Data Design

Contents

Overview

The design of a database - whether it is the *data* within it or the *schema* of it - can be done through several concepts in MySQL.

Data Types

There are three main *data types*, all of which have their own sub-types:

- **Numeric** – data used for calculations
- **Textual** – data used for non-calculated values (postcodes, email addresses, etc.)
- **Date/Time** – handy for rubber-stamping and organising data

These data types are assigned to the fields within database tables.

From here, we can construct a baseline constraint for the type of data we store in each table.

Numeric Data

Numeric data is data which is used for calculations, such as prices, quantity, etc.

Below is a list of the most common numeric data types that can be used in MySQL:

Type	Min storage value	Max storage value	Example
BIT	1	64	63
BOOLEAN	-128	127	0 or 1(f/t)
TINYINT	-128	127	47
SMALLINT	-32768	32767	31337
MEDIUMINT	-8388608	8388607	7145531

Type	Min storage value	Max storage value	Example
INT	-2147483748	2147483747	12
BIGINT	-9223372036854775808	922337203685477507	62547245237
DECIMAL	[lower than big int]	[higher than big int]	35D.30D
FLOAT / DOUBLE	[hardware dependent]	[hardware dependent]	BIT

Textual Data

Textual data is data which isn't used for calculations, but rather just information we wish to contain, such as names, addresses, phone numbers, etc.

Below is a list of some of the different text-based data types that can be used in MySQL:

Type	Format	Style
CHAR	0-255 characters	Fixed
VARCHAR	0-65535 characters	Variable

`CHAR`s are fixed - if a field is of type CHAR(20), the length of that field will always be 20.
`VARCHAR`s are not fixed, which saves on memory and is easier to search for.

There are other text-based data types that are stored in binary format, instead of characters.

These are more efficient due to less overhead, so they don't need to store data in the character set.

Date and Time Data

Date and time data is good for storing information on timestamps, such as order placed date/time, birthdays, etc.

Type	Format	Example
DATE	YYYY-MM-DD	1995-02-29
DATETIME	YYYY-MM-DD HH:MM:SS.SSSSS	2009-08-14 16:20:35.20342
TIMESTAMP	Seconds since epoch	2016-04-21 09:53:23.92381
TIME	HH:MM:SS.SSSSS	15:00:43.00001
YEAR	YYYY	1993

*(note: the **epoch** is a form of computer time that officially began at 1970-01-01 00:00:00.00001(UTC))*

Constraints

Database constraints allow for further control over our data outside of data types.

The most common data constraints include:

- `UNIQUE` – all values entered into this field must not be the same as others

- `NOT NULL` – a field has to be filled in, it cannot be left empty
- `DEFAULT` – this field will have a certain 'standard' value assigned as a default
- `PRIMARY KEY` and `FOREIGN KEY` – a combination of unique and not null – the table can have a primary key to identify it

Unique

The `UNIQUE` constraint prevents two records from having the same values in a column.

For example, the email address field in a customer table could be unique, ensuring that no two customers have the same email.

Not Null

The `NOT NULL` constraint ensures that a field must have an entry.

For example, fields like name and address in a customer table may be required for a record, meaning we would make them `NOT NULL`.

Default

The `DEFAULT` constraint allows us to set a pre-defined value for new records.

For example, the status field in an order table may have the default value of processing, which can then be updated as the order goes through different stages.

Database Keys

Within our database, we typically want a way to uniquely identify a record quickly.

For example, we can't uniquely identify two records that have the name Jeff, and the age of 23; however, if each of these records had a unique field, we can identify them!

Keys allow us to uniquely identify records, and subsequently associate records from different tables to build relationships.

Primary Keys

A *primary key* is a field in a table that uniquely identifies a record in a table.

Generally, the field with the primary key is considered most important piece of information in the table. As such, it is nearly always given to IDs.

customer_id	name	age
1	Jeff	23
2	Cyrus	25
3	John	21

It is a combination of the `UNIQUE` and `NOT NULL` constraints.

A table can only have one primary key.

Primary keys may consist of single or multiple fields; when multiple fields are used as a primary key, they are called a *composite key*.

Foreign Keys

Foreign keys allow us to link tables together and start creating a relational database!

A foreign key is a field in a table that refers to a primary key in a different table.

For example, we could have a field in our order table that refers to the `customer_id` in the `customers` table - this way, we can see which customer has made an order!

order_id	fk_customer_id	date_placed
1	2	2020-03-01
2	1	2020-04-01
3	2	2020-05-01

When setting up a foreign key, you will need to use the `REFERENCES` keyword alongside it:

```
CREATE TABLE orders (
  order_id INT PRIMARY KEY AUTO_INCREMENT,
  fk_customer_id INT NOT NULL,
  placed DATE NOT NULL,
  FOREIGN KEY (fk_customer_id) REFERENCES customers(customer_id)
  );
```

Here, we've created an `orders` table, which contains a foreign key `fk_customer_id` that references the `customer_id` primary key in the `customers` table.

We first created the column with its usual constraints, before then assigning it as a foreign key.

Tutorial

There is no tutorial for this module.

Exercises

Add further data design principles to the games shop database design you created.

This includes adding data types to each field, any relevant constraints and primary/foreign keys.

▶ Solution