08. Relational Database Design

[PM Jat, DAIICT, Gandhinagar]

Contents

Database Design as Process

Database design is a process of determining “good” database schema.

Measure of goodness depends on data model. At conceptual design, in ER, measure of goodness is subjective. It should capture complete and correct representation of database requirement.

In general, in addition to complete and correct representation, minimizing “data redundancies” is striving objective of database design. Data redundancy primarily means a fact/value is occurring at more than one place in the database.

In relational design too, where outcome is good relational schema, minimizing data redundancy is objective. Relational model defines a formal method of measuring goodness of a relation, called Normal Forms - 1NF, 2NF, 3NF, BCNF, 4NF, 5NF. Higher the Normal Form, less the redundancies are, and “better” a Table relation is.

Relational Database Design as process

Input:

- Database Requirement Specifications
- Data items and constraints

Final Output:

- (Good) Relational schema
- Measure of Goodness: Minimum data redundancy

Before getting into details of these formal measures, let us first understand what the “data redundancy” is, and what are undesirable consequences of redundancy?

Database update Anomalies

Considering following set of attributes from XIT database – studid, name, stud_progid, cpi, pid, pname, intake, prog_did, did, dname for have three relations S(studid, name, stud_progid, cpi), P(pid, pname, intake, prog_did), D(did, dname). Hopefully it is “good” design (we will certify later, however). You should get the same schema, if we come through ER route (i.e. have ER model and derive relations using studied ER to relations mapping rules).

studid characte	name character	progid characte	cpi numeric
101	Rahul	BCS	8.70
102	Vikash	BEC	6.80
103	Shally	BEE	7.40
104	Alka	BEC	7.90
105	Ravi	BCS	9.30

Program

pid charac	pname character vary	intake smallint	did charac	DName
BCS	BTech (CS)	30	CS	Computer Sc
BEC	BTech (ECE)	40	EE	Ele
BEE	BTech (EE)	40	EE	Ele
BME	BTech (ME)	40	ME	Mech

Department

did chara	dname character varying(30)
CS	Computer Engineering
EE	Electrical Engineering
ME	Mechanical Engineering

Let us again take the question, “are above relations are good enough”? To answer this question, let us consider some of its alternatives. Compare and see why this is better over other options.

One possibility is, let us say, we merge Program and Department and have a single relation PD (is basically program relation with an additional attribute DName).

Below is combined relation, PD -

pid characte	pname character var	intake smallint	did chara	dname character varying(30)
BCS	BTech(CS)	30	CS	Computer Engineering
BEC	BTech(ECE)	40	EE	Electrical Engineering
BEE	BTech(EE)	40	EE	Electrical Engineering
BME	BTech(ME)	40	ME	Mechanical Engineering
BIT	BTech(IT)	60	CS	Computer Engg.

Does above represent correct set of facts? Try understanding this -

- What will be interpretation of a tuple in combine relation PD?
- What will be the key in PD?
- How do we get tuple set for combine relation from individual relations - Program and Department?

Combined relation PD is basically natural join of both relations Program and Department.A tuple of PD contain all information of as program only, value of attribute dname is interpreted as name of department offering the program. Key of relation is PID. Do you see "data redundancy" here?

If not very obvious, let us also consider another case. Here we have combined relation of Student and Program relations (and let us call it SP), as shown below-

Select pid, pname, intake, did
from SP
where pid = 'BCS';

studid characte	name character	cpi numeric	progid characte	pname character vary	intake smallin	did chara
101	Rahul	8.70	BCS	BTech(CS)	30	CS
102	Vikash	6.80	~~BEC~~ BCS Null	BTech(ECE) Null	40 Null	EE Null
103	Shally	7.40	BEE	BTech(EE)	40	EE
~~104~~	~~Alka~~	~~7.90~~	~~BEC~~ Null	~~BTech(ECE)~~ Null	~~40~~ Null	~~EE~~ Null
105	Ravi	9.30	BCS	BTech(CS)	30	CS
106	Ami	6.70	BCS	BTech(CS)	20	CT

BCS

1

2

~~delete from Prog where Pid = 'BEC'~~

"inconsistent"

What are the undesirable consequences of redundancies?

Considering cost of modern storages, storage should not be of a concern, and even if it is, it probably gets compensated by reduced requirement performing joins wile answering queries (note most queries we need to join).

Select

More serious problem associated with redundancies "database update anomalies".

One anomaly, you hopefully must have already noted, in combined PD, we cannot have a department details unless there is some program offered in that department (in PD). Or we cannot have a program detail unless there is some student attached with it (in SP)? This may be seriously undesirable.

1. Insertion Anomaly
2. Modify Anomaly
3. Deletion Anomaly

Insertion Anomaly

- Attempt inserting a new program in PD?
- Attempt inserting a new Department in PD?

While inserting, a new program, we need to supply all Department details as well. In this case we only have attribute DName; but could be more in other cases – for example in combined SP, attempt adding a new student? This is not only discomfort, but we may also end having inconsistent program details (in SP); inconsistent, since it is to be repeated with every program it offers.

Inserting a new Department in PD, without a program is not possible? We can choose to have null values for program portion of data but since PID is the key; it cannot be NULL.

These are the two examples of Insertion anomaly caused due to data redundancy.

Modification Anomaly

- Attempt changing program for a student in SP? Not only it require we supplying, all information of new program for the student in question, it may also lead database having inconsistent details of program in database; there is possibility of two tuple having same program information do not have same program information.

Deletion Anomaly

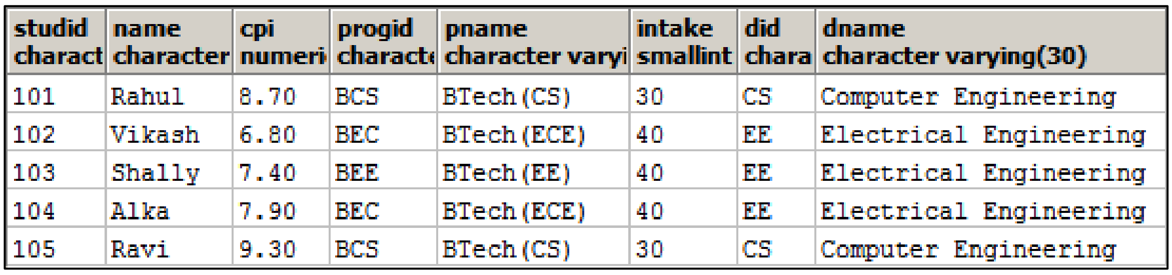
- If we delete program, and that program happens to be only program, department also gets deleted; that is we lose the department details as well!
- How do we delete a department? Deletion basically becomes changing department attributes value of a program to either NULL or some other department data. This is weird.

So, avoiding redundancy and hence avoiding update anomalies is the objective of relational schema design. As already said, normal forms are the measure of goodness of a relation schema. Normal forms are defined in terms of "functional dependencies". In section next we attempt understanding functional dependencies.

Function Dependencies

There are certain dependencies among attributes.

Consider all set of attributes in XIT schema, given below; search dname for a given prog-id, say BCS, you find that it gives you single value "Computer Engineering", even if it occurs at multiple places it gives us a single value. What does returned value indicates? It is name of the department that offers the program 'BCS'.



studid charact	name character	cpi numeri	progid characte	pname character varyi	intake smallint	did chara	dname character varying(30)
101	Rahul	8.70	BCS	BTech(CS)	30	CS	Computer Engineering
102	Vikash	6.80	BEC	BTech(ECE)	40	EE	Electrical Engineering
103	Shally	7.40	BEE	BTech(EE)	40	EE	Electrical Engineering
104	Alka	7.90	BEC	BTech(ECE)	40	EE	Electrical Engineering
105	Ravi	9.30	BCS	BTech(CS)	30	CS	Computer Engineering

On contrary, on searching of you student-id for given prog-id, you do not get a single value; and it is as expected that there are multiple students for a given prog-id. These observations are basically indicative of certain dependencies that occur between attributes in a database. Attribute dependencies come from problem domain and capture database constraints.

We call such dependencies as "Functional Dependencies"

Let us understand functional dependency from the concept of function in math; where you have

$$f: x \rightarrow y$$

Where let X and Y be attribute sets, and interpret it as - "for given the value x for attribute X, the function maps to a single value of attribute Y".If such a mapping exists then we say;X functionally determines Y, and express as $X \rightarrow Y$; then we also say that Y is functionally dependent on X.