Final exam in 1DL301 Database Design I

Department of Information Technology, Uppsala University
April 6, 2020, 08:00 – 20:00

Sample solution

1. See Fig. 1, ignore the *rated by* relationship (which is a solution of Task 2).

2. See the *rated by* relationship in Fig. 1.

```
CREATE TABLE Rating
(
   CustomerId INTEGER NOT NULL,
   TrackId INTEGER NOT NULL,
   Rating INTEGER NOT NULL,
   Review TEXT,
   PRIMARY KEY (CustomerId, TrackId),
   FOREIGN KEY (CustomerId) REFERENCES Customer(CustomerId),
   FOREIGN KEY (TrackId) REFERENCES Track(TrackId)
);
```

3.
```
SELECT COUNT(*)
FROM Track
```

4.
```
SELECT ArtistId, Name
FROM Artist
WHERE Name LIKE "K%"
ORDER BY Name
```

5.
```
SELECT Track.TrackId, Track.Name AS Track, Artist.Name AS Artist,
     Album.Title AS Album
FROM Track, Album, Artist
WHERE Album.AlbumId = Track.AlbumId
  AND Album.ArtistId = Artist.ArtistId
ORDER BY Artist.Name, Album.Title
```

6.
```
SELECT Track.TrackId, Track.Name,
     SUM(InvoiceLine.Quantity*InvoiceLine.UnitPrice) AS Revenue
FROM Track
LEFT JOIN InvoiceLine ON InvoiceLine.TrackId = Track.TrackId
GROUP BY Track.TrackId
ORDER BY Revenue DESC
```

7.
```
SELECT TrackId, Name
FROM Track
WHERE TrackId NOT IN (SELECT TrackId FROM InvoiceLine)
```

8.
```
SELECT Genre.GenreId, Genre.Name, COUNT(Track.TrackId) AS TrackCount
FROM Genre, Track
WHERE Track.GenreId = Genre.GenreId
GROUP BY Genre.GenreId
HAVING TrackCount >= 100
```

9.
```
INSERT INTO Invoice(InvoiceId, CustomerId, InvoiceDate, Total)
VALUES (413, 7, '2020-04-06', 0.99);

INSERT INTO InvoiceLine(InvoiceLineId, InvoiceId, TrackId, UnitPrice, Quantity)
VALUES (2241, 413, 77, 0.99, 1)
```

10. To ensure *atomicity* (which is one of the *ACID* properties) we need to use a *transaction*.

    In SQL we start the transaction with `BEGIN` (or `BEGIN TRANSACTION` or `START TRANSACTION` – depending on which RDBMS we use) and commit it with `COMMIT`:

```
BEGIN;

INSERT INTO Invoice(InvoiceId, CustomerId, InvoiceDate, Total)
VALUES (413, 7, '2020-04-06', 0.99);

INSERT INTO InvoiceLine(InvoiceLineId, InvoiceId, TrackId, UnitPrice, Quantity)
VALUES (2241, 413, 77, 0.99, 1);

COMMIT
```

11. - Track:
      $\{\text{TrackId}\} \rightarrow \{\text{Name, AlbumId, MediaTypeId, GenreId, Composer, Milliseconds, Bytes, UnitPrice}\}$
    - PlaylistTrack:
      No non-trivial full functional dependencies.
    - Genre:
      $\{\text{GenreId}\} \rightarrow \{\text{Name}\}$

12. Genre:

    $\{\text{GenreId}\} \rightarrow \{\text{Name}\}$
    $\{\text{Name}\} \rightarrow \{\text{GenreId}\}$

13. Yes, all of them are in BCNF. To prove this, we need to show that each functional dependency $X \rightarrow Y$ is either trivial or X is a superkey. If we look at the non-trivial full functional dependencies in Task 11, X must be a superkey (and also a candidate key since we use *full* functional dependencies). This is indeed the case for all three tables. (TrackId is a candidate key in Track, and GenreId is a candidate key in Genre. PlaylistTrack has only trivial full functional dependencies.)

14. No. It is not sufficient to look at the current state of the relation, we need to consider all possible valid states. It is indeed possible, that the company might in the future employ two (or more) people with the same first and last name (and in that case, the combination of the first and last name is not able to uniquely identify a single row).

15. A relation is in 2NF if

    (a) it is in 1NF, and

    (b) no non-prime attribute is functionally dependent on a proper subset of any candidate key.

    The first condition is fulfilled by the assumption in the question text.

    Since each candidate key has only one attribute, there is only one proper subset (of any candidate key): the empty set $\{\}$. By assumption no attribute is functionally dependent on the empty set, so also the second condition is fulfilled.

16. We need to index the Email column in the Customer table:

```
CREATE INDEX Customer_Email ON Customer(Email)
```

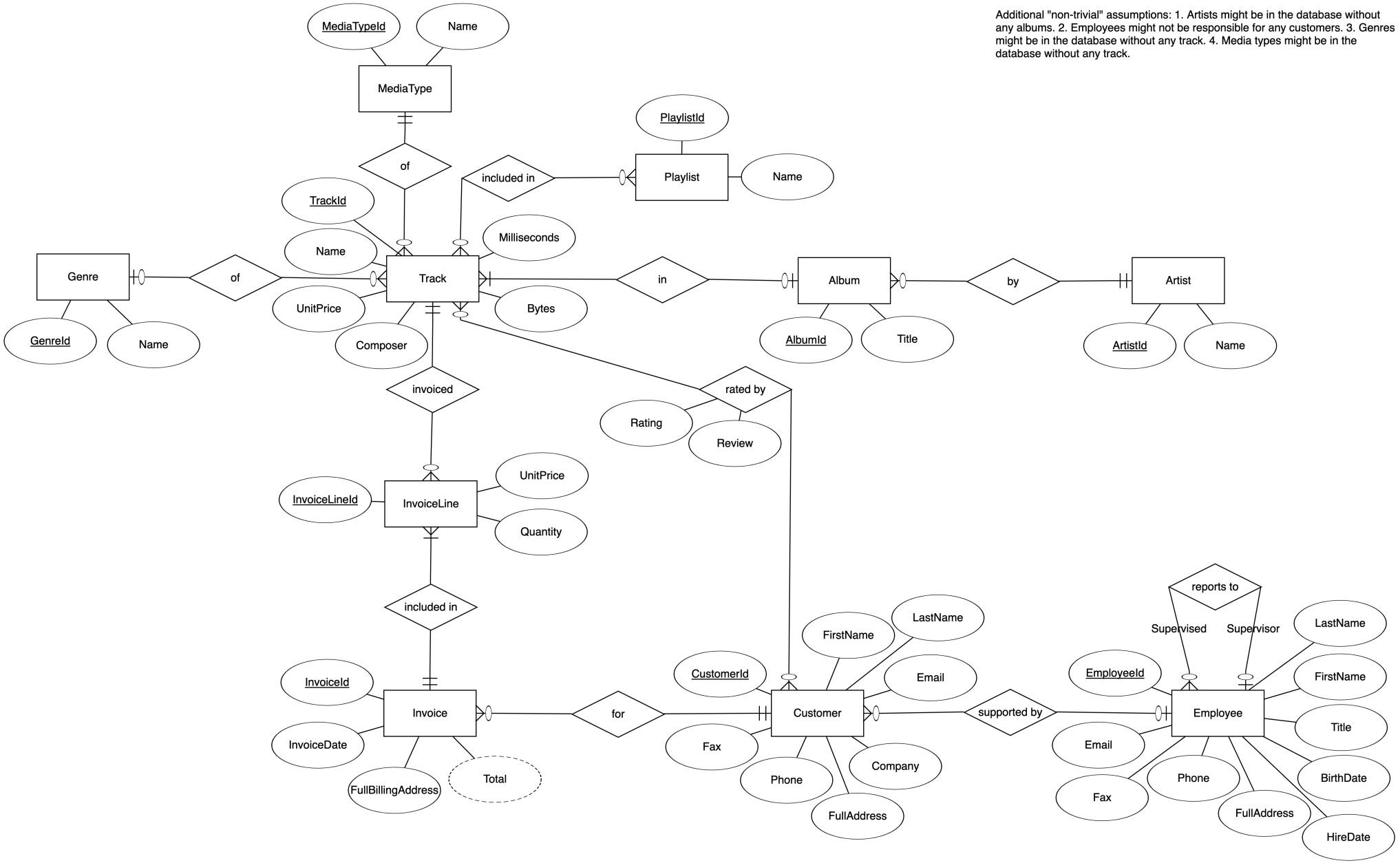

Figure 1: ER model for Tasks 1 and 2