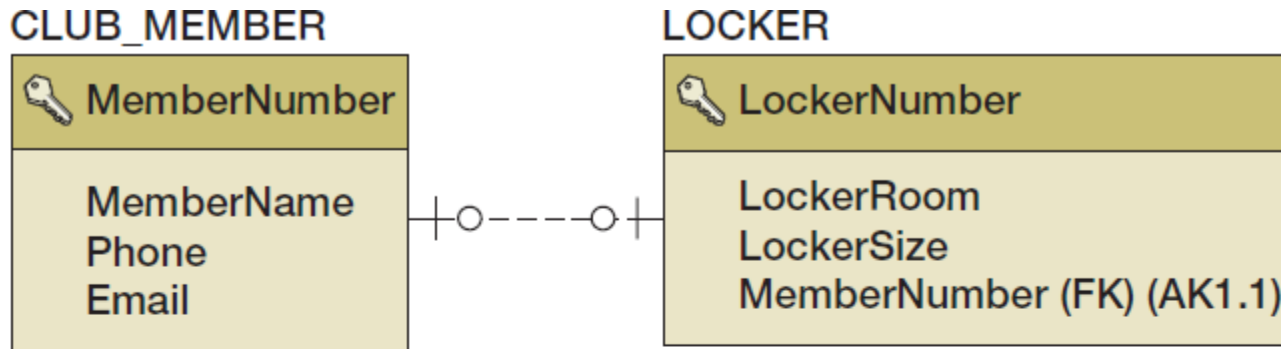
Create Relationships: 1:1 Strong Entity Relationships I

Place the **primary key** of one entity in the other entity as a **foreign key**.

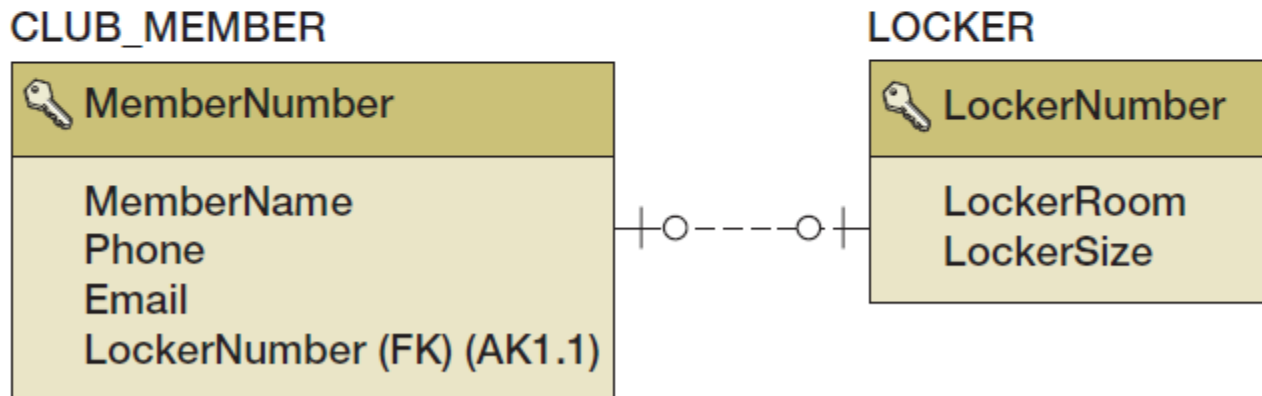
- Either design will work
- Minimum cardinality considerations may be important.

Relationships:

1:1 Strong Entity Relationships II



(a) With Foreign Key in LOCKER



(b) With Foreign Key in CLUB_MEMBER

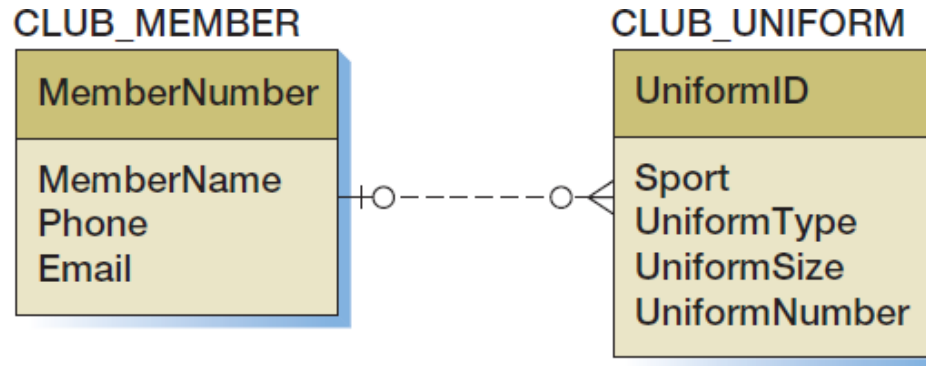
Relationships:

1:N Strong Entity Relationships I

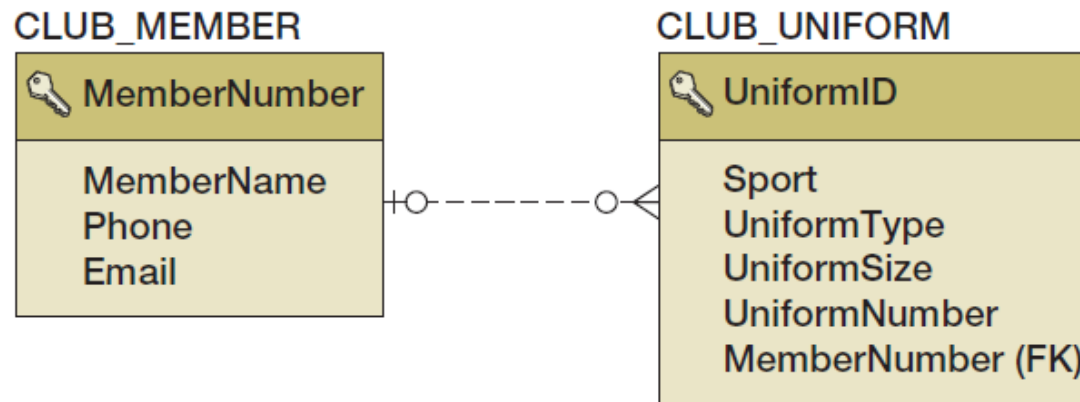
- Place the **primary key** of the table on the *one side* of the relationship into the table on the *many side* of the relationship as the **foreign key**.

Relationships:

1:N Strong Entity Relationships II



(a) 1:N Relationship Between Strong Entities

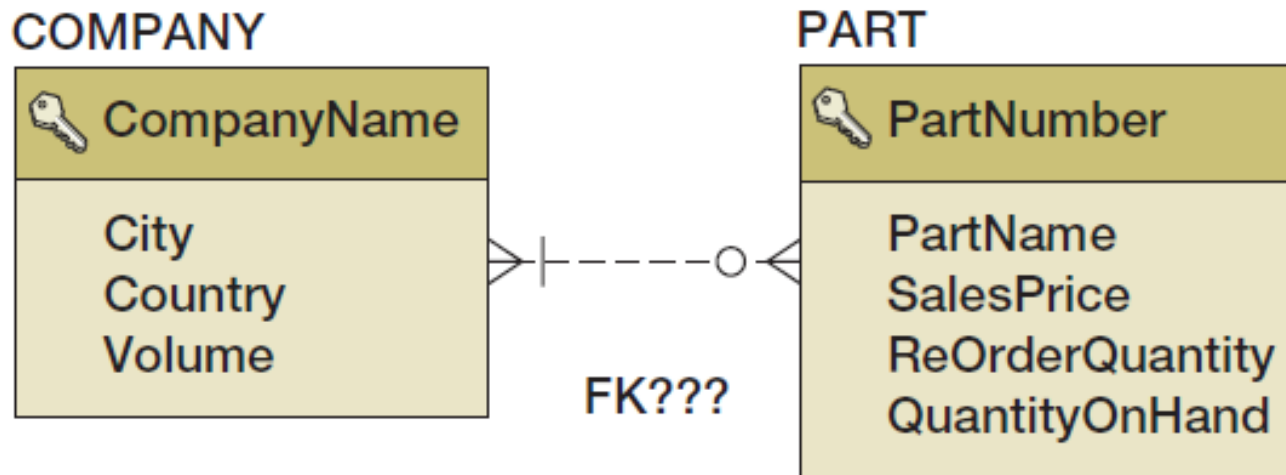


(b) Placing the Primary Key of the Parent in the Child as a Foreign Key

Relationships:

N:M Strong Entity Relationships I

- In an N:M strong entity 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.



Relationships:

N:M Strong Entity Relationships II

- The solution is to create a **new table** that stores data about the corresponding rows from each entity.
- The table consists only of the *primary keys of each table* which form a *composite primary key*.
- Each table's primary key becomes a *foreign key* linking back to that table.

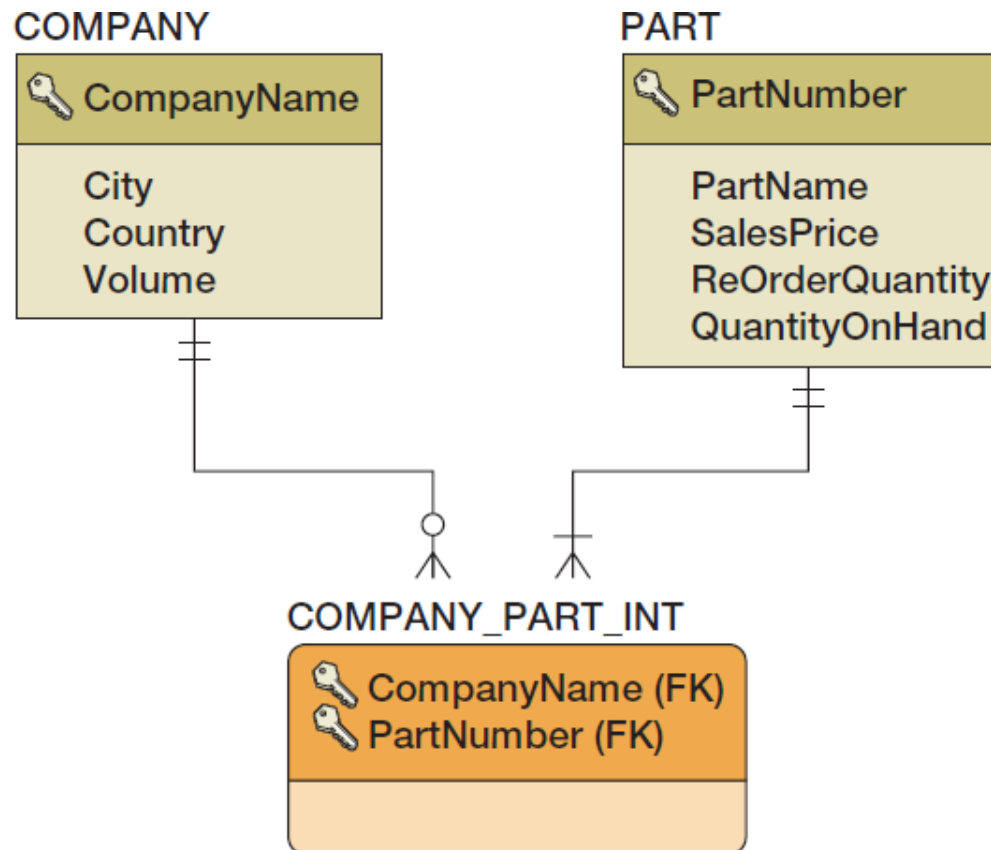
COMPANY_PART_INT (CompanyName, PartNumber)

New key is “concatenation” or joining together of keys of involved entities.

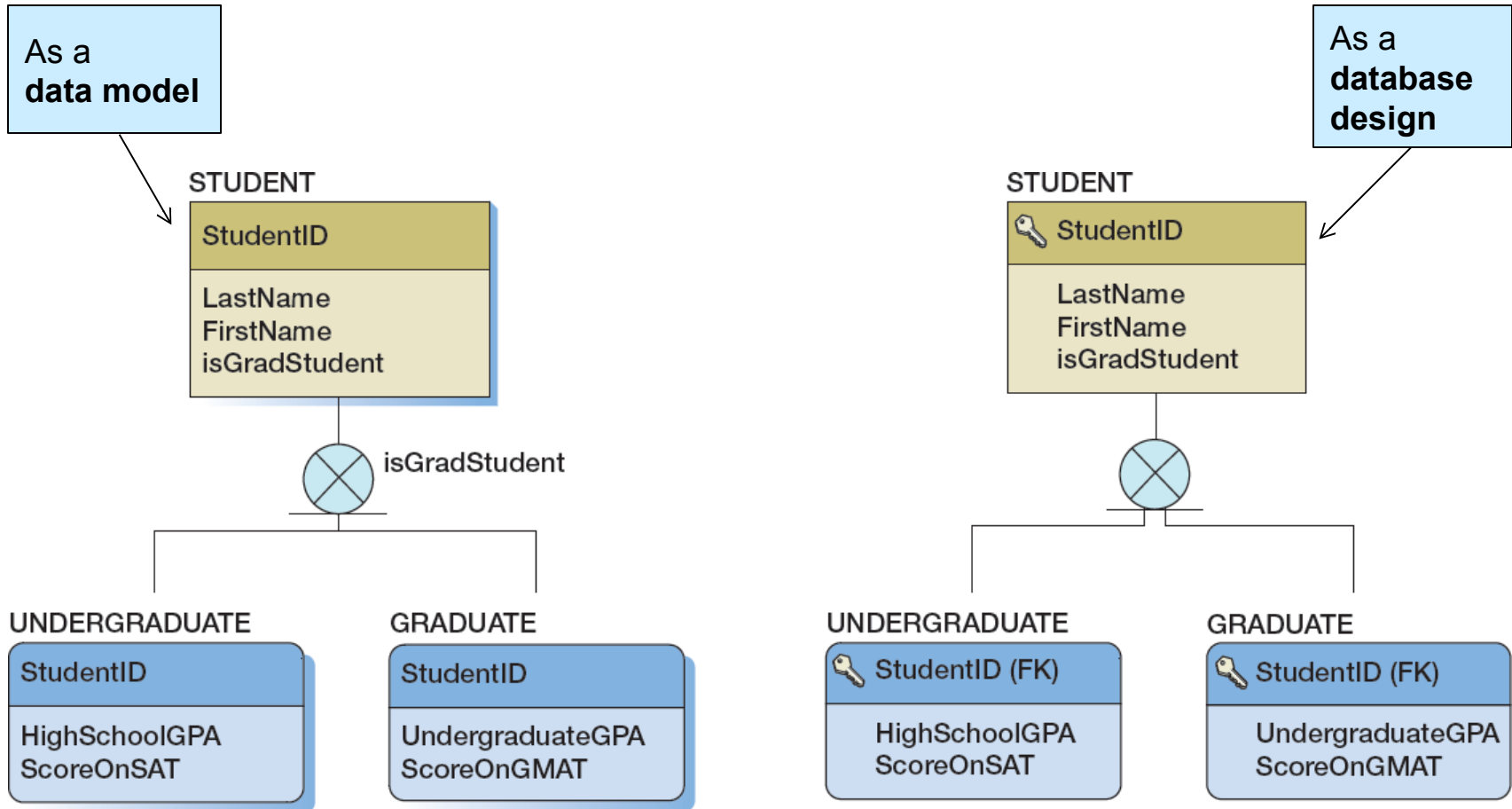
Relationships:

N:M Strong Entity Relationships III

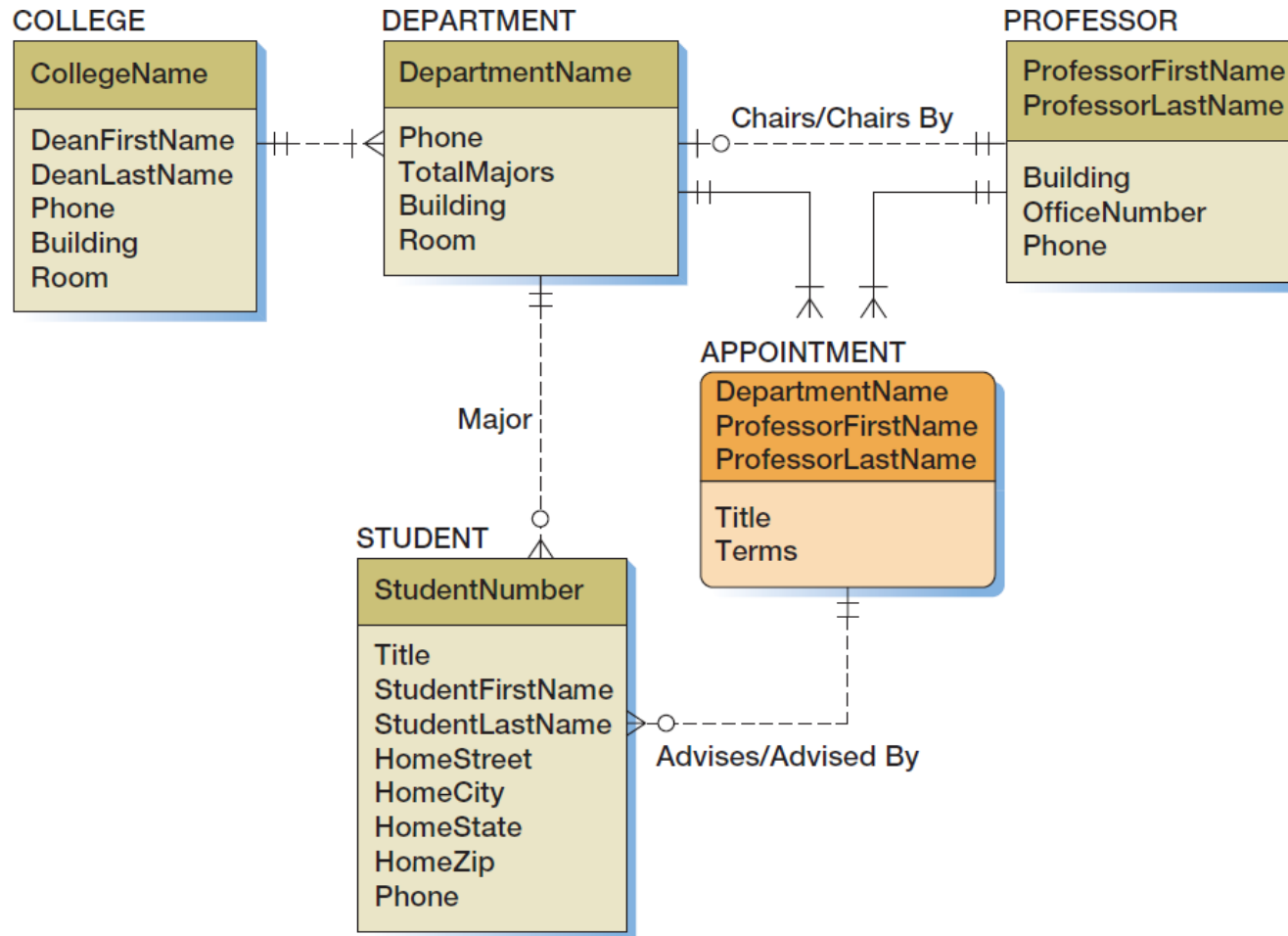
COMPANY_PART_INT (CompanyName, PartNumber)



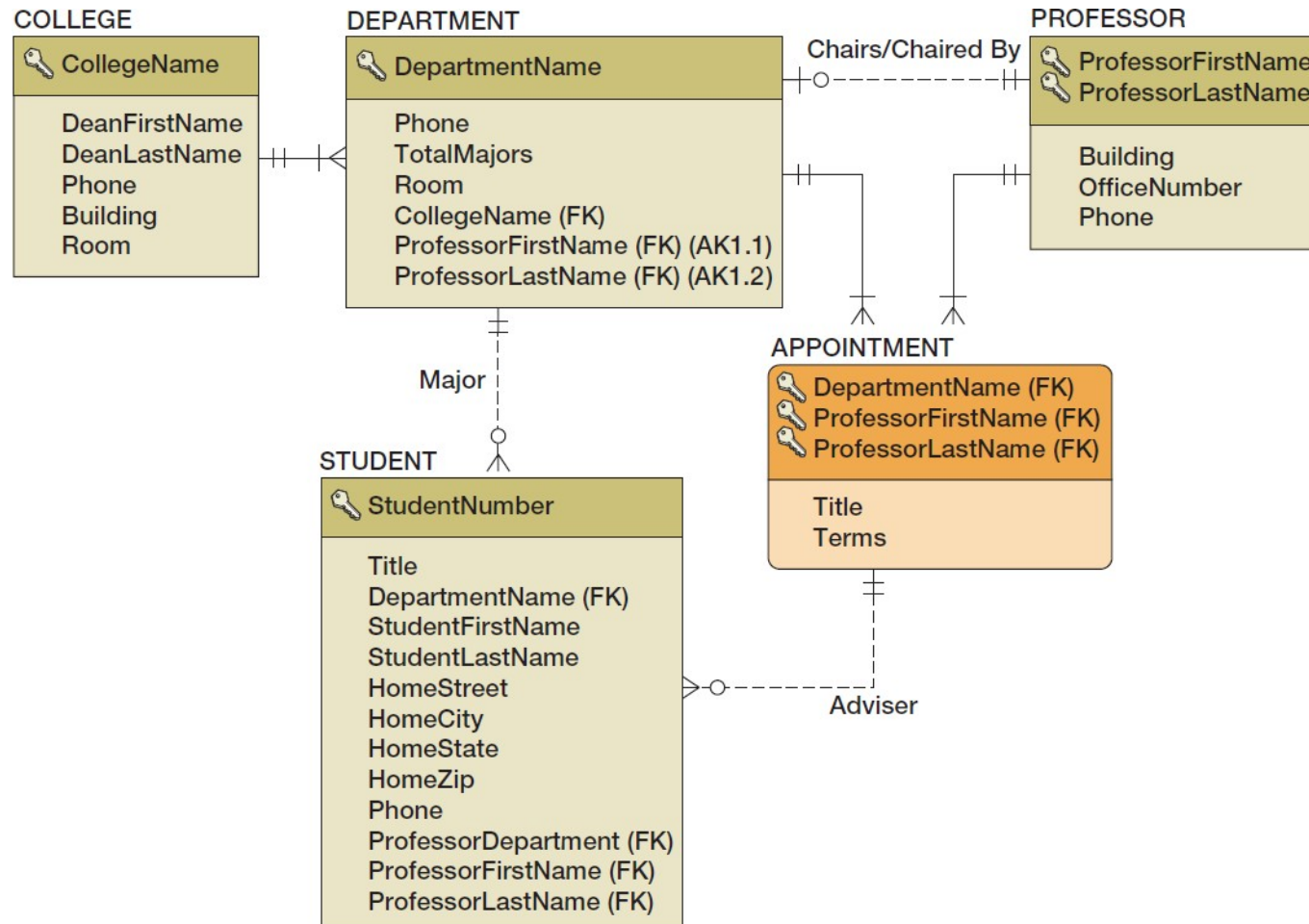
Subtype Relationships



Highline University Data Model



Highline University Database Design



Conclusion

- Relational data model
 - Requirements for design?
- Transformation of conceptual model (e.g., ER with crow's foot notation) to logical model (relational)
 - Set of rules for performing the transformation
 - Specification of data types (for implementation)