INFORMATION SYSTEM DEPARTMENT
INFORMATION TECHNOLOGY FACULTY – HCM
UNIVERSITY OF SCIENCE

ADVANCED DATABASE

Chapter 07

PHYSICAL DESIGN

Dr. VU Thi My Hang

KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

fit@hcmus

Outline

- Physical Database Design
- Define Data Storage Organization
- Define Security Measures
- Determine Performance Measures

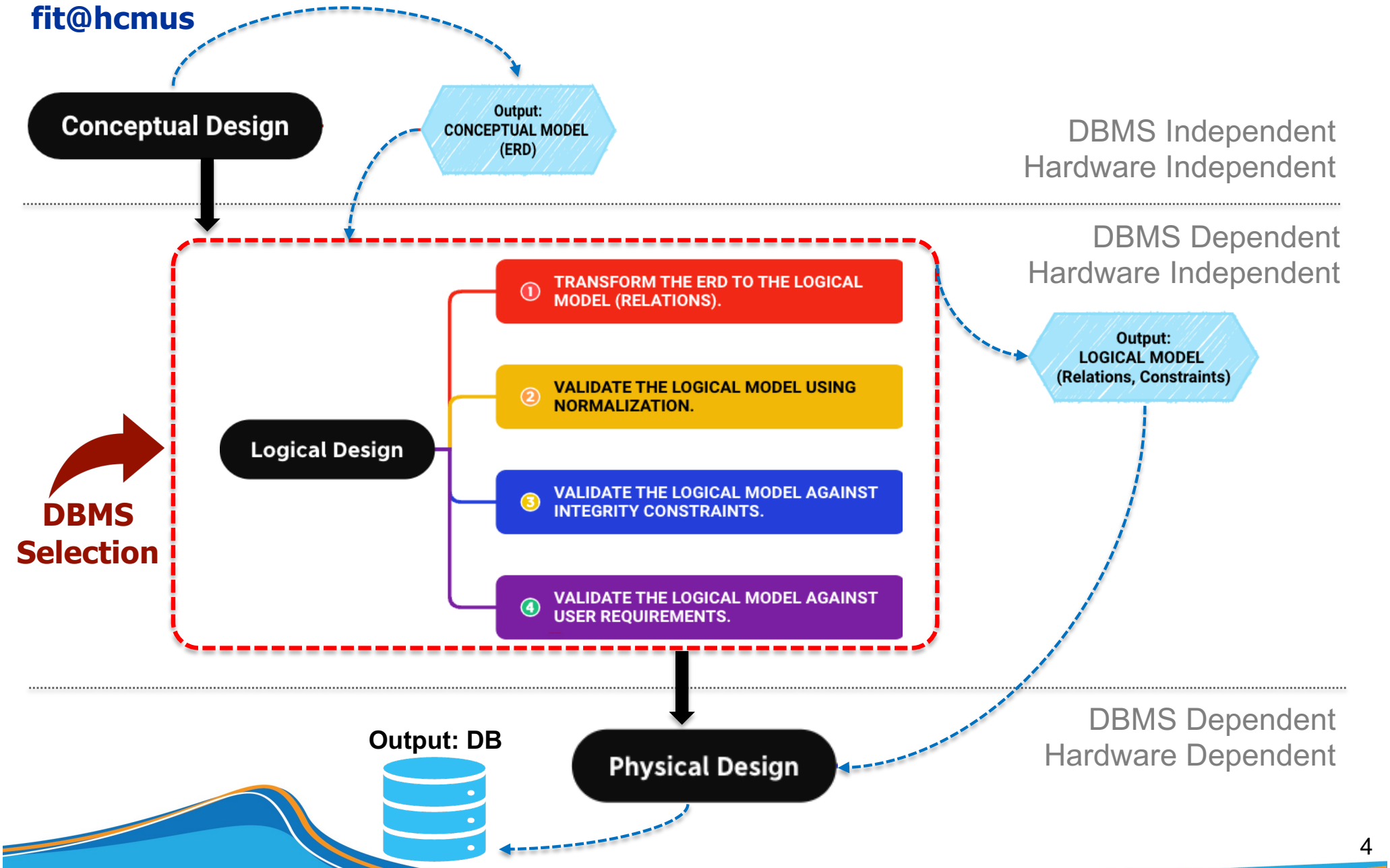
cdio 4.0

fit@hcmus

Outline

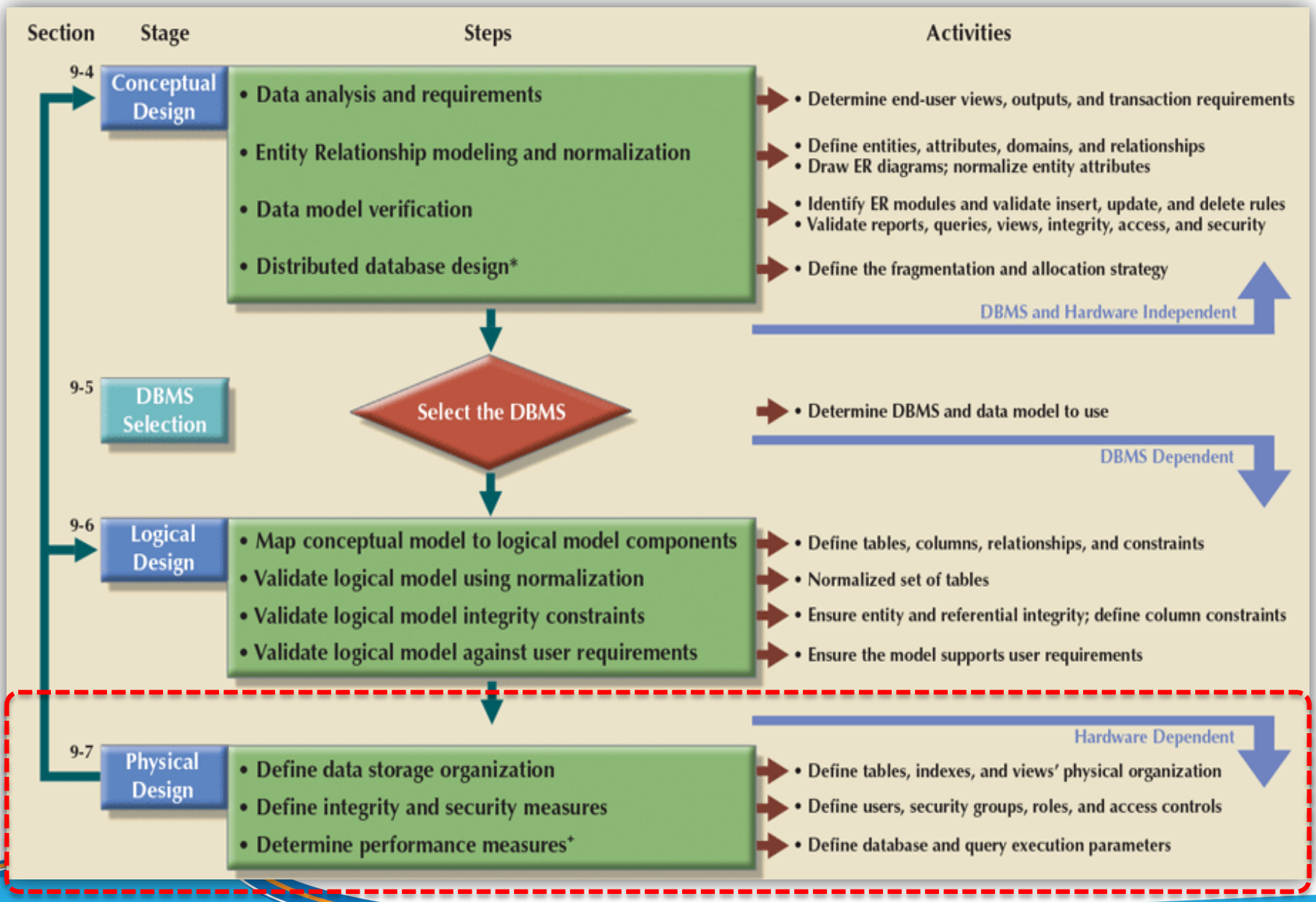
- Physical Database Design
- Define Data Storage Organization
- Define Security Measures
- Determine Performance Measures

Recall: Database Design Process



Recall: Database Design Process

fit@hcmus



(Hoffer et al., 2019)

Physical Design

- **Definition**: Process of transforming the logical model into the **technical specifications** for storing and retrieving data.
- **Goal:** Create a design for storing data that will provide adequate performance and ensure database integrity, security, and recoverability.
- **Output**: physical record descriptions (tables), file organizations, indexes, views, query optimization.

(Coronel and Morris, 2015; Hoffer et al., 2019)

Main Steps

PHYSICAL DESIGN STEPS	
STEP	ACTIVITY
1	Define data storage organization. Tables, Indexes, Views, Data Files
2	Define integrity and security measures. Users, Roles, Access Controls
3	Determine performance measurements. Query Optimization

cdio 4.0

fit@hcmus

Outline

- Physical Database Design
- Define Data Storage Organization
  - Step 1.1 Translate the logical model into tables
  - Step 1.2 Design indexes and types of indexes
  - Step 1.3 Design views and types of views
  - Step 1.4 Design (physical) database file organizations
- Define Security Measures
- Determine Performance Measures

cdio 4.0

fit@hcmus

Outline

- Physical Database Design
- Define Data Storage Organization
  - Step 1.1 Translate the logical model into tables
  - Step 1.2 Design indexes and types of indexes
  - Step 1.3 Design views and types of views
  - Step 1.4 Design (physical) database file organizations
- Define Security Measures
- Determine Performance Measures

Main Tasks for Step 1.1

Translate the logical model into tables

- Design **fields** (i.e., simples attributes of a table)
- Design the representation for **derived data**
- Design the remaining (business) **rules**

(Coronel and Morris, 2015)

Design Fields

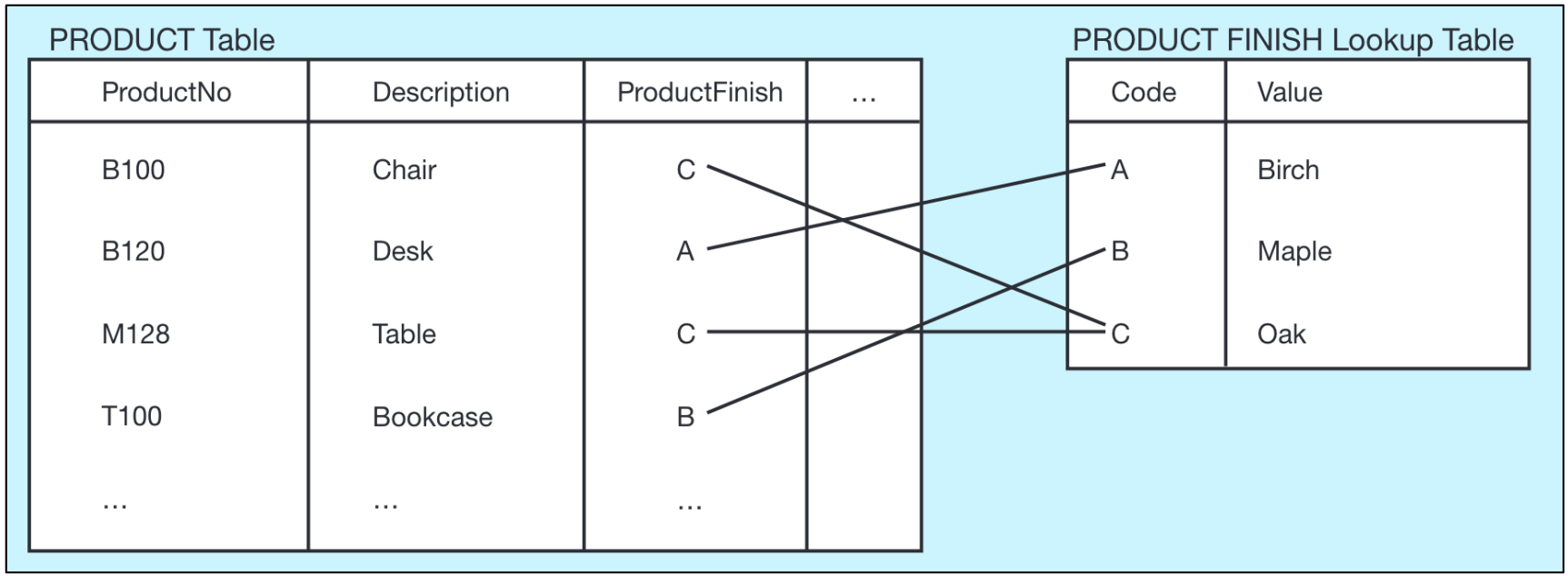
(Simple Attributes)

- Choosing **data types** respecting the following criteria:
  - Represent all possible values.
  - Improve data integrity.
  - Support all data manipulations.
  - Minimize storage space.

(Coronel and Morris, 2015)

Design Fields (cont.)

- **Coding** techniques for translating a field with a limited number of possible values into a code that requires less space.



(Coronel and Morris, 2015)

Design Fields (cont.)

- Controlling **data integrity**:
  - Null/not null constraints.
  - Entity constraints: primary key, unique.
  - Domain constraints: default, check.
  - Referential constraints: foreign key.

```
CREATE TABLE GIAOVIEN (
        MAGV        CHAR(9) PRIMARY KEY,
        HOTEN       NVARCHAR(50) NOT NULL,
        LUONG       INT DEFAULT (1000),
        PHAI        CHAR(3) CHECK (PHAI IN ('Nam', 'Nu')),
        NGAYSINH    DATETIME,
        SONHA       NVARCHAR(10),
        DUONG       NVARCHAR(50),
        QUAN        NVARCHAR(50),
        THANHPHO    NVARCHAR(50),
        GVQLCM      CHAR(9),
        MABM        CHAR(9)
)
```

(Coronel and Morris, 2015)

Design Fields (cont.)

- Handling **missing data**:
  - Substitute an estimate of the missing value.
  - Track missing data so that special reports and other system elements could resolve unknown values quickly.
  - Perform sensitivity testing so that missing data are ignored unless knowing a value might significantly change results.

(Coronel and Morris, 2015)

Design Fields (cont.)

The physical design of the Branch table using an extended DBDL.

```
domain Branch_Numbers       fixed length character string length 4
domain Street_Names         variable length character string maximum length 30
domain City_Names           variable length character string maximum length 20
domain State_Codes          fixed length character string length 2
domain Zip_Codes            fixed length character string length 5
domain Staff_Numbers        fixed length character string length 5

Branch( branchNo      Branch_Numbers    NOT NULL,
        street        Street_Names      NOT NULL,
        city          City_Names        NOT NULL,
        state         State_Names       NOT NULL,
        zipCode       Zip_Codes         NOT NULL,
        mgrStaffNo    Staff_Numbers     NOT NULL)
        Primary Key branchNo
        Alternate Key zipCode
        Foreign Key mgrStaffNo References Staff(staffNo) ON UPDATE CASCADE ON DELETE NO ACTION
```

(Coronel and Morris, 2015)

Design Fields (cont.)

```
CREATE TABLE Customer_T
        (CustomerID                 NUMBER(11,0)      NOT NULL,
        CustomerName                VARCHAR2(25)      NOT NULL,
        CustomerAddress             VARCHAR2(30),
        CustomerCity                VARCHAR2(20),
        CustomerState               CHAR(2),
        CustomerPostalCode          VARCHAR2(9),
CONSTRAINT Customer_PK PRIMARY KEY (CustomerID));

CREATE TABLE Order_T
        (OrderID                    NUMBER(11,0)      NOT NULL,
        OrderDate                   DATE DEFAULT SYSDATE,
        CustomerID                  NUMBER(11,0),
CONSTRAINT Order_PK PRIMARY KEY (OrderID),
CONSTRAINT Order_FK FOREIGN KEY (CustomerID) REFERENCES Customer_T (CustomerID));
```


(Coronel and Morris, 2015)

cdio 4.0

fit@hcmus

Design Representation for Derived Attributes

Advantages and Disadvantages of Storing Derived Attributes		
	Derived Attribute: Stored	**Derived Attribute: Not Stored**
Pros	• CPU processing saving • Data access time saving • Data value is readily available • Keep track of historical data	• Storage space saving • Computation/Calculation always yields current values
Cons	• Must ensure derived value is current, especially if any values used in the calculation change	• CPU processing cycles increasing • Data access time increasing • Requires complex queries

(Coronel and Morris, 2015)

fit@hcmus

Design Representation for Derived Attributes (cont.)

- The **design** of how to represent derived data and the reasons for the selection **should be documented**.
  - List of derived data?
  - How to calculate derived data?
  - Which approaches may be applied?
  - Reasons for your choice.

(Coronel and Morris, 2015)

Design Remaining Rules

- Designing the remaining (business) rules for the target DBMS using **stored procedure**, **triggers** and **other** mechanisms.

```
CREATE TRIGGER TR_SLNha ON  RENTAL FOR INSERT, UPDATE
AS
BEGIN
   DECLARE X int
   SELECT X = COUNT(R.propertyNo)
   FROM INSERTED I, RENTAL R
   WHERE I.clientNo = R.clientNo
   GROUP BY R.clientNo
   IF X >= 1O THEN
   BEGIN
            RAISERROR ('Không thuê quá 10 căn nhà', 0,1)
            ROLLBACK TRANS
   END
END
```


(Coronel and Morris, 2015)

Outline

- Physical Database Design
- Define Data Storage Organization
  - Step 1.1 Translate the logical model into tables
  - Step 1.2 Design indexes and types of indexes
  - Step 1.3 Design views and types of views
  - Step 1.4 Design (physical) database file organizations
- Define Security Measures
- Determine Performance Measures

fit@hcmus

Main Tasks for Step 1.2

Design indexes and types of indexes

- Analyze transactions from the user requirements.

- Choose indexes.

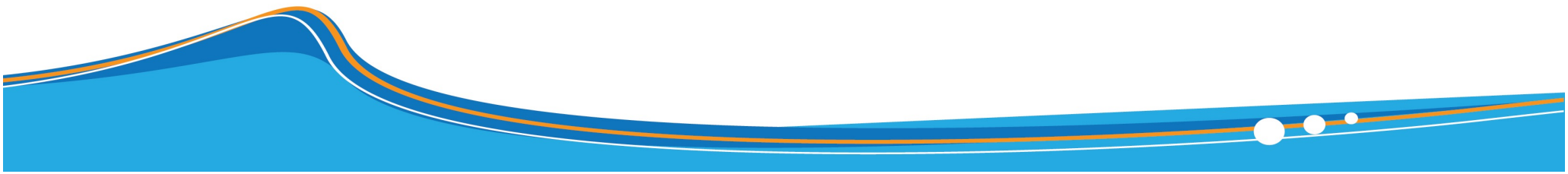
(Coronel and Morris, 2015)

cdio 4.0

fit@hcmus

Analyze Transactions

- Analyze transactions from the user requirements.
  - Input: list of transactions, data usage analysis
  - Output: analysis results involving:
    - Tables/Columns potentially heavily used
    - Tables/Columns accessed by the transactions and the types (R/W) of access such as insert (W), delete (W), update (W), or select (R).
  - Tools:
    - Transaction/table cross-reference matrix
    - Transaction usage maps
    - Transaction analysis form



fit@hcmus

Analyze Transactions (cont.)

Transaction/Table cross-reference matrix

(Ma trận tham chiếu truy vấn/quan hệ)

Transaction/Table (Truy vấn/Quan hệ)	Transaction 1 (Truy vấn 1)				Transaction 2 (Truy vấn 2)				...				Transaction n (Truy vấn n)			
	I	**R**	**U**	**D**	**I**	**R**	**U**	**D**	**I**	**R**	**U**	**D**	**I**	**R**	**U**	**D**
Table 1		X			X	X	X									
Table 2						X										
Table 3		X	X	X					X	X			X	X		
Table 4																
...																
....																
Table n (Quan hệ n)																

I = Insert; **R** = Read; **U** = Update, **D** = Delete

fit@hcmus

Analyze Transactions (cont.)

Transaction/Table cross-reference matrix

(Ma trận tham chiếu truy vấn/quan hệ)

Truy vấn 1: Xác định tổng số nhân viên có trong của mỗi chi nhánh tại TP HCM (tên chi nhánh, tổng số NV).

Truy vấn 2: Cho biết thông tin của các nhà cho thuê của chi nhánh tại TP HCM, kết quả sắp xếp theo giá thuê.

Truy vấn 3: Cho biết thông tin của các nhà cho thuê và tên nhân viên quản lý.

Truy vấn 4: Cho biết số nhà mà mỗi nhân viên tại chi nhánh ABC quản lý.

Truy vấn/Quan hệ	Truy vấn 1				Truy vấn 2				Truy vấn 3				Truy vấn 4			
	I	R	U	D	I	R	U	D	I	R	U	D	I	R	U	D
CHI NHÁNH		X				X								X		
NHÂN VIÊN		X								X				X		
NHÀ CHO THUÊ						X				X				X		

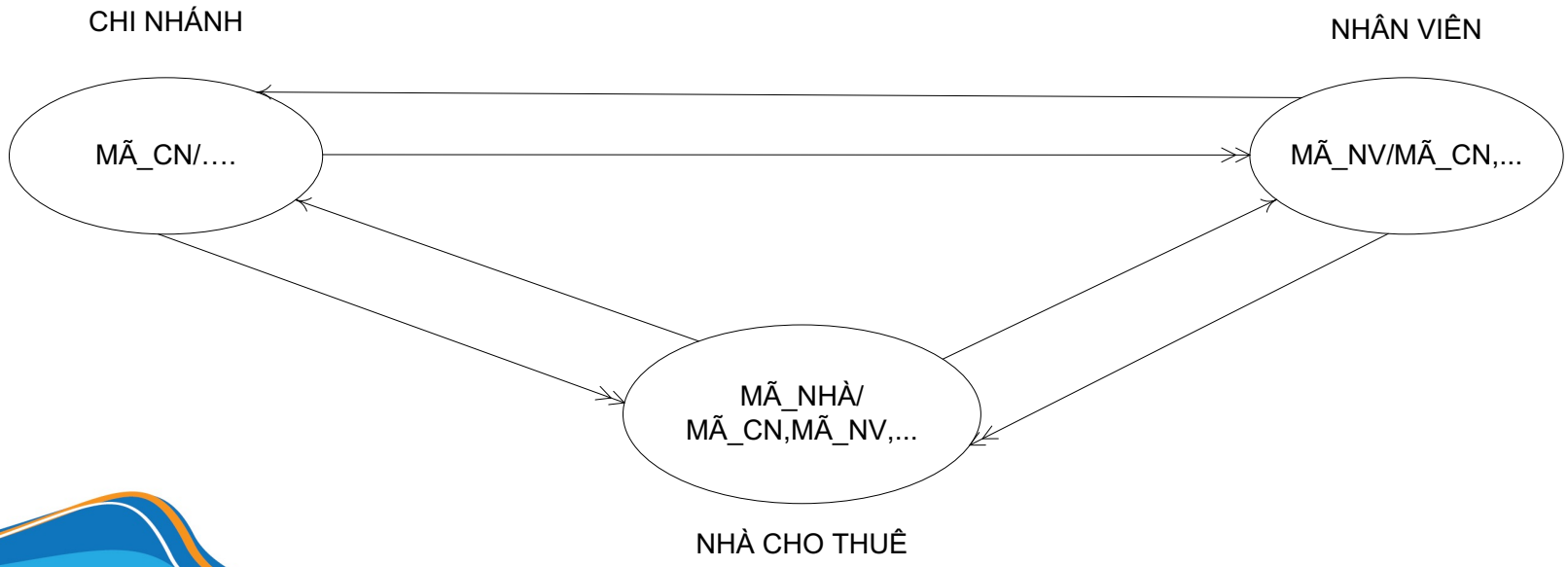
fit@hcmus

Analyze Transactions (cont.)

Transaction/Table cross-reference matrix

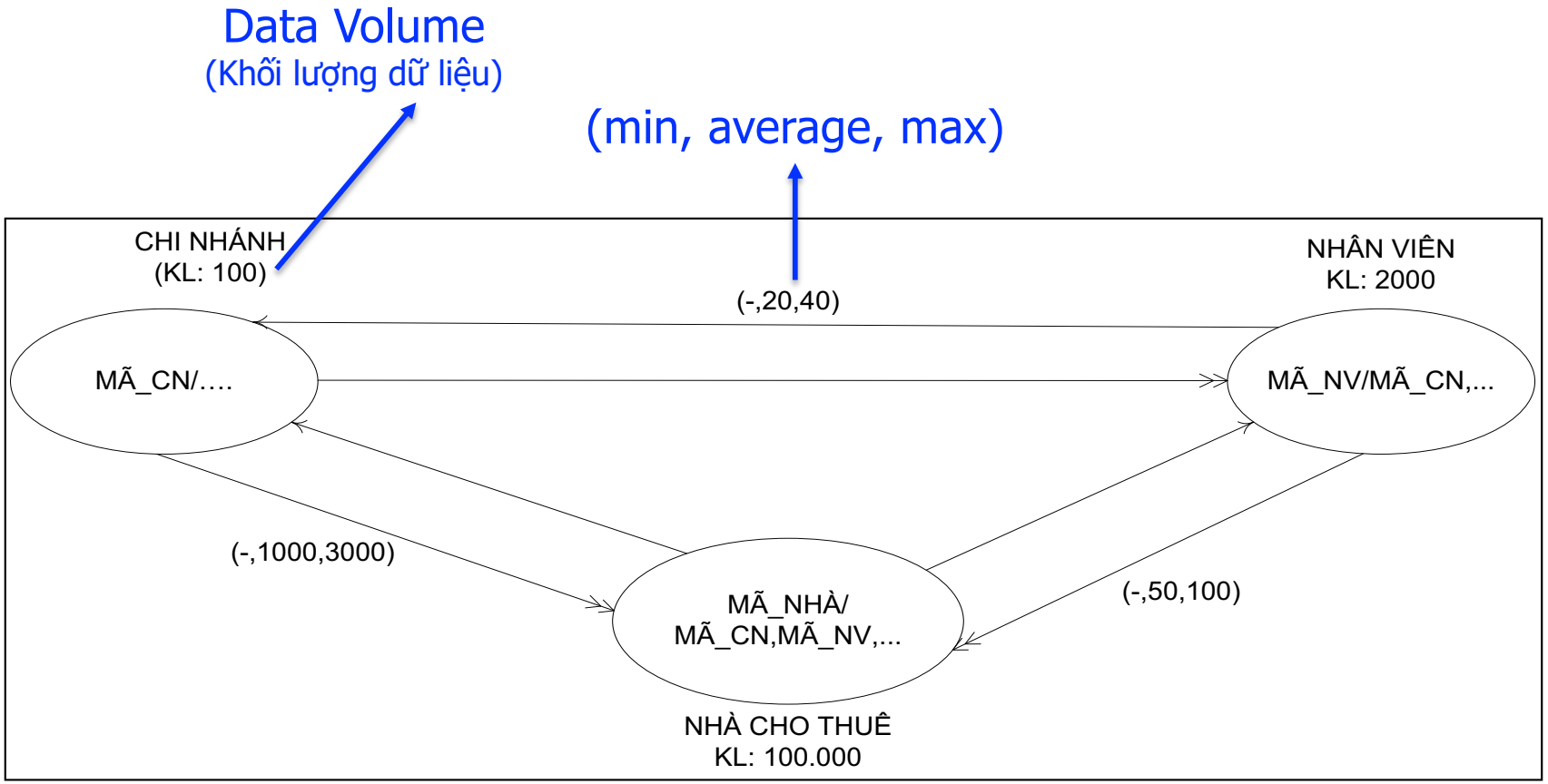
(Ma trận tham chiếu truy vấn/quan hệ)

Truy vấn/Quan hệ	Truy vấn 1				Truy vấn 2				Truy vấn 3				Truy vấn 4			
	I	R	U	D	I	R	U	D	I	R	U	D	I	R	U	D
CHI NHÁNH		X				X								X		
NHÂN VIÊN		X								X				X		
NHÀ CHO THUÊ						X				X				X		



Analyze Transactions (cont.)

Transaction usage maps



Access Frequencies (per hour, month, quarter ...)

Transaction Analysis Forms

fit@hcmus

Analyze Transactions (cont.)

Transaction analysis form

MẪU PHÂN TÍCH TRUY VẤN

16/02/2012

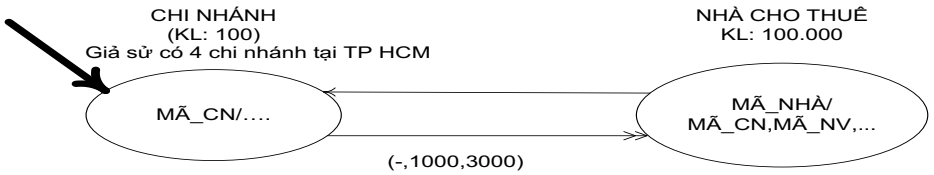
Truy vấn: (2) Cho biết thông tin của các nhà cho thuê của chi nhánh tại TP HCM, kết quả sắp xếp theo giá thuê

Tần suất TV: Trung bình: 50 lần/giờ,

Cao điểm: 100 lần/giờ (từ 17h-19h, Thứ 2 đến Thứ 7)

SELECT FROM CHINHANH c INNER JOIN NHACHOTHUE n ON c.MACN = c.MACN WHERE c.DIACHI LIKE '%TP HCM%' ORDER BY n.GIATHUE	Điều kiện: c.DIACHI LIKE '%TP HCM%'. Thuộc tính kết: c.MACN = c.MACN Thuộc tính sx: GIATHUE Thuộc tính gom nhóm: không Các hàm xây dựng: không Thuộc tính cập nhật: không

Đồ thị con đường truy xuất dữ liệu:



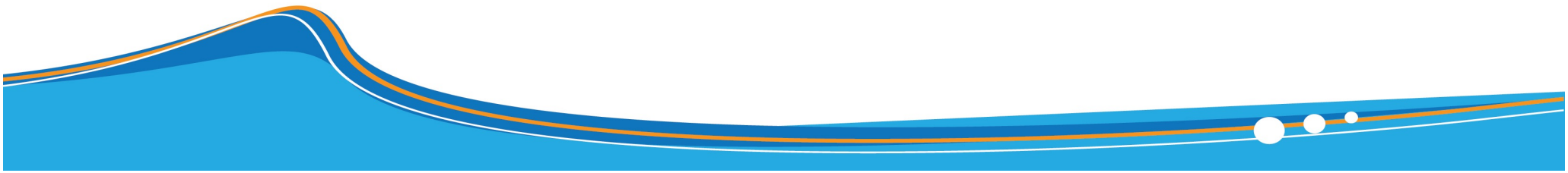
Truy xuất	Quan hệ	Loại truy xuất	Thông số về truy xuất		
			Trên truy vấn	Trung bình/h	Cao điểm/h
1	CHINHANH	R	100	5.000	10000
2	NHACHOTHUE	R	4.000 - 12000	200000 - 600000	400000 - 12000000
Tổng cộng truy xuất			4100 - 12100	205000 - 605000	410000 - 1210000

cdio 4.0

fit@hcmus

Choose Indexes

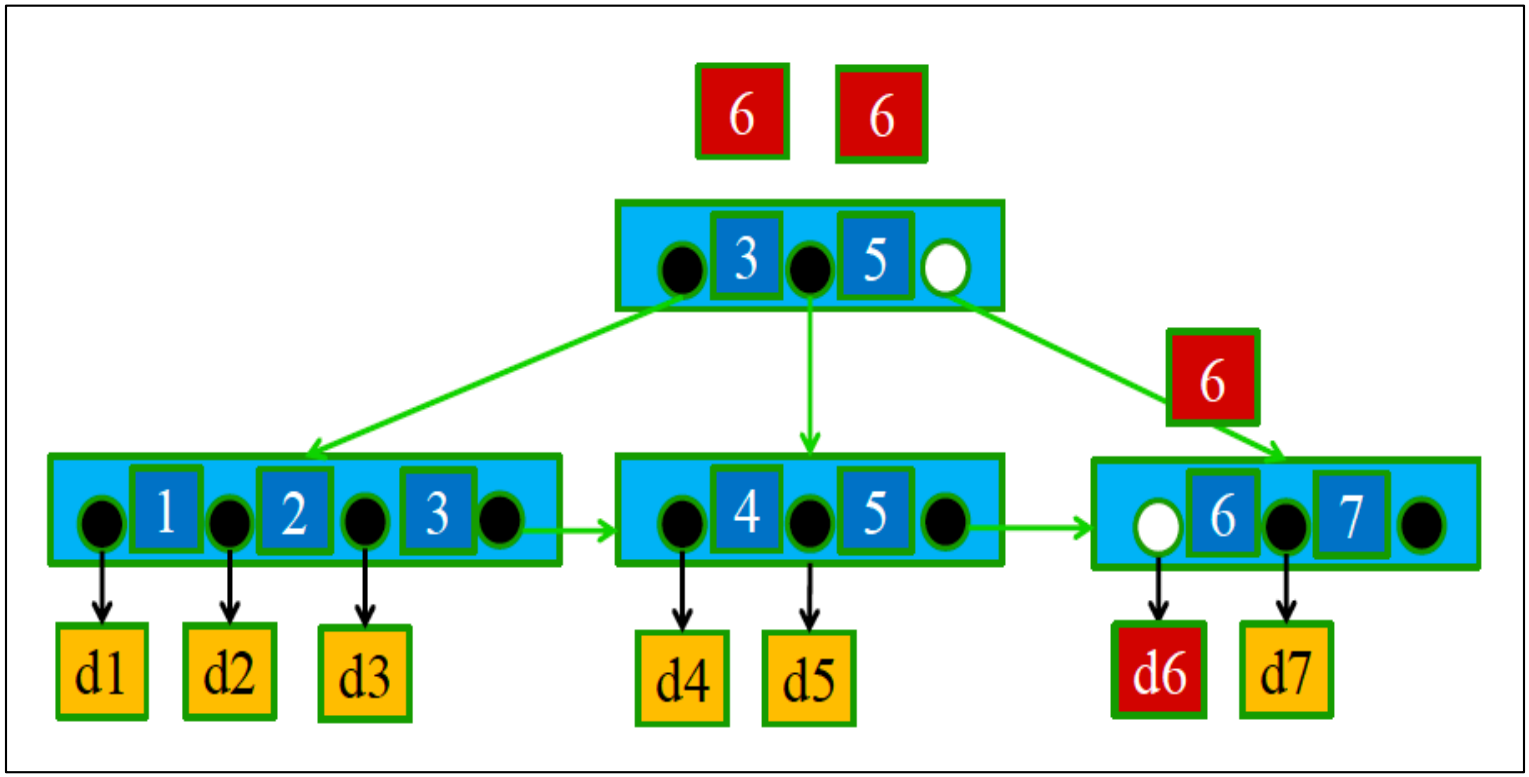
- Index – A separate table that contains organization of records for quick retrieval.
- Primary keys are automatically indexed.
- CREATE INDEX operation used to create index.
- Basic Types of Indexes
  - **B-tree** & **Hash** Index (Index methods)
  - **Non-clustered** & **Clustered** Index (Support of SQL Server)
  - **Single** & **Composite** Index (Nb. of key values)



Choose Indexes (cont.)

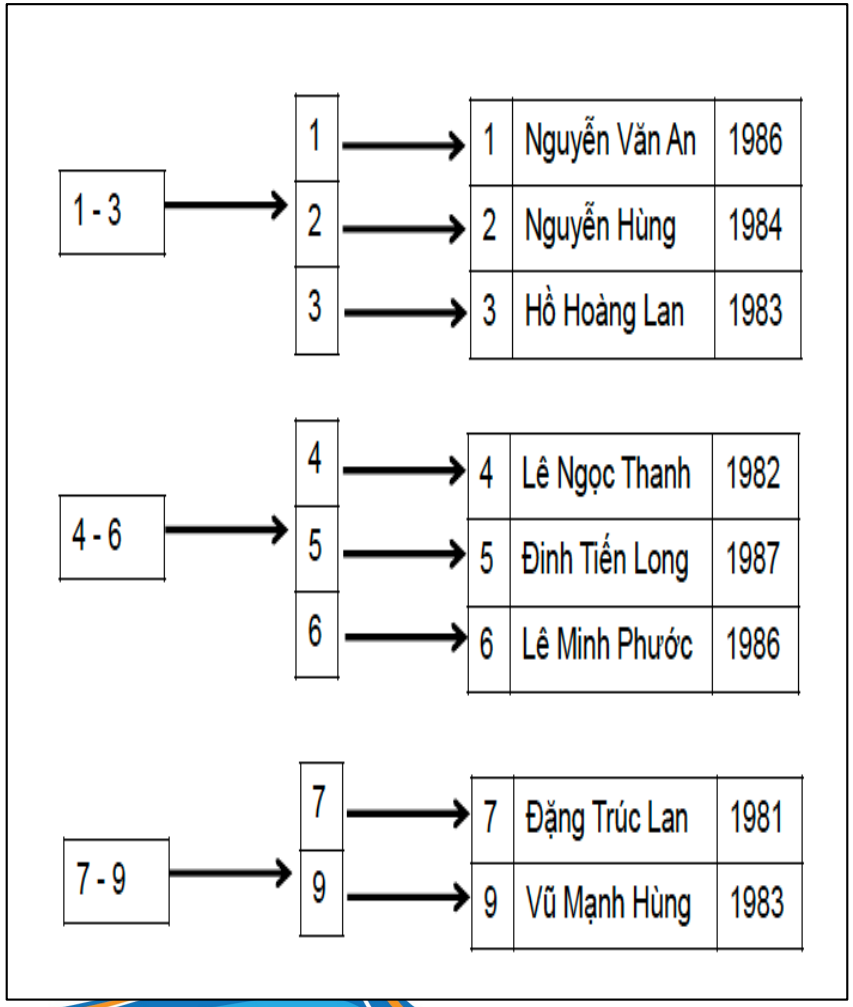
Example of B-Tree Index

(Index methods for primary keys in Oracle, SQL Server)

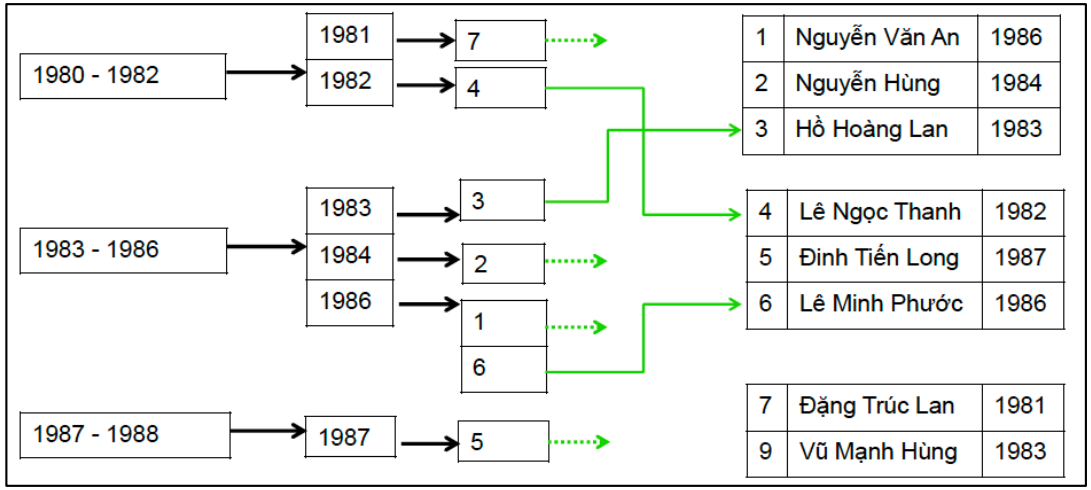


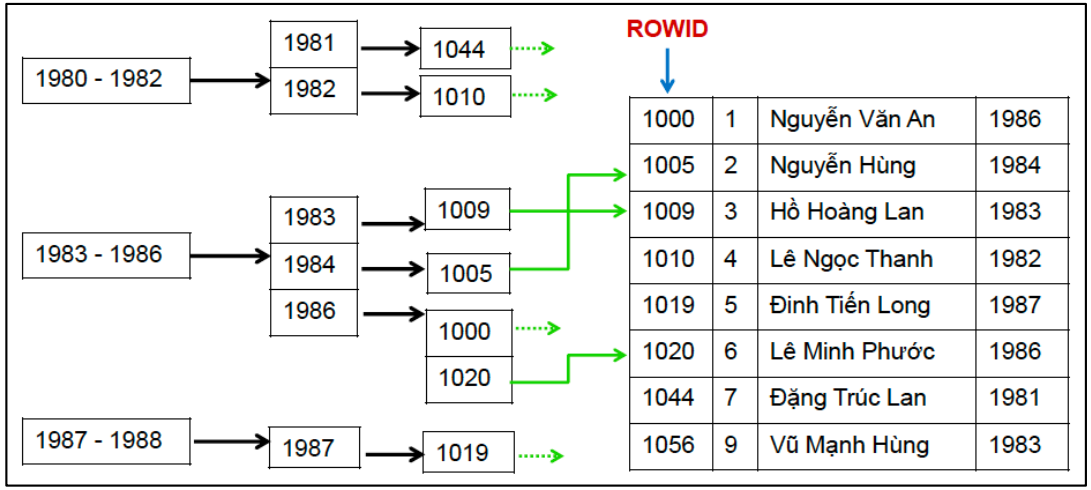
Choose Indexes (cont.)

Example of Clustered Index
(Determine the physical order of data rows)



Example of Non-Clustered Index
(Point to clustered indexes/row identifiers)



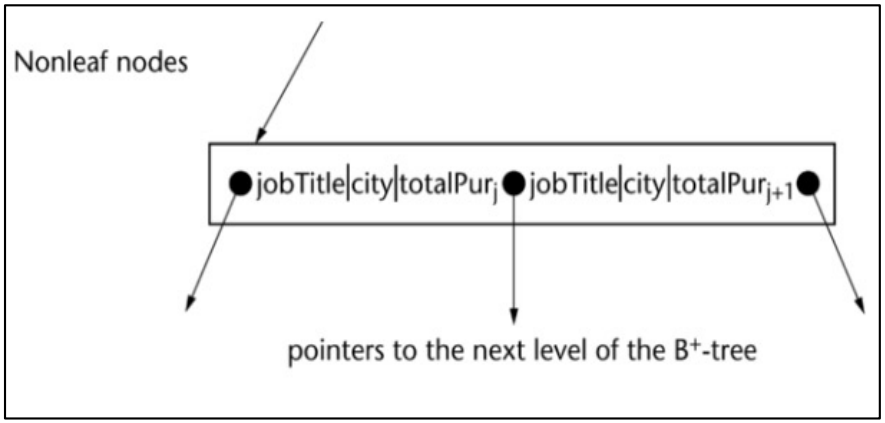


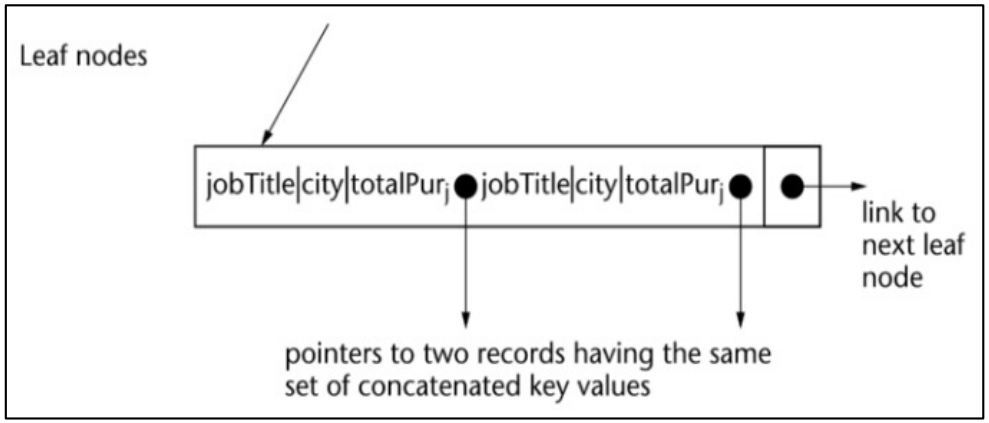
Choose Indexes (cont.)

Example of Composite Index

(Defined based on multiple columns)

```
SELECT empNo, empName, empAddress, jobTitle
    FROM employee
    WHERE jobTitle = 'database administrator'
    AND city = 'Los Angeles'
    AND totalPur < 500;
```





Choose Indexes (cont.)

Hints for index selection

1. Use indexes on larger tables.
2. Index the primary key unless it's done by DBMS.
3. Index search fields (fields frequently in WHERE clause), sorting fields (in ORDER BY clause) and categorizing fields (in GROUP BY clause).
4. Only index fields with more than 100 different values (not for the fields having less than 30 different values).
5. Consider the limitation on number of indexes of DBMS.
6. Consider first creating a compressed version when creating indexes on long-value fields.
7. Null values will not be referenced from an index.
8. Use indexes heavily for non-volatile databases; limit the use of indexes for volatile databases with index modification required.

Outline

- Physical Database Design
- Define Data Storage Organization
  - Step 1.1 Translate the logical model into tables
  - Step 1.2 Design indexes and types of indexes
  - Step 1.3 Design views and types of views
  - Step 1.4 Design (physical) database file organizations
- Define Security Measures
- Determine Performance Measures

SQL for Design Views

```
CREATE VIEW GV_HTTT AS
        SELECT GV.*
        FROM GIAOVIEN GV
        WHERE BM.MABM = 'HTTT'

CREATE VIEW THONGKE_BM AS
        SELECT BM.TENBM, COUNT(GV.MAGV) SL_GV,
                        SUM(GV.LUONG) TONG_LUONG
        FROM GIAOVIEN GV, BOMON BM
        WHERE GV.MABM = BM.MABM
        GROUP BY BM.MABM, BM.TENBM
```


(Coronel and Morris, 2015)

cdio 4.0

fit@hcmus

Outline

- Physical Database Design
- Define Data Storage Organization
  - Step 1.1 Translate the logical model into tables
  - Step 1.2 Design indexes and types of indexes
  - Step 1.3 Design views and types of views
  - Step 1.4 Design (physical) database file organizations
- Define Security Measures
- Determine Performance Measures

File Organization (cont.)

- Physical file: Named portion of physical storage allocated for the purpose of storing physical records.
- Tablespace: Named logical storage unit in which data from multiple tables/views/objects can be stored.
  - Segment: Table, index, or partition.
  - Extent: Contiguous section of disk space.
  - Data block: Smallest unit of storage.

(Coronel and Morris, 2015)

File Organization (cont.)

- Technique for physically arranging records of a file on secondary storage: sequential, indexed, and hashed.

	File Organization		
Factor	**Sequential**	**Indexed**	**Hashed**
Storage Space	No wasted space	No wasted space for data, but extra space for index	Extra space may be needed to allow for addition and deletion of records after initial set of records is loaded
Sequential Retrieval on Primary Key	Very fast	Moderately fast	Impractical, unless use hash index
Random Retrieval on Primary Key	Impractical	Moderately fast	Very fast
Multiple Key Retrieval	Possible, but requires scanning whole file	Very fast with multiple indexes	Not possible, unless use hash index
Deleting Records	Can create wasted space or require reorganizing	If space can be dynamically allocated, this is easy, but requires maintenance of indexes	Very easy
Adding New Records	Requires rewriting file	If space can be dynamically allocated, this is easy, but requires maintenance of indexes	Very easy, except multiple keys with same address require extra work
Updating Records	Usually requires rewriting file	Easy, but requires maintenance of indexes	Very easy

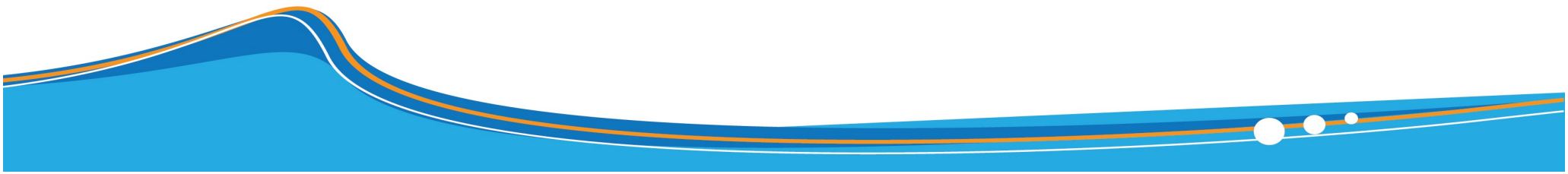
(Coronel and Morris, 2015)

Outline

- Physical Database Design
- Define Data Storage Organization
- Define Security Measures
- Determine Performance Measures

Security Measures

- Activities and measures that ensure the confidentiality, integrity, and availability of an information system.
  - Confidentiality deals with ensuring that data is protected against unauthorized access.
  - Integrity, within the data security framework, is concerned with keeping data consistent and free of errors or anomalies.
  - Availability refers to the accessibility of data whenever required by authorized users and for authorized purposes.



Outline

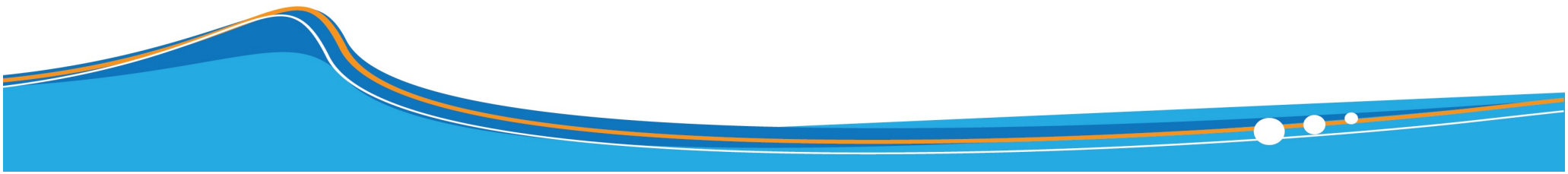
- Physical Database Design
- Define Data Storage Organization
- Define Security Measures
- Determine Performance Measures

cdio 4.0

fit@hcmus

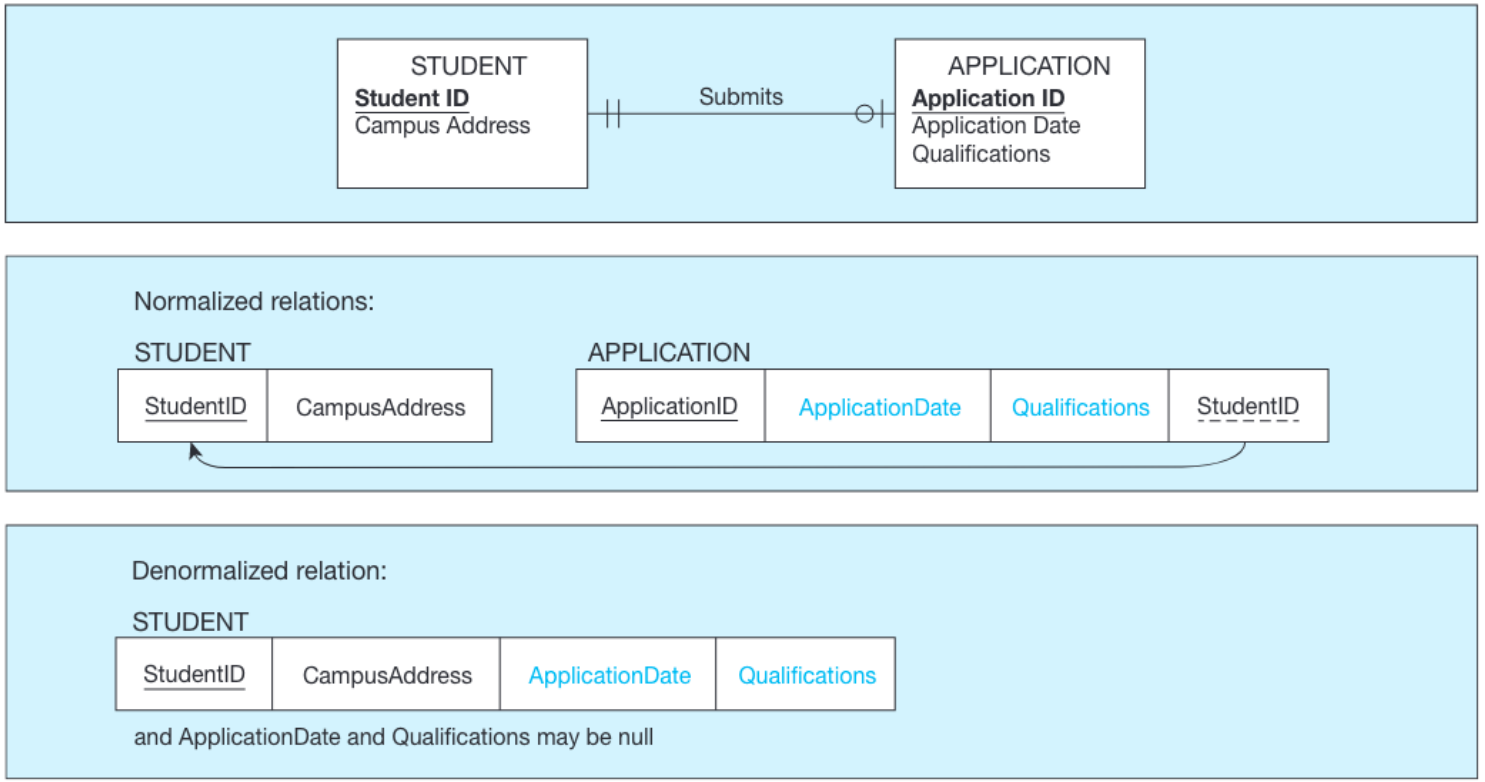
Denormalization

- Denormalization is the process of transforming normalized relations into nonnormalized physical record specifications for the performance purposes.
- Merging relations.
  - Two entities with a 1-1 relationship
  - A N-N relationship (associative entity) with nonkey attributes
  - Reference data exist in an entity on the one side of a one-to-many relationship, and this entity participates in no other database relationships.
- Partitioning data (horizontal & vertical partitions).



Denormalization

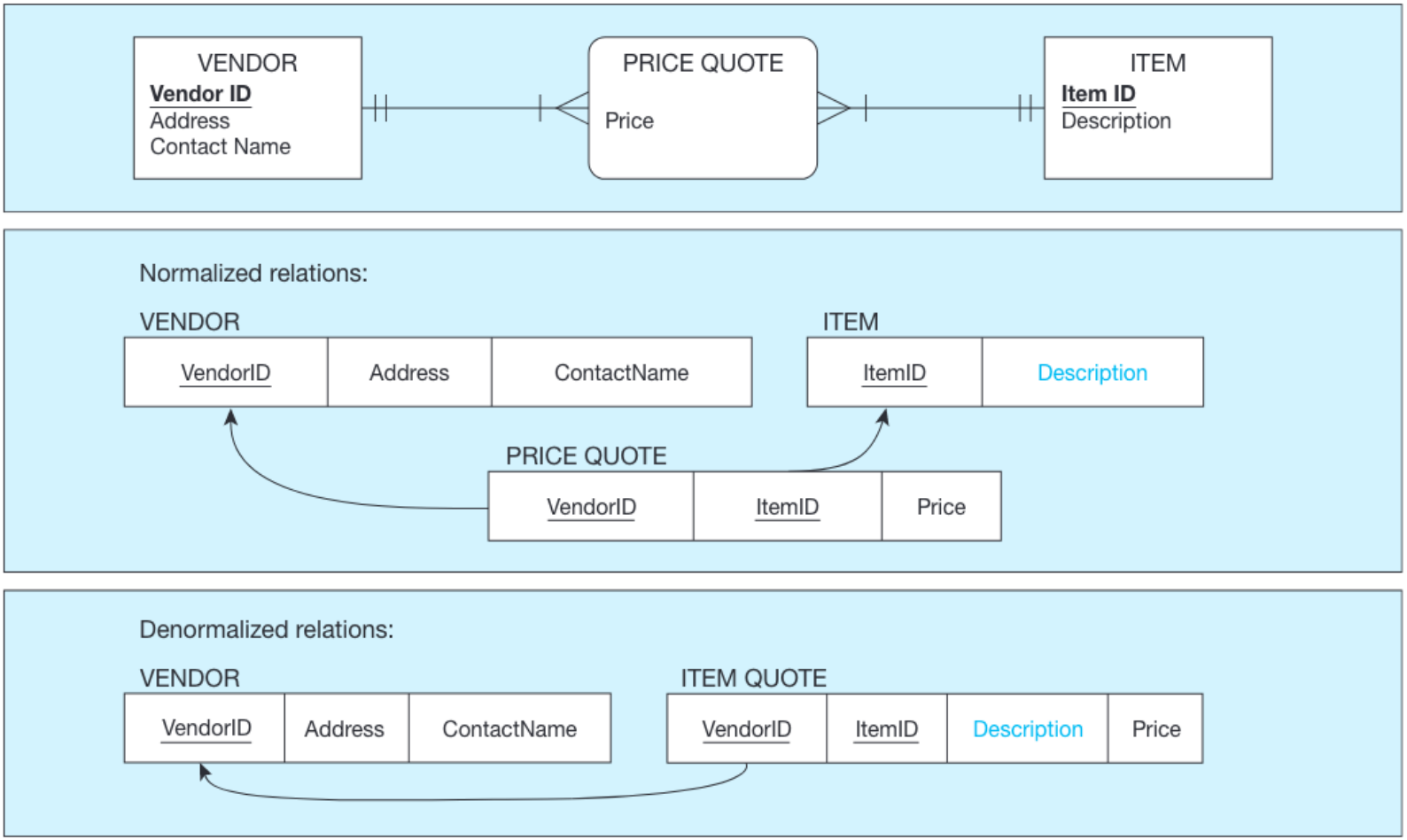
- Two entities with a 1-1 relationship



We assume that ApplicationID is not necessary when all fields are stored in one record, but this field can be included if it is required application data.

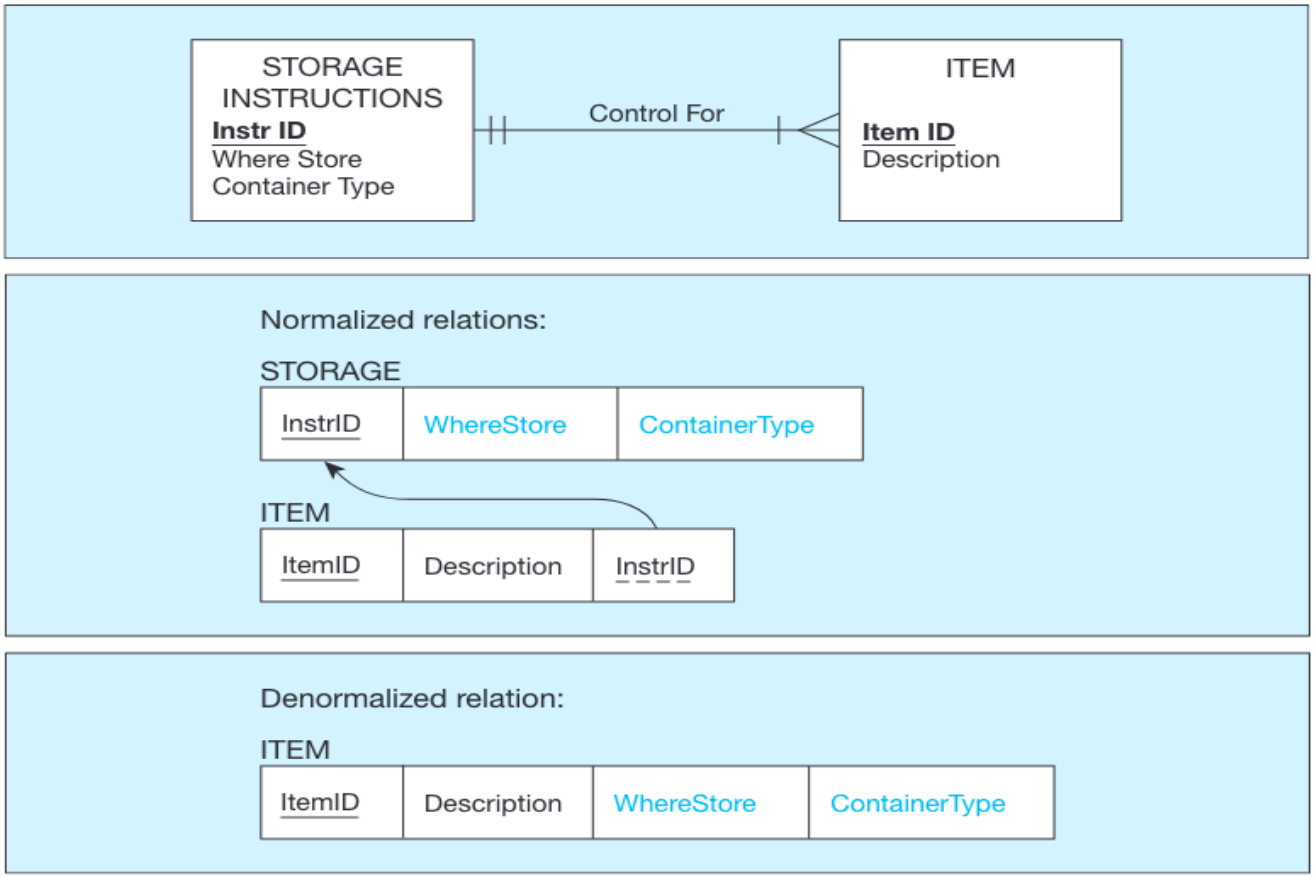
Denormalization

- A many-to-many relationship (associative entity) with nonkey attributes



Denormalization

- Reference data exist in an entity on the one side of a one-to-many relationship, and this entity participates in no other database relationships.



Partitioning

- Horizontal Partitioning: Distributing the rows of a table into several separate files
  - Useful for situations where different users need access to different rows.
  - Three types: Key Range Partitioning, Hash Partitioning, or Composite Partitioning.
- Vertical Partitioning: Distributing the columns of a table into several separate files
  - Useful for situations where different users need access to different columns.
  - The primary key must be repeated in each file.
- Combinations of Horizontal and Vertical

Partitions often correspond with User Schemas (user views)

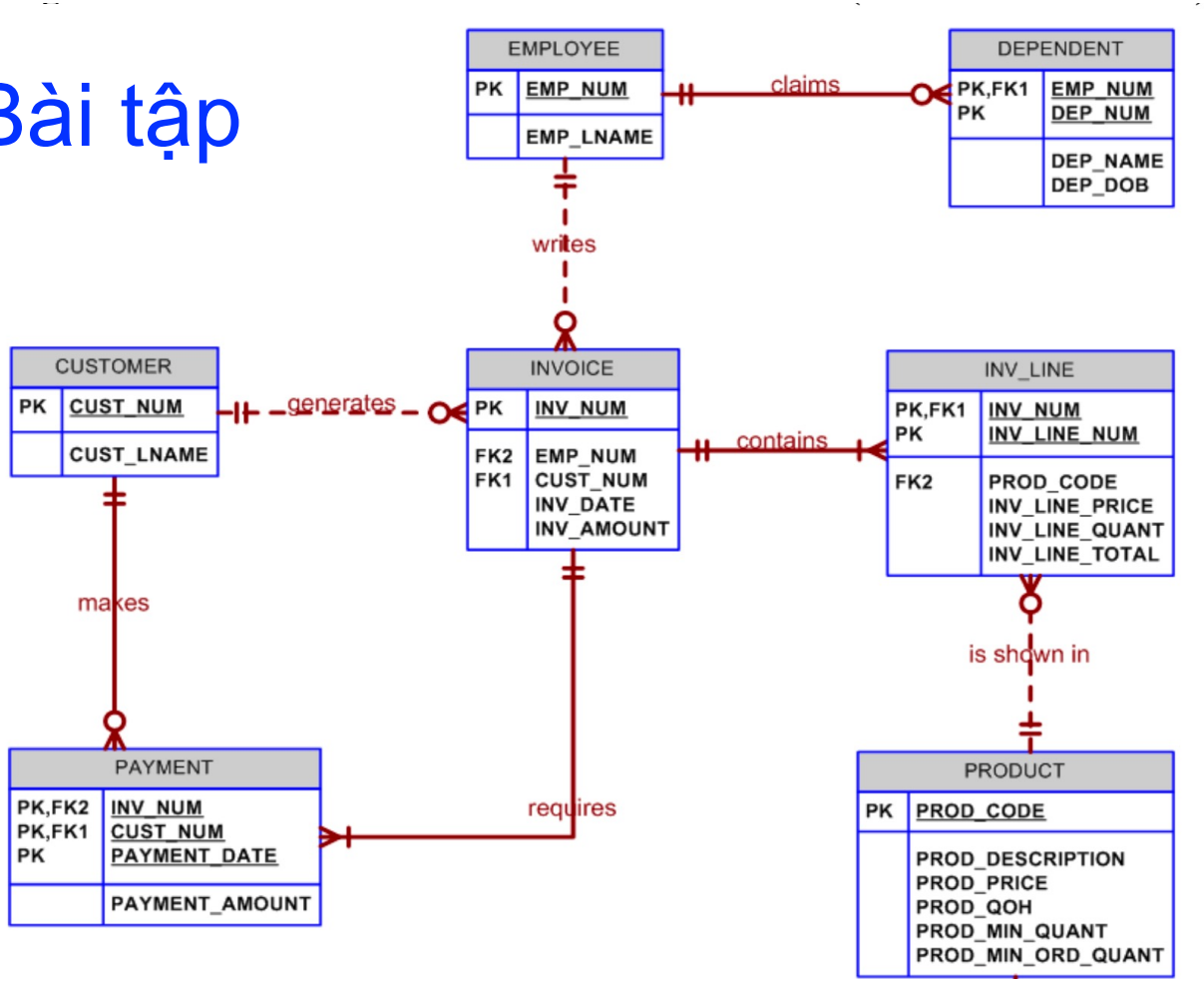
Partitioning

TABLE 8-2 Advantages and Disadvantages of Data Partitioning

Advantages of Partitioning
1. *Efficiency*: Data queried together are stored close to one another and separate from data not used together. Data maintenance is isolated in smaller partitions.
2. *Local optimization*: Each partition of data can be stored to optimize performance for its own use.
3. *Security*: Data not relevant to one group of users can be segregated from data those users are allowed to use.
4. *Recovery and uptime*: Smaller files take less time to back up and recover, and other files are still accessible if one file is damaged, so the effects of damage are isolated.
5. *Load balancing*: Files can be allocated to different storage areas (disks or other media), which minimizes contention for access to the same storage area or even allows for parallel access to the different areas.
Disadvantages of Partitioning
1. *Inconsistent access speed*: Different partitions may have different access speeds, thus confusing users. Also, when data must be combined across partitions, users may have to deal with significantly slower response times than in a nonpartitioned approach.
2. *Complexity*: Partitioning is usually not transparent to programmers, who will have to write more complex programs when combining data across partitions.
3. *Extra space and update time*: Data may be duplicated across the partitions, taking extra storage space compared to storing all the data in normalized files. Updates that affect data in multiple partitions can take more time than if one file were used.

THE END

Bài tập



Sinh viên tự xác định khối lượng dữ liệu và tần suất.

1. Xác định 4 yêu cầu truy xuất thường xuyên trong đó có 2 yêu cầu tìm kiếm và 2 yêu cầu cập nhật.
2. Xây dựng ma trận tham chiếu truy vấn/quan hệ từ phân tích tầm quan trọng của các bảng dữ liệu.
3. Phân tích truy vấn chi tiết cho 2 trong 4 y/c truy xuất ở câu 1 (1 tìm kiếm và 1 cập nhật) từ đó ra quyết định về việc triển khai index.