Michael Balian
Charishma Challurubadri

Restaurant DB Docs

The restaurant MySQL database runs on a local server that is created by the XAMPP software, this software also runs an apache server which helps us serve the created UI documents. The database can be accessed two ways, either through a programming language library like with MySQLi for PHP or through the local host interface called “PHPAdmin”. The admin page has similar functionality as the MySQL Workbench, you can interface with various databases by adding, deleting, updating various tables and rows with MySQL commands. The engine our database utilizes is InnoDB which is the default for databases created in PHPAdmin. The restaurant database consists of three main tables: Location, Section, and Item table.

The first table and most foundational one is the Location table. This table is the main identifier when it comes to bucketing different locations with their own sections and items. The primary key for this table is “ID”, which is set to auto increment when rows are inserted into the table. We have set various constraints to follow the business logic we have in place, for example one constraint in place is the state of the location should be two characters long as this is the valid format for state abbreviation. Another constraint we have in place is, for the phone number to be 10 characters long and to only be characters 0-9 as we do not accept any other characters other than digits for a phone number. We have one unique column definition which is the “Location_Name”, this prevents any duplicate location names created in the database. Lastly for event tracking and auditing purposes we have created two columns which will track the created datetime for the row and also the datetime of when the row is updated.

The second table in the database is the section table, this is where the sections of the menu will be stored. The section table primary key is the “ID” column, which is also set to auto increment. We have a column called “Location_ID” which is the foreign key that corresponds to the Location “ID” column. We have defined this foreign key at the end of the table creation and have set a cascade for delete or update actions on the location row corresponding to this foreign key. For example if the corresponding location row is deleted that action will also cascade down to delete all the sections that have a foreign key “Location_ID” corresponding to the location “ID”. We also have the same timestamp columns as the location table which can help us track the datetime of when the rows were inserted and updated.

Lastly we have the item table, it has a primary key which is the “ID” column which is set to auto increment. We have one constraint set on the “Price” column, this helps check that a row inserted has a positive number for the price as we can never have negative numbers to satisfy the business logic in place. The foreign key for the item table is the “Section_ID” which refers to the “ID” column in the section table. We have also set cascades on the update and delete actions just like the section table. The same timestamps are also in place just like the location and section table.

To close out the details to this database design, we have set some indexes to have efficient performance when querying the database tables. The indexes we have in place are for the “Location_Name”, “Section_Name”, “Item_Name”, and “Price”. Some might wonder why we do not set indexes for the primary and foreign keys as we use that column the most for queries, this is because by default they are already indexed in the database.