

Database Logical Design

CIS 3730

Designing and Managing Data

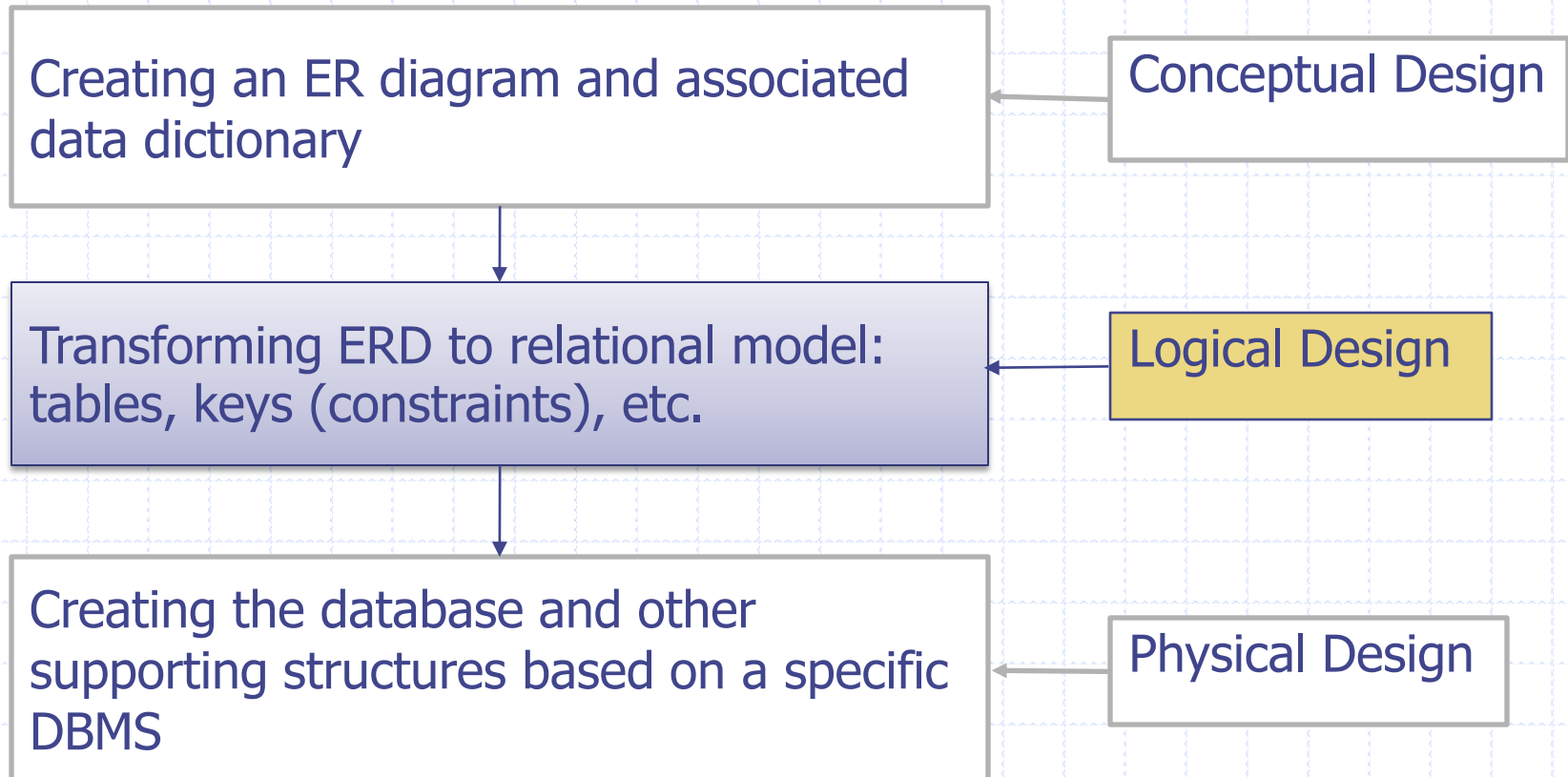
J.G. Zheng
Fall 2010



Overview

- ◆ Relational model is a logical model
 - Based on mathematical theories and rules
- ◆ Two ways to design a relational database model
 - Through normalization
 - Transforming conceptual models (ERD)

3 Level Database Design



Transforming ER to Relations

1. Transforming basic entities, identifiers and attributes
 - Creating tables, columns and keys
 - Defining the data type, length and constraints for each columns

2. Transforming relationships by determining foreign keys
 - Determining foreign keys: which column is foreign key? Which table to add a foreign key? Should a third table be added?
 - Is the value of the foreign key required?
 - Defining referential integrity actions

Database Structure Representation

◆ Text style

[Table Name]([Primary Key], attribute, [*Foreign Key*], attribute, ...)

FK: [Foreign Key] → [Reference Table].[Primary Key]

FK: (if more than one foreign key)

◆ Example

Department (DeptID, DeptName, Location)

Employee (EmpID, EmpName, *Department*)

FK: Department → Department.DeptID

Primary Key(s):
underscored

Foreign Key:
italicized

Foreign Key:
definition

Basic Tables and Columns

- ◆ Each entity becomes a relation (table)
- ◆ Attributes of the entity become fields of that table
- ◆ Identifier becomes primary key

Column Definition

- ◆ Column name
- ◆ Data type: generic data types
 - Character, numeric, date/time, boolean/bit
- ◆ Column length (size)
- ◆ More requirements: constraints
 - Required or not
 - Uniqueness
 - Domain values, range
 - Default values

Data Type

- ◆ Data types
 - Character: string/text, fixed vs. varied length
 - Numeric: integer, float/decimal
 - Date/Time: for date and time related attributes
 - Boolean/bit: for attributes which only have two values

- ◆ Note: use numeric data type only when involving calculations
 - Are these numeric data type attribute?
 - ◆ ZIP code
 - ◆ SSN
 - ◆ Phone number

Derived Attribute

◆ Derived attribute

- An attribute that is generated based on other columns, or rows
- GPA, TotalPayment, Age, etc.

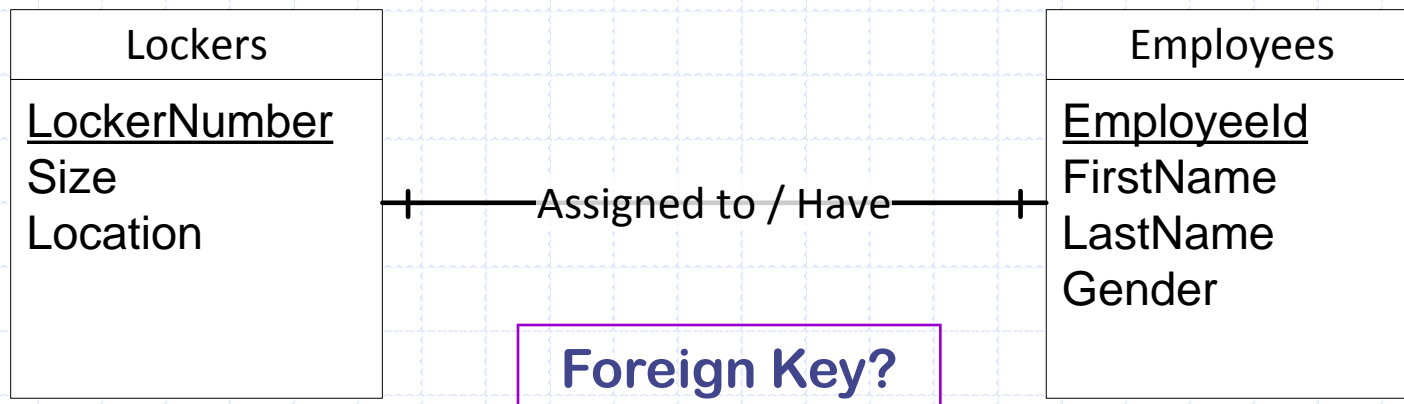
◆ Derived attributes are usually not modeled at this time

Transforming Relationships

- ◆ Relationships are defined by foreign key constraints
 - One foreign key only dereferences one unique key in one other table (one key one table)
- ◆ Designing foreign key constraint
 - The whole purpose is to avoid null values
 - ◆ Which column is the foreign key?
 - ◆ Which table is the referencing table?
 - ◆ When no one existing attribute can be a foreign key, which table should it be added to?

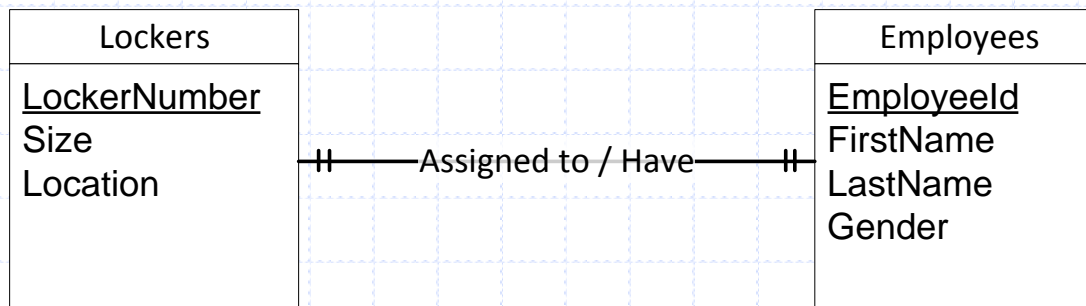
FK in 1:1 Relationships

- ◆ The design of the foreign key depends on minimum cardinality
- ◆ Example:
 - One-to-one relationship between employee and locker



Minimum Cardinality 1:1

- ◆ When both side are mandatory (minimum cardinality 1), the foreign key can be in either table (choose the one that makes more business sense)
- ◆ Example:
 - A locker is required for an employee (an employee must get one)
 - An employee is mandatory for a locker (a locker has to be assigned to someone)



Employees (EmployeeId, FirstName, LastName, Gender)

Lockers (LockerNumber, Size, Location, AssignedTo)

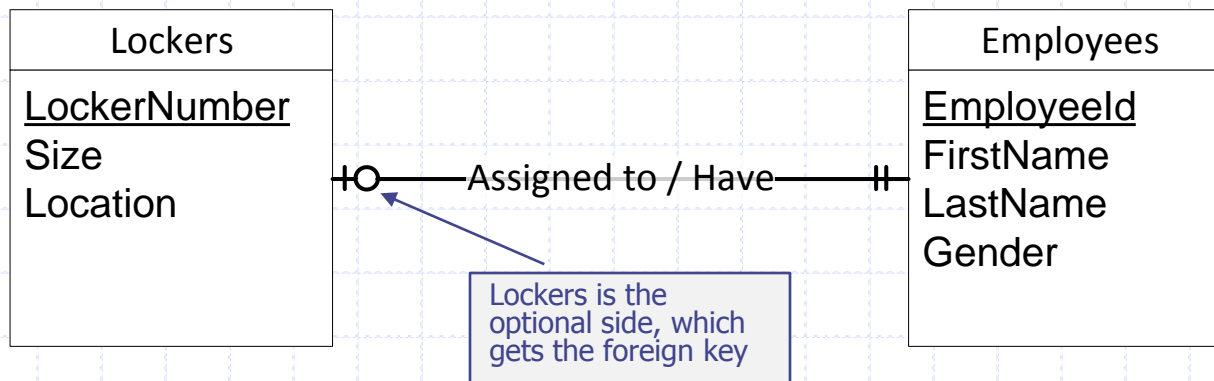
FK: AssignedTo → Employees.EmployeeId

- ◆ Or, consider combining two tables into one table.
 - Choose one identifier as PK, and the other identifier unique.

Employees (EmployeeId, FirstName, LastName, Gender, LockerNumber, Size, Location)

Minimum Cardinality 1:0

- ◆ When only one side is optional, foreign key is placed on the optional side, to avoid null values → kind of like normalization.
- ◆ Example:
 - A locker is optional for an employee (an employee may not get one)
 - An employee is mandatory for a locker (a locker has to be assigned to someone)



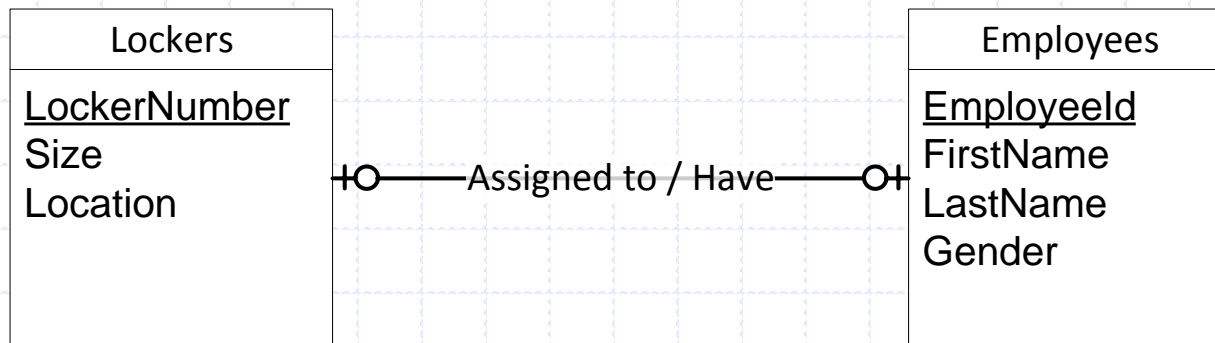
Employees (EmployeeId, FirstName, LastName, Gender)

Lockers (LockerNumber, Size, Location, AssignedTo)

FK: AssignedTo → Employees.EmployeeId

Minimum Cardinality 0:0

- ◆ When both sides are optional, foreign key is placed in the table which causes minimum null values
- ◆ Or, create a new intersection table → kind of like normalization.
- ◆ Example:
 - A locker is optional for an employee (an employee may not get one)
 - An employee is optional for a locker (a locker may not be assigned to someone)



Employees (EmployeeId, FirstName, LastName, Gender)

Lockers (LockerNumber, Size, Location)

LockerAssignment (LockerNumber, EmployeeId)

FK: LockerNumber → Lockers.LockerNumber

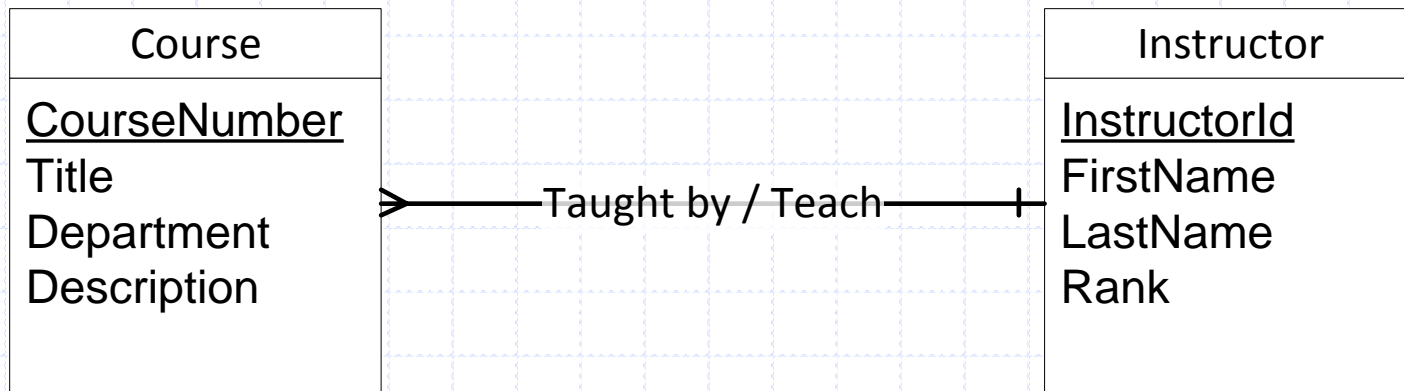
FK: EmployeeId → Employees.EmployeeId

An intersection table
with a composite
primary key

Two foreign keys

FK in 1:N Relationships

- ◆ The primary key from the “One” side is placed in the “Many” side as a foreign key



Instructor (InstructorId, FirstName, LastName, Rank)
Course (CourseNumber, Title, Department, Description, *TaughtBy*)
FK: TaughtBy → Instructor.InstructorId

“Course” is on the many side, which receives the foreign key.

This foreign key is added to course; if the course has a TaughtBy attribute already, then simply make the attribute as a foreign key – no need to add another one.

Why is FK on the many side?

- ◆ The foreign key is always on the "Many" side to avoid redundancy

<u>InstructorId</u>	FirstName	LastName	<u>Teaching</u>
gzheng1	Jack	Zheng	50012
gzheng1	Jack	Zheng	50013
gzheng1	Jack	Zheng	50021

Placing the FK "Teaching" on the "One" side (Instructor) will cause redundancy.

Redundancy

Composite PK (InstructorId, Teaching) and partial dependency: needs to be normalized.

<u>CourseNumber</u>	Title	Taught By
50012	CIS 2010 Intro CIS	gzheng1
50013	CIS 3730 Database	gzheng1
50014	CIS 3300 Analysis	vvaish2

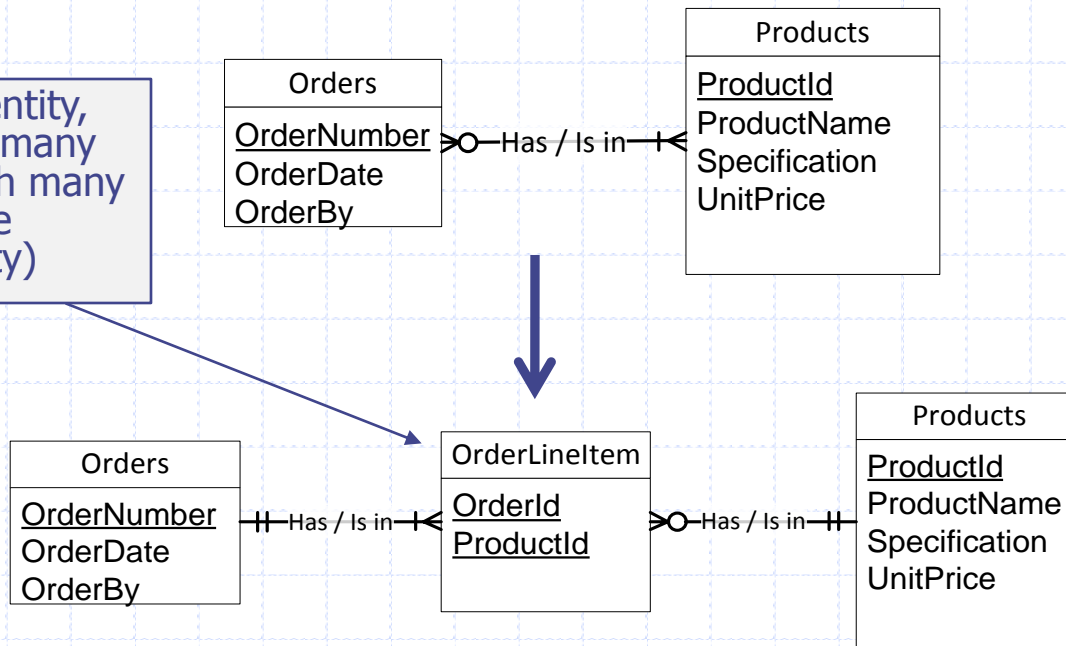
Placing the FK "TaughtBy" on the "Many" side (Course): minimum redundancy.

Transforming N:M Relationships

◆ Many-to-Many

- There is no direct way to map many-to-many relationships in relational database
- A Many-to-Many relationship should be converted to two One-to-Many relationships

An intersection entity, with two one-to-many relationships (both many sides point to the intersection entity)



Creating the Intersection Table

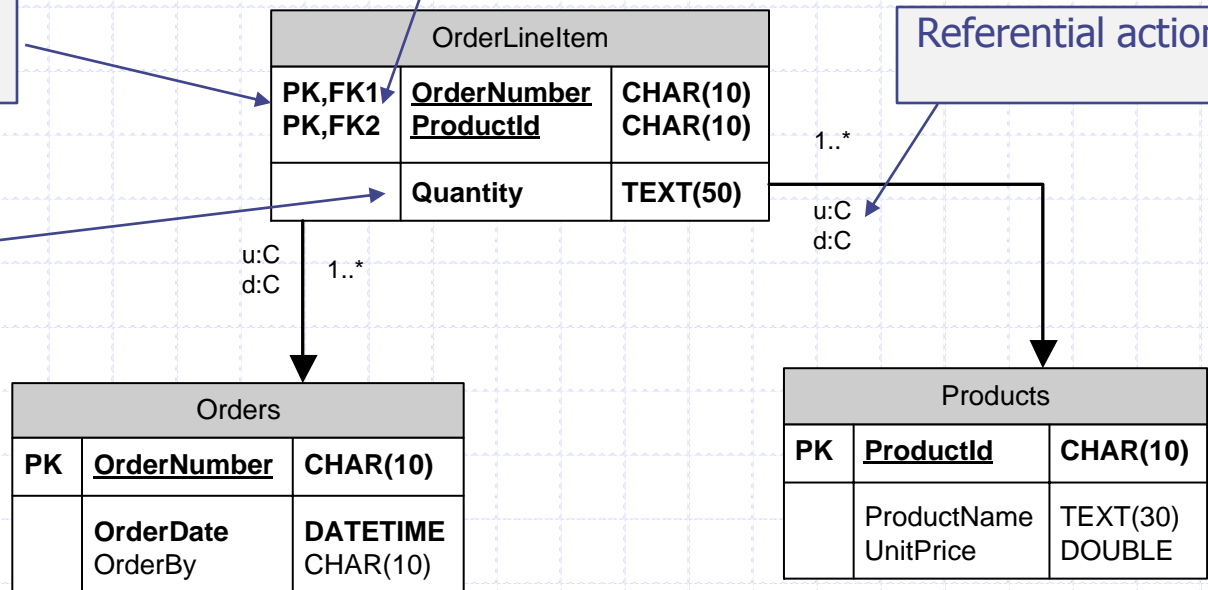
◆ Logical model designed in Visio

The primary key is a composite key consisting of both primary keys from other two tables

Foreign keys are added to the intersection table referencing corresponding tables

Referential actions

If there are more attributes for the relationship, add them to the intersection table as columns



Text Style of the Design

Orders (OrderNumber, OrderDate, OrderBy)

Products (ProductId, ProductName, UnitPrice)

OrderLineItem (OrderNumber, ProductId, Quantity)

FK: OrderNumber_ → Orders.OrderNumber

FK: ProductId → Products.ProductId

Surrogate Key

- ◆ Surrogate keys can be defined at this stage
- ◆ Use surrogate key when a composite primary key will be used as a foreign key
 - This is will much simpler
- ◆ It's common to add a surrogate key to replace the composite key in intersection tables
 - Example

OrderLineItem (LineItemNumber, *OrderNumber*, *ProductId*, Quantity)

FK: OrderNumber_ → Orders.OrderNumber

FK: ProductId → Products.ProductId

Setting FK Nullability

- ◆ Is the foreign key value required?
 - Check the minimum cardinality on the primary key table (parent table) side

Optional

Set foreign key "Optional" or "Null"

Mandatory

Set foreign key "Required" or "Not Null"

◆ FK setting in Visio

Physical Name	Data Type	Req'd	PK
▶ OrderNumber	CHAR(10)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
ProductId	CHAR(10)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Quantity	TEXT(50)	<input checked="" type="checkbox"/>	<input type="checkbox"/>

Check "required" – see slide 17, both minimum cardinalities are mandatory. Also see the Visio tutorial.

Referential Integrity Actions

Actions on the foreign key side table (child table)	Situations: what happens on the primary key side table (parent table)?	
	Update: a primary key value is changed	Delete: a record is deleted
No Action	Nothing will change in the child table. Parent table will NOT be changed.	
<i>Cascading</i>	The FK value of the child table will be changed automatically	All records in the child table matching the deleted row will also be deleted.
Set Null	Set the corresponding FK to null (depending on FK's nullability)	
Set Default	Set the corresponding FK to a default value	
Not Enforced	Nothing will change in the child table. Parent table will be changed (the reference breaks).	

Transformation Steps Summary

1. Transforming basic entities, identifiers and attributes
 - Creating tables, columns and keys
 - Defining the data type, length and constraints for each columns
2. Transforming relationships by determining foreign keys
 - Determining foreign keys and references
 - Defining FK setting: NULL (optional) or NOT NULL (required)
 - Defining referential integrity actions

Summary

◆ Key concepts

- Relational model
- Referential actions (all 5)
- Null

◆ Key skills

- Transform ERD to relational models
 - ◆ Tables, primary keys, data types, and other column constraints
 - ◆ Foreign key, referential actions, and other constraints
 - ◆ Know how to transform three types of relationships (1:1, 1;N, N:M)
- Be able to use the text style of database structure representation
- Use Visio database model diagram to design logical relational models