VANCOUVER ISLAND UNIVERSITY
CSCI 370 — FINAL EXAMINATION
21 April 2023, 13:00 — 16:00

TO BE ANSWERED IN BOOKLETS DURATION: 3 Hours

INSTRUCTOR: H. Liu

Instructions

- Students must count the number of pages in this examination paper before beginning to write, and report any discrepancy immediately to the invigilator.
- This examination paper consists of 9 pages.
- This is a CLOSED BOOK examination. Students are allowed to bring one piece of letter-sized and double-sided note.
- Calculators are NOT permitted.
- Remember to state any assumptions and show rough work.
- Note carefully the weight of each question, and answer appropriately.
- Attempt all questions. All questions relate to material covered in the lectures, labs and assignments.

(This page is left empty intentionally.)

Note: Questions 1 to 8 refer to the following database schema called `TelevisionDB`, which describes a hypothetical relational database used by a television company to service its customers with watching televisions, and collect and manage data from this service.
In the schema, the primary key of each relation is underlined.

```
Accounts(accNo, name, email, otherInfo, status)
Channels(cid, channelName, contentType, mainFeatures, capacity)
Subscribes(accNo, cid, subscribeDate)
ViewingHistory(accNo, cid, viewingDate, length)
```

- Each record in the relation `Accounts` describes a customer account of the TV company. Each account is assigned a unique account number (*accNo*), records the customer's name, email, other relevant information (*otherInfo*), and status (such as 'VIP', 'active', 'inactive', etc).

- Each record in the relation `Channels` describes a television channel. Each channel is assigned a unique ID (*cid*) for internal use, has a channel name, a content category type (such as 'News', 'Music', etc.), and a short description about its main features. Each channel also has an upper limit (*capacity*) indicating the maximum number of customers allowed to subscribe to this channel.

- Each record in the relation `Subscribes` records a relationship that a customer (identified by the customer's account number *accNo*) subscribed to a TV channel (identified by its internal ID *cid*) since a starting date (*subscribeDate*).

- Each record in the relation `ViewingHistory` records how many minutes (*length*) in each day (*viewingDate*) a customer (identified by the customer's account number *accNo*) spent on watching a TV channel (identified by the channels internal ID *cid*) cumulatively.

You might find the following partial class declaration useful in the exam:

```
class Environment {
   public:
      static Environment * createEnvironment();
      static void terminateEnvironment(Environment *env);
      Connection * createConnection(const string &userName,
                                    const string &password,
                                    const string &connectString = "");
      void terminateConnection(Connection *connection);
};

class Connection
{
   public :
      Statement* createStatement( const string  &sql = "");
      void terminateStatement(Statement *statement);
      void commit();
      void rollback();
};

class Statement
{
   public:
      ResultSet * executeQuery( const string &sql = "");
      unsigned int executeUpdate( const string &sql = "");
      void closeResultSet(ResultSet *resultSet);
      void setInt(unsigned int paraIndex, int argument);
      void setString(unsigned int paraIndex, string argument);
};

class ResultSet
{
   public:
      bool next();
      int getInt(unsigned int colIndex);
      string getString(unsigned int colIndex);
};
```

1. (25 Marks) Express each of the following queries in a single SQL statement against the database schema `TelevisionDB` given on the previous page.

   (a) For each channel that has the exact word 'entertainment' contained in its main features, list its channel name, content type and main features.

   (b) For each subscription recorded in `Subscribes`, list the subscriber's name, subscribed channel's name, and the subscribing date.
   Order the result according to channel's name ascending first, and subscribing date descending the next.

   (c) For each customer who hasn't subscribed to any channel yet, list the customer's account number, name, email and status.

   (d) For each channel that has the **lowest capacity**, list its channel name, and the name and email of each of its subscribers.

   (e) For each channel that has more than 500 distinct customers spending time watching from December 24, 2022 to January 3, 2023, list its channel name, and the total number of minutes spent by all customers on watching this channel in the same period of time (from 2022/12/24 to 2023/01/03).

2. (10 Marks) Express the following query in **Relational Algebra** and **Datalog** respectively against the database schema `TelevisionDB`:

   For each channel that has the content type 'News' and is **not** subscribed by any customer(s) whose email is 'donotcare@hotmail.com', list its channel name and main features.

3. (5 Marks) Create a view, called `CustomerSummary`, in the database schema `TelevisionDB`.
   This view should list the name and email of each customer and the number of the TV channels subscribed by this customer.

   If any customer hasn't subscribed to any channel yet, the customer's name and email still should show up in the view result, and list the subscribed channel number as 0.

   The three columns in this view should be called *customerName*, *customerEmail* and *subscriptionNumber* respectively.

4. (10 Marks) Write a C++ function, called `subscribing` that takes a string (accNo) that is a valid customer account number, and a string (cid) that is a valid channel ID as its parameters and tries to let the given customer subscribe to the given channel.

   Your function should use 3 SQL statements to handle this subscribing process based on the following 3 scenarios:

   - if the given customer has already subscribed to the given channel, your function should simply display a proper notice message, and clean up and return;
   - if currently, the number of customers subscribing to the given channel has already reached the capacity of the given channel, your function should display a proper message, and clean up and return;
   - if none of the above cases is true, then your function should record this new subscription, and use the current date (sysdate) as the subscribe date. Don't forget to commit this change to the database, then clean up and return;

   All the parameter data are guaranteed to be properly sanitized before passing into this function and are guaranteed to be valid.

   Your function also takes a database connection as its parameter and this database connection is already properly connected to the database schema `TelevisionDB`.

   The prototype of the function is shown below:

   ```
   void subscribing(Connection *conn, string accNo, string cid);
   ```

5. (5 Marks) The function `subscribing` in the previous question is in a database application program that uses the username "*tvappsub*" and a password to establish the connection to the database `TelevisionDB`.

   Write SQL statement(s) to grant **only the necessary privilege** to the application program so that the function `subscribing` can be executed successfully.

6. (5 Marks) A customer with the account number '*1233755*' decided to unsubscribe all services from the television company that uses the database `TelevisionDB`.
   Write a SQL statement to remove all this customer's channel subscriptions from the database first, then write a SQL statement to change this customer's status to 'inactive'.

7. (5 Marks) Explain the **semantical** difference between the following two SQL queries (a) and (b).

```
    query (a)
SELECT name, email
FROM   Accounts A
WHERE  EXISTS (SELECT *
               FROM   Channels C
               WHERE  NOT EXISTS (SELECT *
                                  FROM   Subscribes S
                                  WHERE  S.accNo = A.accNo
                                    and  S.cid = C.cid));
```

```
    query (b)
SELECT name, email
FROM   Accounts A
WHERE  NOT EXISTS (SELECT *
                   FROM   Channels C
                   WHERE  EXISTS (SELECT *
                                  FROM   Subscribes S
                                  WHERE  S.accNo = A.accNo
                                    and  S.cid = C.cid));
```

8. (5 Marks) Write a SQL statement, and describe how this SQL statement can be used to determine whether the functional dependency
   $email \rightarrow name$
   holds on the current instance of the relation `Accounts` in the database `TelevisionDB`.

9. (5 Marks) Given a relation R and a set of functional dependencies F, answer the following questions and justify your answers:

   (a) Is it possible that this relation R doesn't have any super keys?

   (b) Is it possible that this relation R has multiple candidate keys?

   (c) Is it possible that this relation R has multiple primary keys?

10. (5 Marks) Let R = {ABCDE} be a relation schema, where each letter represents an attribute in R, and each of the functional dependencies in the set $F = \{A \rightarrow BC, C \rightarrow D, D \rightarrow E\}$ holds on R.
    Answer the following questions and justify your answers.

    (a) Is there any benefit to use $(A \rightarrow BC)$ to decompose relation R to {ABC} and {ADE}?

    (b) Is there any benefit to use $(C \rightarrow D)$ to decompose relation R to {CD} and {ABCE}?

11. (5 Marks) Describe a procedure/algorithm that draws a serialization/precedence graph based on a given schedule.
    Explain why an acyclic serialization graph means that its corresponding schedule is a conflict serializable schedule.

12. (5 Marks) Oracle database system can be set to run on one of the following 4 isolation levels:

Isolation Level	**Dirty Reads**	**Non-Repeatable Reads**	**Phantoms**
read uncommitted	allowed	allowed	allowed
read committed	not allowed	allowed	allowed
repeatable read	not allowed	not allowed	allowed
serializable	not allowed	not allowed	not allowed

    On our csci server, Oracle database system runs on the level of "read committed". We may encounter some data inconsistency. But these anomalies can be tolerated in our lab environment where we read public data (in HR schema) and perform updates only in our own individual schema.

    Is there any application environment in which the Oracle database system can and should even run on the level of "read uncommitted"? If your answer is yes, describe a simple example of such an environment. If your answer is no, justify your answer.

13. (10 Marks) The following items describe some information/data we want to capture and store in a relational database about a student assess system used by a university for one term only (similar to a vastly simplified VIU Learn assessment system).

    - There are many students in the university. Each student has a unique student number, a name and an email.
    - There are many courses offered by the university in one term. Each course has a unique combined course ID and course number, and a title.
    - Multiple assignments can be created under each course. Each assignment has a unique assignment name within its course, a max grade, a start date and a due date.
    - Each course has a class list of students who are enrolled to take this course this term. A course certainly can have many enrolled students, and a student can enroll in multiple courses in the term.
    - An enrolled student of a course can submit assignment solutions to the assignments created in the course. To simplify scenario, a student is only allowed to make one submission to any one assignment; and each submission is only allowed to contain one solution file and has a submit date/time. You can use the file name to represent a solution file.

    Your tasks:

    (a) Draw an ER diagram that is consistent with the above description. Clearly show the identifiers and attributes of each entity set, and the cardinality (M-1 or M-M) and attributes (if any) of each relationship set. Make sure that meaningful names are used.

    (b) Translate your ER diagram to its equivalent relational schema. Clearly show the table names, attribute names; underline each table's primary key; and identify foreign keys and the tables they reference.
    (Note that you do NOT need to write the SQL statements to create these tables.)

======= END OF EXAM QUESTIONS =======