Final Project:

Database of Orders for Shipping Corporation

Nguyễn Tuấn Minh (Pigeon)

Database Management and Design

Professor Nanette Veilleux Simenas

January 10, 2022

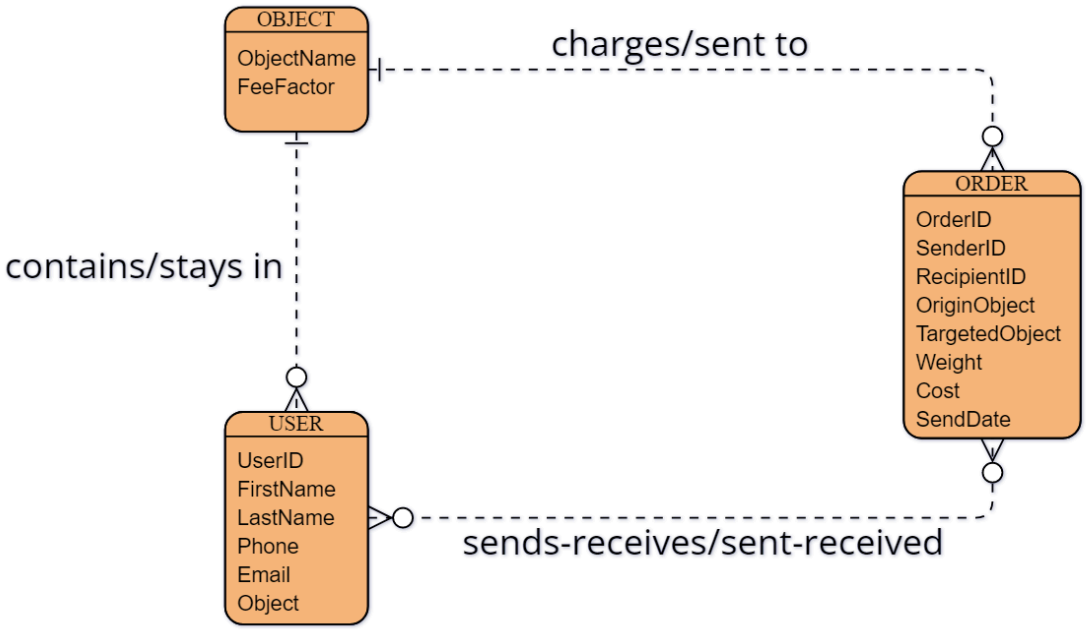
A.Description of the database.

I. The goal of the database.

The Solar System Transporting Corporation handles shipping orders among the three cosmos objects Earth, Moon, and Mars since the success in building habitable environments and waves of migration from Earth in 2378. SSTC's system is separated into Ground Service and Sky Service. For each order, after receiving information from the Ground Team, the Sky Team just has to focus on the targeted cosmos objects, the sender and receiver name, contact info, send dates of orders, and total delivering fee.

Each of the mentioned cosmos objects has its own charging rule based on the weight of arrival order. For example, each package delivered to Earth will be charged $9 per kilogram, while it is $5 on Mars and $15 on Moon.

II. The entity-relationship model.



III. A brief description of the entities, their attributes, and the relationships.

- The entities are OBJECT, ORDER, and USER.

- The attributes of OBJECT are:
    + ObjectName, which is the name of a habitable planet or satellite in the solar system. This is the identifier of OBJECT.
    + FeeFactor, which is the amount of money that a targeted Object charges per kilogram of the weight of each order.
- The attributes of ORDER are:
    + OrderID, which is the unique number identifying each specific order being processed in the system. This is the identifier of ORDER.
    + SenderID, which is the UserID of the person who sent the order from the origin object.
    + RecipientID, which is the UserID of the person who would receive the order in the targeted object.
    + OriginObject, which indicates the object that the order is being sent from. This could be identified through the information of the Sender.
    + TargetedObject, which indicates the object that the order is being sent to. This could be identified through the information of the Recipient.
    + SendDate, which is the time that the order left the origin object.
    + Weight, which is the total weight of items being sent.
    + Cost, which is the fee that Sender has to pay based on the Weight of the order and FeeFactor of the targeted object.
- The attributes of USER are:
    + UserID, which is an unique number identifying a specific person has registered in the system of SSTC and is living on one of the three objects. This is the identifier of USER.
    + FirstName, which indicates the first name of each user.

+ LastName, which indicates the last name of each user.
+ Phone, which indicates the contact channel through the phone number of each user.
+ Email, which indicates the contact channel through the email address of each user.
+ Object, which is the object that the user is currently living on.

- The cardinalities are:
  + OBJECT - USER: 1:N and M:O
  + USER - ORDER: N:M and M:M
  + OBJECT - ORDER:: 1:N and M:O
- The relationships are
  + OBJECT - USER: contains/stays in.
  + USER - ORDER: sends-receives/sent-received.
  + OBJECT - ORDER: charges/sent to.

IV. Schemas of normalized tables.

- The schema for the Solar System Transporting Corporation's order database is:
  + OBJECT(ObjectName, FeeFactor)
    - The column characteristics are:

Column Name	**Type**	**Key**	**Required**	**Remarks**
ObjectName	Character	Primary Key	Yes	
FeeFactor	Integer	No	Yes	

  + ORDER(OrderID, SenderID, RecipientID, SendDate, Weight, Cost)
    - The column characteristics are:

Column Name	Type	Key	Required	Remarks
OrderID	Character	Primary Key	Yes	
SenderID	Character	Foreign Key	Yes	REF: USER
RecipientID	Character	Foreign Key	Yes	REF: USER
SendDate	Date	No	Yes	
Weight	Integer	No	Yes	
Cost	Integer	No	Yes	

+ USER(UserID, FirstName, LastName, Phone, EmailAddress, Object)

- The column characteristics are:

Column Name	Type	Key	Required	Remarks
UserID	Character	Primary Key	Yes	
FirstName	Character	No	Yes	
LastName	Character	No	Yes	
Phone	Character	No	Yes	
EmailAddress	Character	No	Yes	
Object	Character	Foreign Key	Yes	REF: OBJECT

V. A demonstration of the normalization process/check on the tables.

- Normalization check on OBJECT:

+ The functional dependency in this table is:

ObjectName => FeeFactor

FeeFactor => ObjectName

+ 1NF: The table OBJECT is on 1NF because:

        - This table has a defined primary key, which is ObjectName.
        - The rows contain data about the entity OBJECT.
        - The columns contain only data about attributes of OBJECT.
        - All entries in each column are of the same kind, as defined in the table of column characteristics in the section above.
        - Each column has a unique name.
        - Cells of the table hold a single value.
        - The order of the columns is unimportant.
        - There are no identical rows.
    + 2NF: The table OBJECT is on 2NF because:
        - It is on 1NF.
        - The primary key determines all non-key attributes. In this case, the primary key ObjectName functionally determines the non-key FeeFactor.
    + 3NF: The table OBJECT is on 3NF because:
        - It is on 2NF.
        - There is only one non-key attribute in this table so it cannot determine any non-key attribute.
    + BCNF: The table OBJECT is on BCNF because
        - It is on 3NF.
        - Every determinant in the table is a candidate key. Both ObjectName and FeeFactor determine each other, and are candidate keys.
- Normalization check on ORDER:
    + The functional dependency in this table is:
        - OrderID => (SenderID, RecipientID, SendDate, Weight, Cost)

+ 1NF: The table ORDER is on 1NF because:
    - This table has a defined primary key, which is OrderID.
    - The rows contain data about the entity ORDER.
    - The columns contain only data about attributes of ORDER.
    - All entries in each column are of the same kind, as defined in the table of column characteristics in the section above.
    - Each column has a unique name.
    - Cells of the table hold a single value.
    - The order of the columns is unimportant.
    - There are no identical rows.
+ 2NF: The table ORDER is on 2NF because:
    - It is on 1NF.
    - The primary key determines all non-key attributes. In this case, the primary key OrderID functionally determines all of the other non-key attributes including SenderID, RecipientID, SendDate, Weight, and Cost.
+ 3NF: The table ORDER is on 3NF because:
    - It is on 2NF.
    - There is no functional dependency among the non-key attributes, which means that there is no non-key attribute that determines another non-key attribute.
+ BCNF: The table ORDER is on BCNF because
    - It is on 3NF.
    - The only determinant in the table is OrderID - the primary key of ORDER, which means that every determinant in the table is candidate

key.

- Normalization check on USER:
    + The functional dependency in this table is:
        - UserID => (FirstName, LastName, Phone, EmailAddress, Object)
    + 1NF: The table USER is on 1NF because:
        - This table has a defined primary key, which is UserID.
        - The rows contain data about the entity USER.
        - The columns contain only data about attributes of USER.
        - All entries in each column are of the same kind, as defined in the table of column characteristics in the section above.
        - Each column has a unique name.
        - Cells of the table hold a single value.
        - The order of the columns is unimportant.
        - There are no identical rows.
    + 2NF: The table USER is on 2NF because:
        - It is on 1NF.
        - The primary key determines all non-key attributes. In this case, the primary key UserID functionally determines all of the other non-key attributes including FirstName, LastName, Phone, EmailAddress, and Object.
    + 3NF: The table USER is on 3NF because:
        - It is on 2NF.
        - There is no functional dependency among the non-key attributes, which means that there is no non-key attribute that determines another non-key attribute.

+ BCNF: The table USER is on BCNF because
    - It is on 3NF.
    - The only determinant in the table is UserID - the primary key of USER, which means that every determinant in the table is candidate key.

B. The SQL commands. (The C-level)

1. Write a view in your database that makes it easy to access some particularly useful collection of data or particularly painful SQL code, e.g. the JOIN in B-1 below. Query that view.
    - The SSTC discovered that items used in many cases of severe security breaches were sent from Mars in 2422. Therefore, they decided to investigate all shipping orders being sent in this year and the information of related Mars customers. The SQL to display the mentioned data is:
        - ```
          CREATE VIEW TransactionsAndCustomersFromMars2422 AS

          SELECT `ORDER`.OrderID, `ORDER`.SenderID,
          `USER`.FirstName, `USER`.LastName, `USER`.Phone,
          `USER`.EmailAddress, `ORDER.SendDate,
          `ORDER`.Weight, `ORDER`.Cost

          FROM `ORDER` JOIN `USER`

          ON (`ORDER`.SendDate LIKE "2422%" AND
          `ORDER`.SenderID=`USER`.UserID AND `USER`.Object =
          "Mars");
          ```

2. Write a query that checks for functional dependencies. Try (as in previous
          ```

assignments) inserting a typo and find it with this query. Then try something that you know is a correct functional dependency, query it with a correlated subquery to check it and report that it is correct (hint: what is the expected result if the functional dependency exists?)

- The primary key of the BCNF-table USER is UserID, which means that the only functional dependency in this table is UserID => (FirstName, LastName, Phone, EmailAddress, Object).
- The SQL command to check the above functional dependency is:
  - ```
    SELECT  U1.UserID, U1.FirstName, U1.LastName,
    U1.Phone, U1.EmailAddress, U1.Object

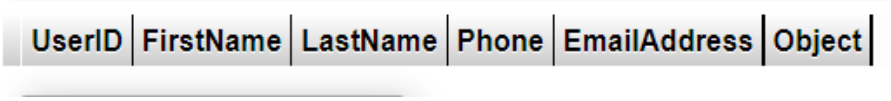
    FROM USER U1

    WHERE U1.UserID IN (SELECT  U2.UserID

    FROM USER U2

    WHERE U1.UserID = U2.UserID

    AND (U1.FirstName <> U2.FirstName OR U1.LastName <>
    U2.LastName OR U1.Phone <> U2.Phone OR
    U1.EmailAddress <> U2.EmailAddress OR U1.Object <>
    U2.Object));
    ```
  - The result here is a empty set.



- The primary key constraint is removed from the USER table for the below test,
    ```

where there are duplicate values inserted into the UserID column and values with typos in corresponding columns to new UserID values.

- `INSERT INTO `USER` (`UserID`, `FirstName`, `LastName`, `Phone`, `EmailAddress`, `Object`) VALUES ('20.210.001', 'Amir', 'Jaskolski', '472.323.935', 'tremayne26@burgas.vip', 'Earth'), ('20.210.001', 'Amie', 'Jaskolski', '472.323.935', 'tremayme26@burgas.vip', 'Earth');`
- The result of functional dependency check is now:

UserID	FirstName	LastName	Phone	EmailAddress	Object
20.210.001	Amir	Jaskolski	472.323.935	tremayne26@burgas.vip	Earth
20.210.001	Amie	Jaskolski	472.323.935	tremayme26@burgas.vip	Earth
20.210.001	Amie	Jaskolski	472.323.935	tremayne26@burgas.vip	Earth

C. The B-level

Your database has at least two entities (tables) in an 1:N relationship and probably you didn't constrain any mandatory children. Otherwise, use VRG, and any table without mandatory children and insert into that table. Don't worry about values for every (can be NULL) column.

1. Query/ SELECT your database for information from at least two tables.
   - The SQL command is: `"SELECT `USER`.UserID, COUNT(`ORDER`.OrderID)AS TotalOrders, `USER`.FirstName, `USER`.LastName, `USER`.Phone, `USER`.EmailAddress, `USER`.Object, `OBJECT`.FeeFactor AS SendingFeePerKG FROM`

```
`USER`,`ORDER`,`OBJECT` WHERE `ORDER`.SenderID =
`USER`.UserID AND `OBJECT`.ObjectName = `USER`.Object AND
UserID = '$my_fn';"
```

- The file name is: “userDataSearching.php”
- The link to access the browser view of the file is: Link
- This file could be used to query data of a specific user from tables USER, OBJECT, and ORDER of the database. There is an input field for the desired UserID, which sends the request to the database after the “Enter” key is pressed or the “Search” button is clicked. Then if there is no error occurred, the database will return the user data related to the input UserID, including the user’s ID, first name, last name, phone number, email address, current space object, the cost for that user per weight (kilogram) of each order, and the total order sent. If could not find any matched UserID or there are errors occurred, there would be error message appears on the screen.

2. Insert a row into the parent table.
    - The SQL command is: `"INSERT INTO USER (UserID, FirstName, LastName, Phone, EmailAddress, Object) VALUES ('".$_POST["id"]."','".$_POST["fname"]."','".$_POST["lname"]."','".$_POST["phone"]."','".$_POST["email"]."','".$_POST["object"]."')"`
    - The file name is: “userInsert.php”
    - The link to access the browser view of the file is: Link
    - This file could be used to INSERT a new User profile into the table USER.

There are input fields for the user"s ID, first name, last name, phone number, email address and current space object. Some of the input fields have their own requirement so that error data, such as ID and phone number with wrong format or text in the number field, could not be inserted into the database. When the "Enter" key is pressed or "Record Profile" button is clicked on, all of the inputed data would be INSERTed into the USER table as a new row and a completion message appears on the screen. If there are errors occurred, there would be an error message.

3. Update a row in that parent table.
    - The SQL command is: `"UPDATE OBJECT SET FeeFactor = ('".$_POST["fee"]."') WHERE ObjectName = ('".$_POST["object"]."')"`
    - The file name is: "feeUpdate.php"
    - The link to access the browser view of the file is: Link
    - This file could be used to update the fee charged per kilogram of the sent order of a specific space object. There are input fields for the name of space object and the new fee. When the "Enter" key is pressed or "Update" button is clicked on, a search for the input object name is performed in the database. If found, the FeeFactor column would be update with the new value and a completion message appears on the screen. Otherwise, there would be error message sent.
    - There would be no change in the children tables because the new fee will only be applied to the new recorded Orders after this change.

D. The A-level

1. Query/SELECT over three tables (include an intersection table) using a JOIN.
    - The SQL, which INNER JOINs 3 table USER, ORDER, OBJECT, is:

```
"SELECT `ORDER`.OrderID, `ORDER`.SenderID,
`ORDER`.RecipientID, `ORDER`.SendDate,
`OBJECT`.ObjectName AS SentFrom, `OBJECT`.FeeFactor,
`ORDER`.Weight, `ORDER`.Cost FROM `ORDER` INNER JOIN
`USER` ON `ORDER`.OrderID = '$my_fn' AND `USER`.UserID =
`ORDER`.SenderID INNER JOIN `OBJECT` ON `USER`.Object =
`OBJECT`.ObjectName ORDER BY OrderID;"
```

    - The file name is: “orderDetailedPrice.php”
    - The link to access the browser view of the file is: Link
    - This file could be used to SELECT/show the more detailed information of specific ORDER. There is a input field for the OrderID of the targeted Order. When the “Enter” key is pressed or “Update” button is clicked on, a search for the input OrderID is performed in the database. If found, all of the data related to the Order including OrderID, SenderID, RecipientID, the sent date, the sent location, the fee factor applied, total weight in kilogram, and the shipping cost appears on the screen. Otherwise, there would be error message on the screen.

2. Insert a row into one of the N:M intersection tables in your database.
    - The INSERT SQL command is: "INSERT INTO `ORDER` (`OrderID`,

`SenderID`, `RecipientID`, `SendDate`, `Weight`, `Cost`) VALUES ('".$_POST["id"]."','".$_POST["sender"]."','".$_POST["recipient"]."','".$_POST["date"]."','".$_POST["weight"]."','".$_POST["cost"]."')"

- The file name is: “orderInsert.php”
- The link to access the browser view of the file is: Link
- This file could be used to INSERT new Order into the ORDER table. There are input fields for the OrderID, Sender’s UserID, Recipient’s UserID, sent date, total weight, and shipping cost. Some of the input fields have their own requirement so that error data, such as date and ID with wrong format or text in the number field, could not be inserted into the database. When the “Enter” key is pressed or “Record Profile” button is clicked on, all of the inputed data would be inserted into the ORDER table as a new row, and the SenderID and RecipientID would refer to the users in USER table. If there are errors, there would be an error message appears on the screen.

References.

Kroenke, David M. Yoder, Robert. Vandenberg, Scott. *Database Processing: Fundamentals, Design, and Implementation*.15th Edition. ISBN-13: 978-0134802749