
FREE eBook
LEARNING
database-design
Free unaffiliated eBook created from
Stack Overflow contributors.
#database-
design

Table of Contents

About

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: database-design

It is an unofficial and free database-design ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official database-design.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapter 1: Getting started with database-design

Remarks

This section provides an overview of what database-design is, and why a developer might want to use it.

It should also mention any large subjects within database-design, and link out to the related topics. Since the Documentation for database-design is new, you may need to create initial versions of those related topics.

Examples

Installation or Setup

Detailed instructions on getting database-design set up or installed.

Read Getting started with database-design online: https://riptutorial.com/database-design/topic/4899/getting-started-with-database-design

Chapter 2: Normalization

Examples

First normal form (1NF)

A relation(or relation schema) in a given database is in `first normal form`, if the domain of all `attributes` of that relation is atomic. A domain is atomic if all the elements of that domain are considered to indivisible units. Suppose a relation *`employee`*, has attribute `name`, then the relation is not in `first normal form`, because the elements of domain of `attribute name` , can be divided into `first name` and `last name`.

In a nutshell, if a relation has composite attributes, then it is not in first normal form. Suppose we have the following relation:

EmpId	first name	last name	salary	position
Deptx-101	John	smith	12000	intermediate
Depty-201	Carolyne	Williams	18900	manager

The `EmpId` of first row can be broken into : `Deptx` (which is used to identiy department) and `101`, is a unique number assigned within the organization. Clearly, the domain of attribute `EmpId` is not atomic and hence our relation is not in `first normal form`.

Disadvantes faced:

1. When such employee id's are used, the department of an employee can be found by writing code that breaks up the structure of `EmpId` into `Deptx` and `101`, which requires extra programming. Also information gets encoded in program rather than in database.
2. Suppose a particular employee has to change department, then the attribute `EmpId`, would have to be updated everywhere it is used.

We can make our relation satisfy `first normal form`, by splitting it into following two relations:

Relation 1

EmpId	first name	last name	department
101	John	smith	Deptx
201	Carolyne	Williams	Depty

Relation 2

EmpId	salary	position
101	12000	intermediate
201	18900	manager

Now, if we have to change the department, we have to do it only once in the relation 1, also determining department is easier now.

Second Normal Form (2NF)

To normalize the database in the second form, there must not be any partial dependency of any column on primary key.

Let's consider the following example:

id	name	dob	subject
1	Mark	1-1-1981	Physics
2	Jack	2-2-1982	Math
2	Jack	2-2-1982	Biology
3	John	3-3-1983	Math

This table is considered to have a composite primary key (*id* and *subject*), but the *name and *dob* columns only depends on the *id*, not the *subject*, so they have partial dependency on the primary key. As a result, we can see the redundancy of information in the table. To normalize the database in the second form, we must split this table into two tables like this:

Students table

id	name	dob
1	Mark	1-1-1981
2	Jack	2-2-1982
3	John	3-3-1983

Subjects table

student_id	subject
1	Physics
2	Math
2	Biology
3	Math

The *student_id* column of the *Subjects* table is a foreign-key that references the primary key *id* of the *Students* table.

Read Normalization online: https://riptutorial.com/database-design/topic/5111/normalization

Credits

S. No	Chapters	Contributors
1	Getting started with database-design	Community
2	Normalization	Racil Hilan, reaanb, sqlvogel, Sumeet Singh