CS 3306-01 Databases 2

Instructor: Professor David Nagus

Name: Ryohei Hayashi

Learning Journal 1

**Question 1**

Since DIA software does not work on my environment (M1 MacBook Air), I used Mermaid.js to create the ER diagram, which allows for lightweight and fast ER diagram generation directly in the browser. This tool operates in a web-based environment and makes it easy to visualize ER data models within the browser.

(a)

In this ER diagram, I have defined the three main entities: "Customer," "Car," and "Accident," and I have clearly outlined the relationships between each entity.

Customer: This entity manages customer information. The key attributes include "CustomerID," "Name," "Address," and "Phone Number." A customer owns one or more cars.

Car: This entity manages car information. The key attributes include "CarID," "Make," "Model," and "Year." Each car may be associated with zero or more accident records.

Accident: This entity manages accident information. The key attributes include "AccidentID," "Date of Accident," and "Damage."

The relationships between these entities are defined as follows:

Relationship between Customer and Car (One-to-Many): Each customer owns one or more cars, so the relationship between Customer and Car is "one-to-many." This means that multiple cars can be associated with a single customer.

Relationship between Car and Accident (Zero-to-Many): Each car can have zero or more accident records, meaning there may be no accidents or multiple accidents associated with a car. Thus, the relationship between Car and Accident is "zero-to-many."

The ER diagram based on this information is shown below. The details of the ER diagram are explained.

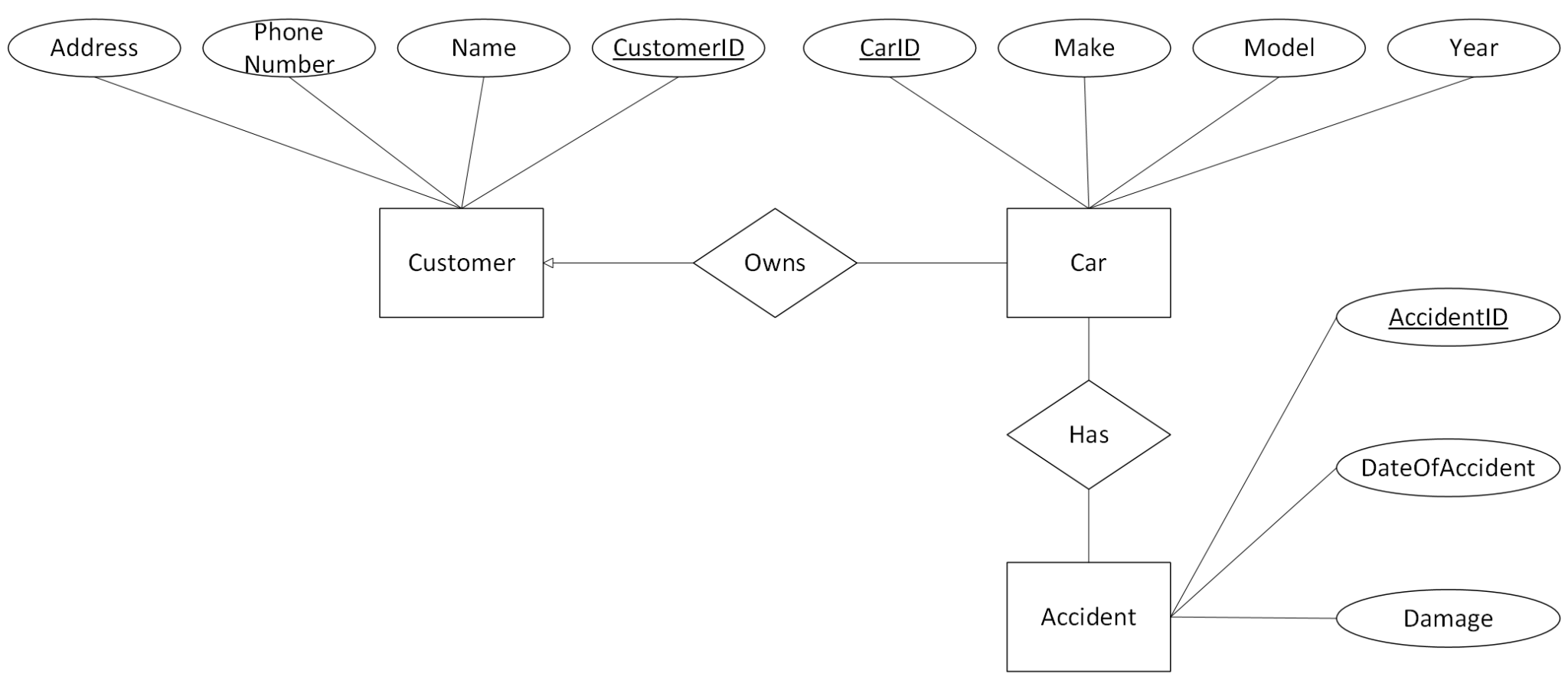


Figure 1. E-R Diagram

(b)

I will explain the three concepts used in the model:

Primary Key: In the "Customer" entity, the "CustomerID," and in the "Car" entity, the "CarID" serve as primary keys. These uniquely identify each record within the entities.

Composite Attribute: In the "Customer" entity, "Name" can be split into multiple components, such as first name and last name, making it an example of a composite attribute.

Multi-valued Attribute: In the "Car" entity, the "Accident" is an example of a multi-valued attribute because one car may be associated with multiple accidents.

**Question 2**

Based on the following table information, I will discuss the SQL queries and their results:

* Customer Table

| CustomerID | Name | Phone-Number | Address |
| --- | --- | --- | --- |
| 1 | John Smith | 123-4567-8910 | 123 Main St |
| 2 | Jane Doe | 098-7654-3210 | 456 Oak Ave |
| 3 | Ryo Haya | 111-2222-3333 | 876 Ota vil |

* Car Table

| CarID | CustomerID | Model | RegistrationNumber |
| --- | --- | --- | --- |
| 101 | 1 | Toyota Camry | ABC123 |
| 102 | 2 | Honda Civic | XYZ456 |

* Accident Table

| AccidentID | CarID | Date | Description |
| --- | --- | --- | --- |
| 1001 | 101 | 2024-01-15 | Rear-end collision |
| 1002 | 101 | 2024-02-20 | Minor fender bender |
| 1003 | 102 | 2024-03-30 | Parking lot accident |

**Left Join**

This query combines all records from the Customer table with corresponding records in the Car table. If a customer exists but does not have a corresponding car, the customer's data will still be included.

```sql

SELECT Customer.Name, Customer.Phone-Number, Car.Model

FROM Customer

LEFT JOIN Car

ON Customer.CustomerID = Car.CustomerID;

```

Result

| Name | Phone-Number | Model |
| --- | --- | --- |
| John Smith | 123-4567-8910 | Toyota Camry |
| Jane Doe | 098-7654-3210 | Honda Civic |
| Ryo Haya | 111-2222-3333 | NULL |

**Right Join**

This query combines all records from the Car table with corresponding records in the Customer table. If a customer does not own a car, their data will still be included, with the `Model` column showing NULL. For example, Ryo Haya does not own a car, so their data is displayed with NULL in the car-related fields.

```sql

SELECT Customer.Name, Customer.Phone-Number, Car.Model

FROM Car

RIGHT JOIN Customer

ON Customer.CustomerID = Car.CustomerID;

```

Result

| Name | Phone-Number | Model |
| --- | --- | --- |
| John Smith | 123-4567-8910 | Toyota Camry |
| Jane Doe | 098-7654-3210 | Honda Civic |
| Ryo Haya | 111-2222-3333 | NULL |

**Full Outer Join**

The Full Outer Join returns all records from both tables, with NULL values where there is no match in one of the tables. This ensures that complete data from both tables is retrieved, even if no corresponding record exists in the other table.

```sql

SELECT Customer.Name, Customer.Phone-Number, Car.Model

FROM Customer

FULL OUTER JOIN Car

ON Customer.CustomerID = Car.CustomerID;

```

Result

| Name | Phone-Number | Model |
| --- | --- | --- |
| John Smith | 123-4567-8910 | Toyota Camry |
| Jane Doe | 098-7654-3210 | Honda Civic |
| Ryo Haya | 111-2222-3333 | NULL |

**Question 3**

Here are examples for the four types of cardinality mappings and why the cardinality is appropriate for each case:

* **One-to-One**

Example: Driver’s license and driver.

Each driver can only have one driver’s license, and each driver’s license is assigned to only one driver, making this a one-to-one relationship.

* **Many-to-One**

Example: Cars and manufacturers.

Multiple cars can be produced by the same manufacturer, so the relationship from the car’s perspective is many-to-one.

* **One-to-Many**

Example: Customer and car.

A customer can own multiple cars. Therefore, from the customer’s perspective, the relationship to cars is one-to-many.

* **Many-to-Many**

Example: Cars and accidents.

One car can be involved in multiple accidents, and one accident can involve multiple cars. Thus, the relationship between cars and accidents is many-to-many.

Additionally, in databases, many-to-many relationships are often represented by using an intermediary table. For example, to manage the many-to-many relationship between cars and accidents, we could create an intermediary table called `Car\_Accident` that records which car was involved in which accident. This approach effectively handles the management of the many-to-many relationship.

Word Count: 663

References

1. Silberschatz, A., Korth, H.F., & Sudarshan, S. (2001). *Database System Concepts (4th ed.).* New York, NY: McGraw-Hill. Available at Database System Concepts 4th Edition By Silberschatz-Korth-Sudarshan.pdf