Agenda

- What is Schema Design
- How to approach Schema Design
- Cardinality
    - How to find cardinality in relations
    - How to represent different cardinalities
- Sparse Relations
- Nuances when representing relations

What is Schema Design

Let's understand what Schema is. Schema refers to the structure of the database. Broadly speaking, schema gives information about the following:

- Structure of a database
- Tables in a database
- Columns in a table
- Primary Key
- Foreign Key
- Index
- Pictorial representation of how the DB is structured.

In general, 'Design' refers to the pictorial reference for solving how should something be formed considering the constraints. Prototyping, blueprinting or a plan, or structuring how something should exist is called Design.

Before any table or database is created for a software, a design document is formed consisting:

- Schema
- Class Diagram
- Architectural Diagram

How to approach Schema Design

Let's learn about this using a familiar example. You are asked to build a software for Scaler which can handle some base requirements.

The requirements are as follows:

1. Scaler will have multiple batches.
2. For each batch, we need to store the name, start month and current instructor.
3. Each batch of Scaler will have multiple students.
4. Each batch has multiple classes.
5. For each class, store the name, date and time, instructor of the class.
6. For every student, we store their name, graduation year, University name, email, phone number.
7. Every student has a buddy, who is also a student.
8. A student may move from one batch to another.
9. For each batch a student moves to, the date of starting is stored.
10. Every student has a mentor.
11. For every mentor, we store their name and current company name.

12. Store information about all mentor sessions (time, duration, student, mentor, student rating, mentor rating).
13. For every batch, store if it is an Academy-batch or a DSML-batch.

Representation of schema doesn't matter. What matters is that you have all the tables needed to satisfy the requirements. Considering above requirements, how will you design a schema? Let's see the steps involved in creating the schema design.

Steps:

1. **Create the tables:** For this we need to identify the tables needed. To identify the tables,

    - Find all the nouns that are present in requirements.

    - For each noun, ask if you need to store data about that entity in your DB.

    - If yes, create the table; otherwise, move ahead.

      Here, such nouns are batches, instructors (if we just need to store instructor name then it will be a column in batches table. But if we need to store information about instructor then we need to make a separate table), students, classes, mentor, mentor session.

      Note that, a good convention about names: Name of a table should be plural, because it is storing multiple values. Eg. 'mentor_sessions'. Name of a column is plural and in snake-case.

2. **Add primary key (id) and all the attributes** about that entity in all the tables created above.

    Expectation with the primary key is that:

    - It should rarely change. Because indexing is done on PK and the data on disk is sorted according to PK. Hence, these are updated with every change in primary key.

    - It should ideally be a datatype which is easy to sort and has smaller size. Have a separate integer/big integer column called 'id' as a primary key. For eg. twitter's algorithm (Snowflake) to generate the key (id) for every tweet.

    - A good convention to name keys is <tablename><underscore>id. For example, 'batch_id'.

      Now, for writing attributes of each table, just see which attributes are of that entity itself. For `batches`, coulmns will be `name`, `start_month`. `current_instructor` will not be a column as we don't just want to store the name of current instructor but their details as well. So, it is not just one attribute, there will be a relation between `batches` and `instructors` table for this. So we will get these tables:

`batches` | batch_id | name | start_month | |----------|------|-------------|

`instructors` | instructor_id | name | email | avg_rating | |---------------|------|-------|------------|

`students` | student_id | name | email | phone_number | grad_year | univ_name | |------------|------|-------|--------------|-----------|-----------|

`classes` | class_id | name | schedule_time | |----------|------|---------------|

`mentors` | mentor_id | name | company_name | |-----------|------|--------------|

`mentor_sessions` | mentor_session_id | time | duration | student_rating | mentor_rating | |-------------------|------|----------|----------------|---------------|

3. **Representing relations:** For understanding this step, we need to look into cardinality.

Cardinality

When two entities are related to each other, there is a questions: how many of one are related to how many of the other.

For example, for two tables students and batches, cardinality represents how many students are related to how many batches and vice versa.

- 1:1 cardinality means 1 student belongs to only 1 batch and 1 batch has only 1 students.
- 1:m cardinality means 1 student can belong to multiple batches and 1 batch has only 1 student.
- m:1 cardinality means 1 student belongs to only 1 batch and 1 batch can have multiple students.
- m:m cardinality means multiple students can belong to multiple batches, and vice versa.

In cardinality, `1` means an entity can be associated to 1 instance at max, [0, 1]. `m` means an entity can be associated with zero or more instances, [0, 1, 2, ... inf].

Quiz 1

What is the cardinality between two users on a social networking site like facebook?

Choices

- ☐ 1:1
- ☐ 1:m
- ☐ m:1
- ☐ m:m

Now, What do you think the cardinality be of a boy and a girl?

How to find cardinality in relations

Between two entites, there can be different relations. For each relation, there are different cardinalities.

For finding cardinality, identify the two entities you want to find cardinality for. Identify the desired relation between them.

A trick to find cardinality: **Example 1:** Let's say we want to find the cardinality between two users on a social networking site. Put them alongside each other and mention the relation between them.

user1	--- friend ---	user2
1	-->	m
m	<--	1

From left to right, 1 user can have how many friends? 1 -> m From right to left, 1 user can be a friend of how many users? m <- 1

If there is `m` on any side, put `m` in final cardinality. Therefore, here the cardinality will be m:m. This trick is based on the concept that, at max how many associations an entity can have.

Example 2: What is the cardinality between ticket and seat in apps like bookMyShow?

ticket	--- books ---	seat
1	-->	m
1	<--	1

In one ticket, we can book multiple seats, therefore, 1 -> m One seat can be booked in only 1 ticket, therefore, 1 <- 1

So, the final cardinality between ticket and seat is 1:m.

Example 3: Consider a monogamous community. What is the cardinality between husband and wife?

husband	--- married to ---	wife
1	-->	1
1	<--	1

In a monogamous community, 1 man is married to 1 woman and vice versa. Hence, the cardinality is 1:1.

Example 4: What is the cardinality between class and current instructor at Scaler?

class	--- assigned to ---	instructor
1	-->	1
m	<--	1

One class can have 1 instructor, therefore, 1 -> 1 One instructor can teach multiple classes, therefore, m <- 1

So, the final cardinality between class and instructor is m:1.

Quiz 2

In a university system, each student can attend various courses during their academic tenure. Simultaneously, courses can be taken by different students every semester. What's the relationship between `Student` and `Course` in terms of cardinality?

Choices

- ☐ One-to-One
- ☐ One-to-Many
- ☐ Many-to-One
- ☐ Many-to-Many

Quiz 3

Considering an e-commerce platform, when a customer places an order, it may contain several products. Many people order popular products. Can you identify the cardinality of the `Order` to `Product` relationship?

Choices

- ☐ One-to-One
- ☐ One-to-Many
- ☐ Many-to-One
- ☐ Many-to-Many

Quiz 4

In a banking application, a customer might have several accounts (like savings, checking, etc.) but each of these accounts can only be owned by a single customer. What type of cardinality does the `Customer` to `Account` relationship exhibit?

Choices

- ☐ One-to-One
- ☐ One-to-Many
- ☐ Many-to-One
- ☐ Many-to-Many

Quiz 5

In an educational institution, a student opts for a major subject. This subject might be the choice of several students, but a student cannot major in more than one subject. How would you describe the cardinality between `Student` and `Major`?

Choices

- ☐ One-to-One
- ☐ One-to-Many
- ☐ Many-to-One

- ☐ Many-to-Many

How to represent different cardinalities

When we have a 1:1 cardinality, the `id` column of any one relation can be used as an attribute in another relation. It is not suggested to include the both the respective `id` column of the two relations in each other because it may cause update anomaly in future transactions.

For 1:m and m:1 cardinalities, the `id` column of `1` side relation is included as an attribute in `m` side relation.

For m:m cardinalities, create a new table called a **mapping table** or **lookup table** which stores the ids of both tables according to their associations.

For example, for tables `orders` and `products` in previous quiz have m:m cardinality. So, we will create a new table `orders_products` to accomodate the relation between order ids and products ids.

`orders_products` | order_id | product_id | | -------- | ---------- | | 1 | 1 | | 1 | 2 | | 1 | 3 | | 2 | 2 | | 2 | 4 | | 3 | 1 | | 3 | 5 | | 4 | 5 |

Sparse relations

The way we were storing 1:m and m:1 cardinalities will lead to wastage of storage space because of sparse relation. A sparse relation is one where a lot of entities are not a part of the relation.

> *A sparse relation refers to a type of relationship or association between tables where there are few matching or related records between the tables. Many records in one table may have no matching records in the related table, resulting in gaps or sparsity in the relationship.*

To solve this, we can create a new table with `id` columns of both tables as done previously. Pros: Saved memory Cons: Need more join operations, may affect performance.

Sparse relations can also happen in 1:1 cardinality. Solution for saving storage space whenever a sparse relation is present is to creat a mapping table.

Nuances when representing relations

Moving forward, there can be cases where we need to store information about the relationship itself. If one of the two tables is storing information about the relationship, it dilutes the purpose of that table.

So, in LLD classes, you will learn about SRP (Single Responsibility Process). The responsibility of everything must be defined. You should always have a separate mapping table for the information about relationship.

Coming back to Scaler example, refer to the tables we created previously:

We did not represent `current_instructor` before. So, what would be the cardinality between `batches` and `current_instructor`? 1 batch can only have 1 `current_instructor`.

But an instructor can be teaching multiple batches at a time (morning batch, evening batch, etc). Hence, this is a m:1 cardinality. So we include the id of `current_instructor` in `batches` table.

`batches` | batch_id | name | start_month | curr_inst_id | |----------|------|------------|--------------|

Similarly, for `batches` and `students`, 1 student can be in 1 batch at a moment, but a batch can have multiple students. So, this is m:1 cardinality, `batch_id` will be included in `students` table.

`students` | student_id | name | email | phone_number | grad_year | univ_name | batch_id | |------------|------|-------|-------------|-----------|-----------|----------|

For `batches` and `classes`, 1 batch can be in multiple classes and 1 class can have multiple batches. Hence, m:m cardinality.

`batch_classes`

batch_id	class_id

Solution to Quizzes:

> *-- Quiz1: Option D (m:m) Quiz2: Option D (Many-to-Many) Quiz3: Option D (Many-to-Many) Quiz4: Option B (One-to-Many) Quiz5: Option C (Many-to-One) --*

Will see you all again, thanks!