Rohit G, Sergio H
Project Step 2 Draft
Course: CS 340

LINK TO WEBSITE: https://web.engr.oregonstate.edu/~hurtadse/projectgroup5_step3_Draft/frontend/

a) **Fixes based on Feedback from Step 1:**

Feedback received from Yuritzia Peraza: "I think there are a few small additions to the project to make it more accessible to users. There can be more challenges faced by cable internet and expand the necessities. Such as showing a customers how multiple unpaid subscriptions can be handled." This feedback was the most valuable to our team because it helped with easier analysis of our database and with more convenience to store and retrieve information.

From this feedback, our team decided to add a **"Payments"** entity that tracks payments made by customers including the date, method of payment, and confirmation number. It helps to keep track of payments to differentiate from the BillingRecords entity.

Changes added to the Draft outline:

Addition to the Entities section:

1. **Payments**

Attributes:

- paymentID: int, auto_increment, unique, not NULL, PK (unique identifier for each payment)
- billingID: int, FK, NULL (Foreign key referencing BillingRecords.billingID. **Nullable** to allow payments not tied to a billing record.)
- paymentDate: date, not NULL
- amountPaid: decimal(10, 2), not NULL
- paymentMethod: enum('Credit Card', 'Debit Card', 'Bank Transfer', 'Cash', 'Check'), not NULL
- confirmationNumber: varchar(50), unique, not NULL

Purpose: Tracks payments made by customers such as the amount, date, method of payment, and confirmation number. Allows for better tracking of payments separate from billing records. *billingID* made nullable, allowing payments to exist independently of a billing record. This supports prepayments, overpayments, or unapplied payments.

Addition to the Relationships section:

1. BillingRecords and Payments
    - Type: 1:M (One billing record can have multiple payments, e.g., if the customer pays in installments).
    - Implementation: billingID is a foreign key in the Payments table.
    - Nullable Relationship: billingID is nullable, enabling payments to exist without being linked to a specific billing record.

b) Fixes based on feedback from step 2:

Grace Kohler pointed out that a payment is required to be made, but that is why we made this option nullable, so that a payment can be made with or without a billingRecord required.

Torsten Bergersen pointed out that the screenshot of the schema was too small and cut out in some parts. We enlarged it for easier viewing.

Torsten also mentioned the potential transitive dependency issue in BillingRecords where paymentStatus is determined by the presence or absence of records in the Payments table. We decided that our current solution is more pragmatic and functional from a business perspective than calculating the sum dynamically each time needed.

c) Project Outline and Database Outline - Updated Version:

This indicates the beginning of the Updated Submission for the Project Step 1 Outline:

Team Members: Rohit Gandham, Sergio Hurtado, TEAM 5

Project Title: Billing and Customer Management System for Cable Internet Provider

a) Overview: This project aims to create a database-driven billing and customer management system for a cable internet provider serving approximately 2,000 customers.
The system will enable the provider to:

1. Track customer data, including personal information and service subscriptions.
2. Generate and manage monthly billing records.
3. Handle payments and balances efficiently.
4. Provide a user-friendly web interface for administrators to perform CRUD operations.

By implementing this system, the provider can improve operational efficiency, reduce billing errors, and enhance customer satisfaction.

b) Database Outline:

Entities

1. **Customers**
   Attributes:
   - customerID: int, auto_increment, not NULL, PK (internal unique identifier)
   - accountNumber: varchar(20), unique, not NULL (External identifier visible to customers. Unique to each account)
   - name: varchar(255), not NULL
   - email: varchar(255), unique, not NULL
   - phone: varchar(15), not NULL
   - address: varchar(255), not NULL
   - joinDate: date, notNull

   Purpose: Stores details of customers, including contact information and service address.

2. **Services**
   Attributes:
   - serviceID: int, auto_increment, unique, not NULL, PK
   - serviceName: varchar(255), not NULL
   - monthlyCost: decimal(10,2), not NULL
   - description: varchar(255), NULL

   Purpose: Defines the services offered, such as internet or cable.

3. **BillingRecords**
   Attributes:
   - billingID: int, auto_increment, unique, not NULL, PK
   - customerID: int, FK, not NULL (references Customers, customerID)
   - billingDate: date, not NULL
   - totalAmount: decimal(10,2), not NULL
   - paymentStatus: enum('Paid', 'Unpaid'), not NULL

   Purpose: Tracks billing records for each customer, such as payment status or balance due.

4. **Subscriptions** (Intersection Table for Many-to-Many Relationship)

Attributes:

- subscriptionID: int, auto_increment, unique, not NULL, PK
- customerID: int, FK, not NULL (References Customers.customerID)
- serviceID: int, FK, not NULL (References Services.serviceID)
- startDate: date, not NULL
- endDate: date, NULL

Purpose: Links customers to subscribed services, allowing multiple subscriptions per customer and multiple customers per service. Tracks subscription start and end dates.

5. **Payments**

Attributes:

- paymentID: int, auto_increment, unique, not NULL, PK (unique identifier for each payment)
- billingID: int, FK, NULL (Foreign key referencing BillingRecords.billingID. **Nullable** to allow payments not tied to a billing record.)
- paymentDate: date, not NULL
- amountPaid: decimal(10, 2), not NULL
- paymentMethod: enum('Credit Card', 'Debit Card', 'Bank Transfer', 'Cash', 'Check'), not NULL
- confirmationNumber: varchar(50), unique, not NULL

Purpose: Tracks payments made by customers such as the amount, date, method of payment, and confirmation number. Allows for better tracking of payments separate from billing records. *billingID* made nullable, allowing payments to exist independently of a billing record. This supports prepayments, overpayments, or unapplied payments.

Relationships:

1. Customers and BillingRecords
   - Type: 1:M (One customer can have multiple billing records).
   - Implementation: *customerID* is a foreign key in *BillingRecords*.

2. Customers and Subscriptions
   - Type: 1:M (One customer can have multiple subscriptions).
   - Implementation: *customerID* is a foreign key in *Subscriptions*.

3. Services and Subscriptions
   - Type: 1:M (One service can have multiple subscriptions).
   - Implementation: *serviceID* is a foreign key in *Subscriptions*.

4. Many-to-Many Relationship

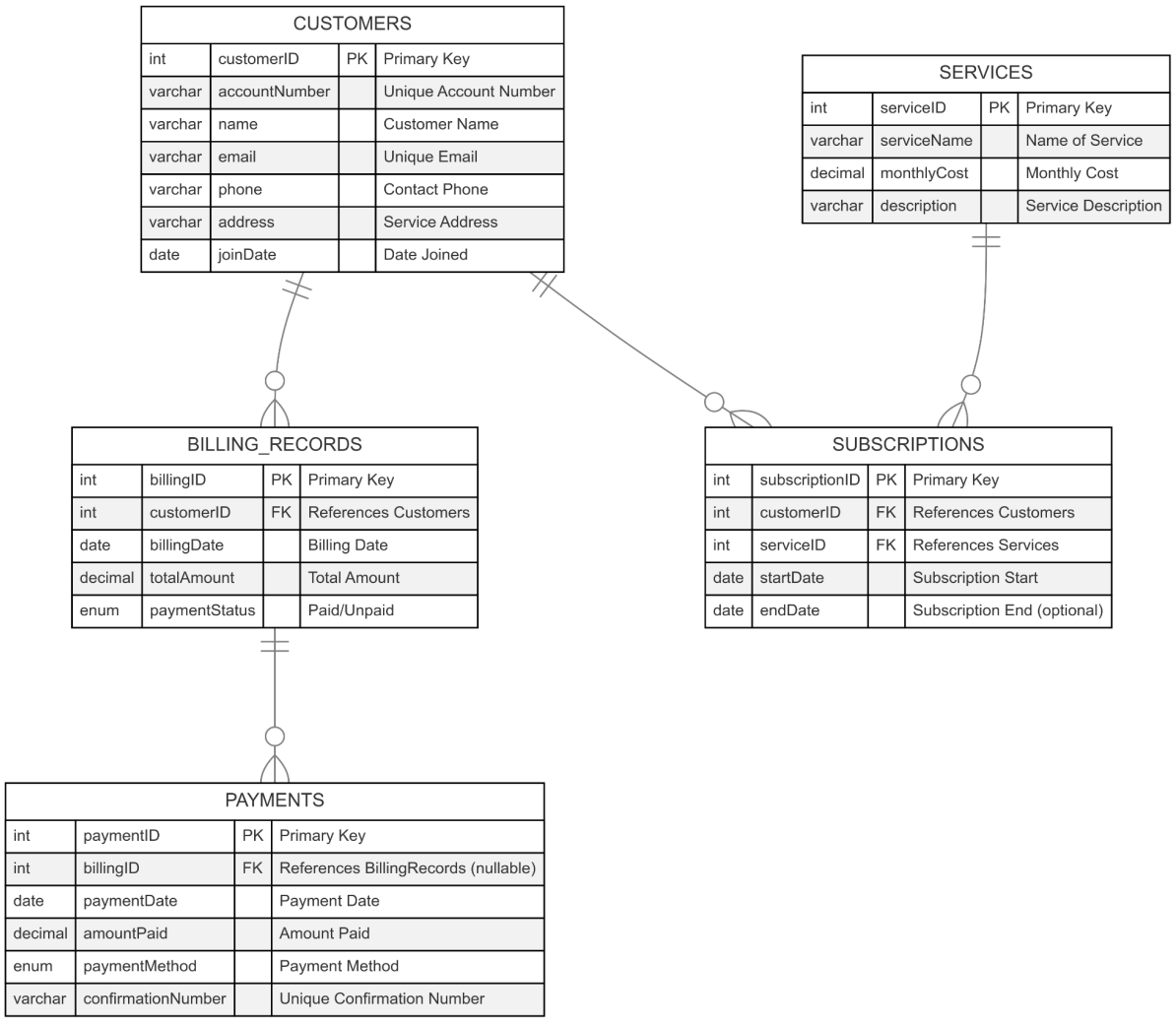
- Entities: Customers and Services
- Intersection Table: *Subscriptions*

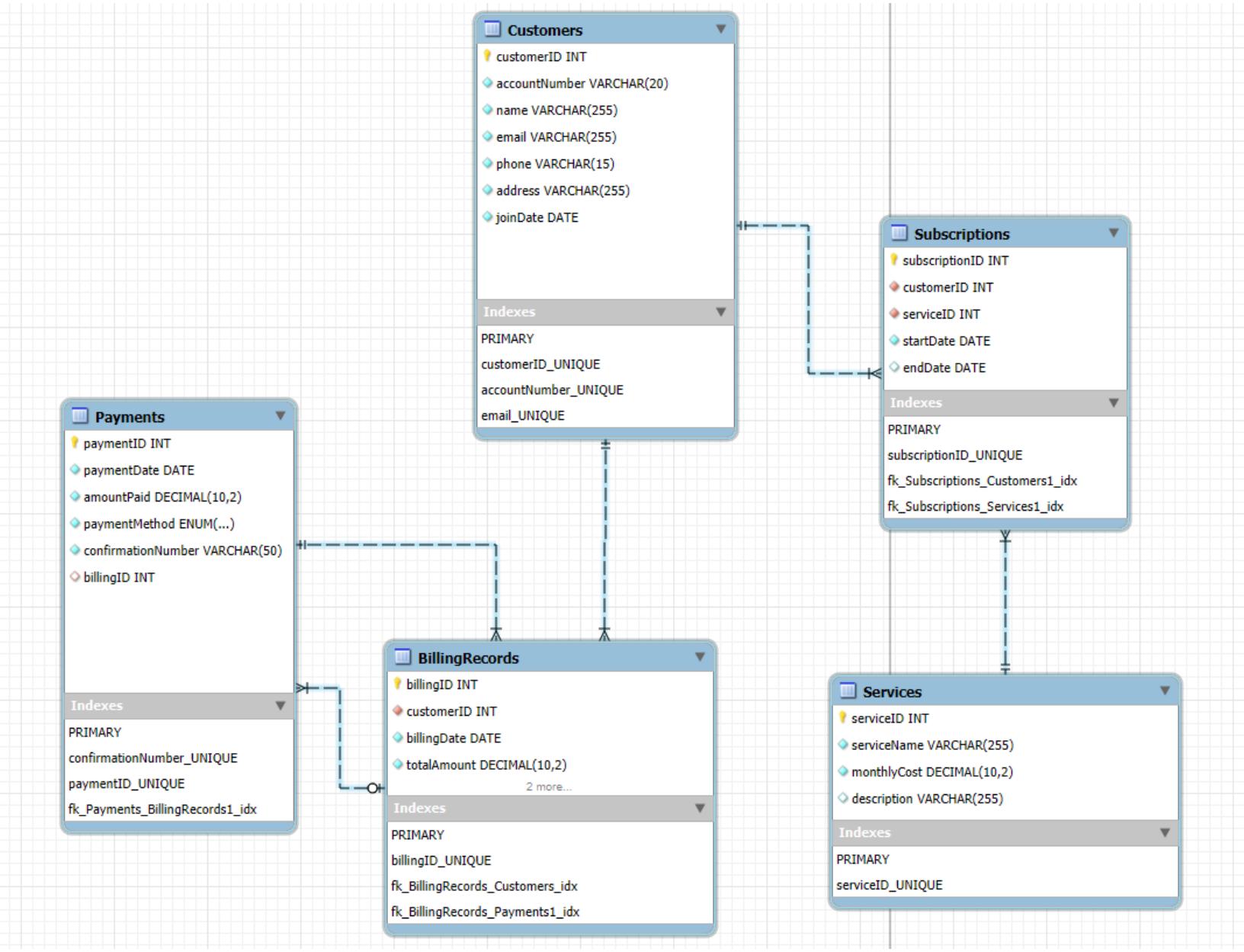
5. BillingRecords and Payments
    - Type: 1:M (One billing record can have multiple payments, e.g., if the customer pays in installments).
    - Implementation: billingID is a foreign key in the Payments table.
    - Nullable Relationship: billingID is nullable, enabling payments to exist without being linked to a specific billing record.

This indicates the end of the Updated Submission for the Project Step 1 Outline.

Normalization: Our outline contains atomic values without repeating groups. Each table has a unique primary key. Every non-key element is dependent on the entire primary key.

c) Entity-Relationship Diagram:




Customers
customerID INT
accountNumber VARCHAR(20)
name VARCHAR(255)
email VARCHAR(255)
phone VARCHAR(15)
address VARCHAR(255)
joinDate DATE
Indexes
PRIMARY
customerID_UNIQUE
accountNumber_UNIQUE
email_UNIQUE
Subscriptions
subscriptionID INT
customerID INT
serviceID INT
startDate DATE
endDate DATE
Indexes
PRIMARY
subscriptionID_UNIQUE
fk_Subscriptions_Customers1_idx
fk_Subscriptions_Services1_idx
Payments
paymentID INT
paymentDate DATE
amountPaid DECIMAL(10,2)
paymentMethod ENUM(...)
confirmationNumber VARCHAR(50)
billingID INT
Indexes
PRIMARY
confirmationNumber_UNIQUE
paymentID_UNIQUE
fk_Payments_BillingRecords1_idx
BillingRecords
billingID INT
customerID INT
billingDate DATE
totalAmount DECIMAL(10,2)
2 more...
Indexes
PRIMARY
billingID_UNIQUE
fk_BillingRecords_Customers_idx
fk_BillingRecords_Payments1_idx
Services
serviceID INT
serviceName VARCHAR(255)
monthlyCost DECIMAL(10,2)
description VARCHAR(255)
Indexes
PRIMARY
serviceID_UNIQUE

d) Schema:

e) Example Data:

Customers Table:

CUSTOMER-ID	ACCOUNT NUMBER	NAME	EMAIL	PHONE	ADDRESS	JOINDATE
1	ACCT001	Alice Johnson	alice.j@example.com	555-1234	123 Elm St	2020-01-15
2	ACCT002	Bob Smith	bob.smith@example.com	555-2345	456 Oak Ave	2020-03-22
3	ACCT003	Carol White	carol.white@example.com	555-3456	789 Pine Rd	2020-06-10

Services Table:

SERVICE-ID	SERVICENAME	MONTHLYCOST	DESCRIPTION
1	Internet	39.99	High-speed broadband internet
2	Cable TV	29.99	Digital cable TV with over 200 channels
3	Phone	19.99	Unlimited local calling

BillingRecords Table:

BILLING-ID	CUSTOMER-ID	BILLINGDATE	TOTALAMOUNT	PAYMENTSTATUS
1	1	2023-01-01	69.98	Paid
2	2	2023-01-01	29.99	Unpaid
3	3	2023-02-01	39.99	Paid

Subscriptions Table:

SUBSCRIPTION-ID	CUSTOMER-ID	SERVICE-ID	STARTDATE	ENDDATE
1	1	1	2020-01-15	NULL
2	1	2	2020-02-01	2022-12-31
3	3	2	2020-03-22	NULL
4	4	3	2020-06-10	NULL

Payments Table:

SUBSCRIPTION-ID	BILLING-ID	PAYMENTDATE	AMOUNTPAID	PAYMENTMETHOD	CONFIRMATION NUMBER
1	1	2023-01-05	69.98	Credit Card	CONF001
2	2	2023-01-07	15.00	Cash	CONF002
3	2	2023-01-20	14.99	Cash	CONF003
4	NULL	2023-02-02	39.99	Bank Transfer	CONF004