Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

CSU34041 Information Management II
Assignment: MySQL Database

Vaccine Delivery System Relational Database

Sean Murphy, 19335002
March, 2022

Contents

Section A: Description of Database Application area and ER Model…… 2

A1. Application Description …… 3
A2. Entity Relationship Diagram …… 4
A3. Mapping to Relationship Schema …… 5
A4. Functional Dependency Diagram …… 6

Section B: Explanation of Data and SQL Code …… 7

B1. Table Creation and Constraints …… 7
B2. Altering Tables …… 8
B3. Trigger Operations …… 8
B4. Views …… 9
B5. Commands to Populate Tables …… 10
B6. Retrieving Information From Database …… 10

Section A: Description of Database Application area and ER Model

A1. Application Description

I decided to create this database based on how I believe that a vaccine delivery system (VDS) may work during the Covid-19 pandemic. The pandemic has been an integral part of our lives for the past 2 years which is why I decided to create a database based on this specific topic. The system is made up of 7 different entities. The design is clear to ensure that the elements of the database are efficient whilst still maintaining integrity. Integrity is maintained through the implementation of constraints.

I will represent the entities of the VDS using tables for vaccines, batches, warehouses, employees, vehicles, hospitals and vaccination centres.

- **Vaccine**
  Each vaccine has a given unique vaccine ID, name and an expiry date in which it cannot be administered after this date. The batch number in which the vaccine is located is also available in this table

- **Batch**
  A batch is made up of multiple vaccines. The batch ID produces the weight of each batch, the manufacture date and expiry date of each batch and also which warehouse each batch is stored in.

- **Warehouse**
  The warehouses are where the batches of vaccines are stored and distributed from. The warehouse ID tells the address of the warehouse, the number of employees in the warehouse and the batch capacity of the warehouse.

- **Employee**
  The employees of the warehouses include warehouse workers and the delivery drivers who will deliver the vaccines to the vaccination centres and hospitals. The employee ID shows the first name, last name, date of birth of the employee and what warehouse that employee is assigned to.

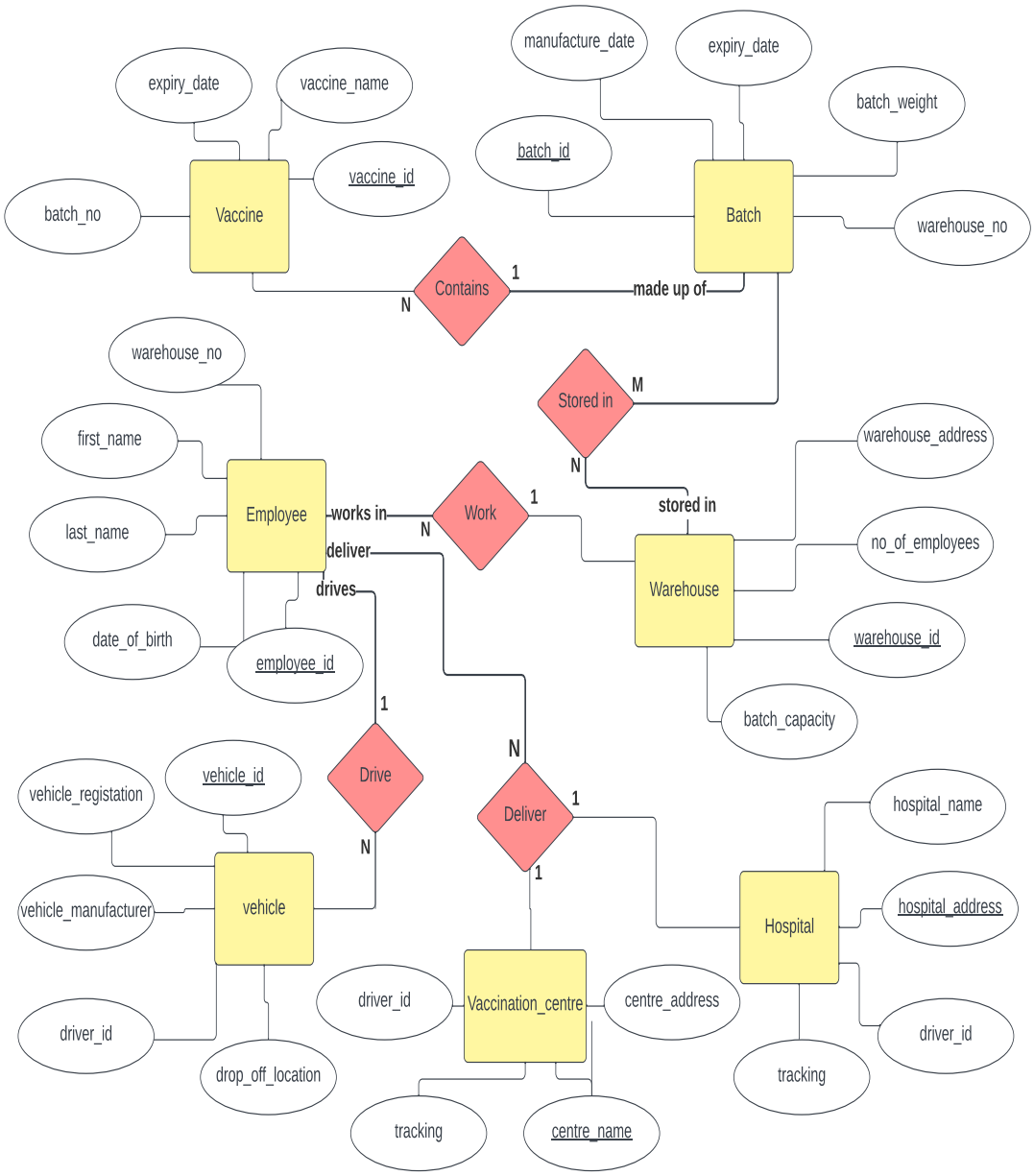
- **Vehicle**
  Each vehicle has unique vehicle ID which displays the vehicle registration, vehicle manufacturer, the id of the driver driving the vehicle and the location in which the driver will drop their load off at.

- **Vaccination centre**
  Deliveries of vaccines are made to vaccination centres by employees of the warehouses. The vaccination centre name identifies the address of the centre, the ID of the driver delivering to that centre and whether or not the vaccines have been delivered or not.

- **Hospital**
  Vaccines are also delivered to hospitals. The hospital address is used to identify its name, the ID of the driver delivering to that centre and whether or not the vaccines have been delivered or not. The reason the primary key for the hospital is the address and not the name is the fact that there exists hospitals in Ireland with the same name.

Entity Relationship Diagram

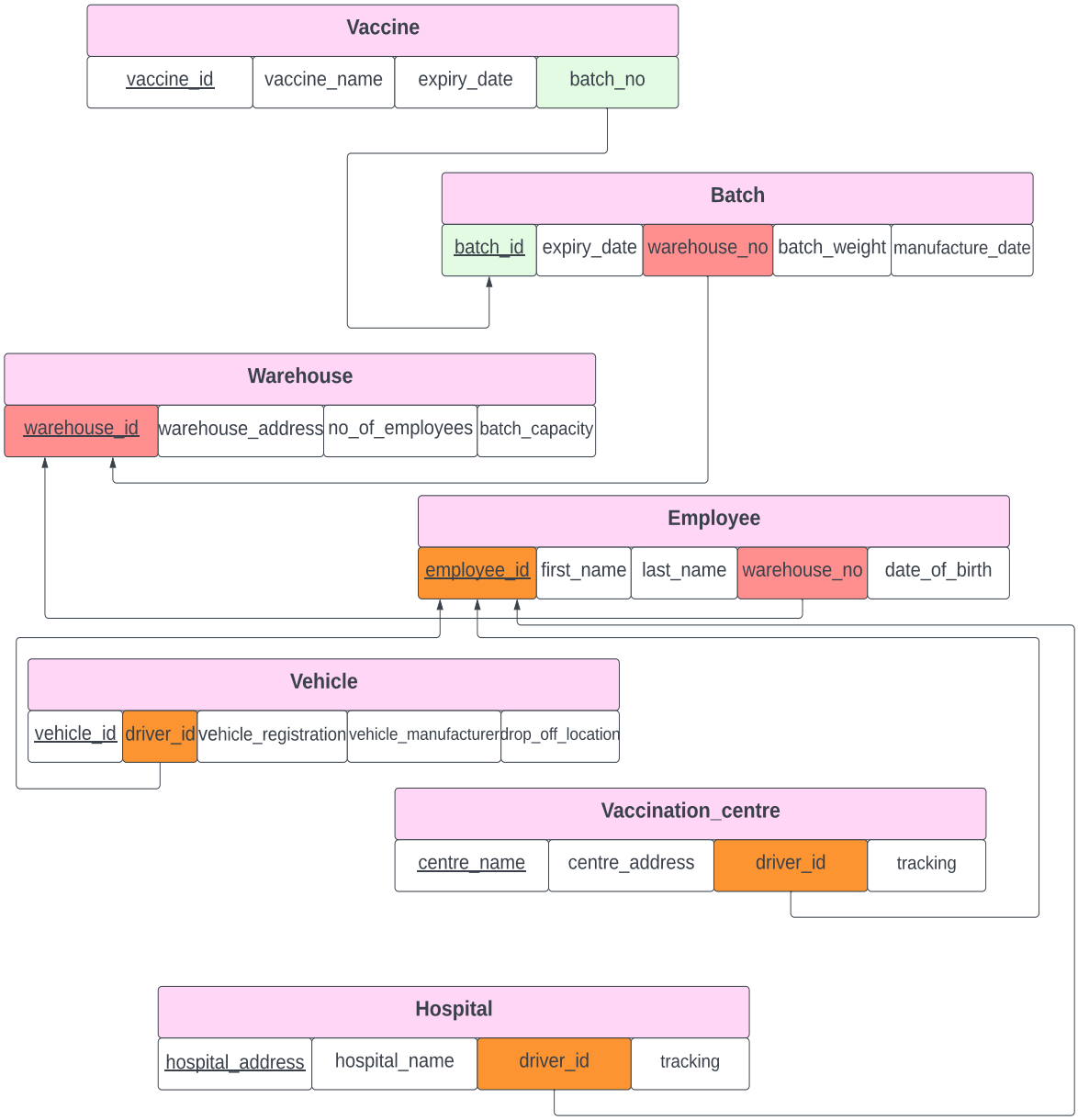
The entity relationship model shows the attributes of each of the different tables and the relationships that exist between each table.



Primary keys are underlined, entities are highlighted in yellow and relationships are highloghted in red

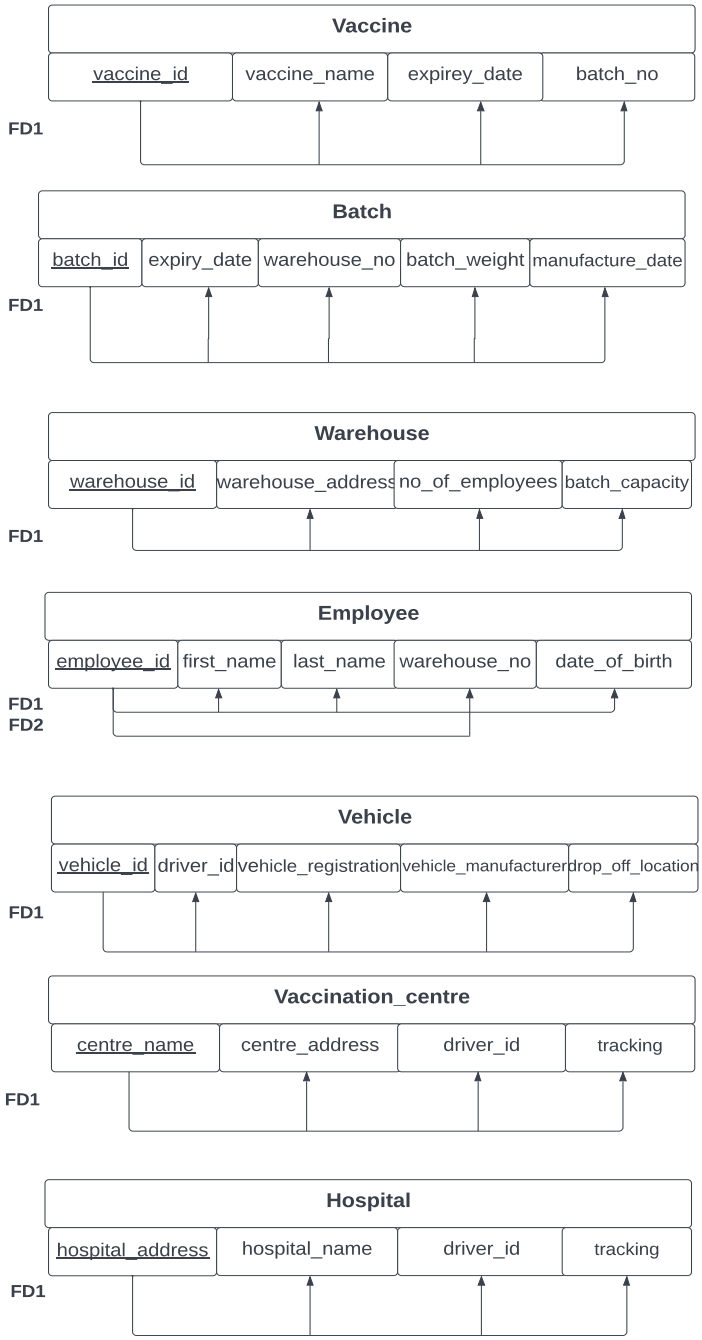
Relational Schema Diagram

The relational schema diagram shows the relationships that entities in the tables have with other tables in the database.



Functional Dependency Diagram

The functional dependency diagram shows how other attributes in each table depend on the primary key of that table.



Section B: Explanation of Data and SQL Code

B1. Table Creation and Constraints

When creating the tables I ensured that all of the entities were added to that table and that if the if the entity was that of variable string that the input length would be maximum VARCHAR length. This would mean that the values entered into the entity would be appropriately constrained. All other attributes within the database are constrained by the data type that they are assigned so no further action is required.

Following this a primary key for each table was selected, this was determined by choosing an entity in which each input to the table would be unique and each primary key was instructed to be NOT NULL. Foreign keys were also prevented from referencing elements that are not in the primary key that they reference.

A number of constraints were implemented into the database in order to insure that the inputs into a number of attributes are valid within the scope of the database which can be seen below.

This constraint ensures that the employees ID is valid as an ID cannot be negative and there are only 300 employees working in the warehouses so the integer cannot be greater than that.

```
CONSTRAINT `check_id_valid`
CHECK (((length(`employee_id`) > 0) and (length(`employee_id`) <= 300)))
```

This constraint ensures that the only strings that can be assigned to the tracking attribute are ‘delivered’ and ‘not delivered’ and nothing else.

```
CONSTRAINT `check_hospital_tracking`
CHECK ((`tracking` in ('delivered', 'not delivered')))
```

This constraint ensures that the ID of each vaccine is valid. As only 5 vaccines are currently approved in Ireland we can only have vaccine ID’s between 1 and 5. A trigger has been implemented in the case that a new vaccine is approved in the country in which case this constraint would need to be updated.

```
CONSTRAINT `check_vaccine_id_is_valid`
CHECK (((length(`vaccine_id`) > 0) and (length(`vaccine_id`) <= 5)))
```

B2. Altering Tables

The only instance of the database that regarded the altering of tables was the implementation of foreign keys into the database in which the ALTER TABLE command was used. This made the database more understandable to see where the foreign keys within the database and the tables in which they reference. Although in the export of the .sql the foreign keys appear as a constraint I used these commands within MySQLWorkbench.

```
ALTER TABLE `vaccine`
ADD FOREIGN KEY (`batch_no`) REFERENCES `batch` (`batch_id`)
```

```
ALTER TABLE `vehicle
ADD FOREIGN KEY (`driver_id`) REFERENCES `employee` (`employee_id`)
```

```
ALTER TABLE `employee`
ADD FOREIGN KEY (`warehouse_no`) REFERENCES `warehouse (`warehouse_id`)
```

B3. Trigger Options

I introduced 2 Triggers into the database which can be seen below:

This trigger will add a new employee into the attribute ‘no_of_employees’ whenever a new employee is employed to work in the warehouse. This will be updated in the warehouse table.

```
DELIMITER ;;
CREATE  DEFINER =`root`@`localhost`
TRIGGER `addEmployee` AFTER INSERT ON `employee` FOR EACH ROW BEGIN
UPDATE Warehouse
SET no_of_employees = no_of_employees + 1
WHERE warehouse_id = new.warehouse_no;
END ;;
DELIMITER ;
```

This trigger will add a new vaccine ID when a new vaccine is approved for use in Ireland into the attribute ‘vaccine_id’, this is an important trigger as it is important to ensure that vaccine ID is always correct as some members of the public, usually based on age are unable to receive certain vaccines so the ID must always be correct

```
DELIMITER ;;
CREATE  DEFINER=`root`@`localhost` TRIGGER `addVaccineID` AFTER INSERT ON
`vaccine` FOR EACH ROW BEGIN
UPDATE Vaccine
SET vaccine_id = vaccine_id + 1
WHERE vaccine_id = new.vaccine_name;
END ;;
DELIMITER ;
```

B4. Views

Views were created in this database in order to provide users with easy access to certain pieces of information within the database.

This view was implemented in order for the user of the database to be able to see the information of the employee and the information of the vehicle that that specific employee is driving.

```
CREATE VIEW `vehicle_driver` AS SELECT
employee.first_name AS `Driver First Name`,
employee.last_name AS `Driver Last Name`,
employee.date_of_birth AS `Driver DOB`,
vehicle.vehicle_id AS `Vehicle Identification`,
vehicle.vehicle_registration AS `Vehicle Registration`,
vehicle.vehicle_manufacturer AS `Vehicle Type`
FROM vehicle
JOIN employee ON employee.employee_id = vehicle.driver_id
```

This view allows the user to access an employee's 'driver_id' based on their 'first name' or generate an employee's 'first name' based on their 'driver_id'. This may be useful for a warehouse manager if they get a complaint about one of their drivers where the person making the complaint only knows the driver ID or the first name of that given driver.

```
CREATE VIEW `driver_credentials` AS SELECT
employee.first_name AS `first_name`,
vehicle.driver_id AS `driver_id`
FROM employee, vehicle
WHERE employee.employee_id = vehicle.driver_id
```

B5. Commands to Populate Tables

I populated my tables using a single INSERT INTO instruction, I found this way to be easiest rather than using single INSERT INTO instructions for each row as that would have been too time consuming.

This is the command I used to populate the 'vaccination_ centre' table. The INSERT INTO instruction can be seen at the top then the values for the centre name, centre address, delivery tracking and driver ID are included, in this order respectively.

```
INSERT INTO `vaccination_centre` VALUES
('Croke Park Vaccination Centre','Jonses Road, Dublin','not delivered',2),
('Kerry Vaccination Centre','Borg Warner, Tralee, Kerry','delivered',5),
('Kilkenny Vaccination Centre','Dublin Road, Lyrath, Kilkenny','delivered',26),
('Letterkenny vaccination centre','Letterkenny Business Park, Letterkenny,
Donegal','delivered',34),
('Mayo Vaccination Centre','Breaffy, Castlebar, Mayo','delivered',22),
('Richmond Barracks Vaccination Centre','St. Michael\'s Estate, Inchicore,
Dublin','not delivered',2);
```

B6. Retrieving Information from Database

There are many ways to retrieve information from the database. One way is by using the views that are seen in section B4. Information on the vehicle driver could be retrieved using the query:

```
SELECT *
FROM vehicle_driver
```

All elements relating to the vehicle driver view would be retrieved.

We can also use general queries to generate information from the database. For example we can retrieve all of the employees that are under the age of 25 (born in 1992 and later) using a SELECT statement as seen below:

```
USE vaccine_delivery_system;
SELECT *
FROM employee
WHERE  date_of_birth > ('1992-01-01')
```

employee_id	warehouse_no	first_name	last_name	date_of_bir...
24	2	Stephen	Kennedy	1999-02-15
112	6	David	Quinn	1996-04-19

We can also join tables together in order to retrieve data from multiple tables. In the below example we use the foreign keys of the warehouse_no attribute from the Employee table and the warehouse_id attribute from the Warehouse table. The query retrieves the information of employees who work in the same warehouse.

```
SELECT e.employee_id, e.warehouse_no, e.first_name,
e.last_name, e.date_of_birth, w.warehouse_address
FROM employee e
JOIN warehouse w
ON e.warehouse_no = w.warehouse_id
```

employee_id	warehouse_no	first_name	last_name	date_of_bir...	warehouse_address
22	2	Conor	Murphy	1976-11-19	Dublin Industrial Esate, Glasnevin, Dublin
23	2	Elaine	Rooney	1982-08-12	Dublin Industrial Esate, Glasnevin, Dublin
24	2	Stephen	Kennedy	1999-02-15	Dublin Industrial Esate, Glasnevin, Dublin