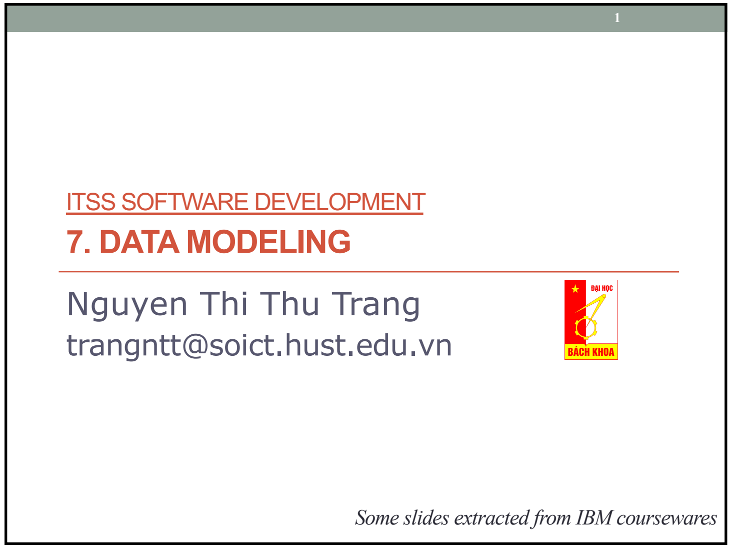
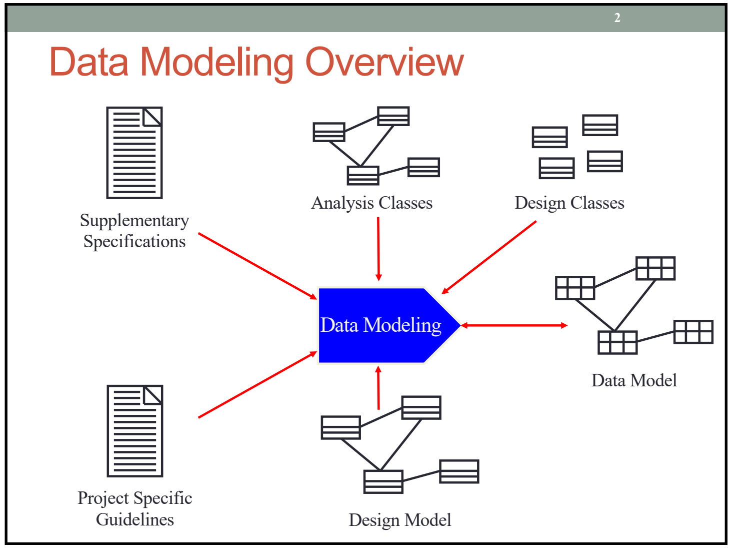


ITSS SOFTWARE DEVELOPMENT

7. DATA MODELING

Nguyen Thi Thu Trang
trangntt@soict.hust.edu.vn

Some slides extracted from IBM coursewares



Data Modeling Overview

Supplementary Specifications

Analysis Classes

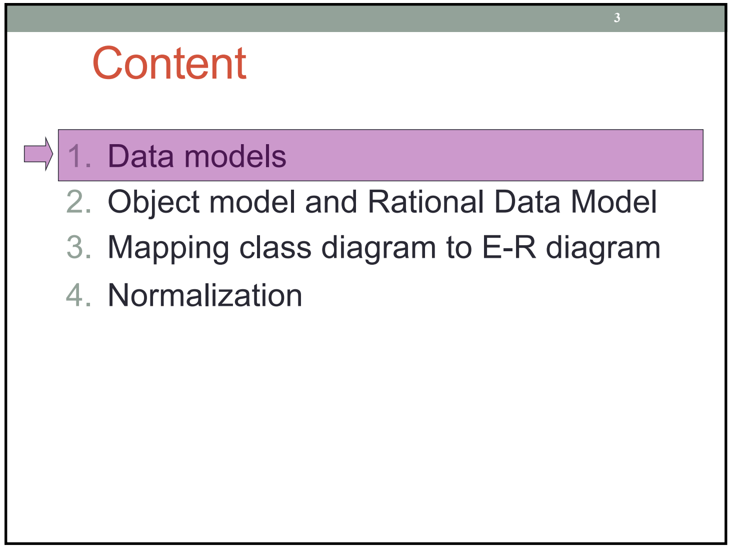
Design Classes

Data Modeling

Data Model

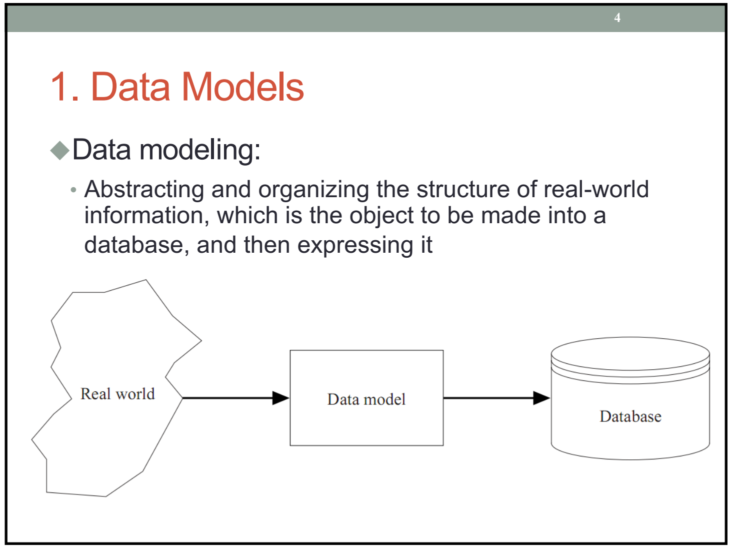
Project Specific Guidelines

Design Model



Content

1. Data models
2. Object model and Rational Data Model
3. Mapping class diagram to E-R diagram
4. Normalization



1. Data Models

- Data modeling:
  - Abstracting and organizing the structure of real-world information, which is the object to be made into a database, and then expressing it

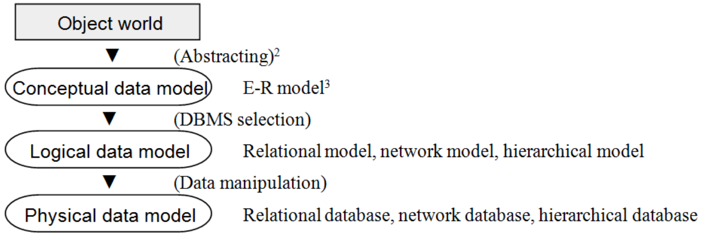
Real world

Data model

Database

1. Data Models (2)

- 3 types of data models



Object world

(Abstracting)[2]

Conceptual data model E-R model[3]

(DBMS selection)

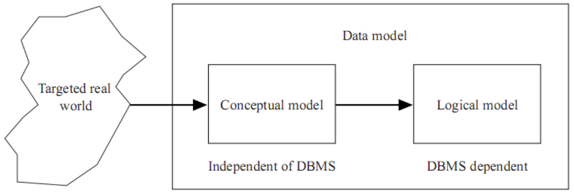
Logical data model Relational model, network model, hierarchical model

(Data manipulation)

Physical data model Relational database, network database, hierarchical database

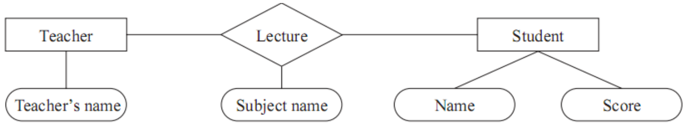
1.1. Conceptual data model

- Natural expressions without constraints imposed by DBMS
- E-R model
  - Expressed by E-R diagram



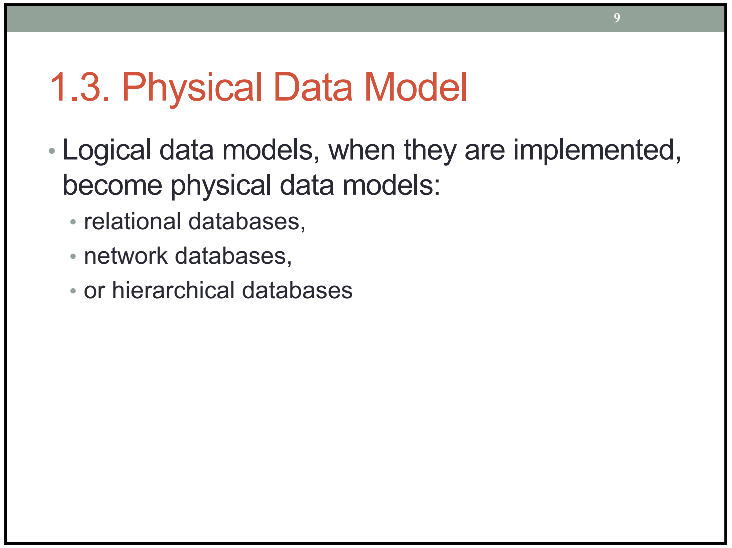
E-R Diagram

- Three elements
  - Entities
  - Relationships
  - Attributes



1.2. Logical Data Model

- 3 types
  - relational model,
  - network model,
  - and hierarchical model

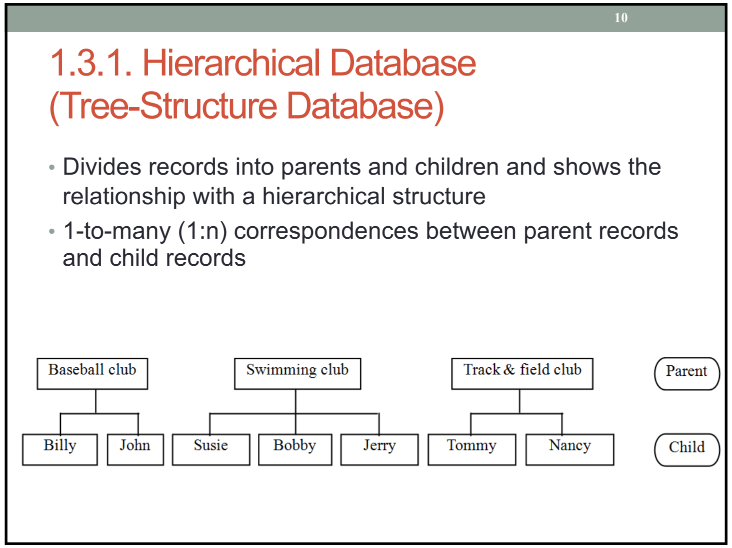


9

1.3. Physical Data Model

- Logical data models, when they are implemented, become physical data models:
  - relational databases,
  - network databases,
  - or hierarchical databases

9



10

1.3.1. Hierarchical Database (Tree-Structure Database)

- Divides records into parents and children and shows the relationship with a hierarchical structure
- 1-to-many (1:n) correspondences between parent records and child records

Baseball club

Swimming club

Track & field club

Parent

Billy

John

Susie

Bobby

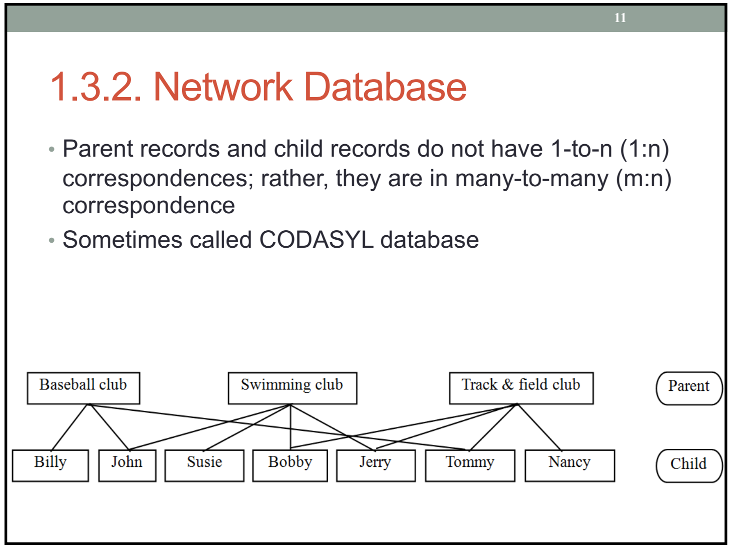
Jerry

Tommy

Nancy

Child

10



11

1.3.2. Network Database

- Parent records and child records do not have 1-to-n (1:n) correspondences; rather, they are in many-to-many (m:n) correspondence
- Sometimes called CODASYL database

Baseball club

Swimming club

Track & field club

Parent

Billy

John

Susie

Bobby

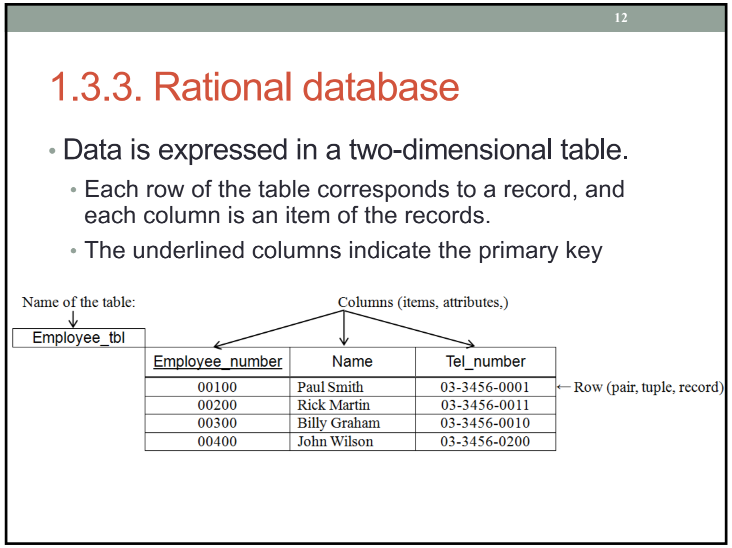
Jerry

Tommy

Nancy

Child

11



12

1.3.3. Rational database

- Data is expressed in a two-dimensional table.
  - Each row of the table corresponds to a record, and each column is an item of the records.
  - The underlined columns indicate the primary key

Name of the table:

Columns (items, attributes,)

Employee_tbl

Employee_number	Name	Tel_number
00100	Paul Smith	03-3456-0001
00200	Rick Martin	03-3456-0011
00300	Billy Graham	03-3456-0010
00400	John Wilson	03-3456-0200

← Row (pair, tuple, record)

12

NOSQL DATABASES

Overview, Models, Concepts, Examples

13

What is NoSQL Database?

- NoSQL (cloud) databases
  - Use document-based model (non-relational)
  - Schema-free document storage
    - Still support indexing and querying
    - Still support CRUD operations (create, read, update, delete)
    - Still supports concurrency and transactions
  - Highly optimized for append / retrieve
  - Great performance and scalability
  - NoSQL == “No SQL” or “Not Only SQL”?

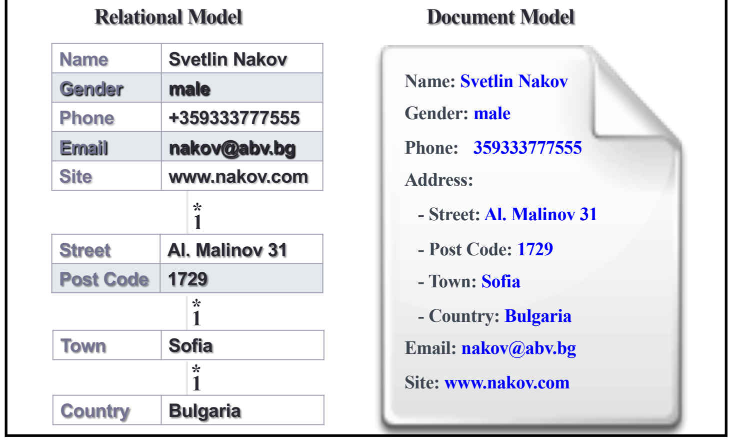
14

Relational vs. NoSQL Databases

- Relational databases
  - Data stored as table rows
  - Relationships between related rows
  - Single entity spans multiple tables
  - RDBMS systems are very mature, rock solid
- NoSQL databases
  - Data stored as documents
  - Single entity (document) is a single record
  - Documents do not have a fixed structure

15

Relational vs. NoSQL Models



Relational Model

Name	Svetlin Nakov
Gender	male
Phone	+359333777555
Email	nakov@abv.bg
Site	www.nakov.com

*
1

Street	Al. Malinov 31
Post Code	1729

*
1

Town	Sofia

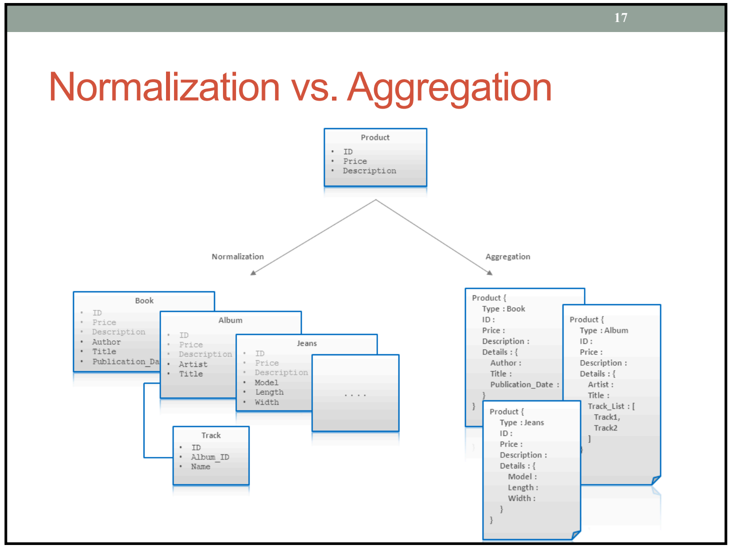
*
1

Country	Bulgaria

Document Model

Name: Svetlin Nakov
Gender: male
Phone: 359333777555
Address:
- Street: Al. Malinov 31
- Post Code: 1729
- Town: Sofia
- Country: Bulgaria
Email: nakov@abv.bg
Site: www.nakov.com

16



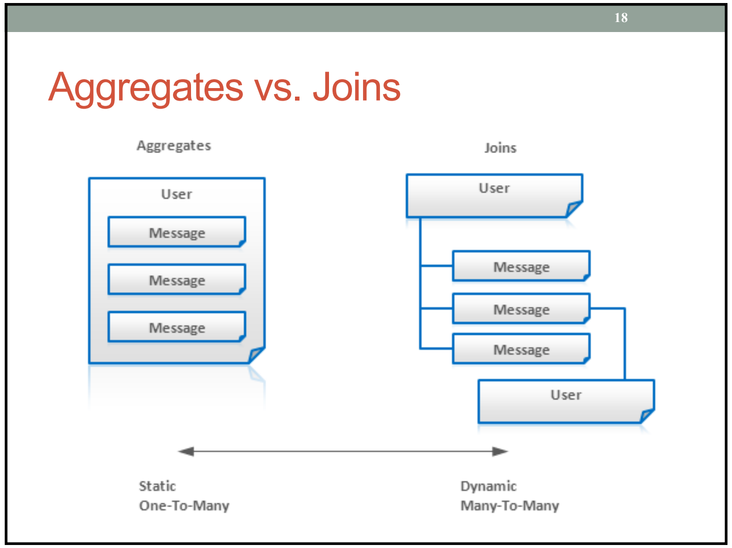
Normalization vs. Aggregation

```
Product {
  Type : Book
  ID :
  Price :
  Description :
  Details : {
    Author :
    Title :
    Publication_Date :
  }
}
```

```
Product {
  Type : Album
  ID :
  Price :
  Description :
  Details : {
    Artist :
    Title :
    Track_List : [
      Track1,
      Track2
    ]
}
```

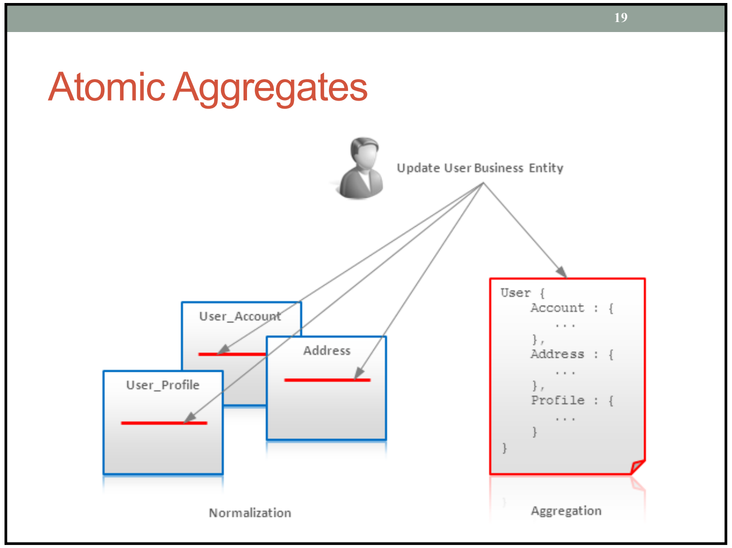
```
Product {
  Type : Jeans
  ID :
  Price :
  Description :
  Details : {
    Model :
    Length :
    Width :
  }
}
```

17



Aggregates vs. Joins

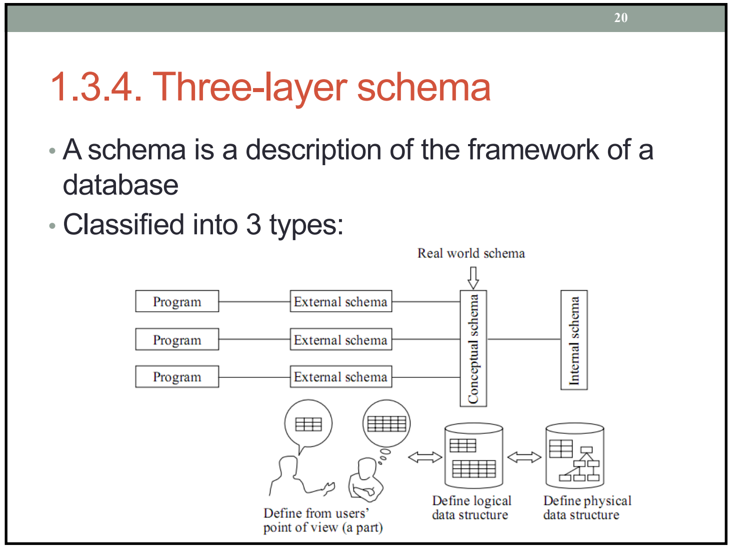
18



Atomic Aggregates

```
User {
    Account : {
      . . .
    },
    Address : {
      . . .
    },
    Profile : {
      . . .
    }
}
```

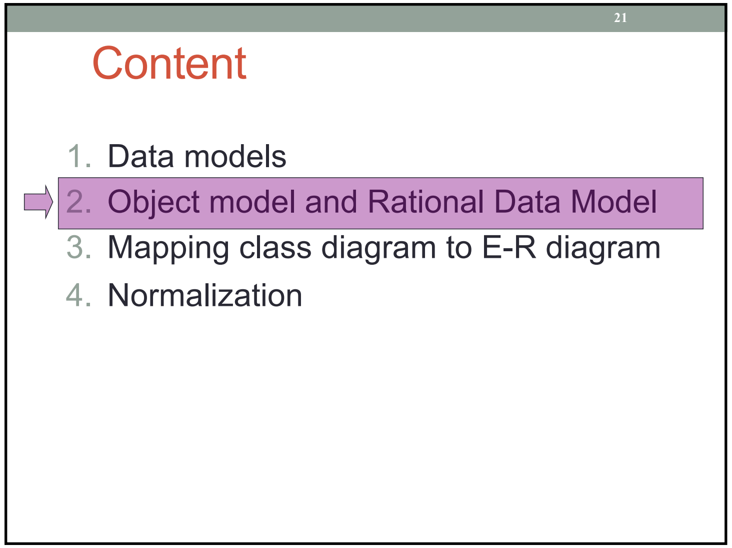
19



1.3.4. Three-layer schema

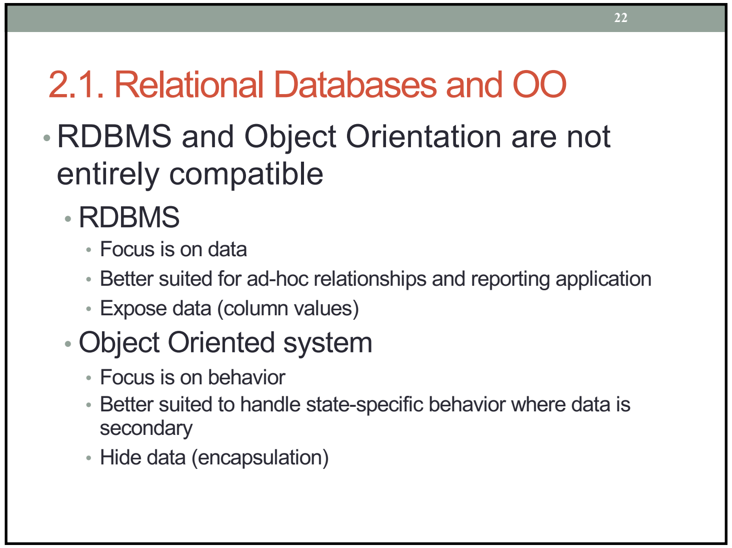
- A schema is a description of the framework of a database
- Classified into 3 types:

20



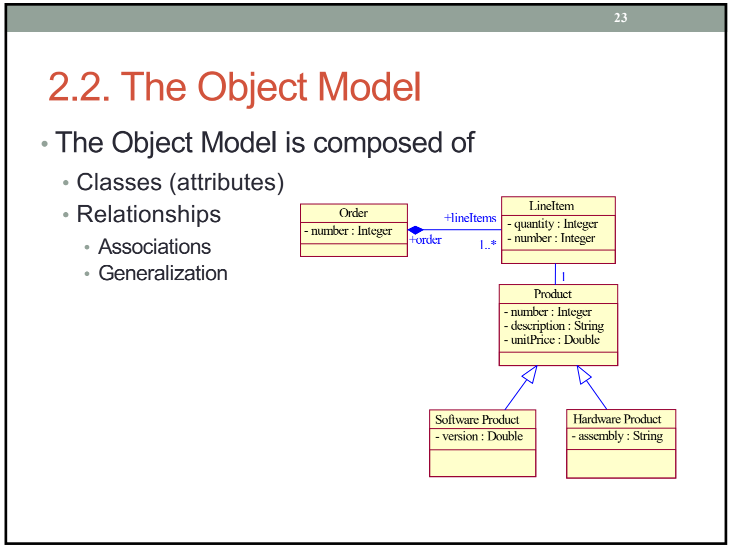
Content

1. Data models
2. Object model and Rational Data Model
3. Mapping class diagram to E-R diagram
4. Normalization



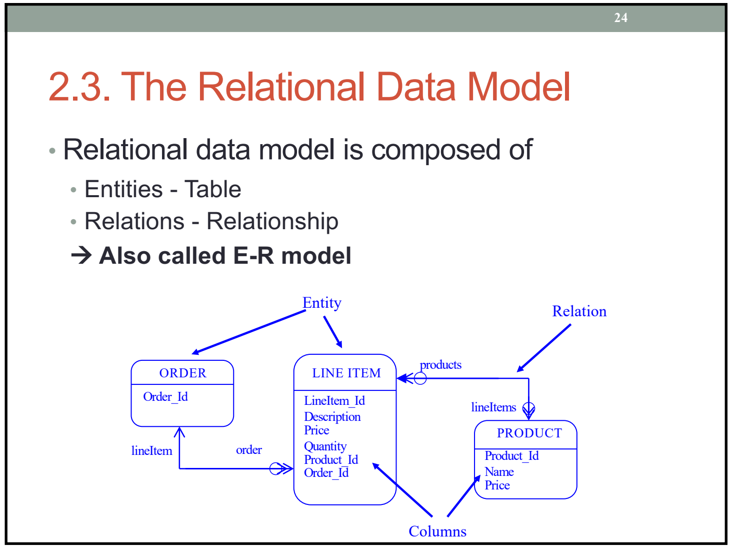
2.1. Relational Databases and OO

- RDBMS and Object Orientation are not entirely compatible
  - RDBMS
    - Focus is on data
    - Better suited for ad-hoc relationships and reporting application
    - Expose data (column values)
  - Object Oriented system
    - Focus is on behavior
    - Better suited to handle state-specific behavior where data is secondary
    - Hide data (encapsulation)



2.2. The Object Model

- The Object Model is composed of
  - Classes (attributes)
  - Relationships
    - Associations
    - Generalization



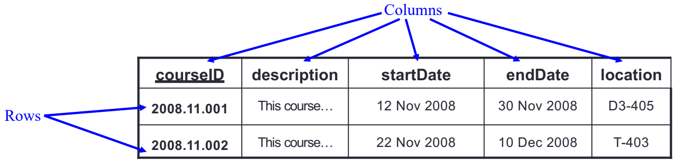
2.3. The Relational Data Model

- Relational data model is composed of
  - Entities - Table
  - Relations - Relationship

  → **Also called E-R model**

2.3.1. Entities/Tables

- Entities is mapped to table when design physical database
- Including
  - Columns: Attributes
  - Rows: Concrete values of attributes


Columns
Rows

courseID	description	startDate	endDate	location
2008.11.001	This course…	12 Nov 2008	30 Nov 2008	D3-405
2008.11.002	This course…	22 Nov 2008	10 Dec 2008	T-403

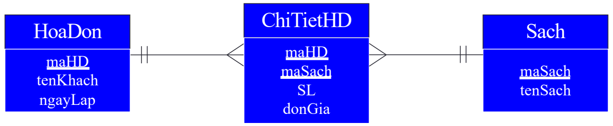
2.3.2. Relations/Relationships

- Relations between entities or relationship between tables
- Multiplicity/Cardinality
  - One-to-one (1:1)
  - One-to-many (1:m)
  - Many-to-one (m:1)
  - Many-to-many (m:n)

  (Normally, many-to-many relation is devided to one-to-many and many-to-one relations)

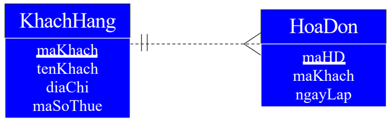
Dependency relationships

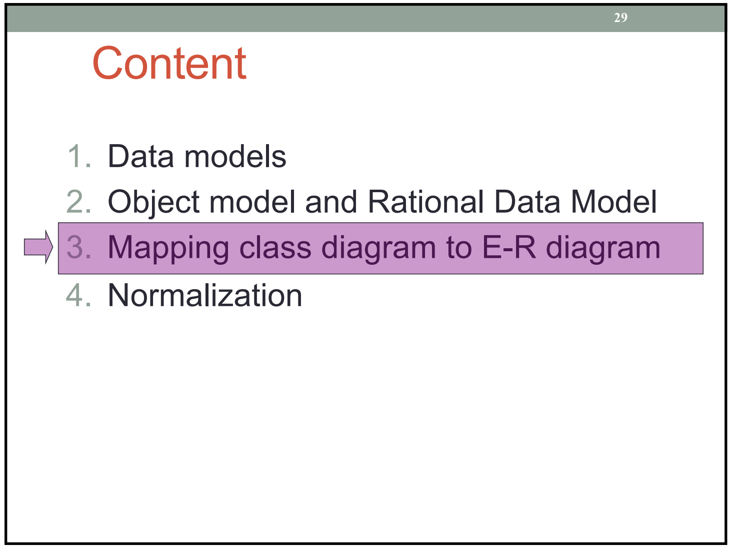
- The child entity can exist only when the parent entity exists
- The child entity has a foreign key referencing to the primary key of the parent entity
- This foreign key is included in the primary key of the child
- Solid line



Independency relationships

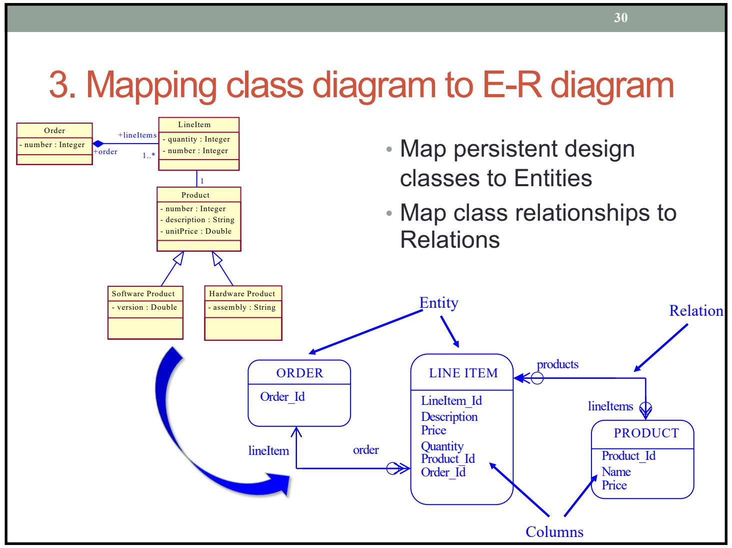
- The child entity can exist even if the parent entity does not exist
- The child entity has a foreign key referencing to the primary key of the parent entity
- This foreign key is not included in the primary key of the child
- Dash line





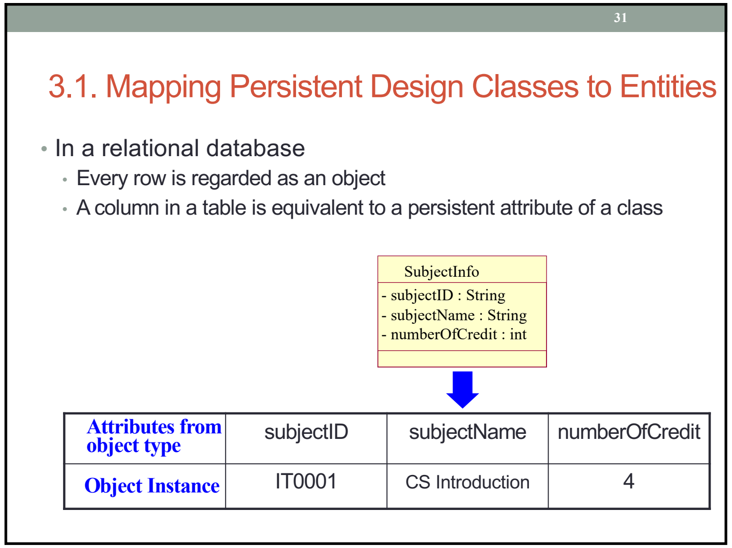
Content

1. Data models
2. Object model and Rational Data Model
3. Mapping class diagram to E-R diagram
4. Normalization



3. Mapping class diagram to E-R diagram

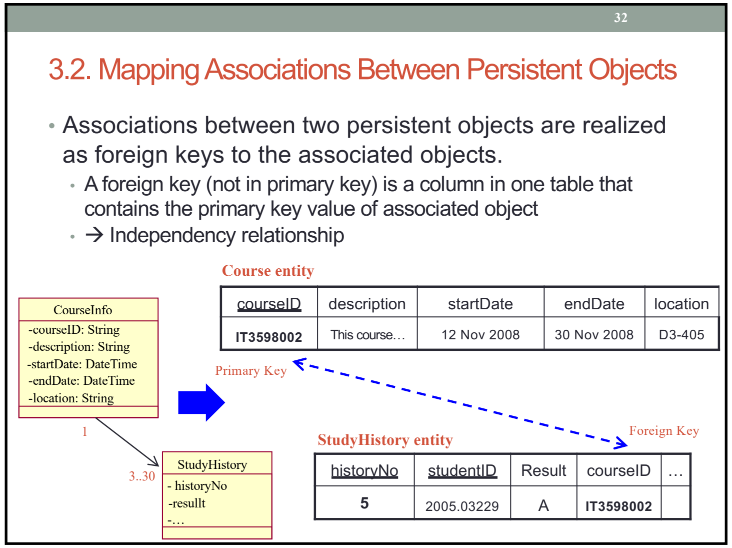
- Map persistent design classes to Entities
- Map class relationships to Relations



3.1. Mapping Persistent Design Classes to Entities

- In a relational database
  - Every row is regarded as an object
  - A column in a table is equivalent to a persistent attribute of a class

Attributes from object type	subjectID	subjectName	numberOfCredit
Object Instance	IT0001	CS Introduction	4



3.2. Mapping Associations Between Persistent Objects

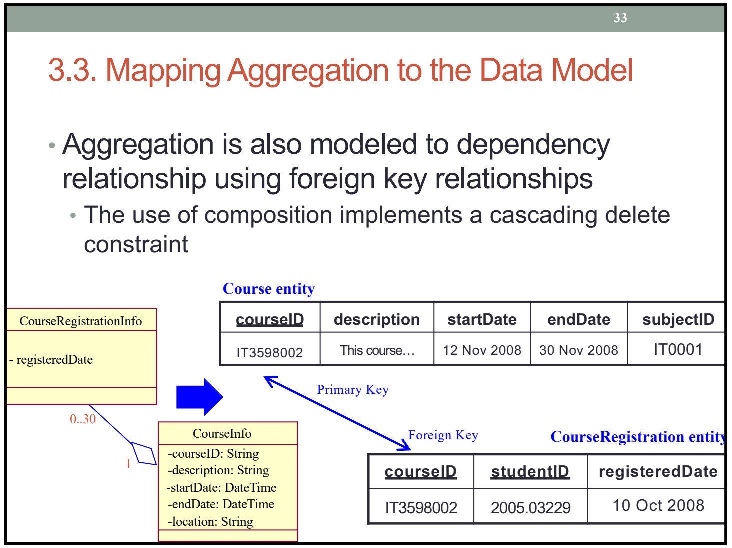
- Associations between two persistent objects are realized as foreign keys to the associated objects.
  - A foreign key (not in primary key) is a column in one table that contains the primary key value of associated object
  - → Independency relationship

Course entity

courseID	description	startDate	endDate	location
IT3598002	This course…	12 Nov 2008	30 Nov 2008	D3-405

StudyHistory entity

historyNo	studentID	Result	courseID	…
5	2005.03229	A	IT3598002	



3.3. Mapping Aggregation to the Data Model

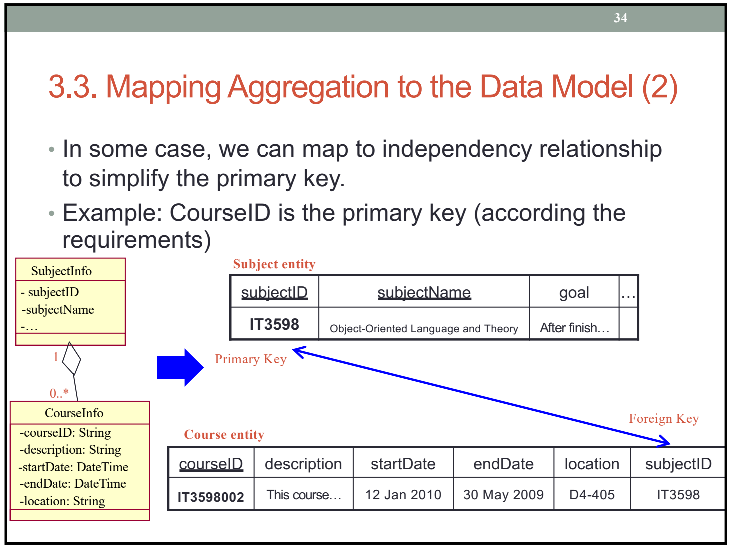
- Aggregation is also modeled to dependency relationship using foreign key relationships
  - The use of composition implements a cascading delete constraint

Course entity

courseID	description	startDate	endDate	subjectID
IT3598002	This course…	12 Nov 2008	30 Nov 2008	IT0001

CourseRegistration entity

courseID	**studentID**	registeredDate
IT3598002	2005.03229	10 Oct 2008



3.3. Mapping Aggregation to the Data Model (2)

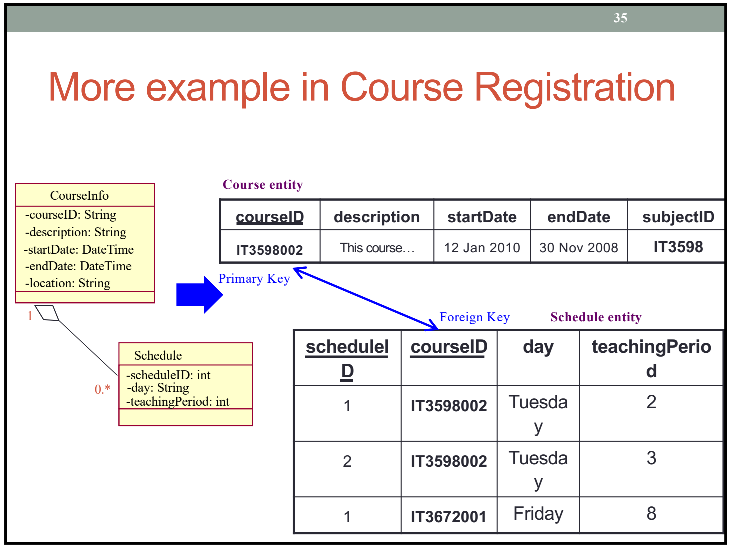
- In some case, we can map to independency relationship to simplify the primary key.
- Example: CourseID is the primary key (according the requirements)

Subject entity

subjectID	subjectName	goal	…
IT3598	Object-Oriented Language and Theory	After finish…	

Course entity

courseID	description	startDate	endDate	location	subjectID
IT3598002	This course…	12 Jan 2010	30 May 2009	D4-405	IT3598



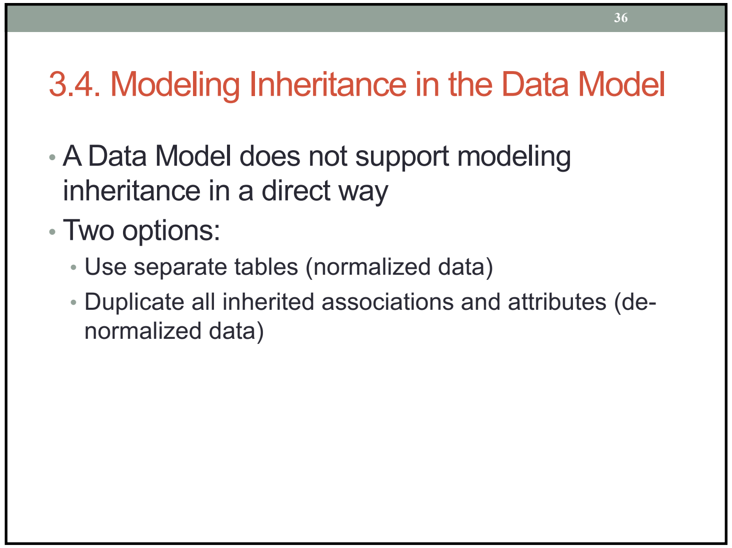
More example in Course Registration

Course entity

courseID	description	startDate	endDate	subjectID
IT3598002	This course…	12 Jan 2010	30 Nov 2008	**IT3598**

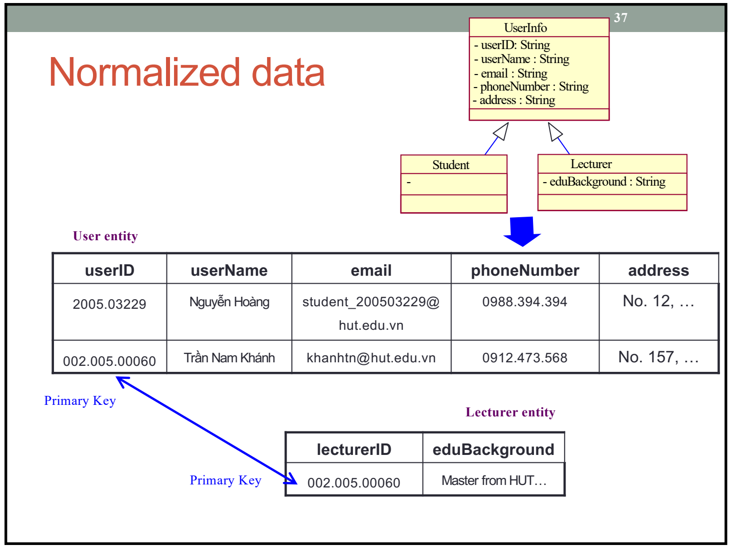
Schedule entity

scheduleID	**courseID**	day	teachingPeriod
1	**IT3598002**	Tuesday	2
2	**IT3598002**	Tuesday	3
1	**IT3672001**	Friday	8



3.4. Modeling Inheritance in the Data Model

- A Data Model does not support modeling inheritance in a direct way
- Two options:
  - Use separate tables (normalized data)
  - Duplicate all inherited associations and attributes (de-normalized data)



Normalized data

UserInfo
- userID: String
- userName : String
- email : String
- phoneNumber : String
- address : String

Student
-

Lecturer
- eduBackground : String

User entity

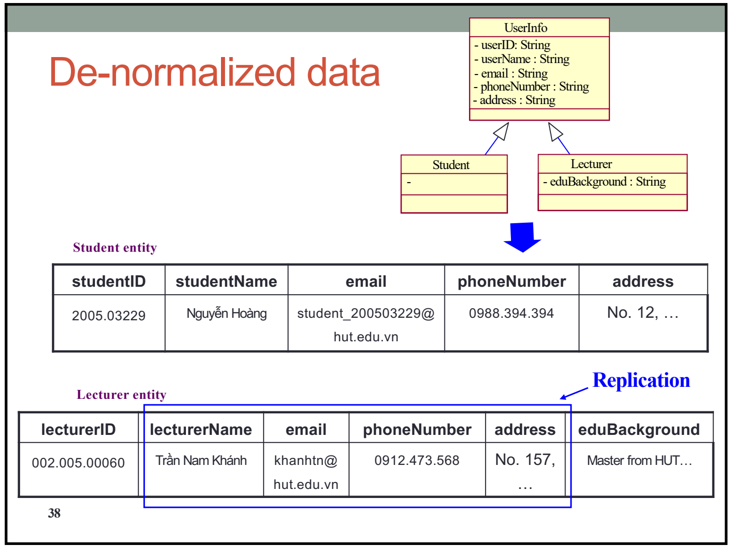
userID	userName	email	phoneNumber	address
2005.03229	Nguyễn Hoàng	student_200503229@ hut.edu.vn	0988.394.394	No. 12, …
002.005.00060	Trần Nam Khánh	khanhtn@hut.edu.vn	0912.473.568	No. 157, …

Primary Key

Lecturer entity

lecturerID	eduBackground
002.005.00060	Master from HUT…

Primary Key



De-normalized data

UserInfo
- userID: String
- userName : String
- email : String
- phoneNumber : String
- address : String

Student
-

Lecturer
- eduBackground : String

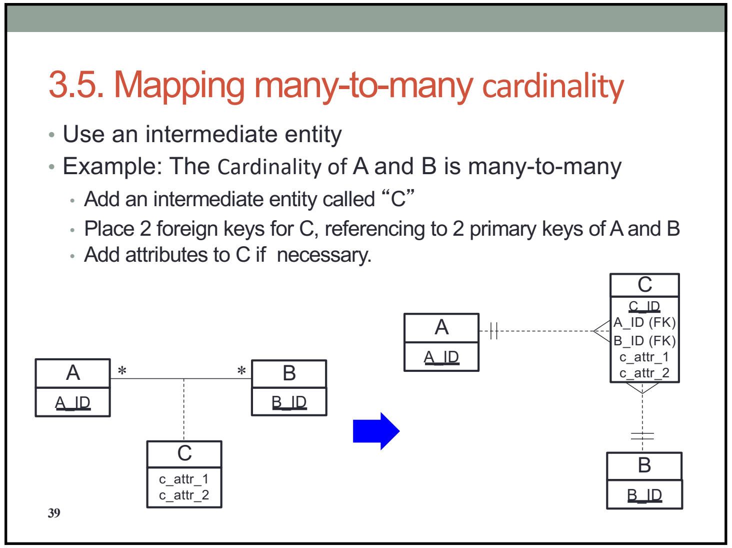
Student entity

studentID	studentName	email	phoneNumber	address
2005.03229	Nguyễn Hoàng	student_200503229@ hut.edu.vn	0988.394.394	No. 12, …

Lecturer entity

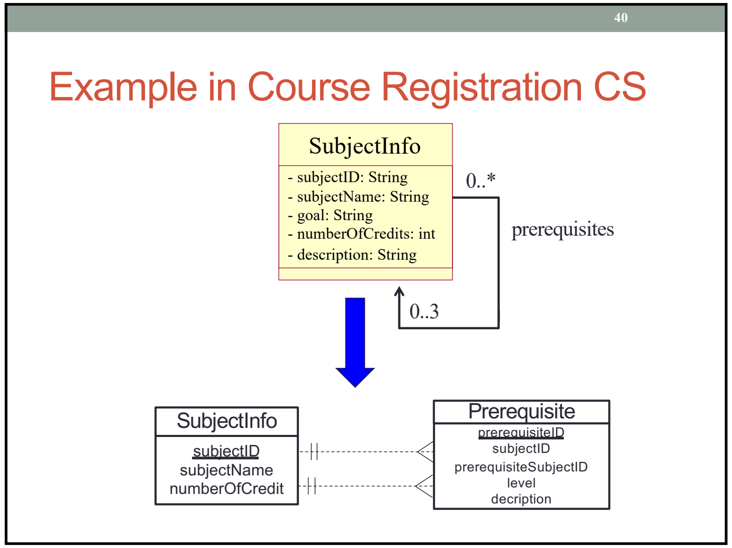
Replication

lecturerID	lecturerName	email	phoneNumber	address	eduBackground
002.005.00060	Trần Nam Khánh	khanhtn@ hut.edu.vn	0912.473.568	No. 157, …	Master from HUT…



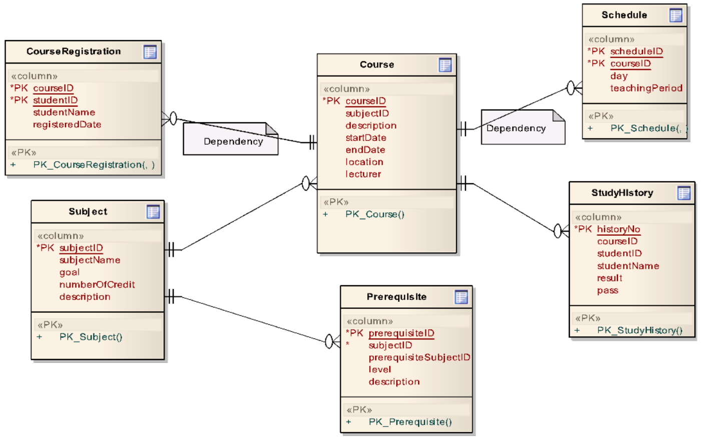
3.5. Mapping many-to-many cardinality

- Use an intermediate entity
- Example: The Cardinality of A and B is many-to-many
  - Add an intermediate entity called "C"
  - Place 2 foreign keys for C, referencing to 2 primary keys of A and B
  - Add attributes to C if necessary.



Example in Course Registration CS

E-R diagram



Content

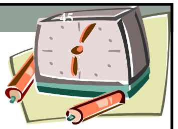
1. Data models
2. Object model and Rational Data Model
3. Mapping class diagram to E-R diagram
4. Normalization

4.1. Overview of Normalization

- Normalization: the process of steps that will identify, for elimination, redundancies in a database design.
- Purpose of Normalization: to improve
  - storage efficiency
  - data integrity
  - and scalability

4.1. Overview of Normalization (2)

- In relational model, methods exist for quantifying how efficient a database is.
- These classifications are called **normal forms** (or **NF**), and there are algorithms for converting a given database between them.
- Normalization generally involves splitting existing tables into multiple ones, which must be re-joined or linked each time a query is issued



4.2. History

- Edgar F. Codd first proposed the process of normalization and what came to be known as the 1st normal form in his paper *A Relational Model of Data for Large Shared Data Banks Codd* stated:

  "There is, in fact, a very simple elimination procedure which we shall call normalization. Through decomposition nonsimple domains are replaced by 'domains whose elements are atomic (nondecomposable) values".

45

4.3. Normal Forms

- Edgar F. Codd originally established three normal forms: 1NF, 2NF and 3NF.
- There are now others that are generally accepted, but 3NF is widely considered to be sufficient for most applications.
- Most tables when reaching 3NF are also in BCNF (Boyce-Codd Normal Form).

46

Functionally determines

- In a table, a set of columns X, functionally determines another column Y…

  X → Y

  … if and only if each X value is associated with at most one Y value in a table.

- i.e. if you know X then there is only **one** possibility for Y.

47

Normal forms so Far…

◆First normal form

- All data values are atomic, and so everything fits into a mathematical relation.

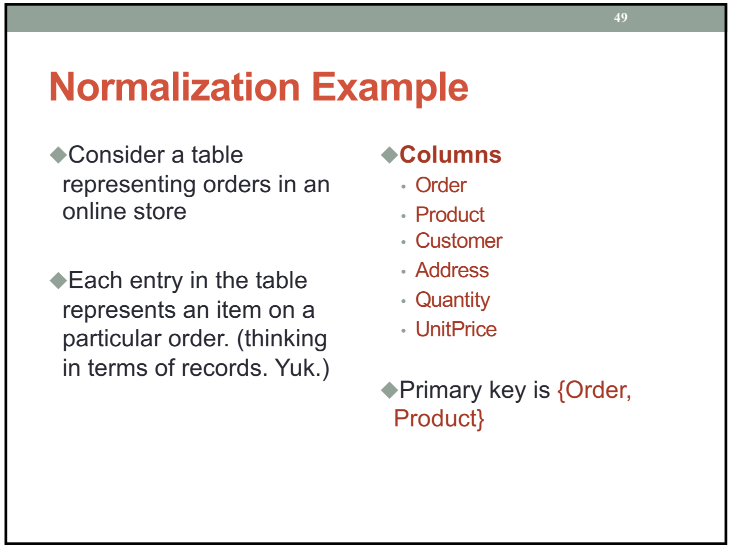
◆Second normal form

- As 1NF plus no *non-primary-key attribute* is partially dependant on the primary key

◆Third normal form

- As 2NF plus no non-primary-key attribute depends transitively on the primary key

48

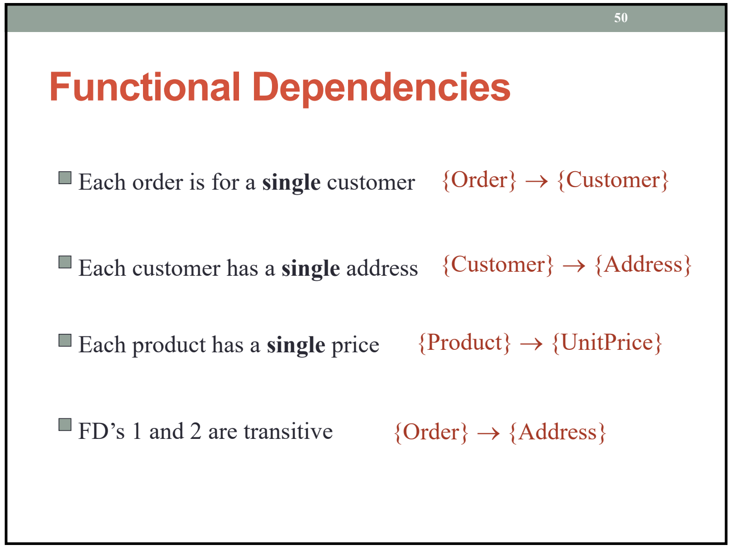


Normalization Example

- Consider a table representing orders in an online store
- Each entry in the table represents an item on a particular order. (thinking in terms of records. Yuk.)

- **Columns**
  - Order
  - Product
  - Customer
  - Address
  - Quantity
  - UnitPrice
- Primary key is {Order, Product}

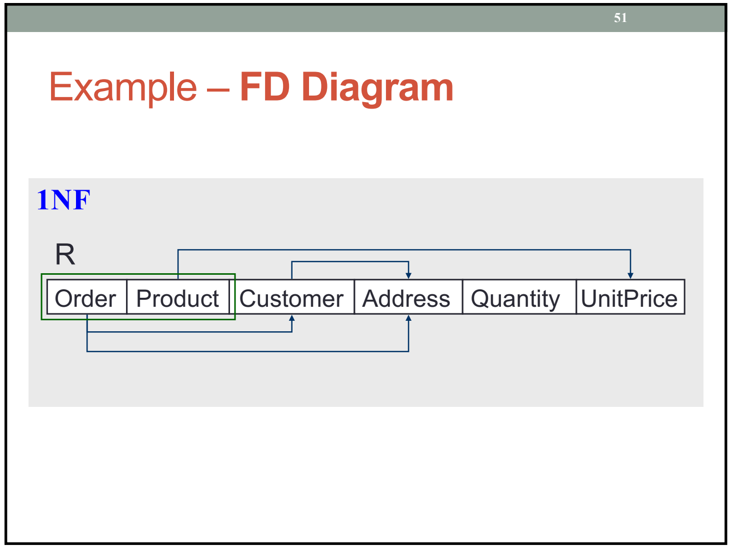
49



Functional Dependencies

- Each order is for a **single** customer {Order} → {Customer}
- Each customer has a **single** address {Customer} → {Address}
- Each product has a **single** price {Product} → {UnitPrice}
- FD's 1 and 2 are transitive {Order} → {Address}

50



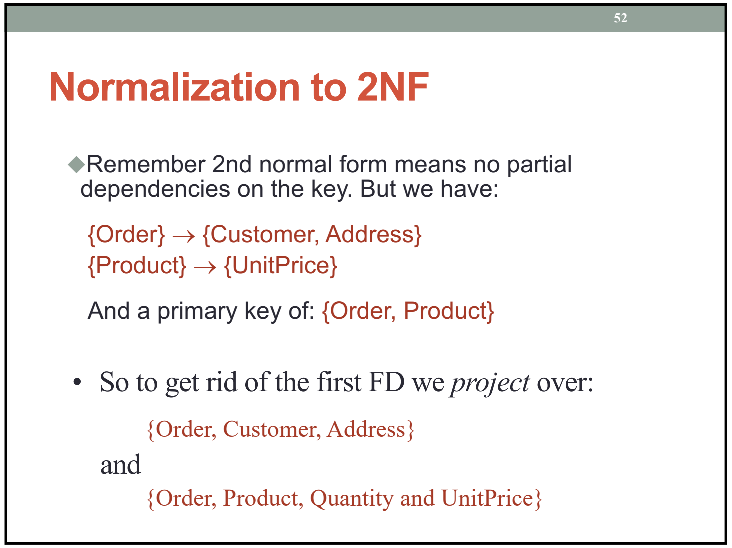
Example – **FD Diagram**

1NF

R

Order	Product	Customer	Address	Quantity	UnitPrice

51



Normalization to 2NF

- Remember 2nd normal form means no partial dependencies on the key. But we have:

  {Order} → {Customer, Address}
  {Product} → {UnitPrice}

  And a primary key of: {Order, Product}

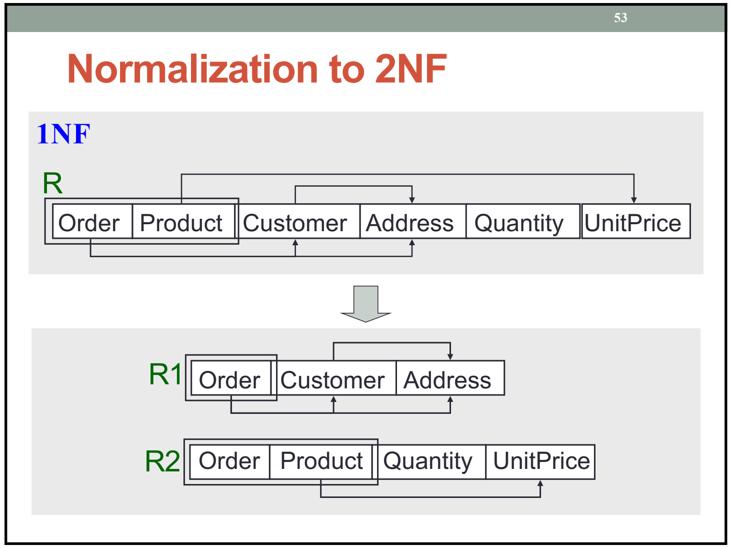
- So to get rid of the first FD we *project* over:

  {Order, Customer, Address}

  and

  {Order, Product, Quantity and UnitPrice}

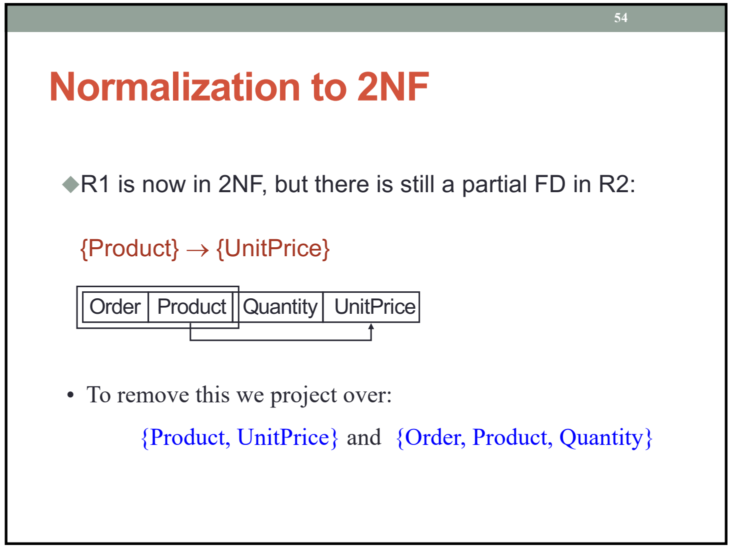
52



Normalization to 2NF

1NF

53



Normalization to 2NF

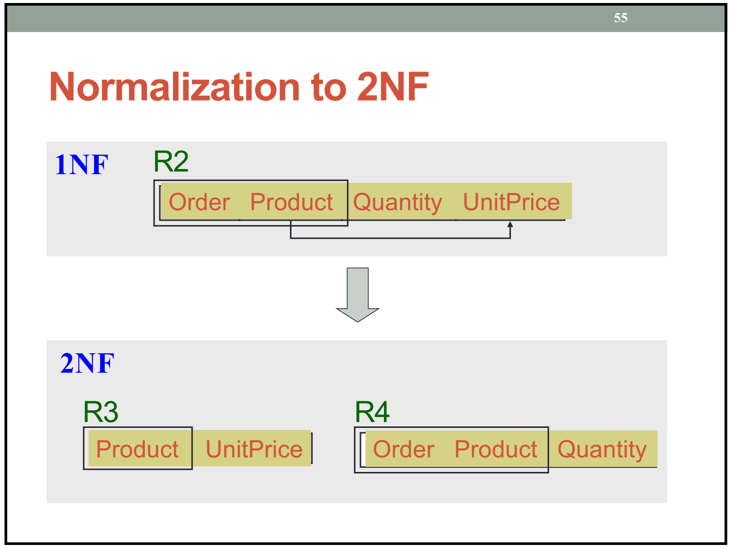
- R1 is now in 2NF, but there is still a partial FD in R2:

{Product} → {UnitPrice}

- To remove this we project over:

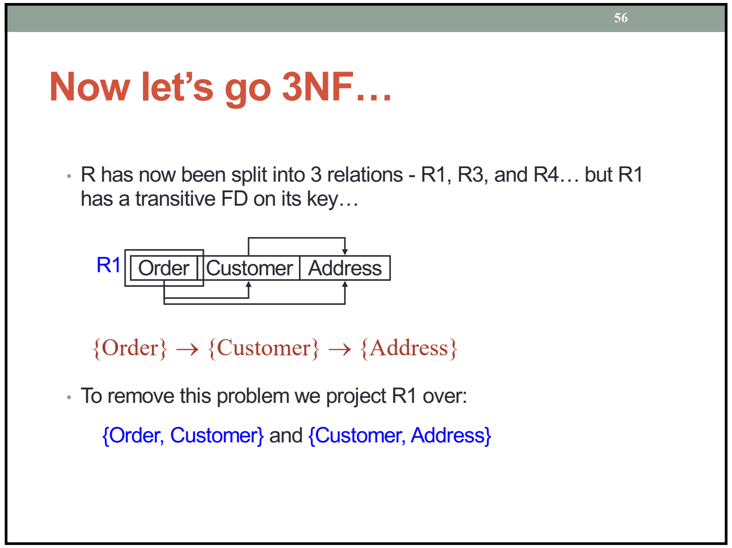
{Product, UnitPrice} and {Order, Product, Quantity}

54



Normalization to 2NF

55



Now let’s go 3NF…

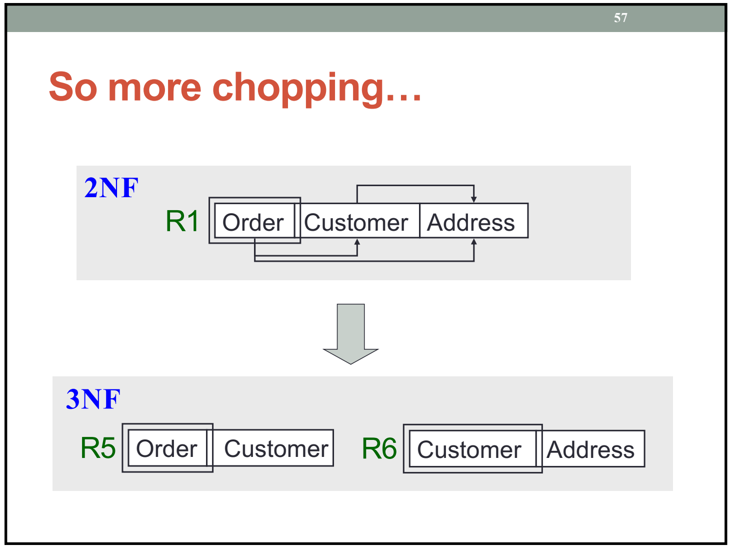
- R has now been split into 3 relations - R1, R3, and R4… but R1 has a transitive FD on its key…

{Order} → {Customer} → {Address}

- To remove this problem we project R1 over:

{Order, Customer} and {Customer, Address}

56



So more chopping…

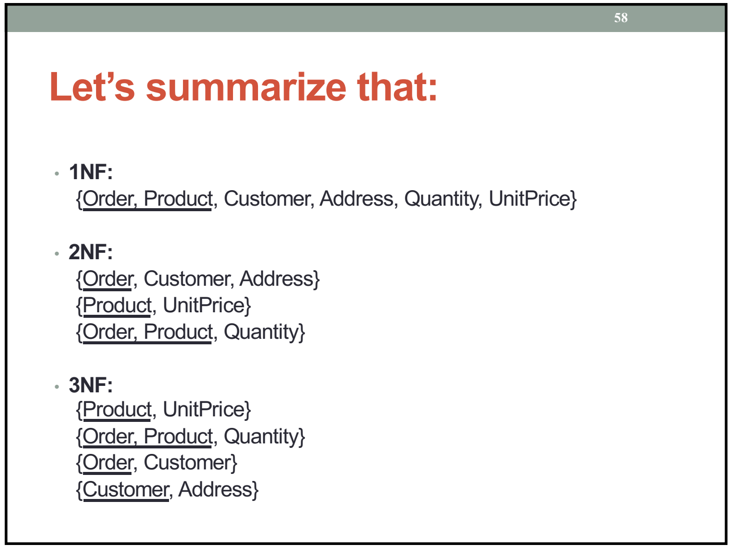
2NF

R1 | Order | Customer | Address

3NF

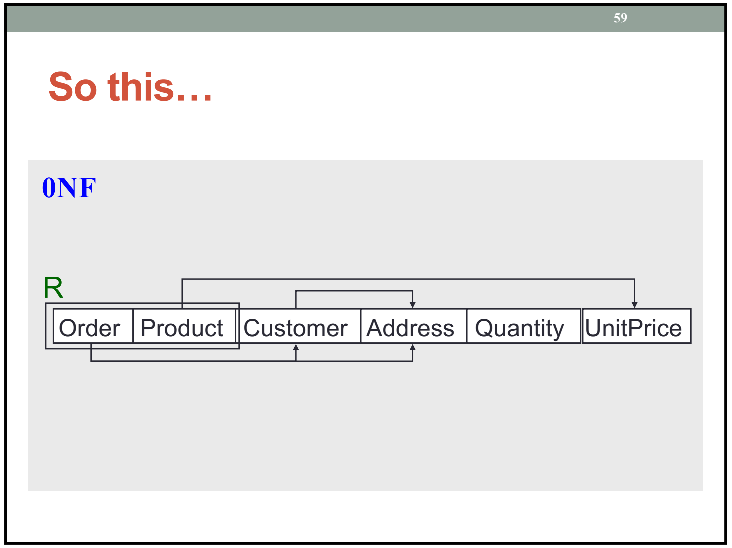
R5 | Order | Customer

R6 | Customer | Address



Let’s summarize that:

- **1NF:**
  {Order, Product, Customer, Address, Quantity, UnitPrice}
- **2NF:**
  {Order, Customer, Address}
  {Product, UnitPrice}
  {Order, Product, Quantity}
- **3NF:**
  {Product, UnitPrice}
  {Order, Product, Quantity}
  {Order, Customer}
  {Customer, Address}

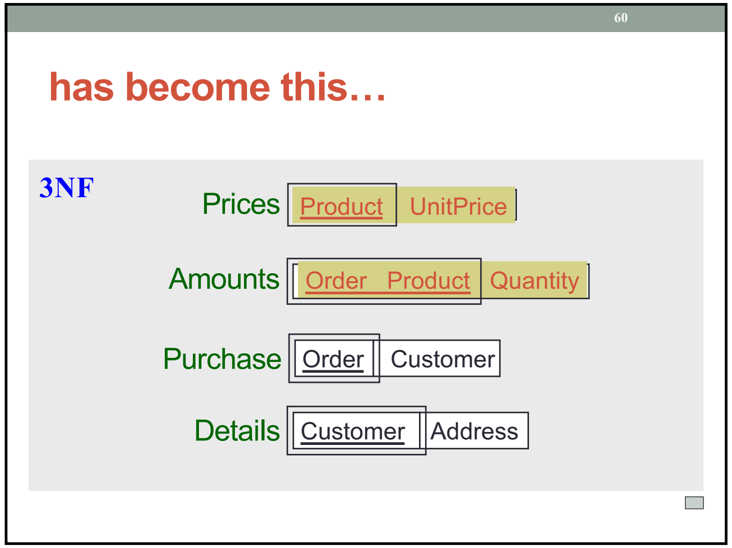


So this…

0NF

R

Order | Product | Customer | Address | Quantity | UnitPrice



has become this…

3NF

Prices | Product | UnitPrice

Amounts | Order Product | Quantity

Purchase | Order | Customer

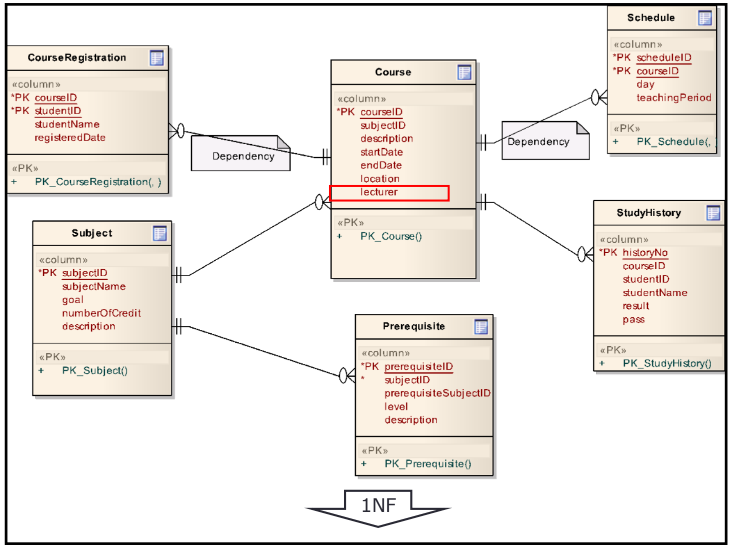
Details | Customer | Address

61

“Register for course” use case

- Make the E-R diagram from the previous step for “Register for course” use case to become:
  - The first normal form
  - The second normal form
  - The third normal form

61



CourseRegistration
«column»
*PK courseID
*PK studentID
studentName
registeredDate
«PK»
+ PK_CourseRegistration(,)

Course
«column»
*PK courseID
subjectID
description
startDate
endDate
location
lecturer
«PK»
+ PK_Course()

Schedule
«column»
*PK scheduleID
*PK courseID
day
teachingPeriod
«PK»
+ PK_Schedule(,)

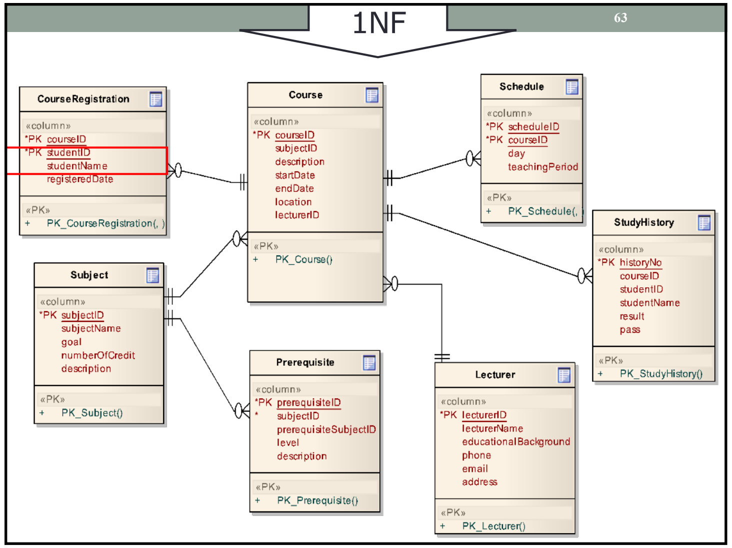
StudyHistory
«column»
*PK historyNo
courseID
studentID
studentName
result
pass
«PK»
+ PK_StudyHistory()

Subject
«column»
*PK subjectID
subjectName
goal
numberOfCredit
description
«PK»
+ PK_Subject()

Prerequisite
«column»
*PK prerequisiteID
* subjectID
prerequisiteSubjectID
level
description
«PK»
+ PK_Prerequisite()

62

63



CourseRegistration
«column»
*PK courseID
*PK studentID
studentName
registeredDate
«PK»
+ PK_CourseRegistration(,)

Course
«column»
*PK courseID
subjectID
description
startDate
endDate
location
lecturerID
«PK»
+ PK_Course()

Schedule
«column»
*PK scheduleID
*PK courseID
day
teachingPeriod
«PK»
+ PK_Schedule(,)

StudyHistory
«column»
*PK historyNo
courseID
studentID
studentName
result
pass
«PK»
+ PK_StudyHistory()

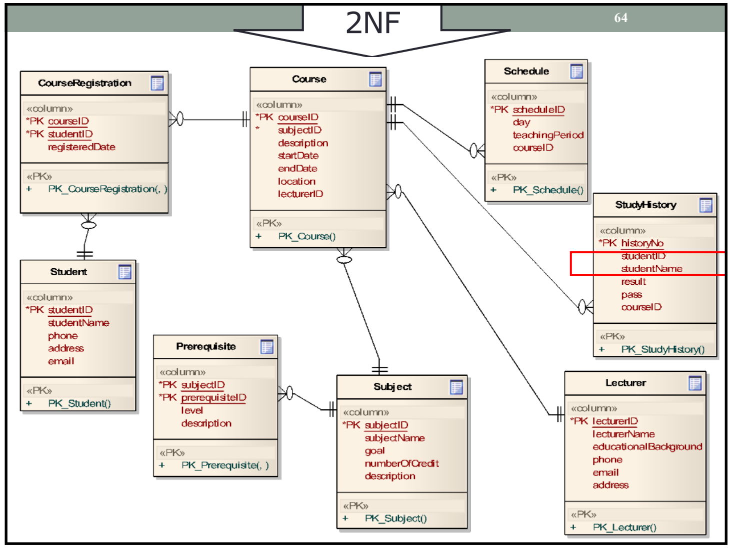
Subject
«column»
*PK subjectID
subjectName
goal
numberOfCredit
description
«PK»
+ PK_Subject()

Prerequisite
«column»
*PK prerequisiteID
* subjectID
prerequisiteSubjectID
level
description
«PK»
+ PK_Prerequisite()

Lecturer
«column»
*PK lecturerID
lecturerName
educationalBackground
phone
email
address
«PK»
+ PK_Lecturer()

63

64



CourseRegistration
«column»
*PK courseID
*PK studentID
registeredDate
«PK»
+ PK_CourseRegistration(,)

Course
«column»
*PK courseID
* subjectID
description
startDate
endDate
location
lecturerID
«PK»
+ PK_Course()

Schedule
«column»
*PK scheduleID
day
teachingPeriod
courseID
«PK»
+ PK_Schedule()

StudyHistory
«column»
*PK historyNo
studentID
studentName
result
pass
courseID
«PK»
+ PK_StudyHistory()

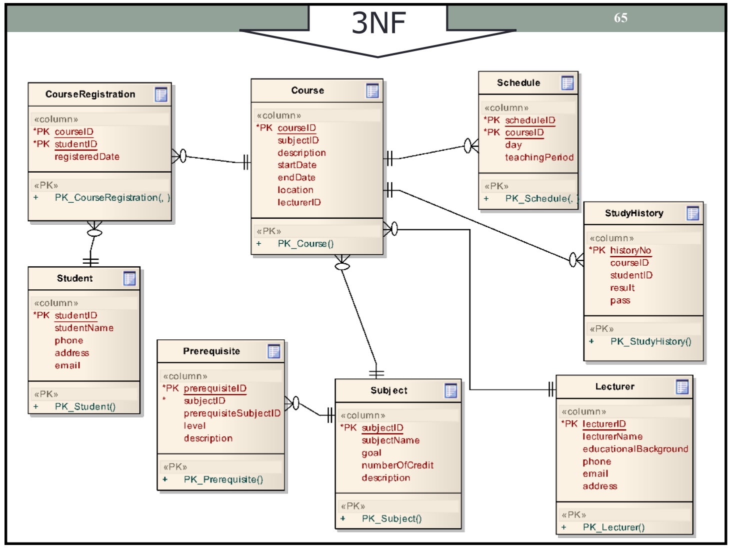
Student
«column»
*PK studentID
studentName
phone
address
email
«PK»
+ PK_Student()

Prerequisite
«column»
*PK subjectID
*PK prerequisiteID
level
description
«PK»
+ PK_Prerequisite(,)

Subject
«column»
*PK subjectID
subjectName
goal
numberOfCredit
description
«PK»
+ PK_Subject()

Lecturer
«column»
*PK lecturerID
lecturerName
educationalBackground
phone
email
address
«PK»
+ PK_Lecturer()

64



65

3NF

CourseRegistration
«column»
*PK courseID
*PK studentID
registeredDate
«PK»
+ PK_CourseRegistration(,)

Course
«column»
*PK courseID
subjectID
description
startDate
endDate
location
lecturerID
«PK»
+ PK_Course()

Schedule
«column»
*PK scheduleID
*PK courseID
day
teachingPeriod
«PK»
+ PK_Schedule(,)

StudyHistory
«column»
*PK historyNo
courseID
studentID
result
pass
«PK»
+ PK_StudyHistory()

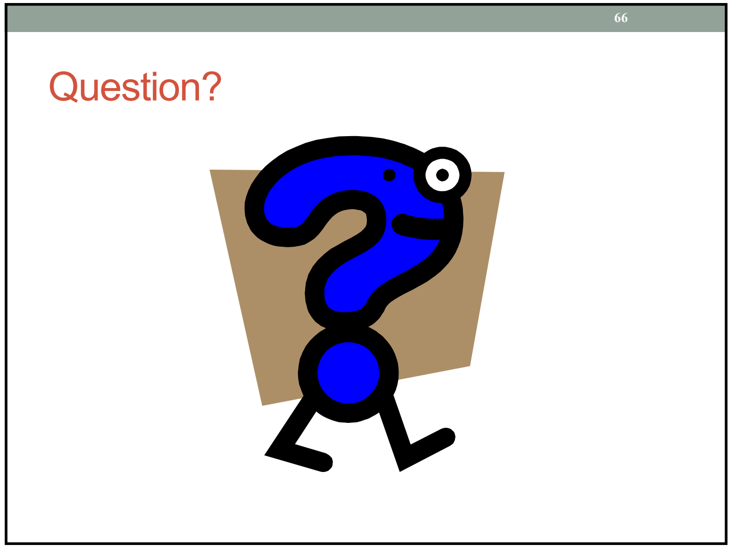
Student
«column»
*PK studentID
studentName
phone
address
email
«PK»
+ PK_Student()

Prerequisite
«column»
*PK prerequisiteID
subjectID
* prerequisiteSubjectID
level
description
«PK»
+ PK_Prerequisite()

Subject
«column»
*PK subjectID
subjectName
goal
numberOfCredit
description
«PK»
+ PK_Subject()

Lecturer
«column»
*PK lecturerID
lecturerName
educationalBackground
phone
email
address
«PK»
+ PK_Lecturer()

65



66

Question?

66