Schema Design

Contents

[1. Schema Design 2](#_Toc185589263)

[Anomalies 2](#_Toc185589264)

[How to do Schema Design in Practice 4](#_Toc185589265)

# Schema Design

* As a software developer, we do prepare design document.
  + Class Diagram - How will you implement application. (LLD)
  + Architectural Diagram - What infrastructure layers will be there. (HLD)
  + Schema Design - What tables will be there in our Database.
* Database Schema:
  + How will you store the data to be able to handle the given set of requirements.
  + Schema is blueprint of a real database. A pictorial representation of how database is going to be structured.
* Why do we need Schema design?
  + Should be able to handle all the requirements.
  + Handle requirements efficiently.
  + Avoid anomalies (issues such as redundancy which can cause inconsistency and more storage).

## Anomalies

* Issue due to redundancy.
* There are 3 types of anomalies...
  + Insertion anomaly
  + Deletion anomaly
  + Update anomaly
* When we have redundancy in database these anomalies happen.

### Insertion Anomaly

* Suppose there is a table Students with id, name, marks, batch\_id, batch\_name as columns as shown below...

A screenshot of a computer screen

Description automatically generated

|  |
| --- |
| CREATE TABLE IF NOT EXISTS students (      id INT PRIMARY KEY,      name VARCHAR(100) NOT NULL,      marks INT CHECK (marks >= 0 AND marks <= 100),      batch\_id INT NOT NULL,      batch\_name VARCHAR(50) NOT NULL  ); |

* New requirement has come: Create a new batch named December 2022. No students in that December 2022 batch yet.
* Since, no students are present in that batch, it is not possible to create a new row (id is primary key, name cannot be null.)
* Is this table Students is correctly created? Answer is no. There is lot of redundancy in the table.
* The batch\_name column has redundancy. Note that 'August 2022' is repeated many times. Because of this redundancy we have insertion anomaly.
* Insertion anomaly is *inability to store data about a particular entity till the time we have data about something else*.

### Deletion Anomaly

* New request has come. We need to delete the student with id 4 (Imagine that this student has registered for June 2022 batch first and is the only student in that batch)
* If we delete that student, the batch information is also deleted. Note that the student with id 4 is in June 2022 and there is only 1 student in that batch.

A screenshot of a computer

Description automatically generated

* Deletion anomaly is, at the time of deleting something, we might end up deleting something else.

### Update Anomaly

* A new requirement has come where we need to change the batch\_name 'August 2022' to 'Aug2022'.
* Imagine, we have written a query to update from 'August 2022' to 'Aug2022'. After updating couple of rows, machine has gone down… The state of table will look as show below…

A screenshot of a computer screen

Description automatically generated

* Update anomaly is, at the time of updating something, we might end up in inconsistencies.

### Database Normalization

* It is the techniques that we use to handle redundancy.
* There is something called normal forms. Normal forms are guidelines used in database design to reduce redundancy and improve data integrity by organizing data into tables.
  + 1NF
  + 2NF
  + 3NF
  + 4NF
  + 5NF
  + 6NF
  + BCNF (Boyce Codd Normal Form)
* We don’t use these NF in practice. We never use normal forms in reality. There is more practical way to approach database design.

## How to do Schema Design in Practice

* Schema design of Scaler.
* Let's list out the requirements.
  + Scaler has multiple students.
  + Scaler has multiple batches. Each student belong to exactly one batch at a time. one batch can have multiple students.
  + Every batch has a current instructor.
  + Every batch has multiple classes.
  + 1 class may involve students from multiple batchs
  + Every student has a student buddy.
  + Every student has a mentor.
  + For every mentor we store their company and number of session the mentor has taken.
  + We have to store for every batch a student belong to, date of joining that batch.
  + For every student for every class we have to store attendance.

#### Steps

1. Find all the nouns that are there in the requirements.
   * Out of all the nouns find the nouns that we want to store information about.
   * Create 1 table for each such noun.
   * Good Practices:
     + Name of each table should ideally plural.
     + Represent table name in snake case. (Example: mentor\_sessions)
   * Nouns identified:
     + students
     + batches
     + instructors
     + classes
     + mentors
     + companies
2. For each of these nouns, find what all we need to store.
   * Create an id column
   * If no realation with another noun, create a column for that (primitive attributes). For example, Students will have name and name has no relation with other nouns. Hence it is a primitive attribute.

|  |  |
| --- | --- |
| CREATE TABLE students (      id INT PRIMARY KEY,      name VARCHAR(100) NOT NULL,      email VARCHAR(100) UNIQUE NOT NULL,      phone\_number VARCHAR(15),      graduation\_year INT  ); | CREATE TABLE batches (      id INT PRIMARY KEY,      name VARCHAR(100) NOT NULL,      start\_date DATE,      number\_of\_students INT  ); |
| CREATE TABLE instructors (      id INT PRIMARY KEY,      name VARCHAR(100) NOT NULL,      email VARCHAR(100) UNIQUE NOT NULL,      average\_rating DECIMAL(3, 2),      years\_of\_experience INT  ); | CREATE TABLE classes (      id INT PRIMARY KEY,      start\_time TIME,      title VARCHAR(100),      endtime TIME  ); |
| CREATE TABLE mentors (      id INT PRIMARY KEY,      name VARCHAR(100) NOT NULL,      email VARCHAR(100) UNIQUE NOT NULL,      number\_of\_mentees INT,      average\_rating DECIMAL(3, 2)  ); | CREATE TABLE companies (      id INT PRIMARY KEY,      name VARCHAR(100) NOT NULL  ); |

1. Now we have to represent relations.
   * But how to represent relation?
   * For relation,
     + Which two tables are related.
     + What relation between entities (Find cardinality).

* What is Cardinality?
  + *Cardinality* refers to the relationship between two entities. It defines how entities in one table relate to entities in another.
  + Let’s say we have Students and Batches
    - 1 Student is allotted to 1 Batches.
    - 1 Batches can have multiple Students.

A graph with a black line

Description automatically generated

* + Let’s say we have Husbands and Wifes table
    - 1 husband has 1 wife.
    - 1 wife has 1 husband.

A graph paper with a rectangle

Description automatically generated

* There are 4 types cardinality :
  1. 1:1
  2. 1:m
  3. m:1
  4. m:m
* Steps to find the cardinality…
  + Let’s say we have Movies and Shows table…
    - Go from Left to right and ask the question 1 movie how many shows?

A graph with a line

Description automatically generated

* + - Go from Right to left and ask the question 1 show can have how many movies?

A graph paper with a grid

Description automatically generated

* + - If there no m, put 1 else put M.

A graph with arrows pointing to the distance

Description automatically generated with medium confidence

* For relation in column, if cardinality is
  + 1:1, id of any 1 side on other side.
  + M:1, id of 1 side on m side. (Because we can have a list on 1 side and we cannot store list)
  + 1:M, id of 1 side on m side. (Because we can have a list on 1 side and we cannot store list)
  + M:M, Mapping table. (We have to create a new table, because there will be list on both sides)

A graph with a line and a line

Description automatically generated with medium confidence

* + - We create a new table, Student\_Course with student\_id, courese\_id as columns.