Relational Model

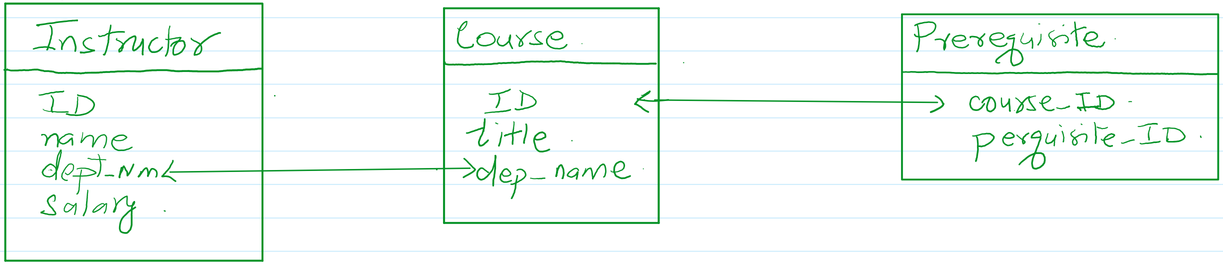
25 August 2024 12:24

The relational model is today the primary data model for commercial data processing application. It attained its primary position because of its simplicity, which eases the job of the programmer compared to earlier data models such as network model or the hierarchical model. In this we first study the fundamentals of the relational model or substantial theory, exist for relational databases.

Structure of Relational Databases:

A relational database consist. of a collection of tables, each of which is assigned a unique name. For example, consider the instructor table which stores the information about instructors. The table has four column headers ID, name, department name and salary. Each row of this table records information about instructor. consisting of the instructors ID name department name and salary. Similarly, the course table stores the information about courses consisting of a course ID, title, department name and credits for each course. Note that each instructor is identified by the value of the column ID. while each course is identified by the value of the course ID. another table pre requisite. which stores the prerequisite course for each course. The table has two columns Course ID and pre-requisite ID. Each row consists of a pair of coarse identifiers, such that second course is a perquisite of the first course.

Does a row in the prerequisite table in? indicates that two courses are related in the sense that one course is the prerequisite for the other. Another example, we consider the table instructor a row in the table that can be thought of representing the relationship between us specified ID and the corresponding values for name department name and salary value.



superkey

ID	Name	Dep_name	Salary
10101	Srinivas	Comp. Sci	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
32323	Amin	History	45000
45464	Partha	Physic	70000
67214	Singh	Comp. Sci	80000
81213	Mohan	Biology	72000

The instructor relation

Course_id	Title	Dep_name	Credits
BIO-101	Introduction to Biology	Biology	4
PHY-101	Principles of physics	Physic	4
CS-101	Introduction to Computer science	Computer science	4
CS-102	Database Systems	Computer Science	4
CS-103	Game design	Computer science	3
BIO-102	Genetics	Biology	3
PHY-102	Quantum Physic	Physic	3

The course relation

Course_id	Preq_id
CS-102	CS-101
CS-103	CS-102
BIO-102	BIO-101
PHY-102	PHY-101

The perquisite relation

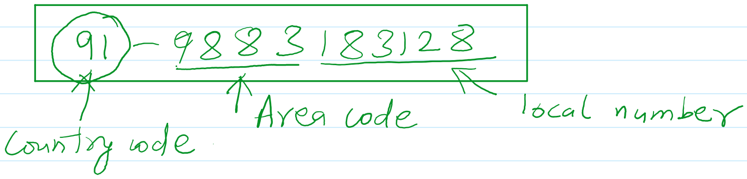
We require that, for all relation r, the domains of all attributes of r be atomic.

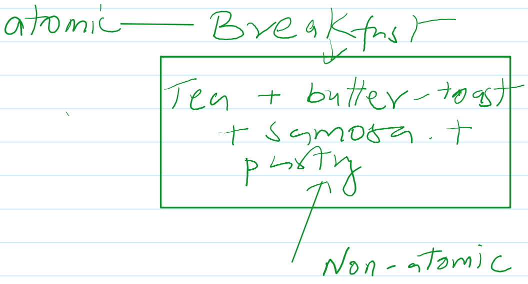
A domain is atomic if elements of the domain are considered to be individual units.

[illegible]

atomic — [illegible]

A domain is atomic if elements of the domain are considered to be individual units.

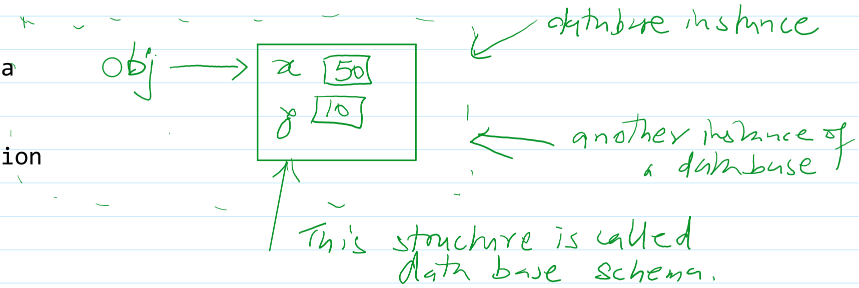




Database schema

When we talk about a database, we must differentiate between the **database schema**, which is a logical design of the database. And the **database instance**, which is a snapshot of the data in the database at the given instant in time the concept of relation corresponds to the programming language, notion of a variable where the concept of a relation schema correspond to the programming language notion of type definition.

In general, a relation schema consists of a list of attributes and their corresponding domains. The concept of relation instance corresponds to a programming language, notion of a value of a variable. The value of the given variable may change with time.


obj
x 50
y 10
attribute instance
another instance of a database
This structure is called data base schema.

department(dept_name, building, budget)

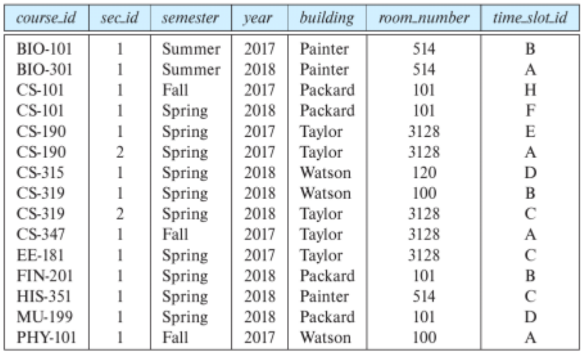
Note that the attribute department name appears in both the instructor schema and the department schema. This duplication is not a coincidence Rather, using common attribute in relation schema is one way of relating tuples of distinct relation. For example, suppose we wish to find the information about all the instructor who works in the Watson building. We look first at the department relation to find the department name of all the department housed in Watson. And then for each such department, we look in the instructor relation to find the information about the instructor associated with the corresponding department name.

Dept_Name	Building	Budget
Biology	Watson	90000
Computer science	Taylor	100000
Finance	Taylor	85000
Biology	Parker	120000
History	Packard	50000
Physic	Watson	85000
Chemistry	Watson	90000

Department relation

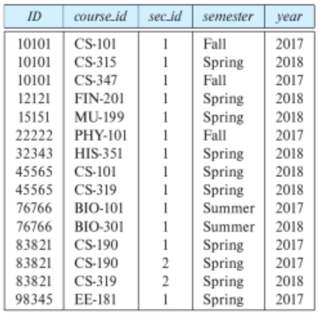
Section(course_id, sec_id, semester, year, building, room_number, time_slot_id)

Teaches(Id, course_id, sec_id, semester, year)



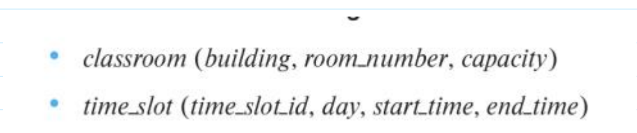
course_id	*sec_id*	*semester*	*year*	*building*	*room_number*	*time_slot_id*
BIO-101	1	Summer	2017	Painter	514	B
BIO-301	1	Summer	2018	Painter	514	A
CS-101	1	Fall	2017	Packard	101	H
CS-101	1	Spring	2018	Packard	101	F
CS-190	1	Spring	2017	Taylor	3128	E
CS-190	2	Spring	2017	Taylor	3128	A
CS-315	1	Spring	2018	Watson	120	D
CS-319	1	Spring	2018	Watson	100	B
CS-319	2	Spring	2018	Taylor	3128	C
CS-347	1	Fall	2017	Taylor	3128	A
EE-181	1	Spring	2017	Taylor	3128	C
FIN-201	1	Spring	2018	Packard	101	B
HIS-351	1	Spring	2018	Painter	514	C
MU-199	1	Spring	2018	Packard	101	D
PHY-101	1	Fall	2017	Watson	100	A

Section relation.

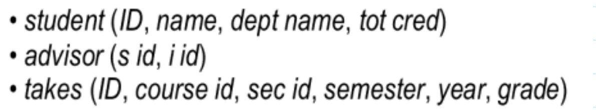


ID	*course_id*	*sec_id*	*semester*	*year*
10101	CS-101	1	Fall	2017
10101	CS-315	1	Spring	2018
10101	CS-347	1	Fall	2017
12121	FIN-201	1	Spring	2018
15151	MU-199	1	Spring	2018
22222	PHY-101	1	Fall	2017
32343	HIS-351	1	Spring	2018
45565	CS-101	1	Spring	2018
45565	CS-319	1	Spring	2018
76766	BIO-101	1	Summer	2017
76766	BIO-301	1	Summer	2018
83821	CS-190	1	Spring	2017
83821	CS-190	2	Spring	2017
83821	CS-319	2	Spring	2018
98345	EE-181	1	Spring	2017

Figure 2.7 The *teaches* relation.



- *classroom* (*building*, *room_number*, *capacity*)
- *time_slot* (*time_slot_id*, *day*, *start_time*, *end_time*)



- *student* (*ID*, *name*, *dept name*, *tot cred*)
- *advisor* (*s id*, *i id*)
- *takes* (*ID*, *course id*, *sec id*, *semester*, *year*, *grade*)

Keys
We must have a way to specify how tuples within a given relations are distinguished. This is expressed in terms of their attributes, that is, the value of the attribute Value of a tuple must be such that they can uniquely identify the tuple. In other words, note to tuple in a relation are allowed to have exact the same value for all the attributes.

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 super key.

A super key may contain extraneous attributes for example. a combination of id and name is a super key for the relation instructor. If K is a super key, then so in any superset of K, we are often interested in super keys for which no proper subset is a super key. Such minimal super keys are called 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 pupils within the relation. A key, whether primary candidate or Super. is a property of entire relation rather than of the individual tuples. Any two individual tuple in the relation are prohibited from having the same value on the key attribute at the same time. The designation of a key represents a constraint in the real world enterprise being modeled.
Primary keys must be chosen with care, as we noted the name of a person is obviously not sufficient, because there may be many people with the same name. In India, Aadhar card number attribute of a person would be a candidate key. Since in India, non-Indian residents usually do not have other card numbers. International enterprise must generate their own unique identifiers.

An alternative is to use some combination of other attributes as a key The primary key should be chosen such that its attribute value are never, or very rarely changed. For instance, the address field of a person should not be the part of the primary key, since it is likely to change. Aadhar card number on the first hand are guaranteed never to change. Unique identifiers generate by enterprise generally do not change, except if two enterprise merge. In such case, same identifier may have issued by both enterprises and a reallocation of identifiers may be required to make sure that they are unique.

It is customary to list the primary key attributes of a relations schema before the other attributes. For example, the department name attribute of the department is listed first, since it is a primary key. Primary key attributes are also underlined. A relation, say, R1 may include

among its attribute the primary key of another relation, say, R2. This. attribute is called a foreign key from R1 referencing R2.
The relation R1 is also called the referencing relation of the foreign key dependency. And R 2 is called the referenced relation of the foreign key. For example, the attribute department name in instructor is a foreign key from the instructor. Referencing department since department name is the primary key of the department. In any database instance, given. any tuple, say ta from the instructor relation, there must be a tuple tb, In the department relations such that the value of the department name attribute of TA is seen as the value of the primary key Department of tb.

PK

Dept_Name	Building	Budget
Biology	Watson	90000
Computer science	Taylor	100000
Finance	Taylor	85000
Biology	Parker	120000
History	Packard	50000
Physic	Watson	85000
Music	Watson	90000

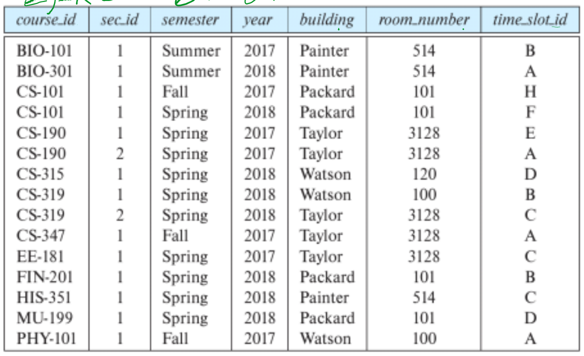
Department relation

FK

ID	Name	Dep_name	Salary
10101	Srinivas	Comp. Sci	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
32323	Amin	History	45000
45464	Partha	Physic	70000
67214	Singh	Comp. Sci	80000
81213	Mohan	Biology	72000

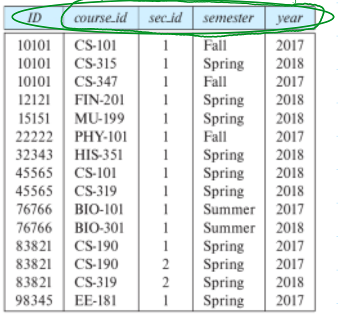
The instructor relation

FK FK FK FK



course_id	sec_id	semester	year	building	room_number	time_slot_id
BIO-101	1	Summer	2017	Painter	514	B
BIO-301	1	Summer	2018	Painter	514	A
CS-101	1	Fall	2017	Packard	101	H
CS-101	1	Spring	2018	Packard	101	F
CS-190	1	Spring	2017	Taylor	3128	E
CS-190	2	Spring	2017	Taylor	3128	A
CS-315	1	Spring	2018	Watson	120	D
CS-319	1	Spring	2018	Watson	100	B
CS-319	2	Spring	2018	Taylor	3128	C
CS-347	1	Fall	2017	Taylor	3128	A
EE-181	1	Spring	2017	Taylor	3128	C
FIN-201	1	Spring	2018	Packard	101	B
HIS-351	1	Spring	2018	Painter	514	C
MU-199	1	Spring	2018	Packard	101	D
PHY-101	1	Fall	2017	Watson	100	A

Section relation



ID	course_id	sec_id	semester	year
10101	CS-101	1	Fall	2017
10101	CS-315	1	Spring	2018
10101	CS-347	1	Fall	2017
12121	FIN-201	1	Spring	2018
15151	MU-199	1	Spring	2018
22222	PHY-101	1	Fall	2017
32343	HIS-351	1	Spring	2018
45565	CS-101	1	Spring	2018
45565	CS-319	1	Spring	2018
76766	BIO-101	1	Summer	2017
76766	BIO-301	1	Summer	2018
83821	CS-190	1	Spring	2017
83821	CS-190	2	Spring	2017
83821	CS-319	2	Spring	2018
98345	EE-181	1	Spring	2017

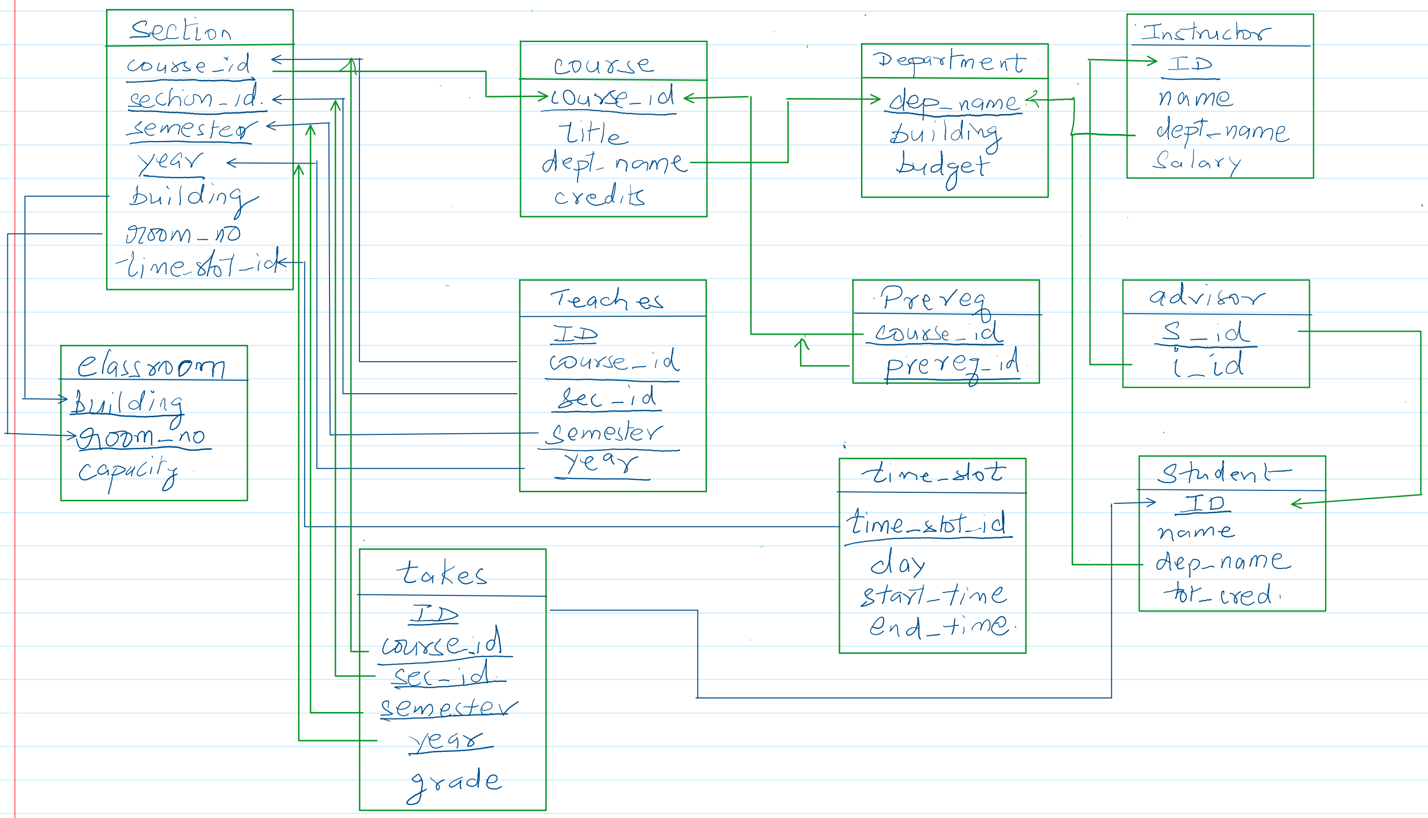
Must be a primary key

Figure 2.7 The *teaches* relation.

Now consider the section and teaches relations. It would be reasonable to require that if a section exists for a course, it must be taught by at least one instructor. However, it could be possibly be taught by more than one instructor. to enforce this constraint, we would require that if a particular (course_ID, section_ID, semester, year) Combination appears in section. then the same combination must appear in teaches. However, this set of values does not form a primary key for teachers. Since. more than one instructor may teach one such section as a result. we cannot declare a foreign key constraint from the section 2 teaches. (although we can define a foreign key constraint in other direction from teachers to section)

The constraint from section to teaches is the example of **referential integrity constraint.** A referential integrity constraint requires the value appearing in specified attributes of any tuple in the referencing relation also appear in specified attribute of at least one tuple in the Reference relations.

Schema diagrams
A database schema along with primary key and foreign key dependencies can be depicted by schema diagram. Each relation appears as a box with the relation name at the top and the. attribute listed inside the box primary. key attributes are shown underlined. Foreign key dependencies appear as arrow from the foreign key attributes of the referencing relation to the primary key of the reference relation.



Schema Diagram for University Database.

Referential integrity constraints and other foreign key constraints are not shown explicitly in the schema diagram.

Home work:
Try to develop Hospital Management System Schema diagram