**Final Project**

**Databases 2**

**Milestone 1 due Wednesday, April 25th**

**Project due Wednesday, May 9th**

In groups of 3 (**required**), you will be developing the database for an internet service provider. This requires creation and setup of the database itself, as well as an application that allows both customers and ISP employees to interact with the database as needed. It has the following requirements:

Data:

* **Customer:** We want to store information about the customers who use our service. Each customer is identified by a username that they choose upon creating an account with the ISP.Each customer has a name, as well as the address of the residence being serviced. They may also have a phone number and email address. When creating an account, customers will choose an internet package. For each customer we want to track the amount of data they upload and download every day.
* **Internet Packages:** Upon registering with the ISP, customers will select one of several internet packages. The package a customer signs up for dictates how much they will pay each month, what their upload and download speed are, how much bandwidth they have each month, as well as what their overage charges are for bandwidth consumed above the limits. Each package has a name, a description, and may have a list of extra features that comes with it.
* **Representatives:** The ISP has multiple different types of representatives for handling specific types of customer interactions. All employees have a name, an employee number, a phone number and an email address. Employees in the billing department handle billing concerns of clients. Employees in the sales department handle sales calls of clients. Employees in the tech support department handle support calls.

For each type of representative, we want to keep logs of which customers they have received calls from. For sales, we want to keep track of calls even if they did not result in a sale (IE: Call for which no customer is created). For billing we want to record the details of the call. For tech support we want to keep track of the type of issue that was reported (IE: Disrupted service, slow speeds), as well as any other details the tech support agent may have noted.

* **Invoicing:** We must track the bills sent out to clients. We want each invoice to report all chargeable services. The cost of the plan, bandwidth overage (per GB over the plan usage limit) and service charges. The overage charge is applied per GB rounded up to the next GB. Other fees that may be applied to invoices include a set up fee for new accounts for installation. Finally, after all other costs have been applied we add on GST (5%) and QST (9.975%), which must be itemized, to the total amount. Each invoice should have a due date, which is 21 days after the end of the invoicing period. We must also keep track payments received, associating each payment with the customer account and tracking any balance unpaid or excess amount. The invoice will report the balance from the previous invoice and all payments received during the invoicing period along with the new balance due.

Daily usage in tracked over the invoicing period and reported on invoice, any overage charged at a rate set determined by the service plan. If a customer's plan changes during the invoice period, the costs and bandwidth usage should be prorated over the invoice period.

* **Installation:** In order to get set up with the ISP, customers must schedule a visit from an technician to perform installation. The customer agrees to a date and time when they can be available, and it is added to the technician's schedule. We need to keep track of these appointments, and should also keep track of any notes that might be taken during an appointment.

In addition, technicians are employees just like other representatives, so we need to keep track of their name, employee number, phone number and email address.

**Mandatory Application Requirements:**

In order to work with the underlying data, you will be in charge of designing multiple applications for different types of user.

Customer Application: The first application you are tasked with designing is an application which customers will use to view and modify their account details

* When the application starts, a prompt should be presented allowing a user to log into an account using their username and password. Until a user is logged in and validated, they cannot do anything.
* Once logged in, the customer should be able to select from a list of options what to do next.
* They should be able to review the details of their existing package. They should be able to use the app to upgrade their package to a more expensive one. Downgrades, however, are not possible without the help of a representative.
* They should be able to view their usage - either for the current pay period or any previous pay period.
* They should be able to view their invoice - either for the current pay period or any previous pay period.
* They should be able to schedule a service request. Three types of request are possible: Removal of service, upgrades, and support requests. For each, the request should be logged in the database where it can be reviewed by the appropriate representative.
* Customers should be able to modify their own password.

Sales Rep Application: The second application you are tasked with designing is an application which sales reps will use to assist customers.

* When the application starts, a prompt should be presented allowing a user to log into an account using their employee identification and password. Until an employee is logged in and validated, they cannot do anything.
* Once logged in, the customer should be able to select from a list of options what to do next.
* Sales reps should be able to register new customers. This includes everything necessary to set up a user account, including assigning them a package as well as scheduling an install.
* Sales reps can modify packages of customers. Unlike customers, sales reps should be able to downgrade a customer’s service package.
* Sales reps can also perform recruiting (cold call sales) via this app. The app should randomly generate a phone number that they can call, and record information about the call in the DB including whether or not it resulted in a new customer.
* Sales reps should be able to modify their own password

**Required Extension:**

Each team is also expected to extend these basic requirements. This involves proposing a new addition the the project. This addition will take the form of a third application, intended for one of the other types of employee mentioned (tech support, billing, or technicians). Alternatively, you can create a new type of person for whom the application is intended. This addition should entail adding new tables and relationships, extending existing ones, as well as new stored procedures, functions and triggers. It should be of the same order of difficulty as the other two mandatory applications.

**NOTE: While these are the basic requirements, you are expected to think carefully about the design of your database. If some relationship makes sense but isn’t explicitly required in this list of requirements, you may still want to add it in. It is expected that if any requirement is unclear, *(and some requirements are unclear by design)* you will discuss it with your client (me) to ensure that your implementation is correct.**

**In groups of 3, you will need to do the following:**

1. **A logical data model ERD of the database.** Can be submitted as hard copy or electronically. (vertabelo.com can make it easy to create a nice looking ERD)
2. **A complete relational table specification in SQL Database Definition Language.** This specification should include appropriate data types for all attributes, constraints (NOT NULL, UNIQUE, etc.) where appropriate. Primary Keys and Foreign Keys should all be defined. Indexes should also be defined where necessary. There should be at least one table with an automatically incrementing Primary Key using a Sequence.
3. **SQL Data Manipulation Statements to load a representative sample of data into your schema.** This sample data should be complete enough to catch fringe (corner) cases. (mockaroo.com can be helpful for creating mock data – it probably won’t be able to completely fulfill your needs but it can save you a bit of time.)
4. **SQL Procedures/Functions/Triggers to carry out database operations.** Most database operations should be implemented as stored procedures and functions. The application should be largely for handling application logic, so there should be several calls done from your java application to stored SQL procedures. You should also have at least some triggers set up. You should make use of Packages to logically group related PL/SQL code.
5. **Application platform to carry out the various tasks required of the database.** Done in Java. Should implement all user interfaces specified in the requirements, and menus/forms/reports for users to navigate through their data. User interfaces can be implemented as command line prompts. They should be relatively easy to use. The application should be set up to connect directly to your database.
6. **A series of test queries/inputs which demonstrate your database's ability to handle their associated sample outputs. Login information for each type of user (so I can easily test your application).**
7. **Security Guidelines for the database.** As a group, you should write up what roles might exist on your database, and indicate which objects each role should have access to, and make it clear which users will be assigned which roles.

**Milestone 1 is due April 25:**

**To submit for Milestone 1:**

Entity-Relationship Diagram

Project Addition Proposal

schema.sql --- for SQL DDL statements

data.sql --- for SQL DML statements

**Final Project is due May 9th**

**To submit for Final Project:**

report --- includes test inputs, sample outputs

schema.sql --- for SQL DDL statements

data.sql --- for SQL DML statements

procs.sql --- for SQL procedures/functions/triggers/packages

source.zip --- a zip file containing application code

readme.txt --- an explanation of how to use your application/database