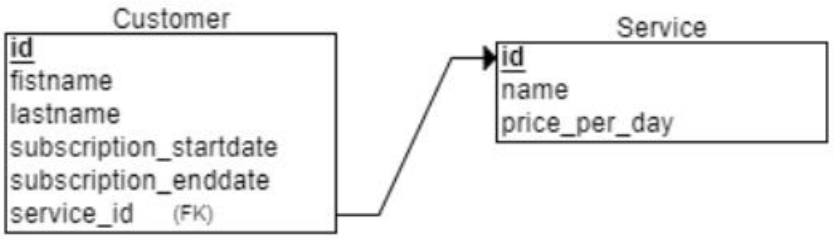
Temporal Databases

Learning goals:	Ability to design and implement a temporal database in MariaDB environment.
Mode of work:	Work preferably in pairs. Individual work is also possible.
Returnables:	Submit the requested statements and screenshots in a single pdf file. If you work in pairs, each participant must submit the same answer separately as an individual submission. Type the name of your pair in Oma's answer box.
Evaluation criteria:	Correctness, scope, and presentation of the answer.
Background material:	Lecture slides, network sources. Specifically, the MariaDB documentation about temporal tables: https://mariadb.com/kb/en/temporal-tables/.

1. Your goal is to design a temporal relational database for customers and their streaming media subscriptions. Each customer can have only one active subscription at a given time. There are three alternative services to subscribe (Gold, Silver, and Bronze). Note: As we use the temporal features of MariaDB, do not model a link table between Customers and Services.



   In MariaDB environment, create an SQL script that generates the database schema (CREATE TABLE statements). Remember the primary and foreign key definitions.

   There are following constraint for saving the subscription history:

   - In the Customer table, the name information may change over time (use *system versioning*).
   - In the Customer table, the subscription period may change. That means, in one period the customer can have the Gold subscription whereas in another period he/she may have the Bronze subscription (add *application-time period versioning* as well; this makes the table bitemporal).
   - The prizes may change in the Subscription table over time (use *system-versioning*).

   Include your CREATE TABLE statements in your answer.

2. Write INSERT statements that insert services (1, Gold, 0.95), (2, Silver, 0.45), and (3, Bronze, 0.00) as well as customers (1, Mary Smith, Silver subscription for period 1.9.2023-9.9.2023) and (2, Ville Puro, Bronze subscription starting on 10.9.2023, and still continuing). Hint: In SQL there is no infinity value, so you can set the ending date to any date in distant future, e.g. in year 9999). Include the INSERT statements in your answer.

3. Make an UPDATE statement that modifies Ville's Bronze subscription into a Gold subscription for the time 3.-8.10.2023. Hint: Use FOR PORTION OF clause. Include your UPDATE statement in your answer accompanied by a screenshot the displays the outcome.

4. Now change Ville's last name into Virtanen, and the prize of the Bronze subscription into 0,10 euros per day. Then, after the updates, write the SELECT statements that display the history information in the Customer and Subscription tables both before and after the UPDATE operations. Include the statements as well as screenshots of the outcome. Hint: if you collect the SQL statements into a script, you can force a break of a few seconds by including a SELECT SLEEP(*n*) statement into your script. In the statement, *n* stands for the duration of the idle time in seconds. This guarantees that you have a conveniently different timestamp for the data before and after the changes.

5. Write a SELECT statement that retrieves the data before the updates. The data to be retrieved includes the first name, last name, the subscription ordered and its daily prize. Thus, you should retrieve the earlier data that was valid before the updates by a query the contains an INNER JOIN clause. Again, include the SELECT statement and a screenshot of the outcome in your answer.