Module -2

Relational Model and E-R Modelling

Dr. T Joshva Devadas
Professor
Scope

Structure of Relational Databases

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

The *instructor* relation.

course_id	*title*	*dept_name*	*credits*
BIO-101	Intro. to Biology	Biology	4
BIO-301	Genetics	Biology	4
BIO-399	Computational Biology	Biology	3
CS-101	Intro. to Computer Science	Comp. Sci.	4
CS-190	Game Design	Comp. Sci.	4
CS-315	Robotics	Comp. Sci.	3
CS-319	Image Processing	Comp. Sci.	3
CS-347	Database System Concepts	Comp. Sci.	3
EE-181	Intro. to Digital Systems	Elec. Eng.	3
FIN-201	Investment Banking	Finance	3
HIS-351	World History	History	3
MU-199	Music Video Production	Music	3
PHY-101	Physical Principles	Physics	4

The *course* relation

- *student* (*ID*, *name*, *dept name*, *tot cred*)
- *advisor* (*s id*, *i id*)
- *takes* (*ID*, *course id*, *sec id*, *semester*, *year*, *grade*)
- *classroom* (*building*, *room number*, *capacity*)
- *time slot* (*time slot id*, *day*, *start time*, *end time*)

- Empid // unique & not null
- Ename // duplicates
- Emp aadhar // unique & Not null
- Emp sal // duplicates
- Emp contact // may have Null values
- Emp email // may have Null values

Super key

- A **superkey** is a set of one or more attributes that, taken collectively, allow us to identify uniquely a tuple in the relation.
- For example, the *ID* attribute of the relation *instructor* is sufficient to distinguish one instructor tuple from another.
- Thus, *ID* is a superkey.
- The *name* attribute of *instructor*, on the other hand, is not a superkey, because several instructors might have the same name.

Candidate Keys

- A superkey may contain extraneous attributes. For example, the combination of *ID* and *name* is a superkey for the relation *instructor*.
- If *K* is a superkey, then so is any superset of *K*.
- We are often interested in superkeys for which no proper subset is a superkey. Such minimal superkeys are called candidate keys.
- Suppose that a combination of *name* and *dept name* is sufficient to distinguish among members of the *instructor* relation. Then, both {*ID*} and {*name*, *dept name*} are candidate keys.

Primary Key

- We shall use the term primary key to denote a candidate key that is chosen by the database designer as the principal means of identifying tuples within a relation.
- Any two individual tuples in the relation are prohibited from having the same value on the key attributes at the same time.
- *student* (*ID*, *name*, *dept name*, *tot cred*)
- *classroom* (building, room number, *capacity*)

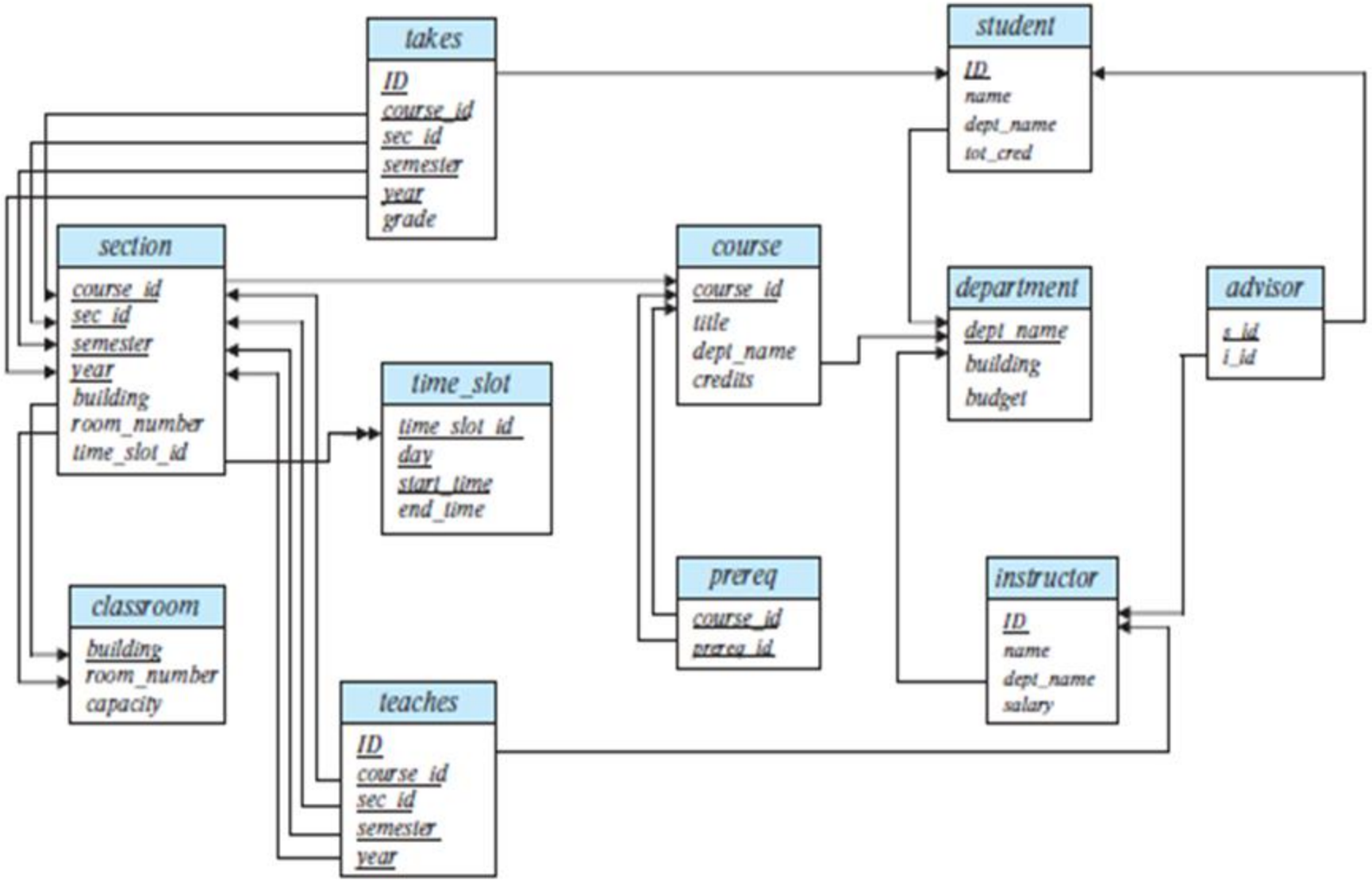
Foreign Key

- A foreign-key constraint from attribute(s) *A* of relation *r*1 to the primary-key *B* of relation *r*2 states that on any database instance, the value of *A* for each tuple in *r*1 must also be the value of *B* for some tuple in *r*2.
- Attribute set *A* is called a foreign key from *r*1, referencing *r*2.

- Alternate key
  - All the candidate key except the primary key
- Unique key
  - As similar to Primary Key but it accepts NULL Values

Schema for University Database

classroom(*building*, *room_number*, *capacity*)
department(*dept_name*, *building*, *budget*)
course(*course_id*, *title*, *dept_name*, *credits*)
instructor(*ID*, *name*, *dept_name*, *salary*)
section(*course_id*, *sec_id*, *semester*, *year*, *building*, *room_number*, *time_slot_id*)
teaches(*ID*, *course_id*, *sec_id*, *semester*, *year*)
student(*ID*, *name*, *dept_name*, *tot_cred*)
takes(*ID*, *course_id*, *sec_id*, *semester*, *year*, *grade*)
advisor(*s_ID*, *i_ID*)
time_slot(*time_slot_id*, *day*, *start_time*, *end_time*)
prereq(*course_id*, *prereq_id*)



Schema diagram for the university database.

Constraints

- we discuss the various restrictions on data that can be specified on a relational database in the form of constraints. Constraints on databases can generally be divided into three main categories:
- **Implicit constraints**
- **Explicit constraints**
- **Semantic constraints**

- Constraints that are inherent in the data model. We call these **inherent model-based constraints** or **implicit constraints**.
- Constraints that can be directly expressed in schemas of the data model, typically by specifying them in the DDL. We call these **schema-based constraints** or **explicit constraints**.
- Constraints that *cannot* be directly expressed in the schemas of the data model, and hence must be expressed and enforced by the application programs. We call these **application-based** or **semantic constraints** or **business rules**.

- Another important category of constraints is ***data dependencies***, which include ***functional dependencies*** and ***multivalued dependencies***.
- They are used mainly for testing the "**goodness**" of the design of a relational database and are utilized in a process called ***normalization***

Functional dependency

A functional dependency (FD) is a relationship between two attributes, typically between the PK and other non-key attributes within a table.
For any relation R, attribute Y is functionally dependent on attribute X (usually the PK), if for every valid instance of X, that value of X uniquely determines the value of Y.

	Attribute1 Name	Attribute2 Age	Attribute3 Sex	Attribute4 EmployeeNumber
Tuple 1	Anderson	21	F	010110
Tuple 2	Decker	22	M	010100
.	Glover	22	M	101000
.	Jackson	21	F	201100
.	Moore	19	M	111100
.	Nakata	20	F	111101
Tuple 7	Smith	19	M	111111

Relationship among or between attributes

Multivalued Dependency

- In a relation **R(A, B, C)**, a **multivalued dependency A ↠ B** means:
- For each value of A, there is a **set of B values** that is **independent** of the set of C values.
- This typically occurs when **two or more attributes depend on a common attribute independently**.

- Lets consider the relation

  Trainer(TrainerID, Skill, Language)

TrainerID	Skill	Language
T1	Python	English
T1	Python	Hindi
T1	Java	English
T1	Java	Hindi

- Trainer **T1** knows skills: **Python**, **Java**.
- Trainer **T1** speaks: **English**, **Hindi**.
- But **Python/Java** and **English/Hindi** are not related — they're **independent**.

So, from:

- T1 ↠ Skill
- T1 ↠ Language
- We derive the **cross product** of all combinations of skills and languages for T1.

Multivalued Dependency Here:

- TrainerID ↠ Skill
- TrainerID ↠ Language
- This is a case of **Multivalued Dependency**.
- Each set (Skill and Language) is independently associated with the **TrainerID**.

Multivalued dependency

Let's Consider a Relation R with attributes Course, Instructor, Textbook_Author. Here columns Instructor and Textbook_Author are dependent on the attribute COURSE and independent of each other. In this case, these two columns can be called as multivalued dependent on Course.

Course	Instructor	TextBook_Author
Management	X	Churchil
Management	Y	Peters
Management	Z	Peters
Finance	A	Weston
Finance	A	Gilbert

Course

Course	Instructor
Management	X
Management	Y
Management	Z
Finance	A

Textbook_Author

Course	TextBook_Author
Management	Churchil
Management	Peters
Finance	Weston
Finance	Gilbert

- The schema-based constraints include
  - Domain constraints,
  - Key constraints,
  - Constraints on NULLs,
  - Entity integrity constraints, and
  - Referential integrity constraints.

Domain Constraints

- Domain constraints specify that 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 typically include standard numeric data types
- for integers (such as short integer, integer, and long integer)
- real numbers (float and double precision float). Characters, Booleans, fixed-length strings, and variable-length strings are also available
- date, time, timestamp, and money, or other special data types also available.

Key Constraints

- no two tuples can have the same combination of values for *all* their attributes.
- Usually, there are other **subsets of attributes** of a relation schema *R* with the property that *no two tuples in any relation state r of R should have the same combination of values for these attributes.*
  - Candidate key (have more than one key)
  - Primary Key (one key – Underlined)

Properties of Keys

- Two distinct tuples in any state of the relation cannot have identical values for (all) the attributes in the key. This first property also applies to a super key.

- It is a ***minimal super key***—that is, a super key from which we cannot remove any attributes and still have the uniqueness constraint in condition 1 hold. This property is not required by a super key.

CAR

License_number	Engine_serial_number	Make	Model	Year
Texas ABC-739	A69352	Ford	Mustang	02
Florida TVP-347	B43696	Oldsmobile	Cutlass	05
New York MPO-22	X83554	Oldsmobile	Delta	01
California 432-TFY	C43742	Mercedes	190-D	99
California RSK-629	Y82935	Toyota	Camry	04
Texas RSK-629	U028365	Jaguar	XJS	04

The car relation with two candidate keys : License_number, Engine_serial_number

Entity integrity constraint

- To identify each row in a table, the table must have a primary key. The primary key is a unique value that identifies each row. This requirement is called the entity integrity constraint .

EMPLOYEE

EMP_ID	EMP_NAME	SALARY
123	Jack	30000
142	Harry	60000
164	John	20000
	Jackson	27000

Not allowed as primary key can't contain a NULL value

Referential Integrity Constraint

- A foreign key constraint (also referred to as a referential constraint or a referential integrity constraint) is a logical rule about values in one or more columns in one or more tables. For example, a set of tables shares information about a corporation's suppliers.

Department Table

DeptID (PK)	DeptName
D01	Computer Science
D02	Mechanical
D03	Civil

Student Table

StudentID (PK)	StudentName	DeptID (FK)
S001	John	D01
S002	Alice	D02
S003	Ravi	D05 ✗

In the Student table, **Ravi** has a DeptID = D05, which **does not exist** in the Department table.

This **violates referential integrity**, because:
The foreign key DeptID in Student must refer to an existing DeptID in the Department table.

```
-- Department Table
CREATE TABLE Department (
  DeptID VARCHAR(5) PRIMARY KEY,
  DeptName VARCHAR(50)
);

-- Student Table with FK constraint
CREATE TABLE Student (
  StudentID VARCHAR(5) PRIMARY KEY,
  StudentName VARCHAR(50),
  DeptID VARCHAR(5),
  FOREIGN KEY (DeptID) REFERENCES
Department(DeptID)
);
```

Null Vs Empty

Aspect	NULL	EMPTY
Definition	Represents the **absence of a value** or **unknown**	Represents a **known value that is blank** or **zero-length**
Meaning	"We don't know what the value is"	"We know the value, and it's blank or 0"
Storage	No actual data stored	Stores a value (e.g., empty string '', or 0)
Use in SQL	IS NULL to check for NULL	= '' or = 0 depending on data type
Example (Text)	NULL – the name is not entered at all	'' – the name is entered as blank
Example (Number)	NULL – salary not known	0 – salary is explicitly zero
In Comparisons	NULL = NULL → FALSE	'' = '' → TRUE

Not NULL

- Another constraint on attributes specifies whether NULL values are or are not permitted.
- For example, if every STUDENT tuple must have a valid, non-NULL value for the Name attribute, then Name of STUDENT is constrained to be NOT NULL.

Handling Null Values

- One way of handling missing values is the deletion of the rows or columns having null values.
- If any columns have more than half of the values as null then you can drop the entire column.
- In the same way, rows can also be dropped if having one or more columns values as null.
- Or You may fill by replacement of null values by meaningful values makes the table easier to interpret for readers

Raw table

name	country	state	phone	email	age	us_taxpayer
Alfred A.	Togo	*null*	1-091-395-4987	alfred@magnolia.com	*null*	*null*
Benny B.	Singapore	*null*	(102)879-0292	benny@ben.co.uk	*null*	*null*
Carla C.	*null*	Kansas	+18143519401	*null*	*null*	yes
Dan D.	France	*null*	(33)610789306	dan@mac.biz	*null*	*null*
Emily E.	Thailand	*null*	907-563-2744	emily@em.net	*null*	*null*
Frederic F.	Nauru	*null*	121-264-0618	freddy@fred.io	*null*	*null*
Gregorio G.	*null*	Florida	+14842989671	greg@ora.biz	*null*	yes
Hector H.	*null*	Washington	+16102448954	hector@hec.biz	*null*	yes
Iliana I.	Nicaragua	*null*		iliana@ili.name	*null*	*null*
John J.	Seychelles	*null*	367-945-7608	john@j.org	*null*	*null*

name	country	state	phone	email	age	us_taxpayer
Alfred A.	Togo	data not available	1-091-395-4987	alfred@magnolia.com	data not available	no
Benny B.	Singapore	data not available	(102)879-0292	benny@ben.co.uk	data not available	no
Carla C.	United States	Kansas	+18143519401	+18143519401	data not available	yes
Dan D.	France	data not available	(33)610789306	dan@mac.biz	data not available	no
Emily E.	Thailand	data not available	907-563-2744	emily@em.net	data not available	no
Frederic F.	Nauru	data not available	21-264-0618	freddy@fred.io	data not available	no
Gregorio G.	United States	Florida	+14842989671	greg@ora.biz	data not available	yes
Hector H.	United States	Washington	+16102448954	hector@hec.biz	data not available	yes
Iliana I.	Nicaragua	data not available		iliana@ili.name	data not available	no
John J.	Seychelles	data not available	367-945-7608	john@j.org	data not available	no