GÖTÜR

Database Management Systems and Big Data Final Project

DECEMBER 27, 2021

528211024 MURAT CAN BEKLER
528211015 İLYAS ÇOBAN

INDEX

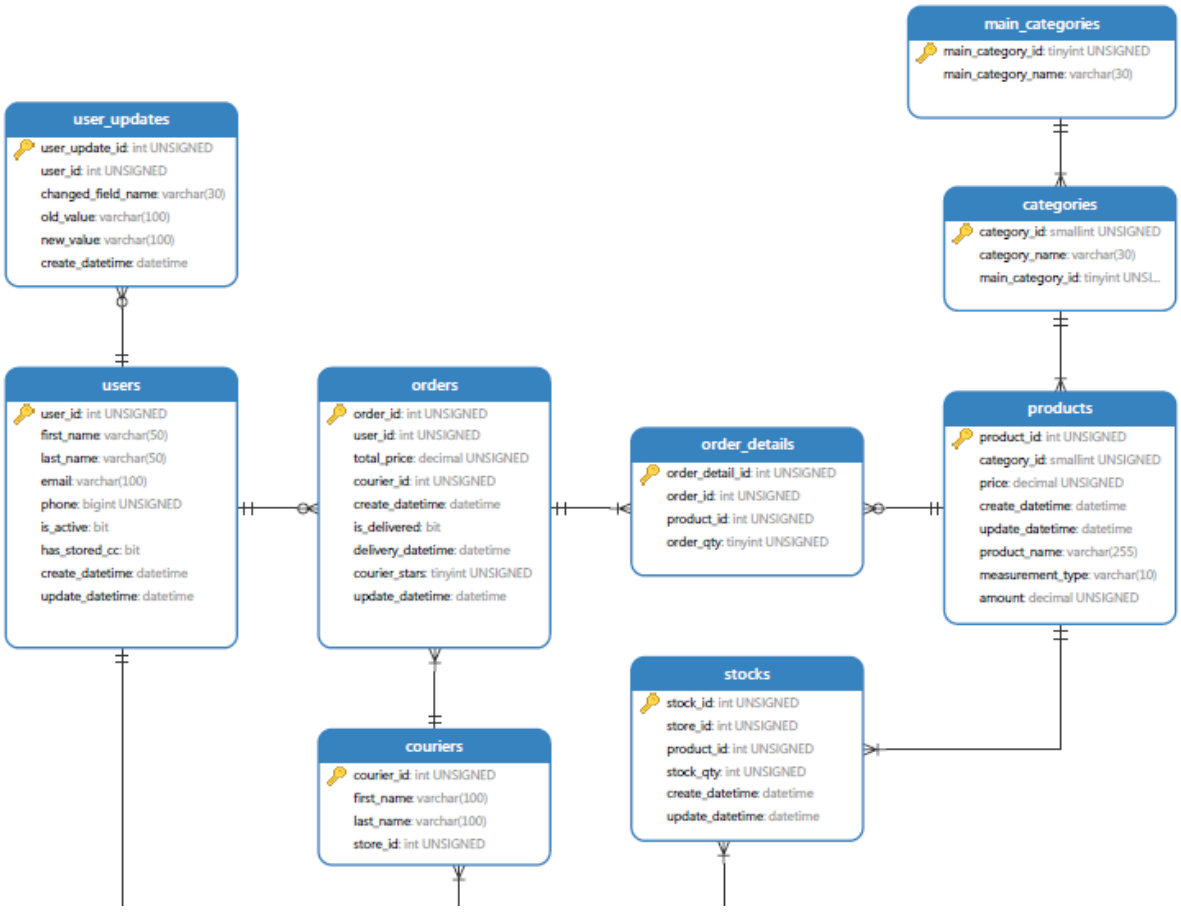
1. Project Description

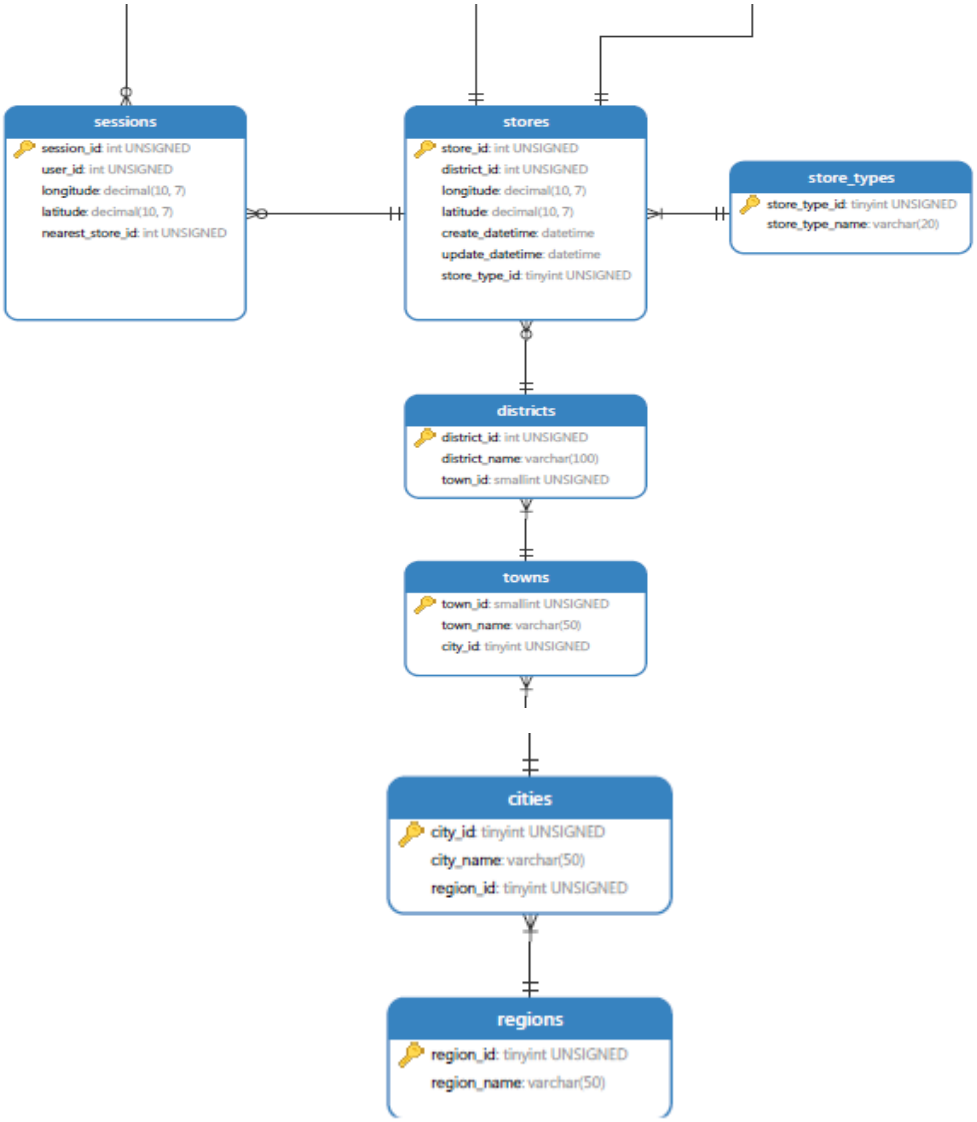
The database has been designed and created for the service called Götür. This database allows the users see the nearest courier on their smart phone, see the nearest Götür shop and the shop's stock, give points to the courier after delivery. Also allows business units monitor daily operations such as order frequency analysis, stock controls for areas.

There are 16 different tables in this database. In the next sections, you can find the information about the names of the columns for each table, properties of the columns, the relation types between tables, the constraints that enable the relations between tables.

Furthermore, you can find the Entity Relational Diagram (ERD), the Indexes, the Stored Procedures and Views which are defined for increasing query performance.

2. Entity Relationship Diagram (ERD)

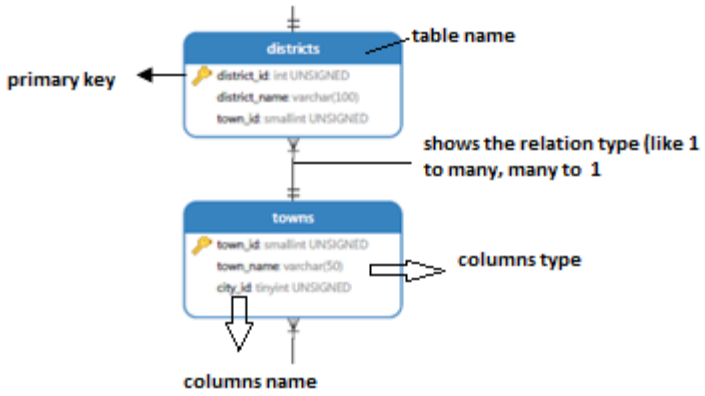




The ERD of the whole table in the database is shown above.[1] You can find an example explanation of the schema structure used in the ERD.

Not: Due to bool data type can not entered in the software used for ERD drawing, bit data type is used instead of bool data type. But in the codes where the tables are created, the bit data type converted as bool data type

[1] ERD has been pasted into the document in separate parts for readability. There is a relationship between the tables: users – sessions, couriers – stores, stocks – stores, towns – cities



3. Tables

3.1 Table's Lists

All the tables in the Götür database are listed below

3.1.1 user_updates: Any changes (email address change, phone number change, etc.) made by each registered user in the system are stored in this table. It is designed in such a way that a connection can be established with the users table via the user_id field

3.1.2 users: It is the table where the information of each registered user in the system is kept.

3.1.3 orders: It is the table where the order information given by the users registered in the system is kept. It is designed in such a way that a connection can be established with the users table via the user_id field.

3.1.4 order_details: It is the table where the basket details of each order placed in the system are found. It is designed in such a way that a connection can be established with the orders table via the order_id column.

3.1.5 products: It is the table containing the products defined in the system and the information about these products. It is designed in such a way that a connection can be established with the order_details table via the product_id column.

3.1.6 categories: It is the table that contains the information about the categories of the products in the system. It is designed to be able to connect with the products table via the category_id column.

3.1.7 main-categories: It is the table that contains the information of the main categories of the categories in the system. It is designed in such a way that the categories table can be linked via the main_category_id column.

3.1.8 stocks: It is the table where the information about the stocks in the system is kept. It can be connected to the products table via the product_id column and to the stores table via the store_id column.

3.1.9 couriers: It is the table that contains the information of the couriers registered in the system. The orders table and courier_id can be linked to the stores table via store_id.

3.1.10 sessions: It is the table where the session information of the users registered in the system is kept. The users table can be linked via user_id, and the stores table via nearest_store_id. (Nearest_store_id column will be found by calculating the latitude longitude information in the tables)

3.1.11 stores: It is the table where the store information registered in the system is kept. It can be connected via the column store_id with the couriers table, the store_id with the stocks table, and the district_id column with the district table.

3.1.12 store_types: It is the table where the type of stores registered in the system is kept. It can be linked to the stores table via the store_type_id column.

3.1.13 districts: It is the table where the information of the districts[2] registered in the system is kept. It can be connected to the stores table via the district_id column.

3.1.14 towns: It is the table where the information of the registered towns[3] in the system is kept. It can be connected with the districsts table via the town_id column.

3.1.15 cities: It is the table where the information about the cities[4] registered in the system is kept. It can be connected with the towns table via the city_id column.

3.1.16 regions: It is the table where the information of the regions[5] registered in the system is kept. It can be connected with the cities table via the region_id column.

3.2 Columns Definitions

user_updates:

user_update_id: Shows the ID information assigned to the change.
user_id: Shows the user_ID who made the change.
changed_filed_name: Shows the name of the modified field.
old_value: Shows the value of the relevant field before the change.
new_value: Shows the value of the relevant field after the change.
create_datetime: It shows the time of the change in datetime format.

users:

user_id: Shows the ID information assigned to the user.
first_name: Shows the first name of the user
last_name: Shows the last name of the user
email: Shows the e-mail of the user
phone: Shows the phone number of the user
is_active: Shows the activity level of the user. (1 active, 0 not active users.)
has_stored_cc: Shows the stored CC information of the user. (1 stored CC, 0 not stored CC.)
create_datetime: Shows the date the user was created
update_datetime: Shows the last date the user made changes to their information. If there is no change, the default value of 1970-01-01 is set.

[2] mahalle
[3] ilçe
[4] şehir
[5] bölge

orders:
order_id: Shows the ID information assigned to the order
user_id: Shows the ID information of the user who created the order
total_price: Shows the total price of the order.
courier_id: Shows the ID information of the courier who will forward the order to the user.
create_datetime: Shows the creation date of the order in datetime format.
is_delivered: Shows the status of the order being delivered to the user (1 delivered, 0 not delivered).
delivery_datetime: Shows the time when the order is delivered to the user in datetime format.
courier_stars: Shows the point information given by the user for the courier after the delivery of the order.
update_datetime: Shows the time of the update in datetime format if any.

order_details:
order_detail_id: Shows the ID information assigned to the order detail.
order_id: Shows the ID information of the order that creates the order detail.
product_id: Shows the product ID information that creates the order detail.
order_qty: Shows the product quantity information that creates the order detail.

products:
product_id: Shows the ID information assigned to the product.
category_id: Shows the category ID information of the product.
price: Shows the price information of the product in TL.
create_datetime: Shows the creation date of the product's record in datetime format.
update_datetime: Shows the update date of the product's record in datetime format.
product_name: Shows the name information of the product.
measurement_type: Shows the unit of measure of the product (indicated in gr/ml).
amount: Shows the weight information of the product.

categories:
category_id: Shows the ID information assigned to the category.
category_name: Shows the name of the category.
main_category_id: Shows the main category ID information under which the category is located.

main_categories:
main_category_id: Shows the ID information of the main category.
main_category_name: Shows the name of the main category.

stocks:
stock_id: Shows the assigned ID information for Stock.
store_id: Shows the ID information of the store where the stock is located.
product_id: Shows the ID information of the product.
stock_qty: Shows the stock quantity.
create_datetime: Shows the creation date of the stock record in datetime format.
update_datetime: Shows the update date of the stock's record in datetime format.

couriers:
courier_id: Shows the assigned ID for the courier.
first_name: Shows the first name of the courier.
last_name: Shows the last name of the courier.
store_id: Shows the store ID information that the courier is connected to.

sessions:
session_id: Shows the assigned ID information for the session.
user_id: Shows the information of which user the session belongs to.

longtitute: Shows the latitude at which the session started.
latitude: Shows the latitude at which the session started.
nearest_store_id: Shows the user the ID of the nearest store when the session starts.

stores:
store_id: Shows the assigned ID information for the Store.
district_id: Shows the ID information of the district where the store is located.
longtitude: Shows the latitude information where the store is located.
latitude: Shows the longitude information where the store is located.
create_datetime: Shows the creation date of the store's record in datetime format
update_datetime: Shows the update date of the store's record in datetime format.
store_type_id: Shows the type of store (2: mobile store, 1 built-in store)

store_types:
store_type_id: Shows the assigned ID information for the Store type.
store_type_name: Shows the name of the Store type.

districsts:
district_id: Shows the assigned ID for the District.
district_name: Shows the name of the district.
tonw_id: Shows the town information where the distributor is located.

town:
town_id: Shows the assigned ID information for the Town.
town_name: Shows the name of the town.
city_id: Shows the city information that the town is connected to.

cities:
city_id: Shows the assigned ID information for the City.
city_name: Shows the name of the city.
region_id: Shows the region information that the city is connected to.

regions:
region_id: Shows the assigned ID information for Region.
region_name: Shows the name of the region.

3.3 Foreign Key Relations

In this section, it is explained on which columns the foreign key relationship between the tables is defined. Relationships will be displayed in the following format: "**fk tanımlanan tablo.fk tanımlanan kolon – fk referans tablosu. fk referans kolon**". For example categories.main_category_id – main_categories.main_category_id expression means is:
"The main_category_id column in the categories table is defined as a foreign key by referring to the category_id column in the main categories table. The main_category_id column in the Categories table cannot take any value other than the values of the main_category_id column in the main_categories table."

Note: Foreign keys are created with an alter table in the syntax of the program called "Navicat" used during the creation of the codes, so foreign key definitions are not included in the code where the tables are created.

categories.main_category_id – main_categories.main_category_id
cities.region_id – regions.region_id
districts.town_id – towns.town_id
order_details.order_id – orders.order_id
order_details.product_id – products.product_id
orders.courier_id – couriers.courier_id
products.category_id – categories.category_id

sessions.user_id – users.user_id
sessions.nearest_store_id – stores.store_id
stocks.store_id – stores.store_id
stocks.product_id – products.product_id
stores.store_type_id – store_types.store_type_id
stores.district_id – districts.district_id
couriers.store_id – stores.store_id
towns.city_id – cities.city_id
user_updates.user_id – users.user_id
orders.user_id – users.user_id

3.4 Columns Data Types

While designing the database, the column types selected for the columns in the tables were chosen according to the purpose of using the data.
Since the lengths of columns such as category_name, old_value are unknown, they are defined as varchar.
Numeric fields that cannot be negative, such as price, ID, are defined as int UNSIGNED.
Numeric fields consisting of 11 characters such as phone are defined as bigint.
Fields that take 1-0 values such as is_active are defined as bool (it is shown as bit because there is no bool in the program used for ERD drawing, it has been changed to bool in sql codes.)

4. Indexes

3 indexes have been designed considering the areas that users will use frequently in the database. The details of the indexes used are shown below.

1. orders.creeate_datetime: The relevant index has been created so that users can quickly examine the orders placed on certain dates in detail. In this way, business units will be able to quickly access the total amount of orders placed on any date and have information about the company's financials.

```
CREATE INDEX idx_order_datetime ON orders (create_datetime);
```

2. users.is_active: The relevant index was created in order to quickly test how many active users are in the system on a certain date. In this way, before making any advertising investment or campaign, business units will be able to see how many users this investment has the potential to reach.

```
CREATE INDEX idx_user_active ON users (is_active);
```

3. orders.delivery_datetime: The relevant index has been created in the system in order to quickly determine the delivery times of orders created on a certain date. In this way, business units will be able to increase customer satisfaction by having information about the average delivery time and by deciding which branch couriers need to increase their performan

```
CREATE INDEX idx_delivery_datetime ON orders (delivery_datetime);
```

Indexes will be updated periodically as new data is added to the system.

5. Store Procedures

3 stored procedures have been designed considering the areas that users will use frequently in the database. The details of the stored procedures used are shown below.

1. sp_StockCheck: The relevant stored procedure will be used to list the products that are less than a certain amount in the system. In this way, products that are decreasing in stocks will be quickly detected and ordered by

business units.

```
DROP PROCEDURE IF EXISTS gotur.sp_StockCheck;

DELIMITER $$
$$
CREATE PROCEDURE gotur.sp_StockCheck(IN quantity int)
BEGIN
                SELECT
                mc.main_category_name AS main_category,
                c.category_name AS category,
                p.product_name AS product,
                s.stock_qty AS stock_quantity,
                s.update_datetime AS last_stock_update_time
        FROM stocks s
        LEFT JOIN products p ON s.product_id = p.product_id
        LEFT JOIN categories c ON c.category_id = p.category_id
        LEFT JOIN main_categories mc ON c.main_category_id = mc.main_category_id
        WHERE s.stock_qty < quantity
        ORDER BY 4, 5;
END$$
DELIMITER ;
```

2. sp_ UserInfoUpdate: The related stored procedure allows the update_datetime and changed field to be updated after the changes to be made in the users table. It calls another stored procedure to add the same record to the

user_updates table.

```
DROP PROCEDURE IF EXISTS gotur.sp_UserInfoUpdate;

DELIMITER $$
$$
CREATE PROCEDURE gotur.sp_UserInfoUpdate(IN user_id_input int, IN column_name varchar(100), IN new_value varchar(100))
BEGIN

	DECLARE old_value varchar(100);
	DECLARE update_time datetime;
	SET update_time = current_timestamp();

	CASE column_name
		WHEN 'email' THEN
			SELECT email INTO old_value FROM users WHERE user_id = user_id_input;

			UPDATE users u
			SET email = new_value, update_datetime = update_time
			WHERE u.user_id = user_id_input;
		WHEN 'phone' THEN
			SELECT phone INTO old_value FROM users WHERE user_id = user_id_input;

			UPDATE users u
			SET phone = CAST(new_value AS UNSIGNED), update_datetime = update_time
			WHERE u.user_id = user_id_input;
		WHEN 'first_name' THEN
			SELECT first_name INTO old_value FROM users WHERE user_id = user_id_input;

			UPDATE users u
			SET first_name = new_value, update_datetime = update_time
			WHERE u.user_id = user_id_input;
		WHEN 'last_name' THEN
			SELECT last_name INTO old_value FROM users WHERE user_id = user_id_input;

			UPDATE users u
			SET last_name = new_value, update_datetime = update_time
			WHERE u.user_id = user_id_input;
		END CASE;
	CALL sp_UserUpdatesInsertNewRecord(user_id_input, column_name, old_value, new_value, update_time);
END
 $$
DELIMITER ;
```

3. sp_UserUpdatesInsertNewRecord: The relevant stored procedure was created in the user_updates table to call from the other stored procedure to discard the old and new records.

```
DROP PROCEDURE IF EXISTS gotur.sp_UserUpdatesInsertNewRecord;

DELIMITER $$
$$
CREATE PROCEDURE gotur.sp_UserUpdatesInsertNewRecord(IN updated_user_id int, IN updated_field_name varchar(30), updated_old_value varchar(100),
                                                     updated_new_value varchar(100), IN update_time DATETIME)
BEGIN
        INSERT INTO     user_updates(user_id, changed_field_name, old_value, new_value, create_datetime) VALUES
        (updated_user_id, updated_field_name, updated_old_value, updated_new_value, update_time );
END $$
DELIMITER ;
```

6. Views

1 view has been designed considering the query patterns that users will frequently use in the database. The detail of the view, in which the top10 cities are listed in descending order according to the order quantity, is shown below.

```
-- Top 10 Cities by Order Amount
CREATE VIEW v_top_10_city_order_amount AS (
        SELECT
                c.city_name
                , count(DISTINCT o.order_id) AS order_count
                , SUM(o.total_price) AS total_order_price
        FROM cities c
        LEFT JOIN towns t ON t.city_id = c.city_id
        LEFT JOIN districts d ON d.town_id = t.town_id
        LEFT JOIN stores s ON s.district_id = d.district_id
        LEFT JOIN couriers co ON co.store_id = s.store_id
        LEFT JOIN orders o ON co.courier_id = o.order_id
        GROUP BY 1
        ORDER BY 3 DESC
        LIMIT 10
);
```