

# Final exam in 1DL301 Database Design I

Department of Information Technology, Uppsala University

April 6, 2020, 08:00 – 20:00

Read this document carefully before you start solving the tasks.

The exam is not anonymous, remember to include your name and your email address in the report.

In this individual home exam you are going to use a small database for a digital media store. You can access the database in two different ways (see below), it is completely up to you which one you use.

You have 12 hours to solve the tasks and submit your solutions. You will not need so much time (we expect that you will need around 5–6 hours to solve all tasks), but we wanted to give you enough time to take breaks and avoid time stress.

If you think that some tasks need further clarification, send an email to Jan Kudlicka (jan.kudlicka@it.uu.se). All clarifications will be published on

<https://db1.course.it.uu.se/exam/>

Make sure to check the page during the exam (don't forget to reload the page).

## Allowed aids

- You are allowed to use all course material and your notes.
- You are allowed to use resources on Internet (such as Wikipedia, W3Schools, RDBMS reference manuals), but you are **NOT** allowed to actively search for the solutions.
- You are **NOT** allowed to get help from other people to solve the tasks.

## Database

To access the database you can

- either use the web interface (SQL Explorer) at:  
<https://db1.course.it.uu.se/sqlexplorer/exam>  
with the following credentials: username *db1*, password *db1*,
- or download the database in the SQLite format from:  
<https://db1.course.it.uu.se/exam/exam.db>

Some useful information about SQLite3 can be found in Appendix B.

You are allowed to use other software that works with SQLite databases, but **only** to execute SQL commands written by you. You are **NOT** allowed to use wizards or any other graphical user interface to prepare SQL or to generate ER diagrams.

The database schema can be found in Appendix A. (You will need the schema to understand what is stored in the database and to solve the first task.)

## Tasks

Unless explicitly specified, each task is worth 1 point.

1. Based on the CREATE TABLE statements in Appendix A (page 5), make an ER model of the database. Give suitable names to the relationships. (Remember cardinality and participation constraints.)

The diagram must use either the notation used in the textbook (and the lectures) or the crow's foot notation.

To save you some time: There are a few tables that include the following address fields: *Address*, *City*, *State*, *Country* and *PostalCode* (and the same fields with the prefix *Billing*-). You are allowed to replace these attributes with a single attribute named *FullAddress* in your diagram.

Note: You are **not** allowed to use any software to connect to a database and generate the diagram automatically.

(5 points)

2. The store owner wants to add a new feature to rate tracks. A customer can rate a track by giving it between 1 and 10 stars. The rating can be accompanied by an optional review (that can be a longer text). One customer can rate one track only once.

Extend your ER model based on this specification. Then convert the new entities and/or relationships to a relational model, and write the SQL statement(s) to create the new tables (remember to include constraints for the primary and foreign keys).

Note: Both the ER diagram and the SQL statement(s) must be included in your submission. You are **not** allowed to use any software to convert the ER model to a relational model (or CREATE TABLE statements).

(3 points)

3. Write an SQL query to return the number of all tracks in the store.

(0.5 points)

4. Write an SQL query to return all artists whose name starts with K. Sort the result set by their name.

Return the following columns in the result set: artist id, artist name.

5. Write an SQL query to return all track names, together with the name of the artist and the title of the album. Sort the result set by the artist name first, then by the album title.

Return the following columns in the result set: track id, track name, artist name, album title.

6. Write an SQL query to return all tracks with their revenue (i.e. how much customers paid for each track; for tracks that were never sold you can use either NULL and 0). Sort the result set by the revenue in the descending order.

Return the following columns in the result set: track id, track name and the revenue.

Hint: The parameter of any aggregate function might also be an arithmetic expression involving several columns.

7. Write an SQL query to return tracks that were never sold.

Return the following columns in the result set: track id and track name.

Hint: You can use a nested query that returns tracks that were sold at least once.

8. Write an SQL query to return all genres with at least 100 tracks in the store. Include the number of tracks for each genre.

Return the following columns in the result set: genre id, genre name and the number of tracks.

9. Write SQL statements that insert a new invoice with the id 413 to the database. The invoice is issued today to the customer with the id 7 for buying a track with the id 77. The customer paid 0.99. (You don't need to insert any values for the NOT NULL fields.)

Note: If you are using the web interface to access the database, you will need to run the statements one by one (i.e., you will not be able to write all statements separated by semicolons and run them together). In your report include all statements separated by semicolons.

10. Related to the previous task, we need to guarantee that either all statements get executed as a unit (and both the invoice and all of its invoice lines get stored), or none does (e.g. in case of a software or hardware failure). What do you need to do in SQL to guarantee this? What database concept(s) is/are related to this?
11. Based on the CREATE TABLE statements in Appendix A, write down all full functional dependencies for the following three tables:

- Track
- PlaylistTrack
- Genre

(1.5 points)

12. How does the answer to the previous task change if we add a unique constraint on the genre name (in the *Genre* table)?

(0.5 points)

13. Are these three tables in Boyce–Codd normal form (BCNF)? Motivate your answer.

(1.5 points)

14. All employees in the *Employee* table have unique names. Does this mean that {FirstName, LastName} is a superkey? Motivate your answer.

15. (Not related to the exam database.) Consider a relation in 1NF and assume that no column depends on the empty set. If the relation's candidate keys have only one attribute each, the relation must be at least in 2NF. Explain why.

16. When a customer logs in to the online store (using their email as the username), the following query is executed to be able to show a greeting:

```
SELECT FirstName, LastName, Company FROM Customer WHERE Email = ?
```

(The question mark represents a placeholder that is replaced by the customer's email address.)

The database server needs to go through all records in the *Customer* table to execute the query. What can you do to avoid this and significantly reduce the execution time? Include an SQL statement in your answer.

## Submission instructions

### Files in your submission

You need to submit the following files:

- A report with your solutions **in the PDF format**. The report **must** include your name and your email address on the front page.
- ER diagrams in the PNG or JPEG format. Note: The ER diagrams **must** be included in the report as well.
- SQL submission file (sqlsubmission.txt). Go to   
<https://db1.course.it.uu.se/gensub/>  
(username *db1*, password *db1*). Fill out your name, and copy and paste your SQL queries and statements into the form. Click the button below the form to generate and download the submission file. Note: All SQL queries/statements **must** be included in the report as well.

### ER diagrams

- Avoid crossing lines and overlapping elements. Make sure that all text is easy to read.
- You can draw the diagram (and export it to a PNG image) using ERD plus (<https://erdplus.com>). You are allowed to use ERD plus only to draw the diagrams, you are not allowed to use it for generating relational models. Note: Logged-in users can save their work on the server, but we strongly recommend to back up your work often by exporting the diagram (and saving it on your own computer). You can use the tool as an unregistered user, but if you accidentally click on a link in the top of the page or close the window, there is no confirmation dialog and you will lose your work.
- You can draw the diagram by hand. If you don't have a scanner at home, you can use your mobile phone or tablet to take a picture. Make sure that the whole diagram is included, the picture is sharp, and that all text is easy to read.

### SQL

- In your report, make sure that you wrap long lines so that they do not run off the page. (SQL queries and statements that run off the page will not be graded.)
- Do not include the result sets in your report.
- Remember to include sqlsubmission.txt (see above).

### Submission

**If you are registered to a Fall 2017, 2018 or 2019 instance of the course:**

Go to the course page in the Studentportalen and use the assignment page *Exam April 6, 2020* (under *Exam submission* in the left menu) to submit your solution.

**All other students:**

Send your solution by email to [jan.kudlicka@it.uu.se](mailto:jan.kudlicka@it.uu.se). The subject of the mail **must** be *IDL301 exam submission*. Remember to attach the files. Your submission must be received before the exam deadline. We will send you a confirmation email after the submission deadline.

## A. Database schema

```
CREATE TABLE Album
(
    AlbumId INTEGER NOT NULL,
    Title NVARCHAR(160) NOT NULL,
    ArtistId INTEGER NOT NULL,
    PRIMARY KEY (AlbumId),
    FOREIGN KEY (ArtistId) REFERENCES Artist (ArtistId)
);

CREATE TABLE Artist
(
    ArtistId INTEGER NOT NULL,
    Name NVARCHAR(120),
    PRIMARY KEY (ArtistId)
);

CREATE TABLE Customer
(
    CustomerId INTEGER NOT NULL,
    FirstName NVARCHAR(40) NOT NULL,
    LastName NVARCHAR(20) NOT NULL,
    Company NVARCHAR(80),
    Address NVARCHAR(70),
    City NVARCHAR(40),
    State NVARCHAR(40),
    Country NVARCHAR(40),
    PostalCode NVARCHAR(10),
    Phone NVARCHAR(24),
    Fax NVARCHAR(24),
    Email NVARCHAR(60) NOT NULL,
    SupportRepId INTEGER,
    PRIMARY KEY (CustomerId),
    FOREIGN KEY (SupportRepId) REFERENCES Employee (EmployeeId)
);

CREATE TABLE Employee
(
    EmployeeId INTEGER NOT NULL,
    LastName NVARCHAR(20) NOT NULL,
    FirstName NVARCHAR(20) NOT NULL,
    Title NVARCHAR(30),
    ReportsTo INTEGER,
    BirthDate DATETIME,
    HireDate DATETIME,
    Address NVARCHAR(70),
    City NVARCHAR(40),
    State NVARCHAR(40),
    Country NVARCHAR(40),
    PostalCode NVARCHAR(10),
    Phone NVARCHAR(24),
    Fax NVARCHAR(24),
    Email NVARCHAR(60),
    PRIMARY KEY (EmployeeId),
    FOREIGN KEY (ReportsTo) REFERENCES Employee (EmployeeId)
);

CREATE TABLE Genre
(
    GenreId INTEGER NOT NULL,
    Name NVARCHAR(120),
    PRIMARY KEY (GenreId)
);
```

```

CREATE TABLE Invoice
(
    InvoiceId INTEGER NOT NULL,
    CustomerId INTEGER NOT NULL,
    InvoiceDate DATETIME NOT NULL,
    BillingAddress NVARCHAR(70),
    BillingCity NVARCHAR(40),
    BillingState NVARCHAR(40),
    BillingCountry NVARCHAR(40),
    BillingPostalCode NVARCHAR(10),
    Total NUMERIC(10,2) NOT NULL,
    PRIMARY KEY (InvoiceId),
    FOREIGN KEY (CustomerId) REFERENCES Customer (CustomerId)
);

CREATE TABLE InvoiceLine
(
    InvoiceLineId INTEGER NOT NULL,
    InvoiceId INTEGER NOT NULL,
    TrackId INTEGER NOT NULL,
    UnitPrice NUMERIC(10,2) NOT NULL,
    Quantity INTEGER NOT NULL,
    PRIMARY KEY (InvoiceLineId),
    FOREIGN KEY (InvoiceId) REFERENCES Invoice (InvoiceId),
    FOREIGN KEY (TrackId) REFERENCES Track (TrackId)
);

CREATE TABLE MediaType
(
    MediaTypeId INTEGER NOT NULL,
    Name NVARCHAR(120),
    PRIMARY KEY (MediaTypeId)
);

CREATE TABLE Playlist
(
    PlaylistId INTEGER NOT NULL,
    Name NVARCHAR(120),
    PRIMARY KEY (PlaylistId)
);

CREATE TABLE PlaylistTrack
(
    PlaylistId INTEGER NOT NULL,
    TrackId INTEGER NOT NULL,
    PRIMARY KEY (PlaylistId, TrackId),
    FOREIGN KEY (PlaylistId) REFERENCES Playlist (PlaylistId),
    FOREIGN KEY (TrackId) REFERENCES Track (TrackId)
);

CREATE TABLE Track
(
    TrackId INTEGER NOT NULL,
    Name NVARCHAR(200) NOT NULL,
    AlbumId INTEGER,
    MediaTypeId INTEGER NOT NULL,
    GenreId INTEGER,
    Composer NVARCHAR(220),
    Milliseconds INTEGER NOT NULL,
    Bytes INTEGER,
    UnitPrice NUMERIC(10,2) NOT NULL,
    PRIMARY KEY (TrackId),
    FOREIGN KEY (AlbumId) REFERENCES Album (AlbumId),
    FOREIGN KEY (GenreId) REFERENCES Genre (GenreId),
    FOREIGN KEY (MediaTypeId) REFERENCES MediaType (MediaTypeId)
);

```

## B. SQLite

Note: This section is relevant only if you decide not to use the web interface.

SQLite is a simple RDBMS that does not require a database server, each database is stored in a single file on your computer. If you use Mac or Linux, or you have installed Anaconda on your computer, you should be able to use SQLite3 without installing any additional packages. If you use Windows and don't have Anaconda, you can download SQLite3 from:

<https://www.sqlite.org/download.html>

Once the SQLite3 is installed on your computer, run

```
sqlite3 exam.db
```

in a terminal (or cmd in Windows). The parameter (`exam.db`) is the filename where the database is stored. Make sure that you run the command in the same directory where you have downloaded the file.

For historical reasons, the foreign key constraints are disabled by default. To enable them, enter

```
PRAGMA foreign_keys = ON;
```

every time you start sqlite3.

### Some additional tips

To make the result sets easy to read, change the output mode to column and switch on the headers:

```
sqlite> .header on
sqlite> .mode column
```

You can use `.tables` to list all tables, and `.schema` followed by the table name to show its schema.