Unit II - Relational Data Model
Entity and Referential Integrity

Dr. GEETHA MARY A
ASSOCIATE PROFESSOR,
SCSE, VITU

Sources:
Pearson Education, Inc. 2011, Elmasri/Navathe, Fundamentals of Database Systems, Sixth Edition
McGraw Hill Education , 2010, Silberschatz, Korth and Sudarshan, Database System Concepts, Sixth edition

What is a Relation?

- A Relation is a 2-dimensional table of values (rows and columns)
- each row, or **tuple**, is a collection of related facts
- the degree of the relation is the number of attributes in the relation
- each column represents an **attribute**
- each row is an **instance** of the relation

What is a Relation (cont'd)?

- So, a relation is a big table of facts.
  - Each column contains the same attribute data with the same data type
  - Each row describes a real-world instance of the relation
- A Relational database contains one or more relations (or tables).

Schema vs. Instance

- the name of the relation and the set of attributes is called the **schema** (or the **intension**)

- the current values in the relation represent an **instance** (or **extension**) of the data

More formally……

- A *domain* **D** is a set of atomic values
  - local phone number – The set of 7-digit numbers
  - names – The set of names of persons
  - date of birth – Possible dates of birth for people
- A *relation schema* $\mathbf{R(A_1, A_2, ..., A_n)}$ is a:
  - relation name (**R**)
  - list of attributes ($\mathbf{A_1, A_2, ..., A_n}$)
  - each attribute $\mathbf{A_i}$ is the name of a role played by some domain **D** in the relation schema **R**

More formally (cont'd)

- a *relation* **r(R)** is a subset of
  $dom(A_1)$ X $dom(A_2)$ X … X $dom(A_n)$

- each element in a relation, called a *tuple*, is a collection of n values

Student (name, address, phone number)

Attribute

Tuple

Name	Address	Phone Number
Bob	Johnston St.	533-3333
Mary	Union St.	533-4444
Fred	Clarence St.	533-5555

DEFINITION SUMMARY

Informal Terms		Formal Terms
Table		Relation
Column		Attribute/Domain
Row (record)		Tuple
Values in a column		Domain
Table Definition		Schema of a Relation
Populated Table		Extension

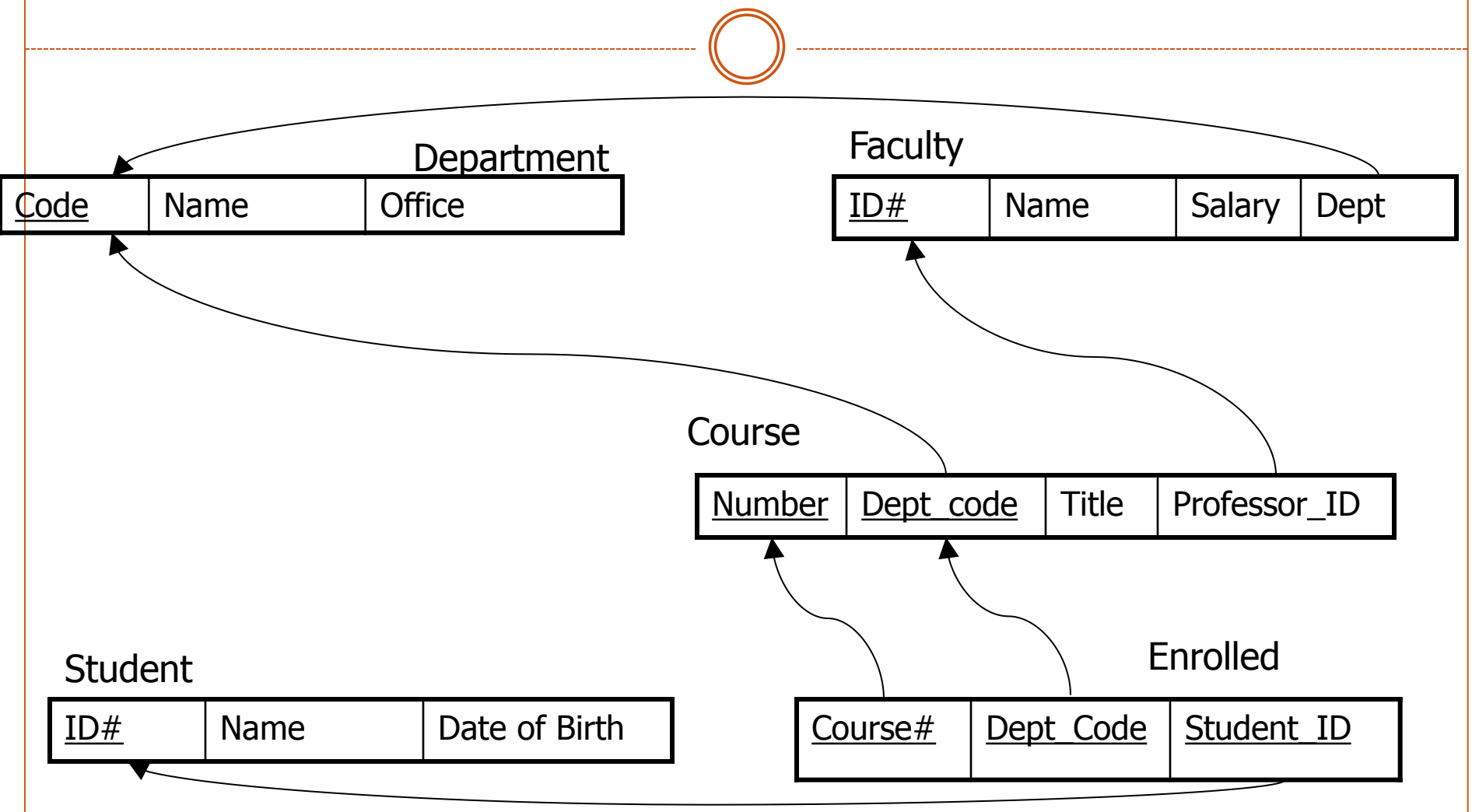
Characteristics of Relations

- tuples have no particular order
- ordering of attributes not important
- all values belonging to a particular attribute are from the same **domain**
- attributes are atomic
- attributes may have a *null value*

Operations on Relations

- Operations include **insert**, **delete**, **modify** and **retrieval**.

Example – Referential Integrity



Insert

- Provide a list of attribute values to be inserted (ie. A new tuple)

- Example

  insert values (554433, “Bob”, 25143.56, “ENGL”) into faculty

Insert (cont'd)

Inserts may violate constraints.

Key Constraint:

insert values (554433, "Bob", 25143.56, "ENGL")
into employee

(Will fail if the employee number "554433" is already in the table)

Entity Integrity Constraint:

insert values (NULL, "Bob", 25143.56, "ENGL")
into employee

(primary key cannot be NULL)

Insert (con't)

Referential Integrity Constraint:

insert values (554433, "Bob", 25143.56, "ENGL")

into employee

(Will fail if the "ENGL" is not a code for a department)

Delete

Faculty

ID#	**Name**	**Salary**	**Dept**
1234	Mary	2345.67	ENGL
2345	Jane	3246.87	HIST
3456	Fred	2876.32	COMP

delete the **faculty** tuples with name="Fred"

- Why is this not a good idea?

Delete (con't)

- The only constraint which can be violated is the referential integrity constraint (i.e. A tuple in another relation references the tuple that is slated for deletion).

  delete from Faculty where **name** = "Fred"

  (referenced by tuples in **Course**)

- Also, what if there are two people named "Fred"?

Modify

- Change the value for one or more attributes in a relation

  Example:

  modify SALARY of Faculty where ID# = 1234 to 30000

- Modifying a primary key is like deleting a tuple and adding a new one. (Same violations may apply).

Types of Constraints

- Domain constraints
- Key constraints
- Integrity constraints
  - Entity Integrity Constraint
  - Referential Integrity Constraint
  - Semantic Integrity Constraint

Domain Constraints

- The value of each attribute, A, must be an atomic value from the domain of A
- So, if an attribute is from the domain of a phone number, then the attribute must be a phone number.

Key constraints

- value of a key uniquely identifies a tuple in a relation
- a **superkey** *K* is subset of attributes of **R** such that:
  - no 2 tuples have same values for *K*
- Every relation has at least one superkey;

Keys (cont'd)

- A **key** is a minimal superkey; a superkey from which we cannot remove any attributes and still be able to uniquely identify tuples in a relation

- common keys – ID number, Social Insurance Number, etc.

Keys (cont'd)

- A relational schema may have more than one key
  - each key called a ***candidate key***
  - one designated as the ***primary key***

Integrity Constraints

- Integrity constraints are specified on a schema and hold for every instance of the schema
- **Entity integrity constraint**
  - no primary key value can be null
- **Referential integrity constraint**
  - if R_1 refers to R_2 then $t_1 \in r_1(R_1)$ must refer to an existing $t_2 \in r_2(R_2)$

Foreign Keys

- a ***foreign key*** in **R** is a set of attributes **FK** in **R** such that **FK** is a primary key of some other relation **R'**
- a foreign key is used to specify a referential integrity constraint

- Example?

Key examples

Department (code, name, phone)

Faculty (name, number, office, dept_code)

Course (name, number, dept_code)

Semantic Integrity Constraints

- Constraints on data values such as:
  - The salary of an employee must not exceed that of her supervisor.
  - The total of available seats must be > 0 in order for a reservation to be made.
  - A person's date of birth must be before the current date.