02. Relational Model

[PM Jat, DAIICT, Gandhinagar]

Content:

What is a relation?

Origin of relational model lies in “relation” in mathematics. The model was first proposed by Dr. E.F. Codd of IBM Research through paper "A Relational Model for Large Shared Data Banks" in Communications of the ACM, June 1970.

To understand what the *relation* in relational model is, let us begin with relation in math.

- Consider four sets A, B, C, and D;
- Relation is defined as (a, b, c, d) where
  $a \in A$, $b \in B$, $c \in C$, and $d \in D$.
- Or, relation is subset of (A x B x C x D).

Now let us try using this concept in some real situation. Assume that

- Set A is universal set of student IDs.
- Set B is universal set of student names
- C as universal set of Program Codes that the university offers, and
- D is set of real values (0 to 10).

With this premise shown below is a relation drawn from said four sets

(101,Rahul,BCS,7.5)

(102,Vikash,BIT,8.6)

(103,Shally,BEE,5.4)

(104,Alka,BIT,6.8)

(105,Ravi,BCS,6.5)

Each tuple represents a student entity.

Let us do some adaption of math relation to relational relation.

1. We say that each tuple expresses some fact about “something”; in this case student entity - each tuple represents a student entity.

2. We bring in a notion of "attribute"; we say that each value in the tuple is value for corresponding attribute of the entity or event (next example). First is value for ID attribute, second is value for name attribute, and so forth.
3. Let us call corresponding sets of value in a relation as "domain" for the attribute
4. We give name to relation based on what it represents; in this case we name this relation as "Student".

Below is tabular view of Student relation. Looks like a table; that is why probably relation is also called as *table*. A tuple is also called as row in a table. Attribute is called as column.

Terms: table, column, row are used in Structured Query Language (SQL). While relation, attribute, tuple are more used in theoretical discussions.

We often, in tabular view, show header information of table; i.e. attribute or column names. We may also give domain for each attribute.

ID	Name	ProgID	CPI
101	Rahul	BCS	7.5
102	Vikash	BIT	8.6
103	Shally	BEE	5.4
104	Alka	BIT	6.8
105	Ravi	BCS	6.5

Another example:

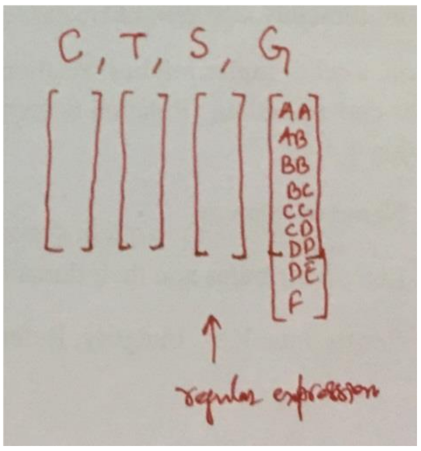
Consider four sets C, T, S, G; where

C is set of course numbers (offered in an institute)

T is set of term-ids

S set of student IDs

G is set of grades



Following is a relation drawn from these sets -

(IT214, Autumn-2017, 201501123, BB)

(IT214, Autumn-2016, 201501123, CD)

…

Each tuple, in this case represents fact of an event of "student registering in a course offered in a term". We name to this relations as Registers. Below is depiction of some snapshot of relation instance.

CourseNo	Term	StudentID	Grade
IT214	Autumn-2017	201501123	DE
IT214	Autumn-2016	201501123	BC
…			

Relation Characteristics

A **relation** is a set of tuples.

Order of tuple is not important in a relation.

All tuples are distinct from one another "in terms of values".

A tuple is an ordered list of values (for respective attribute) drawn from corresponding domain for the attribute.

Tuple can also be represented as set of (Attribute-Name, value pairs).
Example: {(ID,101), (ProgID, BCS), (Name,Rahul), (CPI,7.5)}

Relation instance and relation schema

Relation, a set of tuples implies "relation instance" in relational model. Relations instance holds data.

Every relation also has corresponding "Relation Schema". The relation schema basically describes following –

- Name of relation
- List of attributes and their domain, and
- Constraints: Key, Integrity, Referential Integrity, and so

Example:

Student (Studentid [int], name [varchar(20)], progid [smallint], batch [smallint], cpi, [numeric(5,2]) with other details.

Note: in SQL, domains are defining through data-types.

Notations:

Schema: R(A1, A2, An), where R is name of relation and A1 through An are name of attribute in relation R

Domain for attribute Ai is defined as dom(Ai)

r(R) for relation [instance] of schema R

Tuple: Two ways to represent a tuple

As ordered list of n-values; t = <v1, v2, ..., vn> ; or

As set of n (attrib-name, attrib-value) pairs: t = {(A2,v2), (A1,v1), ..., (An,vn)}

Notationally, value of attribute Ai in tuple t, can be expressed as t[Ai] or t.Ai

Similarly, t[Au, Av, ..., Aw] refers to the sub-tuple of t containing the values of attributes Au, Av, ..., Aw, respectively in t

Relation constraints

Constraints are data existential rules, and are also called "Database Integrity Constraints".

Recall: constraints are part of schema, and DBMS is to ensure compliance of constraints on every valid state of database.

Following are types of constraints specified on a relation-

1. Domain constraints
2. Key constraints and Entity Integrity constraints
3. **NOT NULL** value constraints
4. Referential Integrity Constraints
5. Other constraints – sometimes difficult to specify as a part of schema.

Terminology note: the term "relation" implies "relation instance"; for relation schema, we explicitly augment "relation" with term "schema". However sometime, people use term "relation" while they mean "relation schema". In most cases by context, we should be able detect if relation refers to "relation schema".

Here we are talking about constraints; constraints are defined as part of schema.

Domain constraints

- Attribute values are drawn from their corresponding domain.

NOT NULL constraints

- It is specified for an attribute, and implies that the attribute cannot have NULL value. That is, value for such attribute mandatory in a tuple.

Key Constraints

Primarily this constraint requires that every tuple in a relation is distinct.

In other words, we say this as following. Suppose, r is a relation; and R is its schema. Key constraints requires that there are no two tuples t1 and t2 in relation r, where t1=t2. It can also be expressed as t1[R] != t2[R], where R is basically set of all attributes of the relation r.

However, for checking uniqueness of a tuple, we may not check values for all attributes; checking for some subset often suffices the purpose.

Super-key

Some subset SK of R such that no tuple t1 and t2 can be same in a relation r; i.e.
t1[SK] != t2[SK]. Super Key is subset of attributes that are enough to check uniqueness of tuples in a relation.

Key

Some minimal subset K of R such that no tuple t1 and t2 can be same in a relation r; i.e.
t1[K] != t2[K]. Key is minimum subset of attributes that are enough to check uniqueness of tuples in a relation.

Super key can be defined in terms of Key: any superset of K (and subset of R) is super key.

Key can also be defined in terms of Super Key; minimal subset K of super-key such that
t1[K] != t2[K].

Motivation of define Key is that we do not have to check for all attributes of relation for ensuring uniqueness of tuples in a relation. Checking attribute in Key are enough.

Remember Keys here are sets of attributes; even though in most cases it will singleton set.

A relation might have multiple keys. Also key might be "composite", that is having more than one attribute.

Exercise:

What is the key of following relation schema?

> Student(StudentID, UUID, Name, DOB, CPI)
> Offers(TermID, CourseNo,InstrutorID)
> Registers(StudentID, TermID, CourseNo, Grade)

There is another term for Key is candidate key. A relation might have more than one key, and candidate keys;

Entity Integrity Constraint:

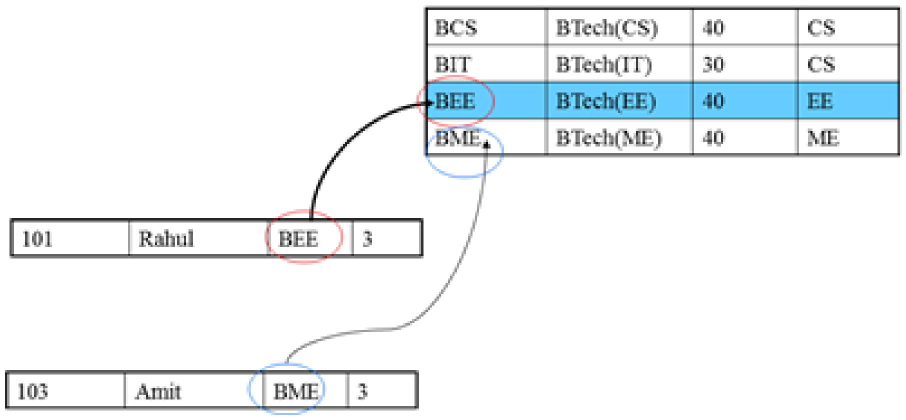
Values for Key Attribute cannot be NULL. If we allow it is no more possible identifying a tuple in its set (relation). So, this constraint requires that entity is always identifiable.

Referential Integrity Constraint

Foreign Key

In relations, we use foreign key to capture associations between entities.

Figure below depicts student entities associated with program entities. This association is captured by referring corresponding tuple.


BCS | BTech(CS) | 40 | CS
BIT | BTech(IT) | 30 | CS
BEE | BTech(EE) | 40 | EE
BME | BTech(ME) | 40 | ME
101 | Rahul | BEE | 3
103 | Amit | BME | 3

In this diagram, we have two student entities represented by two tuples in student relation. Refer encircled values; they are values for ProgID attribute. Values BEE, and BME are basically reference to program tuples. <u>The attribute having reference value is called foreign key</u>. The relation having reference is called referencing relation. Relation in which referenced tuple is, called referencing relation.

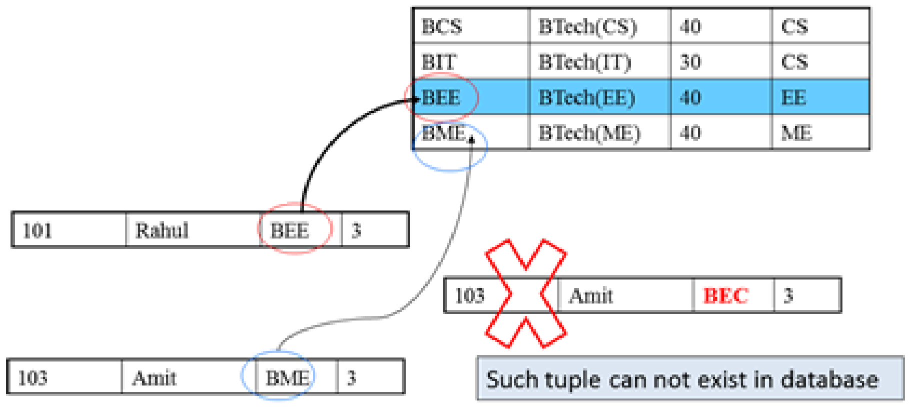
Foreign key is an attribute (or attribute set) in a relation, that is used for referencing purpose. It is used to refer to a tuple in some relation (can be in same relation). Referencing happens by storing value of key attributes of target relation. Recall that key values are enough to identify a tuple.

In the schema, conceptually we define foreign keys as following:

Suppose we have a relation schema R that has attributes A,B,C,D, E. Let us say there is another relation schema S with attributes P,X,Y,Z. Suppose value of attribute D in relation r (instance of R) is to refer attribute p in relation s (instance of S); then we express Foreign Key as following- D is a foreign key in R that refers to attribute P of relation S. It should be obvious that attribute P should be key of relation schema S. Why? That is when we will be able to unambiguously refer to a tuple in s.

Referential Integrity Constraint

Referential Integrity constraint requires that a value in a foreign key in a tuple should refer to some existing tuple; a tuple that is identified by value in foreign key. Figure below shows a student tuple that has "invalid" value in foreign key. Invalid because it attempts to refer a tuple that does not exist. Such a tuple cannot exist in the relation.



Formally we can explain it as following: Suppose relation R has a foreign key FK, that refers to attribute K in relation S. Suppose t1 is a tuple in r; foreign key constraints requires that there should be a tuple t2 in s such that t1[FK] = t2[K] or t1[FK] is null.

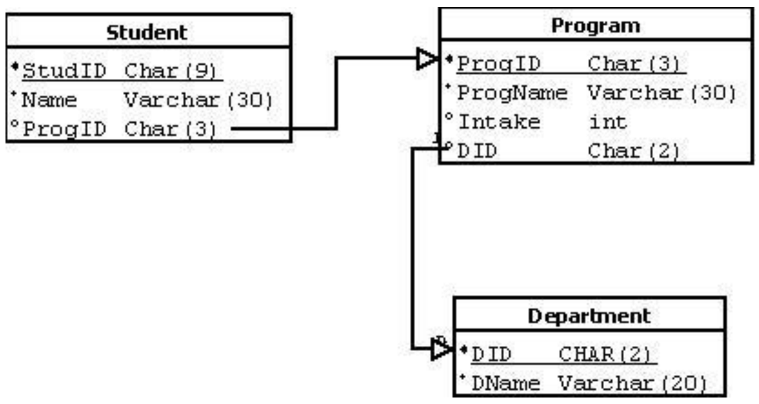
Summary:

1. We have seen how mathematical relation has been adapted to represent databases.
2. A relation in relational model is set of tuples. A tuple is ordered list of values for respective attributes. A tuple can also be represented as set of attribute value pairs.
3. Defined: Key (candidate key), Super key, and Foreign Key.
4. Learned various types of Constraints: Domain, Key, Not Null, Entity Integrity, and Referential Integrity.
5. Foreign keys are means of recording "associations". Value in a foreign key refers to another tuple. FK specification tells following: attribute that is FK; attribute (and in which relation) to which the foreign key refers.
6. Referential integrity constraint tells us that value in a foreign key must refer to some existing tuple in referred relation.

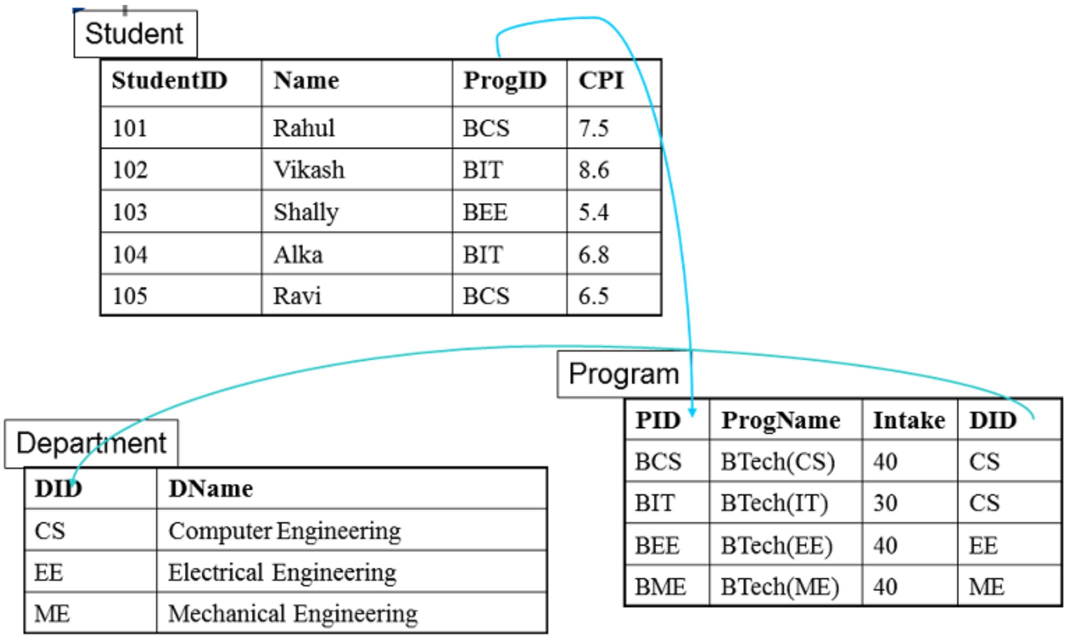
Examples

1. ProgID in student referring refering to ProgID in program relation [fig below]
2. DID in program referring to DID in Department [fig below]
3. Relation: Offers(CouseNo, TermID, InstructorID), all three attributes referring to key of respective relation. Key of this relation is?
4. Relation: Register(CouseNo, TermID,StudentID,Grade), {CourseNo, TermIID} compositely referring to Offers, and StudentID referring to StudentID in Student relation.

Here is complete schema of XIT database:



An instance XIT schema



Student

StudentID	Name	ProgID	CPI
101	Rahul	BCS	7.5
102	Vikash	BIT	8.6
103	Shally	BEE	5.4
104	Alka	BIT	6.8
105	Ravi	BCS	6.5

Program

PID	ProgName	Intake	DID
BCS	BTech(CS)	40	CS
BIT	BTech(IT)	30	CS
BEE	BTech(EE)	40	EE
BME	BTech(ME)	40	ME

Department

DID	DName
CS	Computer Engineering
EE	Electrical Engineering
ME	Mechanical Engineering

Relational Database manipulation

1. Relational Algebra
2. Relational Calculus
3. Standard Query Language (SQL)

Relational Algebra

Relational algebra allows expressing queries as algebraic expression, where operands are relations. In example below, sigma (σ) is an operation, employee is operand relation, and logical expression is parameter to the operation.

$$\sigma_{\text{DNO = 4 AND SALARY > 50000}}\ (\text{employee})$$

This expression outputs tuples that member of employee relation, and value of salary attribute is > 50000 and value of dno attribute is equals to 4.

Relational algebra expressions have closure property that is result of an expression is a valid relation.

Relational algebra was introduced in original proposal of relational model.

There are definite set of operations defined on relations. Some operations are unary while others are binary.

Complex queries are expressed by sequencing multiple operations.

Relational Calculus

A query in relational calculus is expressed as a logic expression; above relational algebra query is expressed in relational following is a query, expressed in tuple relational calculus, as following.

{t| EMPLOYEE(t) AND t.salary > 50000 AND t.dno=4) }

Below is another query that outputs specific attributes of tuple tuple t, where t is such that member of employee, and there t.dno and d.dno matches where d is tuple of department relation, and d.name is 'Research'

$$\{t.\text{Fname}, t.\text{Lname}, t.\text{Address} \mid \text{EMPLOYEE}(t) \text{ AND } (\exists d)(\text{DEPARTMENT}(d) \text{ AND } d.\text{Dname}=\text{'Research' AND } d.\text{Dnumber}=t.\text{Dno})\}$$

It may be noted that in calculus expressions are more declarative (rather than procedural) in nature; no concept of operation and their order.

Standard Query Language (SQL)

Structured Query Language (SQL) was not part of original proposal of Relational Model. Some people pronounce SQL as "SEQUEL".

Originally developed at IBM with System R. System R was first implementation of Relational Model in Early 70s.

SQL has evolved since then, and now a standard language for relational databases.

ANSI has published SQL-86, SQL-89, SQL-92, SQL-99, SQL-2003, SQL-2006, and SQL-2008, 2013 or so. SQL that discussed in most texts is SQL-99.

Despite ANSI SQL standard, RDBMS often do some kind of deviation to it. Postgres's SQL is probably most ANSI compliant.

SQL provides following categories of commands:

- Data Definition Language (DDL)
  - Used for defining database schema.
- Data Manipulation Language (DML)
  - Update commands
  - Querying commands
- Physical level commands
  - Used for improving performance of database.
- Transaction Control
  - Begin transaction, commit transaction, rollback etc
- Authorization
  - Grant permission to other users for access (read/update /delete, etc.)

Note: relational algebra and calculus do not allow expressing data definitions, and data update operations. SQL being the language that is used in practice, it provides full stack of operations.

SQL was initially influenced by tuple relation calculus, however later updates incorporated most algebraic concepts as well. We will learn most of SQL in algebraic perspective.