ĐẠI HỌC BÁCH KHOA

25 YEARS ANNIVERSARY

SOICT

ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

IT3090E - Databases

Chapter 2: Relational Databases

Muriel VISANI
murielv@soict.hust.edu.vn

Contents

- Chapter 1: Introduction
- **Chapter 2: Relational databases**
- Chapter 3: Relational algebra
- Chapter 4: Structured Query Language (SQL)
- Chapter 5: Database Design
- Chapter 6: Indexing
- Chapter 7: Query processing and optimization
- Chapter 8: Constraints, rules and triggers
- Chapter 9: Security
- *(Optional) Chapter 10: Transactions: concurrency and recovery*

Learning points

- 1. Relational data model
- 2. Constraints

Learning objectives

- Upon completion of this lesson, students will be able to:
  - Recall some basic concepts of relational data model.
  - Show some constraints of relational data model.

Keywords and descriptions

Keyword	Description
Data model	A set of concepts used to describe the structure of a database: data types, relationships, constraints, semantics…
Relation	Is thought of as a table of values, each row in the table represents a collection of related data values.
Key	An attribute or a set of attributes in the relation, which can identify a tuple uniquely.
Integrity constraints	Provide a way of ensuring that changes made to the database by authorized users do not result in a loss of data consistency.

1. Relational data model

- 1.1. Introduction
- 1.2. Database Basic concepts

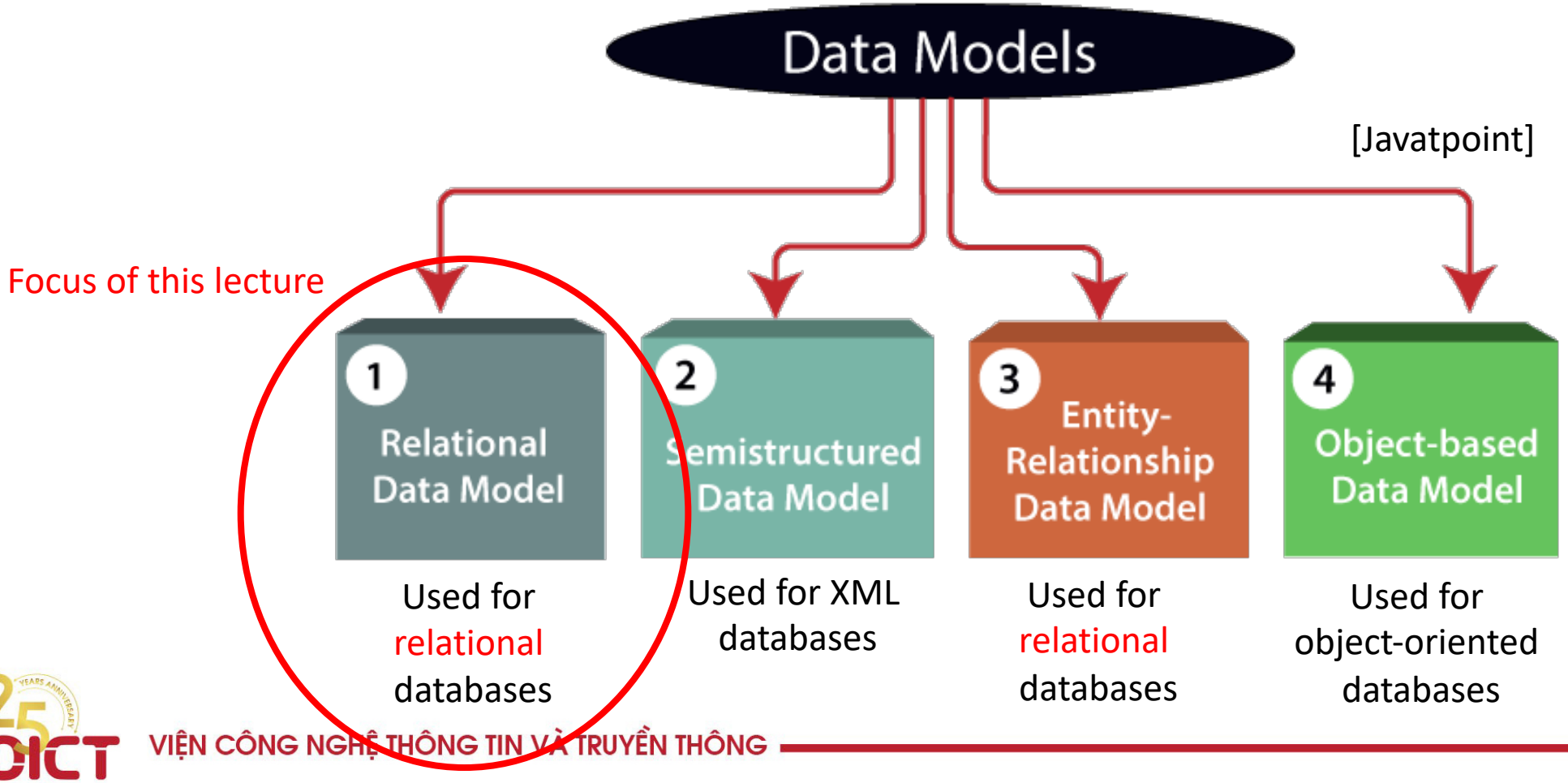
1.1. Introduction

- A data model is the modeling of the data description, data semantics, and consistency constraints of the data.
- It provides the conceptual tools for describing the design of a database at each level of data abstraction.

[Javatpoint]

1.1. Introduction

- There exist numerous possible data models, including:



1.1. Introduction

- Relational data model:
  - Is very simple model, was first introduced by Ted Codd of IBM Research in 1970
  - Used by most of commercial database systems
  - There exist a number of DBMS for efficient implementations
    - with high-level querying languages
  - Based on mathematical theory, close to file structure and data structure, there are three sets of terminology:

Relation	Table	File
Tuple	Row	Record
Attribute	Column	Field

1.2. Basic concepts

Relations	• are presented in the form of tables, which have rows (tuples) and columns (attributes). The lowercase letters Q, R, S denote relations.
Relation instance/state	• actual contents at given point in time. The lowercase letters q, r, s denote relation states.
Database	• a set of named relations (or tables).

1.2. Basic concepts

Tuple	• a single row of a table, which contains a single record for that relation. The letters t, u, v denote tuples.
Cardinality	• is the number of tuples in a relation.
Degree (arity)	• is the number of attributes in a relation.

1.2. Basic concepts

- An example

student(student_id, first_name, last_name, dob, gender, address, note, clazz_id)

Relation name

Attributes

student

student_id	first_name	last_name	dob	gender	address	note	clazz_id
1234	David	Beckham	12/21/1997	Male	London, UK		1
1238	Theresa	May	08/06/1998	Female	London, UK		1
1452	David	Cameron	07/06/1997	Male	Bangor, UK		1
1497	Tony	Blair	03/01/1999	Male	Bath, UK		2
1516	John	Major	03/01/1998	Male	Bradford		2
1542	Margaret	Thatcher	05/08/1997	Female	Cambridge		2

Tuples

Cardinality = 6

Degree = 8

1.2. Basic concepts

- Relational schema: structural description of relations in database.
  - A relation schema R of degree n, denoted by $\mathbf{R(A_1, A_2, \ldots, A_n)}$, is made up of a relation name R and a list of attributes $A_1, A_2, \ldots, A_n$
  - Each attribute A_i has values belong to domain D_i of A_i, denoted by $\mathbf{dom(A_i)}$
  - An n-tuple t in a relation r(R) is denoted by $\mathbf{t = <v_1, v_2, \ldots, v_n>}$, where v_i is the value corresponding to attribute A_i. Both $t[A_i]$ and $t.A_i$ (and sometimes t[i]) refer to the value v_i in t for attribute A_i

1.2. Basic concepts

- Notice that the uppercase letters Q, R, S denote relation names.
- A relation (or relation state) r of the relation schema $R(A_1, A_2, ..., A_n)$, also denoted by r(R), is a set of m n-tuples $r = \{t_1, t_2, ..., t_m\}$. Each n-tuple t is an ordered list of n-values $t = <v_1, v_2, ..., v_n>$, where each value v_i, $1 \le i \le n$, is
  - either an element of $dom(A_i)$,
  - or a NULL value.
- A relation (or relation state) r(R) is a mathematical relation of degree n on the domains $dom(A_1)$, $dom(A_2)$, ..., $dom(A_n)$, which is a subset of the Cartesian product of the domains that define R:

$$r(R) \subseteq (dom(A_1) \times dom(A_2) \times ... \times dom(A_n))$$

1.2. Basic concepts

- Example of Relational schema
  - We can use 4 relations (tables) to logically store the products, suppliers, orders, purchase_line
    - But other schemas are possible to store exactly the same information (see course on Database Design)

- **PRODUCTS:** product reference, designation, unit price, supplier number
- **SUPPLIERS:** supplier number, name, city
- **ORDER:** order number, order date, total amount
- **ORDER DETAILS:** order number, product reference, quantity, unit price

  - **Question**: why using 4 tables instead of 3?

1.2. Basic concepts

- Extract of an instance of the relation PRODUCTS

- **PRODUCTS:** product_reference, designation, unit_price, supplier_number

Product_reference	designation	Unit_price	Supplier_number
139	lamp	30.00	1623
248	Stove	75.00	1623
258	Backpack Northface	87.00	1369
416	Backpack Sakos	100.00	1359
426	Shoes	155.00	1369
765	tent	300.00	1502

1.2. Basic concepts

- Each table has a unique name:
  - Ex: PRODUCTS

- Each table consists of columns (or attributes)
  - The name of the attributes is unique **in the table**
  - But, two different attributes from two different relations might have the same name

⚠ Name of tables and attributes
  - Avoid accents, special characters and spaces (replace them with '_')!!!
  - On the other hand, the capital letters / small letters do not matter

- Each attribute has a particular domain / type (number, text, date, etc.)
  - Ex: designation: text, unit price: number (see after for real types)

1.2. Basic concepts

- Real-life example of a relational schema:

SHOPS (no_shop, city, name_manager)
CUSTOMERS (no_customer, name, country, city, type)
ITEMS(no_item, name, weight, color, qty_stock, BuyingPrice, SellingPrice, no_supplier#)
DELIVERY (no_delivery, date_del, no_customer#, no_shop#)
ORDERS (no_order, date, no_customer#, no_shop#)
SUPPLIERS (no_supplier, supplier_name)

DELIVERY_DETAILS (no_delivery#, no_item#, qty, no_order#)
ORDER_DETAILS (no_order#, no_item#, qty, no_delivery#, total_amount)

- **Questions:**
  - Why did I put an attribute called no_suppler inside the relation ITEMS?
  - Why did I put the attribute called no_order inside the relation DELIVERY_DETAILS, and not DELIVERY?
  - According to you, what do the # represent?

2. Constraints

- 2.1. Introduction
- 2.2. Types of constraints
- 2.3. An example

2.1. Introduction

- Every relation has some conditions that must hold for it to be a valid relation.
  - These conditions are called Relational Integrity Constraints.
- Provide a way of ensuring that changes made to the database (by authorized users) do not result in a loss of data consistency.

2.2. Types of constraints

- Key constraints
- Domain constraints
- Referential integrity constraints

2.2. Types of constraints

- A (unique) key is an attribute or a set of attributes in the relation, which can identify a tuple uniquely.
    - In a relation with a key, no two tuples can have identical values for key attribute(s)

2.2. Types of constraints

- Some types of key:
  - Candidate Key:
    - Unique key (K) such that no proper subset subset of K has the uniqueness property (irreducibility)
    - A candidate key can not have NULL values.
  - Primary Key (PK): Candidate key <u>chosen</u> to refer to tuples in a relation. Underlined in the relation schema.
  - **Example**:

STUDENT (Student_number, student_last_name, student_middle_name, student_first_name, student_email)

  - **Questions**:
    - Which attributes/set of attributes are keys?
    - Which attributes are candidate keys?
    - Which attribute(s) is/are primary key?

2.2. Types of constraints

- Some types of key:
  - Alternate Keys: Candidate keys that are not selected to be the primary key.
  - Foreign Key (FK):
    - Attribute, or set of attributes, within one relation that matches candidate key of some relation
    - Used to model relationships (associations) between relations (tables)
    - Each key attribute of foreign key has either its name italic, or a hashtag # following its name

ITEM
no_item … num_supplier #

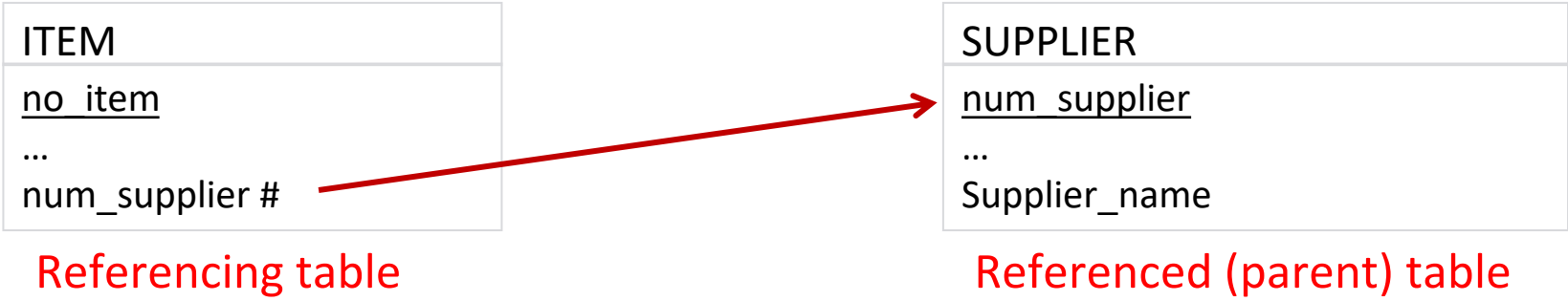
Referencing table

SUPPLIER
num_supplier … Supplier_name

Referenced table

2.2. Types of constraints

- Foreign Key (FK):
  - The # (or italics) appear in the referencing table
  - The referenced attribute in the referenced table - here SUPPLIER(num_supplier) – is **unique and NOT NULL** in the referenced table, but it does not need to be minimal (in practice we try, though…)
  - The referenced attribute in the referenced table - here SUPPLIER(num_supplier) – is often a **primary key** of the referenced table, but **not always**
  - In the referencing table, the foreign key attribute is **not necessarily NOT NULL** nor **unique**
    - I can still store an item instance in my DB, even if I don't know which supplier it comes from…
    - I can have multiple items from the same supplier



1.2. Basic concepts

- Real-life example of a relational schema:

SHOPS (no_shop, city, name_manager)
CUSTOMERS (no_customer, name, country, city, type)
ITEMS(no_item, name, weight, color, qty_stock, BuyingPrice, SellingPrice, no_supplier#)
DELIVERY (no_delivery, date_del, no_customer#, no_shop#)
ORDERS (no_order, date, no_customer#, no_shop#)
SUPPLIERS (no_supplier, supplier_name)

DELIVERY_DETAILS (no_delivery#, no_item#, qty, no_order#)
ORDER_DETAILS (no_order#, no_item#, qty, no_delivery#, total_amount)

- **Questions:**
  - Is there any alternate key for the table CUSTOMER?
  - Why do I need no_delivery AND no_item to as the primary key of DELIVERY_details?
  - Is it a problem that no_delivery and no_item are both foreign keys, and part of the primary key of the table DELIVERY_DETAILS?

2.2. Types of constraints

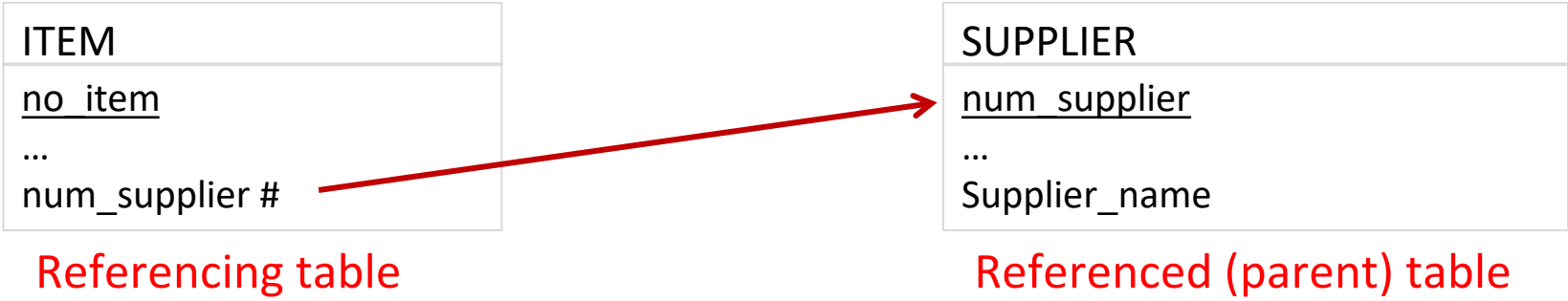
- Domain Constraints:
  - Attributes have specific values in real-world scenario. Every attribute is bound to have a specific range of values.
  - Within each tuple, the value of each attribute A must be an atomic value from the domain dom(A).
  - The data types associated with domains:
    - standard numeric data types for integers (short integer, integer, and long integer) and real numbers (float, double precision float).
    - Characters, Booleans, fixed-length strings, and variable-length strings, date, time, timestamp, and money, or other special data types.
    - a subrange of values from a data type.
    - an enumerated data type in which all possible values are explicitly listed.

2.2. Types of constraints

- Another domain constraints: NOT NULL
  - States that an attribute cannot have any NULL value within all the tuples of the relation
  - Null value:
    - Represents value for an attribute that is currently unknown or not applicable;
    - deals with incomplete or exceptional data;
    - ⚠ represents the absence of a value and is not the same as zero or spaces

2.2. Types of constraints

- Referential integrity Constraints:
  - Referential integrity constraints work on the concept of Foreign Keys. A foreign key is a key attribute of a relation that can be referred in other relation.
  - Referential integrity constraint states that if a relation refers to a key attribute of a referenced table, then that key element must exist in the referenced table



2.3. An example

student(**student_id**, first_name, last_name, dob, gender, addresse, note, *class_id#*)

subject(**subject_id**, name, credit, percentage_final_exam)

results(***student_id#, subject_id#, semester,*** midterm_score, final_score)

- **Questions:**
  - Why do I need semester in the PK of the table results?
  - By default, can the attributes student_id and subject_id be NOT NULL in the relation results?
  - In practice, should I set the attributes student_id and subject_id to be NOT NULL in the relation results?

2.3. An example

student

student_id	first_name	last_name	dob	gender	address	note	clazz_id
1234	David	Beckham	12/21/1997	Male	London, UK		1
1238	Theresa	May	08/06/1998	Female	London, UK		1
1452	David	Cameron	07/06/1997	Male	Bangor, UK		1
1497	Tony	Blair	03/01/1999	Male	Bath, UK		2
1516	John	Major	03/01/1998	Male	Bradford		2
1542	Margaret	Thatcher	05/08/1997	Female	Cambridge		2

subject

subject_id	name	credit	percentage_final_exam
IT3090	Databases	3	0.7
IT4843	Data integration	3	0.7
IT4868	Web mining	2	0.6
IT2000	Introduction to ICT	2	0.5
IT3020	Discrete Mathematics	2	0.7
IT3030	Computer Architectures	3	0.7

Foreign key

Foreign key

results

student_id	subject_id	semester	midterm_score	final_score
1234	IT3090	20171	7	8
1238	IT3090	20171	9	8
1452	IT3090	20171	6	6
1234	IT2000	20162	5	8
1234	IT3020	20171	8	9
1452	IT3030	20171	7	9
1238	IT3020	20162	7	7

Primary key

Summary

- Relational data model
  - Relations, relation instance/state, relation schema
  - Database, tuple
  - Cardinality, degree

- Constraints
  - Key constraints
  - Domain constraints
  - Referential integrity constraints

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

Thank you for your attention!

soict.hust.edu.vn/ fb.com/groups/soict

Questions