Learn PISIO

Relational Database Design

Practical Approach

Драган Ћајић

2015.

http://learn-pisio.byethost15.com/

Contents

Relational Database Design

00. Introduction

Relational Database Design

01. What is a Database?

Database

data

front-end and back-end / website example

spreadsheet vs. database

02. What is a Relational Database?

Relation is, basically, **connection** between data (entity and attribute values for that entity).

Mathematical concept -> combining set(s)

Combining attributes in relational databases, rather than numbers, in combining sets.

An **entity** is anything we store data about, and **attributes** are the thing we store.

So, entity is what we store data about, attribute is what we store.

When we connect this with relation, we have attribue values and attribute types.

set of attributes in tables

Table is graphical way of illustrating how we store data in database.

We enter specific values (in table) for each one of attributes to make **relation** between attributes and the entity.

A row (in table) is all of the attribute values for specific entity, so, row is pointing to (specific) entity.

An column are all of the values for specific attribute type.

Entity type is category of entities we are storing.

Attribute type is category of attributes.

Basically, type is a category.

In a table, the columns are the attributes (id, password, username, email, address, phone number, ...), and an row is where we give a value for every single one of these attributes, all talking about one entity.

So, attributes are columns, entities are individual rows. Then the hole thing is the table and that is the entity type (e.g. user), because every single row within this table should be a user.

Another name, in mathematics, for a row is **tuple**, so tuple is row.

03. RDBMS

Data**b**ase **M**anagement **S**ystem (DBMS)

Relational **D**ata**b**ase **M**anagement **S**ystem (RDBMS)

Database can be used to store tons of informations.

RDBMS is just subcategory of DBMS, a specific kind that is designed to work with relational databases.

view mechanism – the way of data is presented by RDBMS to the user

View mechanism allows us to change the surface appearance of our data.

Well, with a view mechanism we are able to get different views of our data at the surface level.

View mechanism allow us to create different **views**, that is the layer of security.

huge database for a business

Not every single person who uses the database has the privilege of creating a new view, just because you can access the database does not mean you can do things such as update data, change the way it's structured, and so forth. That's probably something that only the administrator or the owner of the business is going to be able do.

So, that is a really good security feature.

transactions

Examples of RDBMS are: MySQL, Microsoft SQL Server, Oracle, ...

RDBMS takes data on the hard drive (on a server, or your home computer) and puts that into presentable tables for people like administrator to view.

The other thing that RDBMS is going to do is to create a consistency for the front-end (e.g. website form).

With a database, we can have consistency on the front-end (to make our users happy!), on the back-end though things can change (e.g. the way we store dates, timezones, or character sets, ...) and generally that's not going to affect the front-end unless the data from back-end affects the front-end (e.g. deleting user column is going to affect website form field).

SQL – language used to communicate to our RDBMS

04. Introduction to SQL

SQL is the programming language used to communicate to database. Basically, we speak English, database do not, so SQL is some kind of mediator between computer, database and human English. It's almost English.

SQL is not like specific RDMS programming language, it is general language for every single Relational Database Management System. From vendor to vendor, such as from MySQL to Microsoft SQL Server, or something else, it may vary little bit, but the general concepts of SQL state the same.

So, SQL first is used to define the database structure (DDL), and then it manipulates the data within (DML).

`CREATE` -> DDL

`UPDATE` -> DML

Another thing that SQL can do is what's known Join, and joins are very important with Relational DBMS, because with RDBMS we break things up into separate tables for

simplicity sake, rather than having a spreadsheet with like tons of information. We break it of like entity, e.g. so table about the User, the table about the Orders, the table about the Customers. Then we can join those values to get a new larger table – a new view that looks like a huge unorganized table.

User table: `id`, `name`

Sale table: `id`, `user_id`, `item`

We can do Join here and get new view which has: `name`, `item`.

05. Naming Conventions

Naming convention is just a pattern that people do (or You do!) to keep things consistent.

So, not only that this naming convention will help keep my database consistent across different tables and columns, but this naming convention is not belonging to one person – it is used by other people. That means that different person who is reviewing my database already understand what's going on.

There is multiple naming conventions, there is not one way to do everything, and it is not required. We are gonna use this naming convention:

`SELECT`

`user` table: `id` (or `user_id`)

06. What is Database Design?

Database Design

Data integrity – all of your data is correct, up-to-date, there is no disconnected data, e.g. if for some reason link between two connected tables is broken, then when one table updates, another table is left behind – data becomes incorrect, that's **data integrity problem**, because the data is not managed properly.

In good designed database we prevent data integrity issues, so all of our data is up-to-date; we don't have repeating data, we don't have old data that don't should be in there – we want database always be up-to-date, unless for some reason we are storing archives, e.g. you have store and you want to store every single order that has been made, or sold. Well, then you can store all of that in a database, and it doesn't need to be updated because it is archive.

But, if you have a customer in your database and the address is out-of-date, it is not updated – then you have a problem.

If your database is designed badly and you have repeating data, well then one example of data can update and the other one is left behind. So, you have like two addresses for one person which just should be one address. That's another example of data integrity problem.

If we are talking about database design as a whole, people usually break it up into 2 or 3 sections:

- ✓ Conceptual,
- ✓ Logical,
- ✓ Physical.

07. Data Integrity

Data integrity is just having correct data in your database; we don't want repeating values, we don't want incorrect values, we don't want broken relationships between tables, e.g.

`User` table – user buys stuff, the stuff is bought by a user

`Sale` table – user buys a product, sale needs a user, so sale is dependent on the user and we have **relationship**.

Do keep in mind that name "relational database" does not come from the word relationship, it comes from relations, which is the mathematical connection of sets, but relational database does have realtionships between tables.

Three main types of Data Integrity:

- ✓ Entity integrity – uniqe identity

  Enitity is anything we store data about, e.g. User has `id` (known as key) used to enforce **uniqueness among entities**, table `USER`: `name`, `phone` – problem if we have users with the same name living on the same address, possessing one home phone number – one person repeating (twice), or two rows for two different persons?! Solution is either to add more columns where the row would be unique (such as SSID), or we can add an ID column. In that case we have two different persons, that's entity integrity – having uniqueness among entities, we have uniqueness with the rows and we have only one user table.

- ✓ Referential integrity – allows us to connect tables with something known as foreign key constraints

When we reference the `id` in another table, e.g. `comments` table and comment that is posted by person from `user` table (`user_id`); if link is broken and person who posted comment is deleted – we have referential integrity problem, old data.

- ✓ Domain integrity – range of what we are storing in the database

Acceptable values for a column for sake, for the range of values we are storing, e.g. phone number should be 10 places/digits (3 area code + 3 + 4 digits for USA) and should be numbers. If we put word "cake" in phone number column – that is domain integrity error, because it is not 10-digit number, it's just a word, that's not proper domain – it is NOT number, and it is not 10 places long.

When we don't have data integrity we have errors, when we have data integrity we do something to correct those errors, such as saying phone number values NEED to be numbers (integer with 10 digits). That's how we implement data integrity by seting database rules.

Data Types: numbers, text, dates – three basic categories

`Char(20)` – implementing data integrity for characters, domain integrity

We are implementing/enforcing connection between tables in the database by using Foreign Key constraints:

`USER` (parent)

`COMMENT` (child) – the child can't exists without the parent, you can't have a comment without a commenter

With using foreing key constraints we can say: "if this user gets removed also remove all of the comments posted by that user", and that is example of foreign key constraint, as for non-repeating data – that is done with just designing you database in the best way.

08. Database Terms

Data – anything we store in our database

Database – that is where we store our data in

Relational Database – specific kind of database that stores data in tables

DBMS – manage the database, using code we control our database

RDBMS – specific kind of DBMS, use to control our tables and values within tables of relational database

Null – e.g. we don't have `fax_number`, that emptiness is said to be null, empty value although by definition it means **no value** – there is no data in that specific field

Anomalies – error within our data integrity, something that goes away from what we are expecting, or from the normal, e.g. when we update something and instead updating one column it updates ten, and it was not supposed to – that's an anomaly

Integrity – we implement database integrity to protect against anomalies:

- ✓ Enity integrity – uniqueness among the table
- ✓ Referential integrity – keeps the connections, the keys – foreign keys, primary keys, they keep them connected across multiple tables
- ✓ Domain integrity – a column within a table has all of the expected values, e.g. if we have phone number we would expect numbers not text, or not a date, and the word "domain" is basically the range of values that are acceptable to store within a column

Those are data terms. Design terms are:

Entity – is anything we are store data about, e.g. user, mortgage, transaction, credit card information, comment

Attributes –

to be continued ...