11 important database designing rules which I follow

Shivprasad koirala

26 Feb 2014 CPOL

This article will discuss about 11 important database designing rules.

Table of Contents

Courtesy: Image from Motion pictures

Introduction

Before you start reading this article let me confirm to you I am not a guru in database designing. The below 11 points are what I have learnt via projects, my own experiences, and my own reading. I personally think it has helped me a lot when it comes to DB designing. Any criticism is welcome.

The reason I am writing a full blown article is, when developers design a database they tend to follow the three normal forms like a silver bullet. They tend to think normalization is the only way of designing. Due this mind set they sometimes hit road blocks as the project moves ahead.

If you are new to normalization, then click and see 3 normal forms in action which explains all the three normal forms step by step.

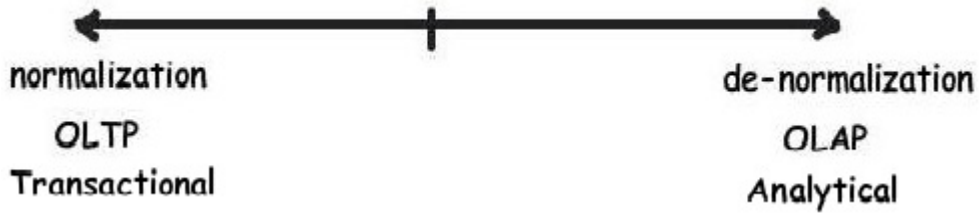
Said and done normalization rules are important guidelines but taking them as a mark on stone is calling for trouble. Below are my own 11 rules which I remember on the top of my head while doing DB design.

Rule 1: What is the nature of the application (OLTP or OLAP)?

When you start your database design the first thing to analyze is the nature of the application you are designing for, is it Transactional or Analytical. You will find many developers by default applying normalization rules without thinking about the nature of the application and then later getting into performance and customization issues. As said, there are two kinds of applications: transaction based and analytical based, let's understand what these types are.

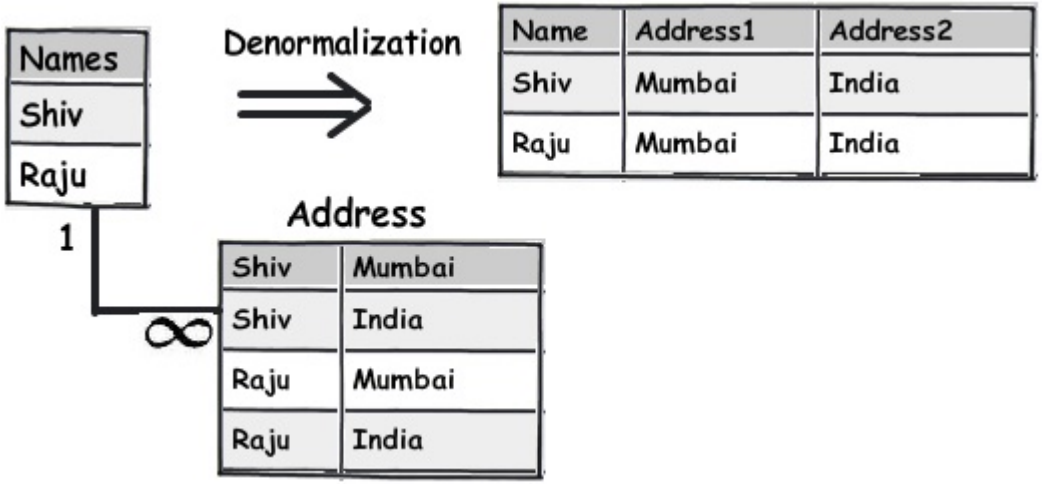
Transactional: In this kind of application, your end user is more interested in CRUD, i.e., creating, reading, updating, and deleting records. The official name for such a kind of database is OLTP.

Analytical: In these kinds of applications your end user is more interested in analysis, reporting, forecasting, etc. These kinds of databases have a less number of inserts and updates. The main intention here is to fetch and analyze data as fast as possible. The official name for such a kind of database is OLAP.



In other words if you think inserts, updates, and deletes are more prominent then go for a normalized table design, else create a flat denormalized database structure.

Below is a simple diagram which shows how the names and address in the left hand side are a simple normalized table and by applying a denormalized structure how we have created a flat table structure.

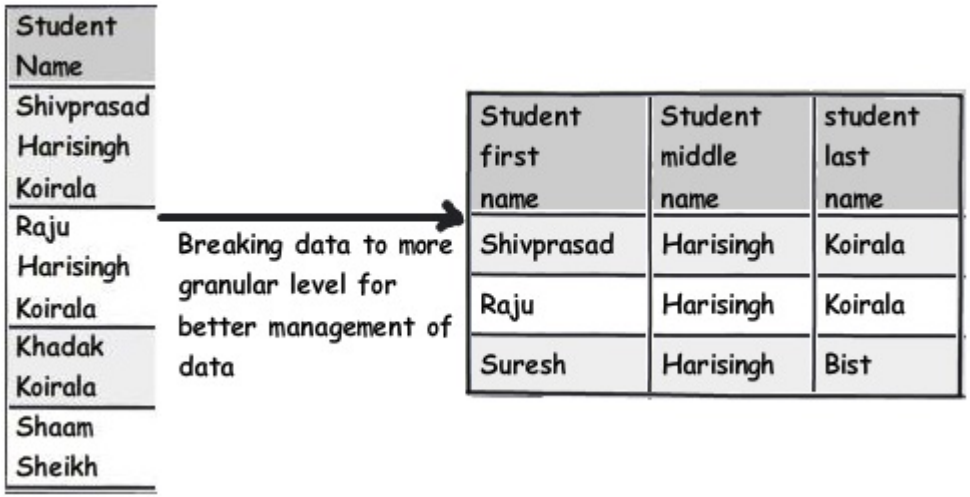


Rule 2: Break your data into logical pieces, make life simpler

This rule is actually the first rule from 1st normal form. One of the signs of violation of this rule is if your queries are using too many string parsing functions like substring, charindex, etc., then probably this rule needs to be applied.

For instance you can see the below table which has student names; if you ever want to query student names having “Koirala” and not “Harisingh”, you can imagine what kind of a query you will end up with.

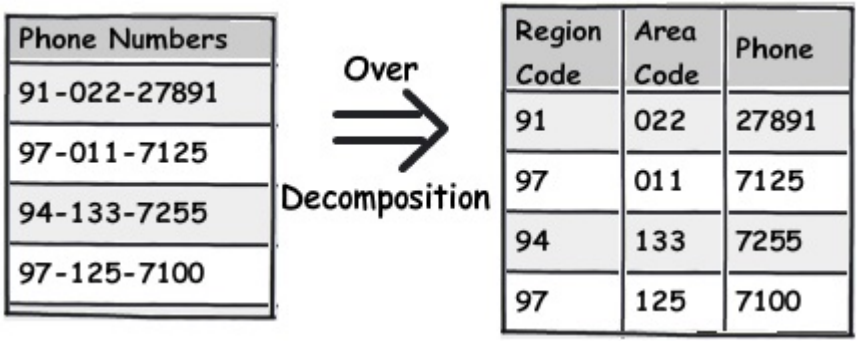
So the better approach would be to break this field into further logical pieces so that we can write clean and optimal queries.



Rule 3: Do not get overdosed with rule 2

Developers are cute creatures. If you tell them this is the way, they keep doing it; well, they overdo it leading to unwanted consequences. This also applies to rule 2 which we just talked above. When you think about decomposing, give a pause and ask yourself, is it needed? As said, the decomposition should be logical.

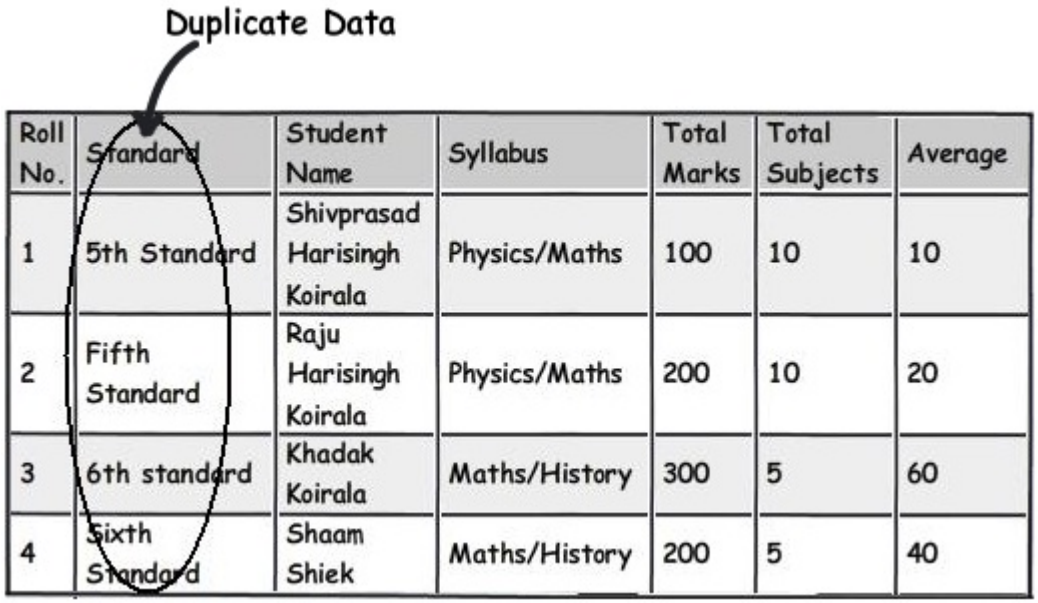
For instance, you can see the phone number field; it's rare that you will operate on ISD codes of phone numbers separately (until your application demands it). So it would be a wise decision to just leave it as it can lead to more complications.



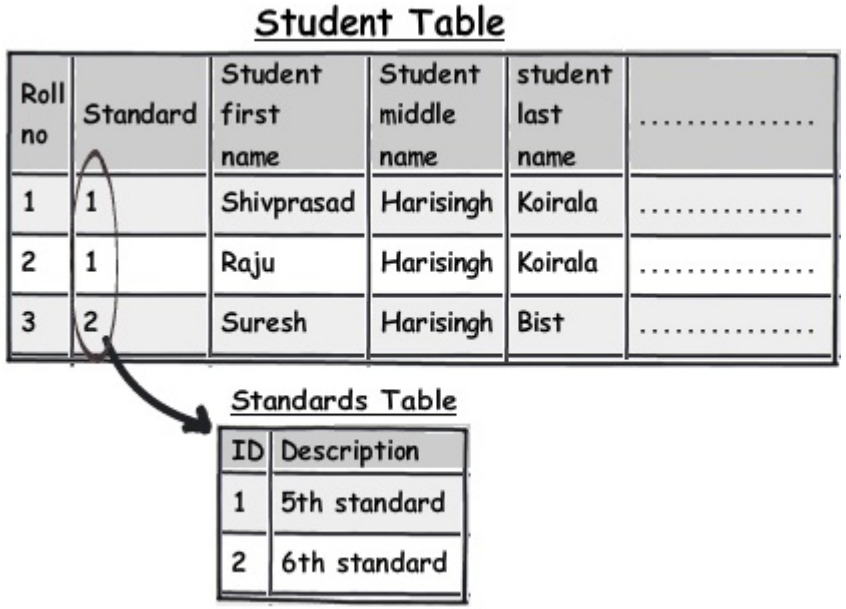
Rule 4: Treat duplicate non-uniform data as your biggest enemy

Focus and refactor duplicate data. My personal worry about duplicate data is not that it takes hard disk space, but the confusion it creates.

For instance, in the below diagram, you can see "5th Standard" and "Fifth standard" means the same. Now you can say the data has come into your system due to bad data entry or poor validation. If you ever want to derive a report, they would show them as different entities, which is very confusing from the end user point of view.



One of the solutions would be to move the data into a different master table altogether and refer them via foreign keys. You can see in the below figure how we have created a new master table called "Standards" and linked the same using a simple foreign key.

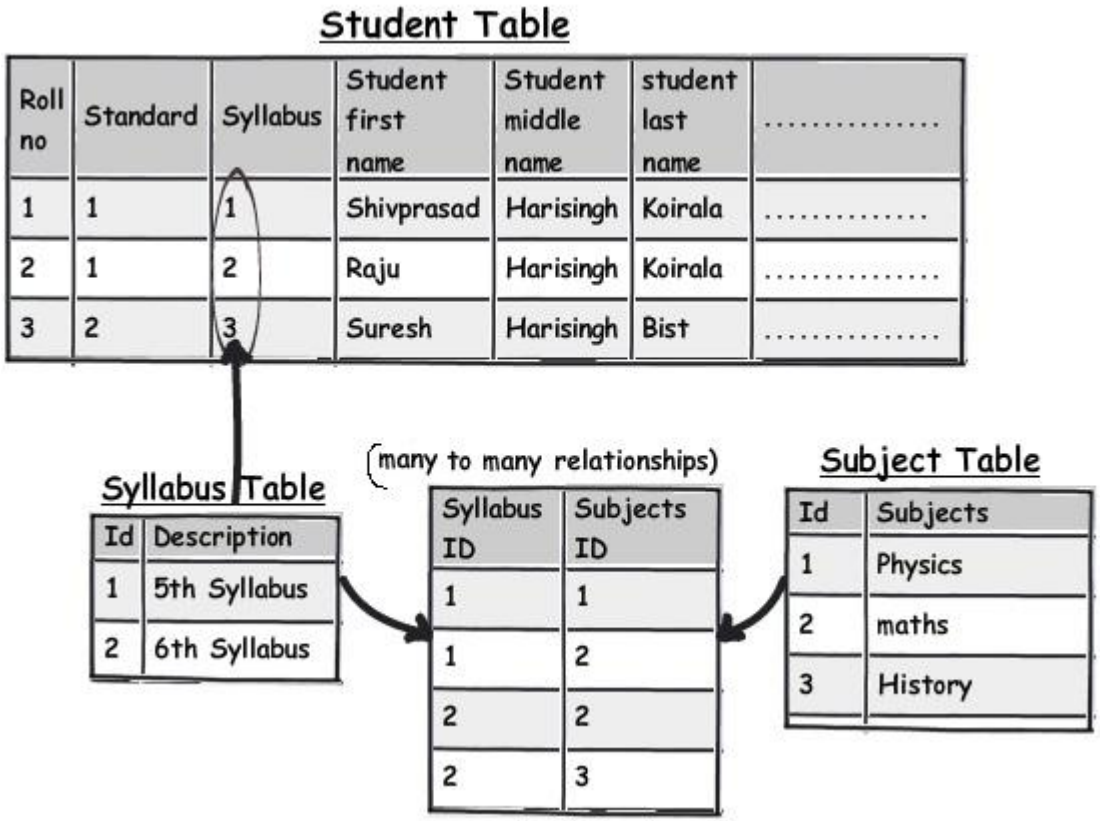


Rule 5: Watch for data separated by separators

The second rule of 1st normal form says avoid repeating groups. One of the examples of repeating groups is explained in the below diagram. If you see the syllabus field closely, in one field we have too much data stuffed. These kinds of fields are termed as “Repeating groups”. If we have to manipulate this data, the query would be complex and also I doubt about the performance of the queries.

Roll No.	Standard	Student Name	Syllabus	Total Marks	Total Subjects	Average
1	5th Standard	Shivprasad Harisingh Koirala	Physics/Maths	100	10	10
2	Fifth Standard	Raju Harisingh Koirala	Physics/Maths	200	10	20
3	6th standard	Khadak Koirala	Maths/History	300	5	60
4	Sixth Standard	Shaam Shiek	Maths/History	200	5	40

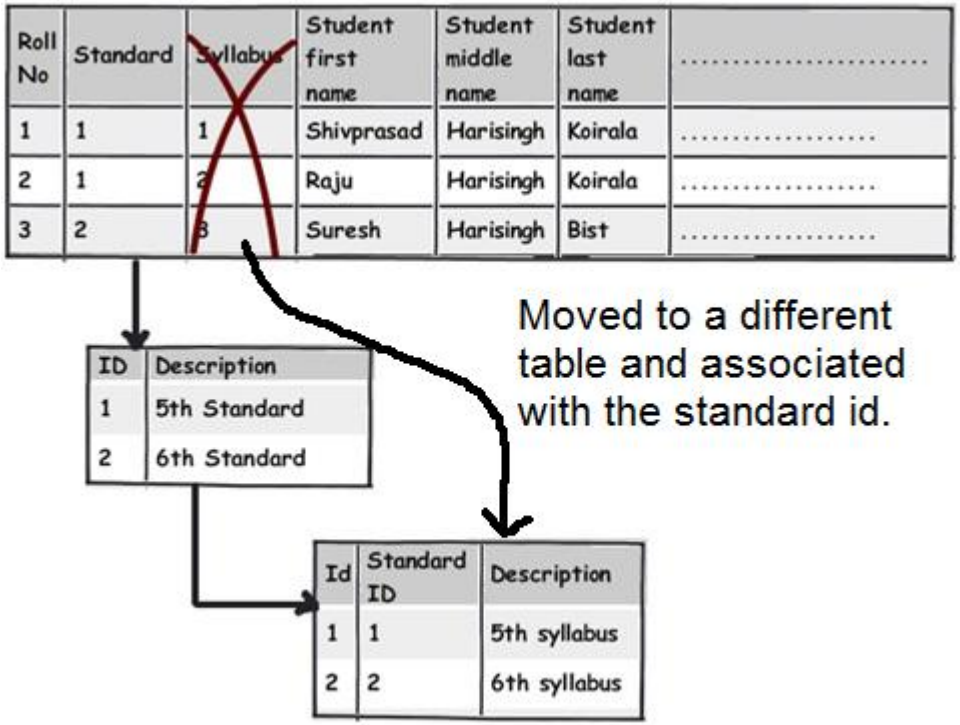
These kinds of columns which have data stuffed with separators need special attention and a better approach would be to move those fields to a different table and link them with keys for better management.



So now let's apply the second rule of 1st normal form: "Avoid repeating groups". You can see in the above figure I have created a separate syllabus table and then made a many-to-many relationship with the subject table.

With this approach the syllabus field in the main table is no more repeating and has data separators.

Rule 6: Watch for partial dependencies



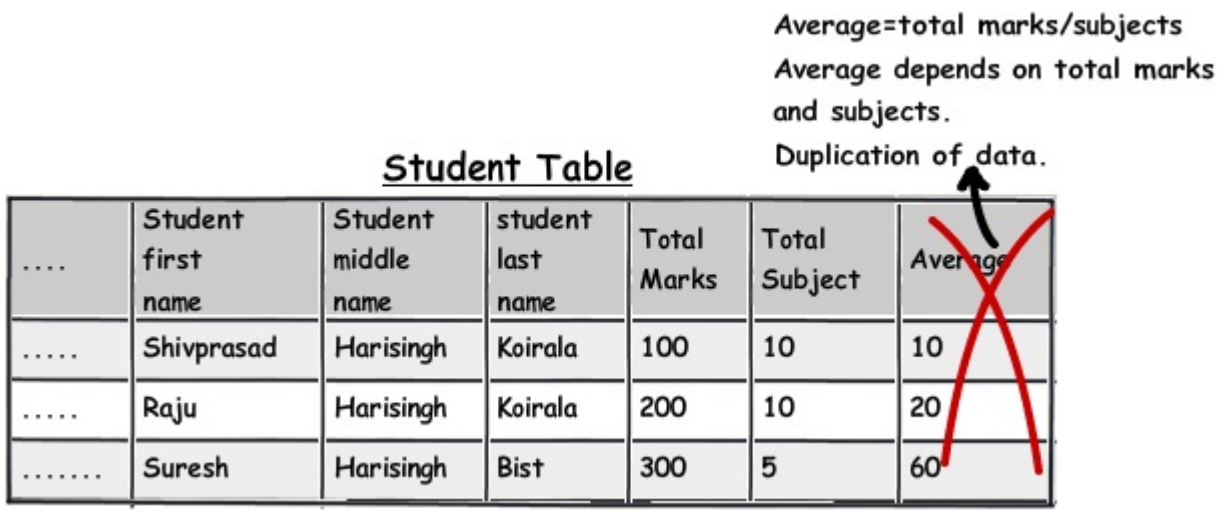
Watch for fields which depend partially on primary keys. For instance in the above table we can see the primary key is created on roll number and standard. Now watch the syllabus field closely. The syllabus field is associated with a standard and not with a student directly (roll number).

The syllabus is associated with the standard in which the student is studying and not directly with the student. So if tomorrow we want to update the syllabus we have to update it for each student, which is painstaking and not logical. It makes more sense to move these fields out and associate them with the Standard table.

You can see how we have moved the syllabus field and attached it to the Standards table.

This rule is nothing but the 2nd normal form: "All keys should depend on the full primary key and not partially".

Rule 7: Choose derived columns preciously

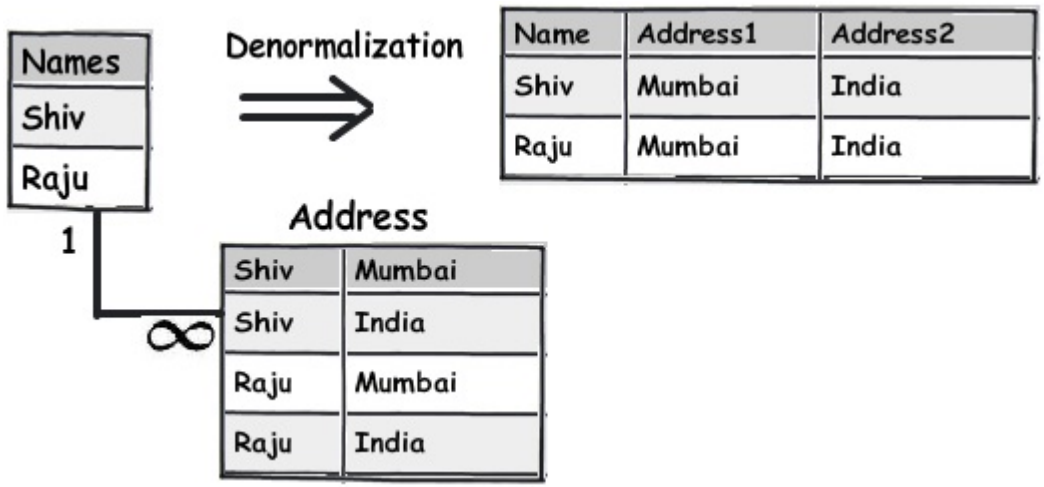


If you are working on OLTP applications, getting rid of derived columns would be a good thought, unless there is some pressing reason for performance. In case of OLAP where we do a lot of summations, calculations, these kinds of fields are necessary to gain performance.

In the above figure you can see how the average field is dependent on the marks and subject. This is also one form of redundancy. So for such kinds of fields which are derived from other fields, give a thought: are they really necessary?

This rule is also termed as the 3rd normal form: "No column should depend on other non-primary key columns". My personal thought is do not apply this rule blindly, see the situation; it's not that redundant data is always bad. If the redundant data is calculative data, see the situation and then decide if you want to implement the 3rd normal form.

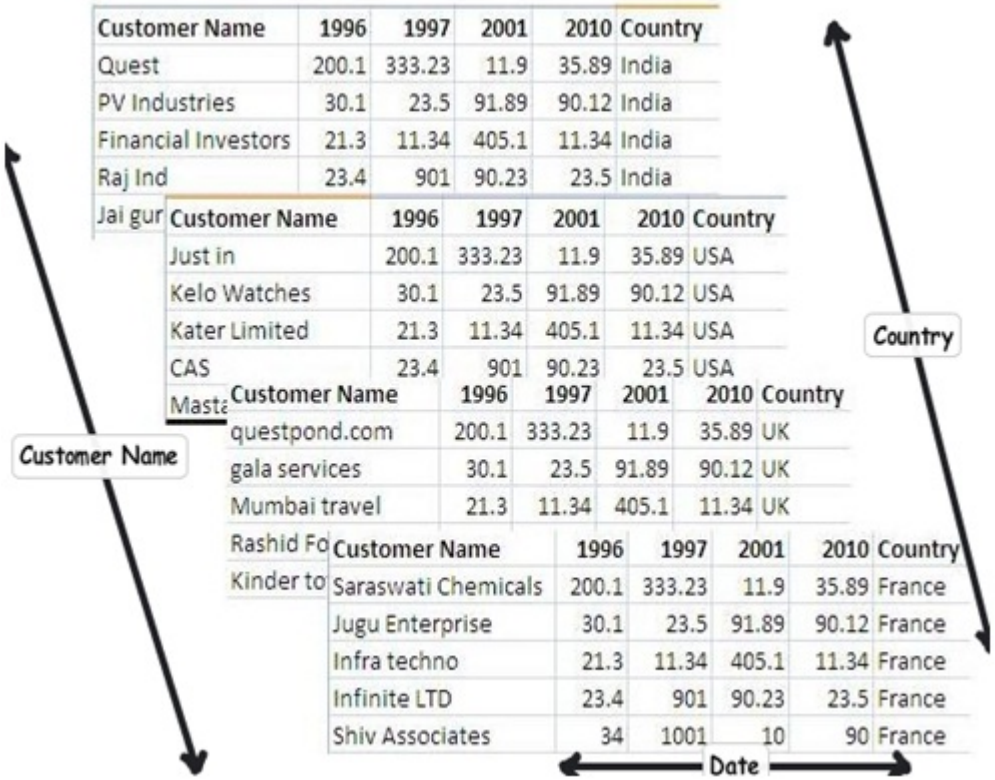
Rule 8: Do not be hard on avoiding redundancy, if performance is the key



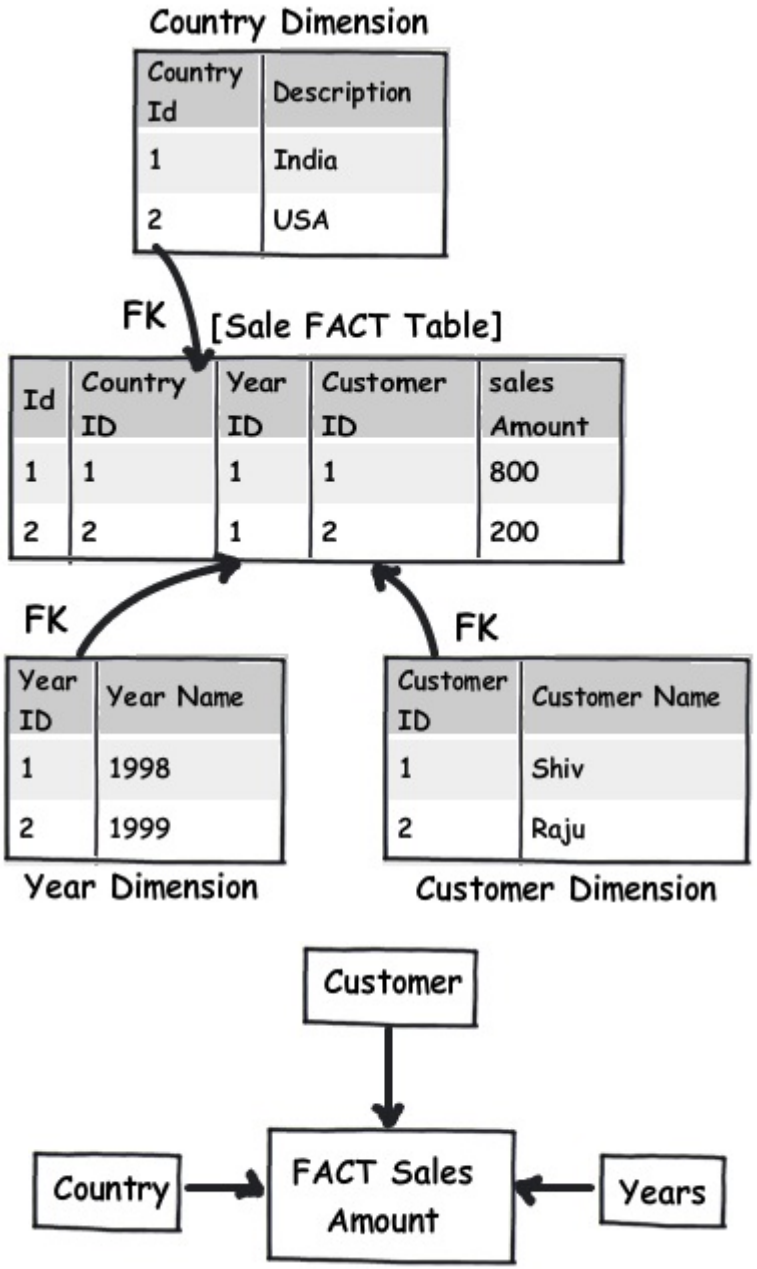
Do not make it a strict rule that you will always avoid redundancy. If there is a pressing need for performance think about de-normalization. In normalization, you need to make joins with many tables and in denormalization, the joins reduce and thus increase performance.

Rule 9: Multidimensional data is a different beast altogether

OLAP projects mostly deal with multidimensional data. For instance you can see the below figure, you would like to get sales per country, customer, and date. In simple words you are looking at sales figures which have three intersections of dimension data.

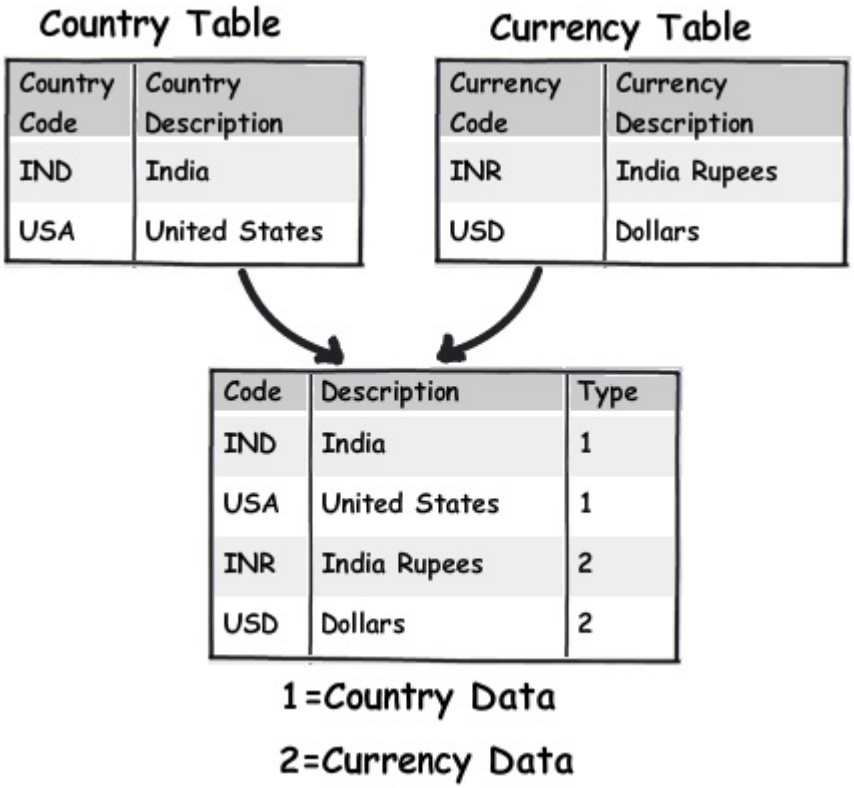


For such kinds of situations a dimension and fact design is a better approach. In simple words you can create a simple central sales fact table which has the sales amount field and it makes a connection with all dimension tables using a foreign key relationship.



Rule 10: Centralize name value table design

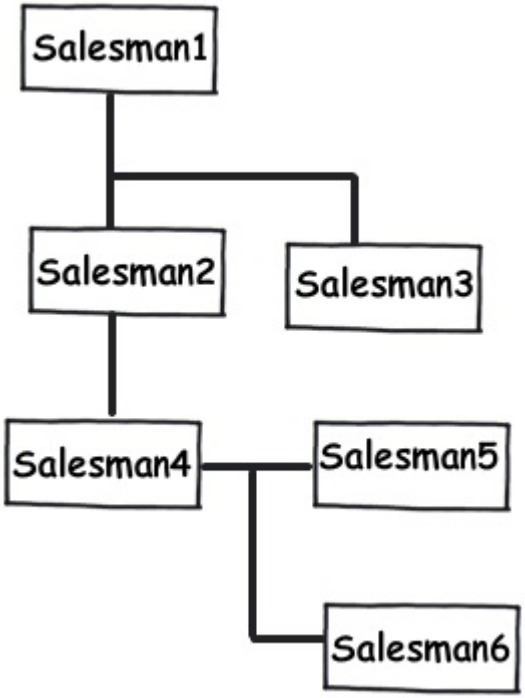
Many times I have come across name value tables. Name and value tables means it has key and some data associated with the key. For instance in the below figure you can see we have a currency table and a country table. If you watch the data closely they actually only have a key and value.



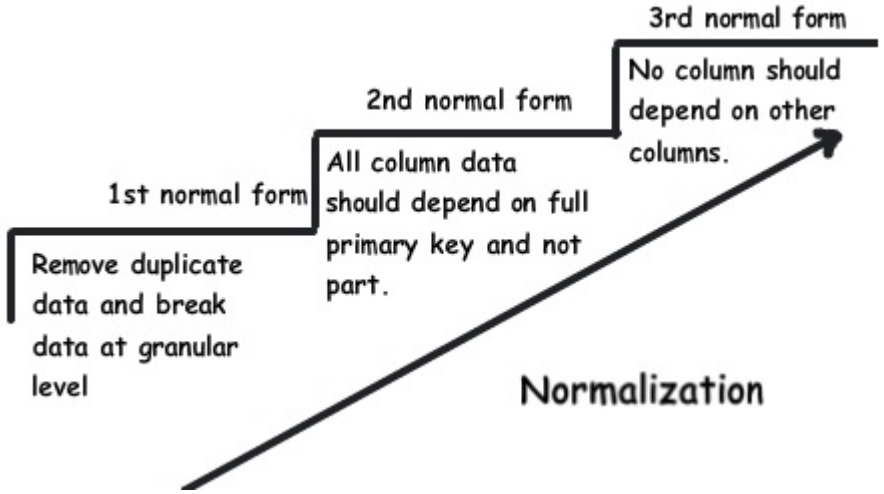
For such kinds of tables, creating a central table and differentiating the data by using a type field makes more sense.

Rule 11: For unlimited hierarchical data self-reference PK and FK

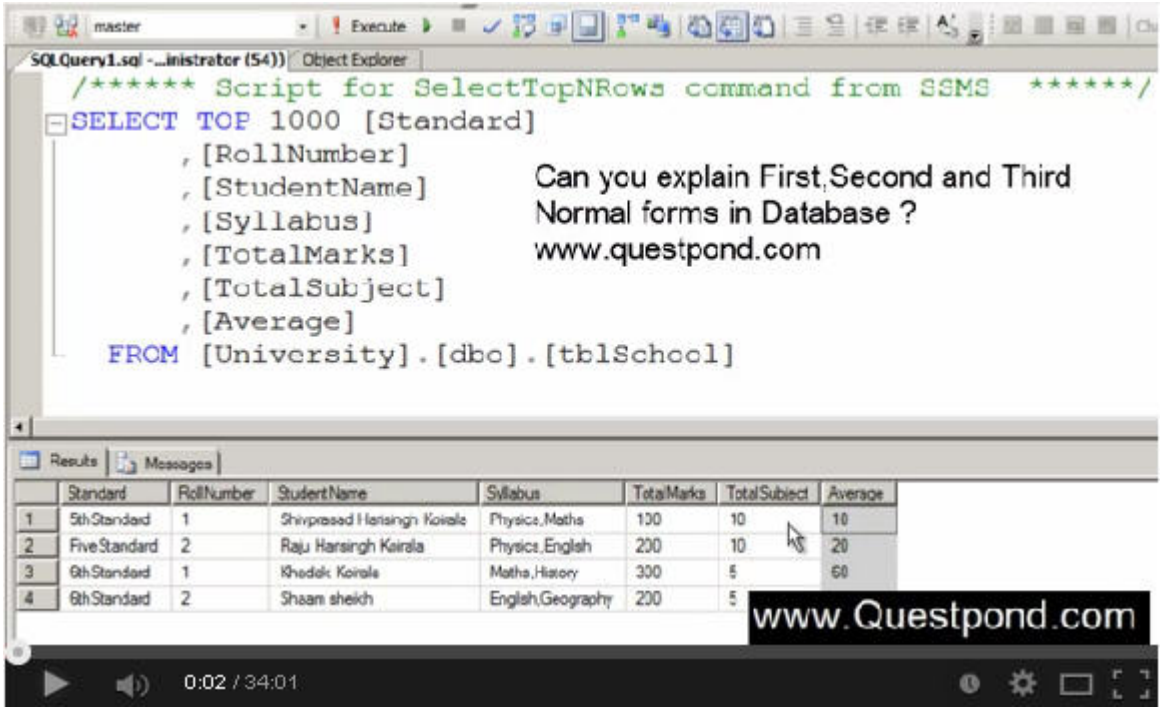
Many times we come across data with unlimited parent child hierarchy. For instance consider a multi-level marketing scenario where a sales person can have multiple sales people below them. For such scenarios, using a self-referencing primary key and foreign key will help to achieve the same.



This article is not meant to say that do not follow normal forms, instead do not follow them blindly, look at your project's nature and the type of data you are dealing with first.



Below is a video which explains the three normal forms step by step using a simple school table.



You can also visit my website for step by step videos on Design Patterns, UML, SharePoint 2010, .NET Fundamentals, VSTS, UML, SQL Server, MVC, and lots more.

License

Share

About the Author

Shivprasad koirala

Architect https://www.questpond.com
India

Follow this Member

Do not forget to watch my Learn step by step video series.

Learn MVC 5 step by step in 16 hours
Learn MVC Core step by step
Learn Angular tutorials step by step for beginners
Learn Azure Step by Step
Learn Data Science Step by Step
Step by Step Mathematics for Data Science
Learn Xamarin Mobile Programming Step by Step
Learn Design Pattern in 8 hours
Learn C# in ...

show more

Comments and Discussions

Add a Comment or Question

Email Alerts | Search Comments

Spacing Relaxed | Layout Normal | Per page 25 | Update

First Prev Next

11 important database designing rules which I follow	**XolaniG**	**20-Jun-17 14:08**
MS SQL Server developer download	**Member 12541993**	**24-May-16 17:33**

nice	BillW33	3-Mar-16 0:37
Greate	Aladár Horváth	18-Aug-15 18:27
[My vote of 2] rule 10 is one of common database design mistakes.	seyyed hamed monem	22-Jun-15 14:53
My vote of 5	Pratik Bhuva	31-Mar-15 16:41
design	Member 11218475	18-Nov-14 18:02
Rule #10 has to be put down	opc3	19-Sep-14 8:31
Re: Rule #10 has to be put down	Anurag Gandhi	17-Aug-16 17:19
My vote of 5	Akiii_Lethal	25-Jul-14 17:03
Good one	Shahriar Iqbal Chowdhury/Galib	9-Jun-14 15:28
My vote of 5, but one doubt	Thava Rajan	28-Feb-14 17:21
Good Article	AzuriVN	28-Feb-14 10:03
My vote of 5	David Days	27-Feb-14 23:57
here a question is here! what do you think? which decision you will made?	Phu Hiep DUONG	27-Apr-13 4:05
Rule 8: Do not be hard on redundancy	Jan Zumwalt	1-Dec-12 0:19
My vote of 5	zeego	14-Nov-12 13:18
My vote of 5	Xavi Rius	12-Sep-12 4:24
My vote of 5	Jasmine2501	11-Sep-12 3:49
My vote of 2	Member 8528409	8-Sep-12 4:02
My vote of 4	brother.gabriel	29-Apr-12 4:45
My vote of 4	Uilleam	9-Apr-12 22:39

Re: My vote of 4	**Shivprasad koirala**	10-Apr-12 17:58
My vote of 5	**Manoj Kumar Choubey**	**6-Apr-12 0:45**
Great	**rajeshrisharma**	**4-Apr-12 20:25**

Last Visit: 9-Jul-19 0:18 Last Update: 9-Jul-19 0:19 Refresh 1 2 Next »

General News Suggestion Question Bug Answer Joke Praise Rant Admin

Use Ctrl+Left/Right to switch messages, Ctrl+Up/Down to switch threads, Ctrl+Shift+Left/Right to switch pages.