**Final Project**

**Databases 2**

In groups of 2 or 3 (**required**), you will be developing the database for a local post office. This requires creation and setup of the database itself, as well as an application that allows post office employees to interact with the database as needed. It has the following requirements:

Data:

* **Mail:** We want to store information about mail. There are 2 main classes of mail handled by the post office – registered and unregistered mail. Registered mail is managed individually at the post office before being sent and at each point along its route to safeguard against loss, theft, or damage. All mail has a delivery address indicating where it should be delivered. It also has a weight, a total amount of postage. Mail may also have a return address (this is required for registered mail). We also want to track status of mail, so we know where it is at any given time.
* **Mail Routes:** Mail routes are a list of postal codes that a mail carrier follows when delivering mail. Prior to delivering mail, the carrier acquires all of the mail for a given route and loads it in a truck. Each route has a name, and an id. Each route should also keep track of a “last delivered” property, which indicates when mail was last delivered on that route.
* **Schedule:** Each route must be fulfilled daily, so we require a schedule which shows which carriers are working which routes. In the schedule, we should keep track of when carriers start and finish their routes as well.
* **Vehicles:** The post office manages several delivery vehicles. Each vehicle has an ID, a license plate number. We want to track each vehicles status as well, noting if it is in use by a carrier, available at the post office, or decommissioned for maintenance.
* **Employees:**  We have three types of employees we want to keep track of, Carriers, Clerks and Postmasters. All employees have a first and last name, a phone number, an email address and an employee id. Carriers also have specific routes that they deliver mail on. Clerks and Postmasters each have an office number where they work, and an extension where they can be reached at the branch.

Application Requirements:

* When the application starts, a prompt should be presented allowing a user to log into an account using their employee id and password. Until a user is logged in and validated, they cannot do anything
* The application should have 3 tiers of users: Carrier, Clerk, and Postmaster. Upon logging in, the tier of the user should dictate what options are presented to the user.
* New users are only created by those in the Postmaster position.

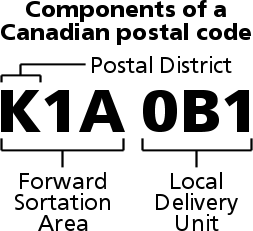
Carriers:

* Carriers should be able to view the mail route and truck they are assigned to.
* Carriers should be able to see the list of mail they must deliver. They should be able to view it in 3 ways: All the mail they must deliver, all the mail to be delivered in a specific postal code that is part of their route, all the mail addressed to a specific building that is part of their route.
* For registered mail, the carrier must be able to mark each individual parcel or letter as “delivered”.
* A carrier must be able to mark a route as started – indicating that they are currently delivering mail on that route.
* A carrier must be able to mark their mail route as completed. Upon completion, the carrier must specify any letters that were unable to be delivered an mark their status as “undeliverable”.
* Mark themselves as unavailable to work on a particular day – if a carrier cannot work on a given day due to illness or unforeseen circumstances they can register it via the system – allowing a replacement to be found
* Modify their own password

Clerks:

* Add new mail to the system.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Local ($)** | **USA ($)** | **International ($)** |
| 0-30g | 0.85 | 1.20 | 2.50 |
| 30-50g | 1.20 | 1.80 | 3.60 |
| 50-100g | 1.80 | 2.95 | 5.90 |
| 100-200g | 2.95 | 5.15 | 10.30 |
| 200-500g | 5.05 | 10.30 | 20.60 |

* They should be able to obtain shipping cost based on weight and location, and reject mail whose postage is insufficient or which is over 500g.
* Prepare mail to be delivered on a given route (load it on truck, set it aside).
* Mark mail as “sent to xxx” for neighbouring mail facilities if within the same Forward Sortation Area, or to “Sent to Airport” if it is within a different postal district.
* View all mail, by route, postal code, address and name of recipient or sender.
* Modify their own password.

Postmaster:

* Immediately upon logging in: Should see a list of all (non-empty) routes that do not have either a truck or a carrier assigned to them for today or tomorrow.
* Via a submenu: All clerk options.
* Create a new empty mail route.
* Modify an existing mail route by adding or removing postal codes from it. (Note, postal codes should only ever be associated with at most one route.)
* View the average amount of time it takes for a route to be completed.
* Add new employees – and associated accounts.
* Remove employees. For carriers, should be removed from all associated routes.
* Set up a delivery route by associating it with both a carrier and a vehicle.
* Add or remove vehicles, or temporarily decommission vehicles for maintenance.
* Modify their own password

**Required extras:**

For each team member, propose 1 significant expansion of the problem (seen by me by April 30th at latest). Possible ideas may include expanding the problem to a larger area – covering multiple post offices, adding a new user type, handling large package delivery, or just increasing the range of options available to current users. (As a general rule each expansion should require at least 1 new entity to be managed by the database)

**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, you will discuss it with your client (me) to ensure that your implementation is correct.**

**In groups of 2 or 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 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 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 the load their associated sample outputs. Login information for each of the three tiers of user (so I can easily test your application).**

**To submit:**

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