5.2. Task 2, Logical and Physical Model

How To Prepare

- Understand the lectures on normalisation (given live and recorded) and on logical and physical models (only recorded, not given live). You can start working on the task before the normalisation lecture, but you have to understand that lecture before completing the task.
- Read the document tips-and-tricks-task2.pdf
- Download tips-and-tricks-task2.pdf
- .

When To Solve

You're recommended to start working on this task as soon as possible after seminar 1, but remember to first do the preparations mentioned above. The deadline for submitting the report is found on the seminar 2 assignment page.

Intended Learning Outcomes

- Model needs for information based on an organizational description and convert the model to a functioning database.
- Use relational databases and query languages.

Mandatory Part

Translate the conceptual model from task one into the same kind of model that was created in the lecture on logical and physical models, that is a logical model with enough physical aspects to enable creating a database. You are allowed to change the conceptual model if you discover flaws, but only change if it is necessary. The model that is created must cover the entire description of the Soundgood music school company in section one above. The diagram must be made in a crow foot notation (for example IE notation). Also create a database based on the model, the database must be exactly as described in the model. If you discover flaws when creating the database, and want to change it, also the model must be changed. You are advised to also look at tasks three and four already now, to get an understanding of how your database will be used. Below follows guidelines for what shall be written in the report.

- In the *Method* chapter of your report, mention diagram editor(s) and other tool(s) you used, and explain the procedure you followed to create the model. You do not have to mention all eleven steps covered in the videos on on logical and physical models, but it must be clear how you proceeded. *Do not explain the result of each step of your procedure*, only explain the steps themselves.

- In the *Result* chapter of your report, show and briefly explain your model. Also include a link to a git repository where you have stored SQL scripts that create your database. There shall be one script that creates the database, and another script that inserts data. HINT: You might want to use an online generator to create the data, for example https://generatedata.com/
- Links to an external site.
- , instead of creating the data manually. You're also not required to write the script that creates the database manually. If you create the database using a GUI tool, you may generate the script using for example pg_dump (note that pg_dump is a Postgres tool, but there are similar tools for MySQL). You may explain the SQL in the scripts if you wish, and think it clarifies the model, but it is not required to write such an explanation.
- In the *Discussion* chapter of your report, evaluate your model. Suggested assessment criteria are found in assessment-criteria-seminar2.pdf
- Download assessment-criteria-seminar2.pdf
- , you do not have to cover them all. **These same criteria will also be used to grade your project report**.

Higher Grade Part

All three requirements below must be met to get the higher grade points. You can not get part of the points by meeting just some of the requirements.

1. **The report must include a relevant and extensive Discussion** chapter about the mandatory part of the task.
2. **Make sure that the models created in this task contain all data required by Soundgood, and don't require an application using the database to manage any data at all**. It could for example be tempting to create a database that doesn't store the number 2, which is mentioned in the text *each student can rent up to two specific instruments*, and instead hardcode the number 2 into the application calling the database. Carefully read the description of the Soundgood music school in section one above, and make sure your models include all data mentioned in that text. You don't have to consider any data that's not clearly mentioned in section one. Discuss advantages and disadvantages of storing all data in the database, as is done here, instead of having some data in the application. This discussion shall be placed in the Discussion chapter of the report.
3. **This task is to handle the fact that prices can change.** Say for example that a student takes a certain kind of lesson, and that the cost of that lesson changes before the student has payed. In this situation, it must be possible to find the cost of the lesson when the student took it, not just the current cost, since the student shall pay the cost that was used when the lesson was taken. You have to solve this task by creating a model which allows adding a new price, when a price changes, instead of updating the

existing price. This also means the model must make it possible to tell which of all stored prices for a certain lesson is currently valid. Discuss advantages and disadvantages of keeping multiple versions of data, as is done here. This discussion shall be placed in the Discussion chapter of the report. HINT: A solution to this problem often includes storing the time a certain value is inserted into the database. You're however not required to do that, there are also other possible solutions.

5.3. Task 3, SQL

How To Prepare

- Understand the lecture on SQL (given live and recorded).
- Read the document tips-and-tricks-task3.pdf
- Download tips-and-tricks-task3.pdf
- .

When To Solve

You're recommended to start working on this task as soon as possible after seminar 2, but remember to first do the preparations mentioned above. The deadline for submitting the report is found on the seminar 3 assignment page.

Intended Learning Outcomes

- Use relational databases and query languages.
- Describe and explain basic concepts, principles and theories in the area of data/databases/data storage and in information administration and database design

Mandatory Part

The goal here is to create OLAP, Online Analytical Processing, queries and views. Such queries serve to analyze the business and to create reports. **You're also required to analyze the efficiency of one of the queries with EXPLAIN ANALYZE**. Even though that's not standard SQL, it's available in both PostgreSQL and MySQL. You also have to make sure the database contains sufficient data to check that all queries work as intended. If needed, update the script that inserts data, created in task two. You're allowed to change the database you created in task two if needed. The queries that shall be created are explained below, only OLAP queries will be created here. The OLTP (Online Transaction Processing) queries used by the business itself, which in the case of Soundgood is to rent out instruments, register taken and given lessons, etc, will be created in task 4, together with the program executing them.

HINT: You might want to use an online generator to create the database content, for example https://generatedata.com/

Links to an external site.

, instead of creating the data manually.

The following queries will be executed manually, to generate analysis reports.

1. **Show the number of lessons given per month during a specified year**. This query is expected to be performed a few times per week. It shall be possible to retrieve the total number of lessons per month (just one number per month) and the specific number of individual lessons, group lessons and ensembles (three numbers per month). It's not required that all four numbers (total plus one per lesson type) for a particular month are on the same row; you're allowed to have one row for each number as long as it's clear to which month each number belongs. However, it's most likely easier to understand the result if you do place all numbers for a particular month on the same row, and it's an interesting exercise, therefore you're encouraged to try. Table 1 below is an example result of such a query, illustrating the expected output.

	Total	**Individual**	**Group**	**Ensemble**
Oct	2	1	0	1
Nov	3	0	2	1
Dec	10	4	4	2

2. **Show how many students there are with no sibling, with one sibling, with two siblings, etc**. This query is expected to be performed a few times per week. The database must contain students with no sibling, one sibling and two siblings, but doesn't have to contain students with more than two siblings. *Note that it's not allowed to solve this by just adding a column with sibling count (maybe called no_of_siblings or something similar) to the student table*. Such a solution would be almost impossible to maintain since it doesn't tell who's a sibling of who. If a student quits, there wont be any way to update the sibling count of that student's siblings. Table 2 below is an example result of such a query, illustrating the expected output.

No of Siblings	**No of Students**
0	3

1	2
2	5

3. **List ids and names of all instructors who has given more than a specific number of lessons during the current month**. Sum all lessons, independent of type, and sort the result by the number of given lessons. This query will be used to find instructors risking to work too much, and will be executed daily. Table 3 below is an example result of such a query, illustrating the expected output.

Instructor Id	First Name	Last Name	No of Lessons
13	Albus	Dumbledore	5
15	Pomona	Sprout	5
2	Gilderoy	Lockhart	4

The following queries will be performed programmatically, and the results will be displayed on Soundgood's web page. You only have to create the queries, not the web page.

4. **List all ensembles held during the next week**, sorted by music genre and weekday. For each ensemble tell whether it's full booked, has 1-2 seats left or has more seats left. HINT: you might want to use a CASE statement in your query to produce the desired output. Table 4 below is an example result of such a query, illustrating the expected output.

Day	Genre	No of Free Seats
Tue	Gospel	No Seats
Wed	Punk	1 or 2 Seats
Wed	Rock	No Seats
Fri	Rock	Many Seats

Below is specified what shall be written in the report.

- In the **Method** chapter of your report, mention which DBMS you use, which tool is used to develop SQL queries, and how you have verified that your SQL queries work as intended. You shall just tell which method you

used for verifying that the queries work as intended, not explain in detail how you verified the functionality of each query..

- In the **Result** chapter of your report, include a link to a git repository where you have stored a script with all queries. *Also explain each query and show that all queries work as intended by* ***including the output of each query***. The git repository must also contain the scripts that create the database and insert data. It shall be possible to test your solution by executing first the script that creates the database, then the script that inserts data, and finally any of the queries created in this task.
- In the **Discussion** chapter of your report, evaluate your queries. Suggested assessment criteria are found in assessment-criteria-seminar3.pdf
- Download assessment-criteria-seminar3.pdf
- . You don't have to cover them all, but **you must at least cover the bullet on EXPLAIN, which means you have to analyze at least one of your queries. The assessment criteria in this document will also be used to grade your project report**.

Higher Grade Part

Both the following two requirements must be met to get the higher grade points. You can not get half of the points by meeting just one requirement.

1. **The report must include a relevant and extensive Discussion chapter** about the mandatory part of the task.
2. **This task concerns historical data and denormalization**. Historical data is normally a strong candidate for denormalization, since it will not be updated. That makes write performance uninteresting, and also strongly reduces the risk of data anomalies. You will first have to read about denormalization (for example in the text book), since that is not much covered in the lectures.

   The task is that, for marketing purposes, the Soundgood music school wants to be able to see which lessons each student has taken, and at which cost, since they first joined the school. This means you are required to keep records of all given lessons, including participants and price for each lesson. One way to do this would be to just save a copy of the entire database each day, or each time a price is changed. This is however unnecessarily complicated, since it will be slow to generate reports from such a database, and since you would have to add some kind of time interval for the prices, instead of just keeping the current price. A better way to solve the problem is to create a denormalized historical database. One example of an appropriate denormalization is to store prices of lessons in a column in the lessons table, instead of keeping prices in a separate table, even though this will lead to duplicated data, with the same price appearing in all lessons of the same type. Another possible

denormalization is to merge individual, group and ensemble lessons into one single lesson table.

Create a historical database like the one described above, and also create SQL statements for copying data from your present database to the historical database. **The historical database shall contain the following data: lesson type (group, individual or ensemble), genre (empty if it's not en ensemble), instrument (empty if it's an ensemble), lesson price, student name, student email.** You're not required to write a program that automates moving data to the historical database, it's sufficient to create SQL statements that are executed manually. Discuss advantages and disadvantages of using denormalization. This discussion shall be placed in the Discussion chapter of the report.