Seminar 2 - Logical and Physical Model

Data Storage Paradigms, IV1351

Seema Bashir

23/11-2023

1 Introduction

The aim of this task is to transform the conceptual model from task one into a logical model, incorporating sufficient physical aspects as discussed in the lecture on logical and physical models. This logical model intends to possess the necessary elements to facilitate the creation of a database. The model also meticulously adheres to the data specified in the requirements and the descriptions which highlights how the Soundgood Music School manages its routines. Thereby, the task also necessitated following the algorithms and steps crucial for transforming the conceptual model into a logical one.

Creating the logical model, also makes it imperative to ensure its ability to support a diverse range of operations- a crucial step before deeming the database complete. Therefore, a thorough examination of the business description becomes essential to discern the operations required.

In collaboration with Razan Yakoub, this report details the approach employed and executed, providing an overview of the collaborative results in meeting the specified requirements.

2 Literature Study

In order to prepare for this seminar, chapters 9.1, 14 and 15.1-15.2 in the 7th edition of Fundamentals of Database Systems by Elmasri and Navathe are evaluated. The relational database design produces a set of relations. The ultimate goals is create such a database in which information is preserved, resulting in minimum redundancy. Reducing the need for multiple updates to maintain consistency. Thereby, the process of normalisation is used to improve data integrity, minimize data duplication, and enhance overall database performance.

The First Normal Form serves as the starting point for organizing a database systematically. It mandates that every piece of data in a table should be indivisible and singular, eliminating any instances of repeating groups or arrays. This ensures that each attribute holds only non-composite values, laying the groundwork for a well-structured database that addresses redundancy and allows for more straightforward data manipulation.

Moving to the Second Normal Form, its primary goal is to remove partial dependencies within a relational database. A table achieves 2NF if it already meets the requirements of 1NF, and all non-prime attributes are fully dependent on the entire primary key. This entails that each non-prime attribute must be connected to the complete primary key, avoiding dependencies on only a part of it. By doing so, 2NF not only enhances data integrity but also streamlines the organization of information, minimizing redundancy, and facilitating more efficient querying and maintenance of the database.

In the context of the Third Normal Form (3NF), it represents a pivotal phase in the normalization process, focusing on refining the structure of relational databases. The emphasis in 3NF is on eradicating transitive dependencies, ensuring that non-prime attributes (those not constituting the primary key) do not indirectly rely on each other. Compliance with 3NF mandates that each non-prime attribute is functionally dependent on the primary key, and there should be no partial dependencies. This stringent criterion ensures that all attributes are fully reliant on the primary key, fostering data integrity and promoting a well-structured relational database.

Furthermore, lectures provided by Leif Lindbeck including, *Logical and Physical Model* and *a*re watched and evaluated inorder to grasp an understanding of the steps required to create the logical model for this task. The logical model is a representation of a database's structure and organization without specifying how data is physically stored or accessed. It serves as an intermediary step between the conceptual model (which captures the high-level concepts and relationships) and the physical model (which defines the actual implementation details).

In the logical model, the aim is to model a database instead of the reality. Thus, important steps of creating such a model include first on deciding on which relations (tables) to use for the database. Thereby, it is necessary to verify that the models are normalised such that all specified operations can be performed. Additionally, all attributes datatypes must be specified (not DBMS specific).

Furthermore, the physical model specifies everything related related to how the data defines in the logical model is stored. Thereby it is adapted to a specific DBMS. Additionally, it also specifies storage details such as the coloumn types, keys, index and the view.

Lastly, the "Tips and Tricks 2" document provided vital details regarding the logical model. The document highlights the importance of the cardinality of a relation. It illustrates how cardinality on one side indicates how many instances on that side can be related to a single instance on the other side. Additionally, the document also emphasizes on the creation of surrogate primiary keys. Thus, using *INT GENERATED ALWAYS AS IDENTITY'. a*s the data type for surrogate primary keys is a recommended practice in database design. This combination ensures that primary key values are automatically generated, eliminating the need for manual assignment and reducing the risk of inserting non-unique values.

The document further mentions the importance of avoiding duplicated information. In data management, dealing with duplicated information involves the risk of inconsistency and operational challenges. The document emphasizes the necessity of extracting duplicated data to a separate table linked to the original, aiming to enhance data accuracy, simplify updates, prevent errors, and facilitate the introduction of new data elements. Moreover, A common practice is to name them as ($< tablename >_i d$) to minimize confusion and enable natural joins in SQL queries. Despite the drawback of duplicating the table name in the column, this convention is often favored for its query simplification benefits. An alternative approach, if avoiding the inclusion of the table name in a column name is preferred, is to simply name the surrogate primary key as 'id' or 'key.'

Additionally, another rule of hand that is highlighted in the document are the possible mistakes that can be avoided such as the primary key of a table representing multi-valued attribute. When creating a separate table for the attribute with a foreign key referencing the original table's PK, it may inadvertently prevent two entities from sharing the same attribute value. To address this limitation and allow for shared values, the recommendation is to include the foreign key in the primary key. The summary also cautions against an incorrect solution where uniqueness is compromised, leading to an effectively non-multivalued attribute. The overarching message is to understand the implications of chosen solutions when dealing with primary keys for tables representing multi-valued attributes.

Furthermore, another fundamental considerations regarding the design of the database is considering the choice between fewer and larger tables versus more and smaller tables. The key advantage of favoring fewer and larger tables lies in streamlining SQL queries through reduced JOIN operations. However, it is crucial to recognize potential drawbacks, such as the information duplicated.

Lastly, the documents also details the use of enumerated types. Commonly handled through a separate table (lookup table), enums are associated with columns in the main table via foreign keys. Advocates of using enums in SQL argue for simplified queries, but this can be addressed by creating a view that combines the lookup table with the main table.

3 Method

In order to facilitate the application of the Logical Model and the smooth progression from the Conceptual Model to the Logical Model, the Astah diagram editor was consistently utilized. The adoption of the IE notation (crow foot) remained unchanged. Moreover, PostgreSQL is used as the underlying relational database management system, enabling the creation and management of the database through command-line interactions.

For the creation of the logical model, a series of 11 steps systematically are systematically followed in order to transform the conceptual model into a model with logical model with enough physical traits. Thereby, the first step involves converting entities into tables, a necessity prompted by distinct naming conventions. Given the restriction on multivalued attributes, each such attribute is treated as an independent entity, thus distinct tables for such multivalued attributes are created with a relational connection to its original entity.

The subsequent step centered on assigning specific datatypes, such as CHAR, VARCHAR, INT, and TIMESTAMP, to individual attributes.After establishing the datatypes for each attribute, the next step involved assessing the respective column constraints. The approach prioritized practicality, real-world relevance when considering on deciding on the appropriateness of constraints. The goal is to ensure that constraints align with the inherent nature of the data, avoiding unnecessary complexities. For instance, the importance of attributes not being NULL is recognized as a relevant consideration in enhancing the overall effectiveness of the database design.

The following stage consists of assigning primary keys to strong entities. With a preference for surrogate keys. Surrogate keys are synthetic keys that are introduced into a database to serve as the primary key of a table.The primary purpose of surrogate keys is to uniquely identify records within a table, and they are often implemented using auto-incrementing integers or other unique identifiers generated by the database management system.

Weaker entities typically lack surrogate keys unless they possess significance on their own or exert strength over another entity. The primary key, inherited as a foreign key in the weaker entity, serves either as the primary key for the weaker end in a one-to-one relationship. Thereby, another important step involves handling foreign key contriants. Handling foreign key constraints involves defining and enforcing rules for actions such as deletion and updating of records. For instance, specifying whether to cascade changes (propagate changes to related records) or restrict deletion (prohibit deletion of a record with dependent records) helps prevent unintended consequences and ensures the integrity of the database relationships.

Moreover, another vital step involes the handling of many to many relations. In a many-to-many relationship, each record in one entity can be linked to multiple records in another, and vice versa. To represent this relationship in a relational database, a new entity (the cross-reference table) is introduced. The primary key of this new entity is typically a combination or set of the primary keys from the two entities involved in the relationship.

Lastly, one approach utilized to adress the ENUM's in the logical model is with the creation of look-up tables. In the logical modeling phase of database development, the creation of lookup tables for enums is a fundamental process for structuring enumerated types. To begin, a separate table is introduced to exclusively house the potential values of the enum. Each row within this table is dedicated to a unique value of the enum, creating a systematic representation of the entire set. These are then connected to the entities with a one to one relation, such that the weaker entities posses the primary key of the look-up table.

In the final phase of bringing the conceptual database model to life, the logical diagram is translated into executable SQL scripts, marking a crucial step in the database implementation process. This transformation involves connecting a command line interface to the PostgreSQLraza database application, allowing for direct interaction with the database server. The SQL code is meticulously crafted to reflect the entities, attributes, relationships, and constraints outlined in the logical diagram. The ER diagram, generated through Astah, serves as a visual guide for this process, enabling the seamless conversion of the conceptual model into tangible database structures. Once the database schema is created, the next step involves populating it with sample data for testing and development purposes. Notably, the generated data is not arbitrary; it is systematically produced using a specialized random data generator tailored for SQL and specifically designed to align with the PostgreSQL database. This comprehensive approach ensures that the database not only mirrors the intended structure but is also enriched with realistic and diverse data, setting the stage for effective application development and testing.

4 Results

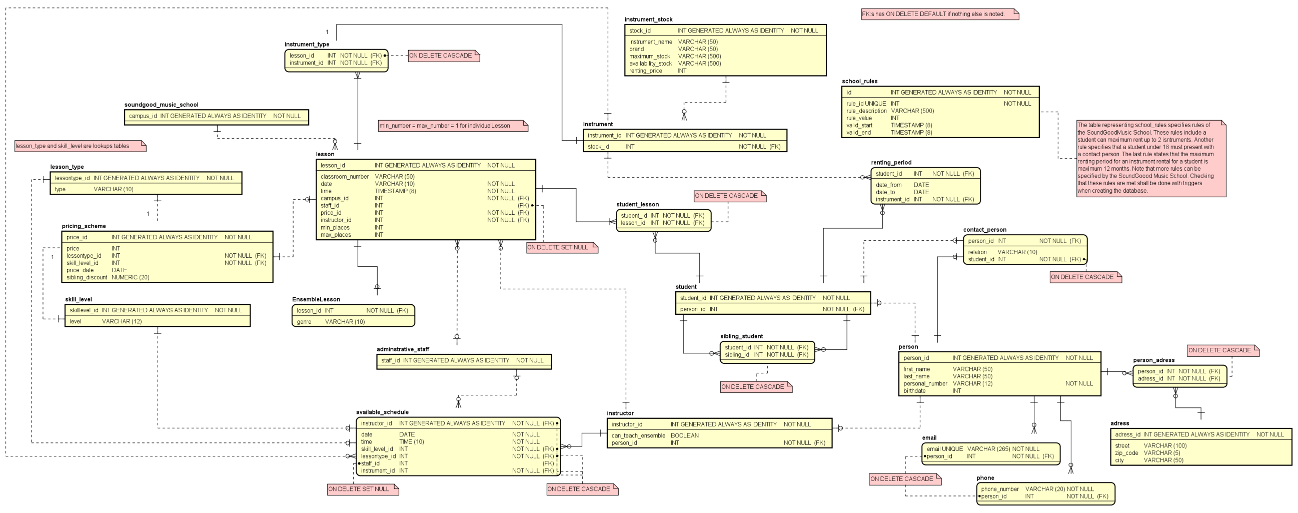

Figure 1: The Logical/Physical Model of Sound-Good Music School Database

Github Repository including relevant SQL scripts `https://github.com/Seemamian/Data_Lagring.git`

Figure 1 depicts the logical model for the SoundGood Music School, featuring various tables that form the foundational structure of the database. The conceptual model exhibits several strong entities, one of which is "lesson." The establishment of the "lesson" table is rooted in its self-sufficiency, distinguishing it from its associated weak entity, "ensemble," and facilitating distinctive identification through the designated primary key known as "lessonid." The "lessonid" is defined as INT GENERATED ALWAYS AS IDENTITY, signifying that it functions as an auto-incrementing integer. This primary key, being an auto-incrementing integer, ensures a dependable and uniquely identifiable identifier for each record within the table.

Given that the "lesson" is related to the "ensemble" through a 1-to-0 or many relationship, the ensemble inherits the primary key, "lessonid," as a foreign key, establishing a relational connection between the two entities.

A consistent structure is evident across the tables "instrumentstock," "instrument," "rentingperiod," and "student." The entity "instrumentstock" serves as a parent to "instrument," establishing a foreign key relationship denoted as "stockid" in the "instrument" table. Despite "instrument" being a child entity concerning "instrumentstock," it functions as a strong entity in relation to the "rentingperiod" entity. Consequently, the "instrument" table possesses its primary key, which is referenced as a foreign key in the "rentingperiod" table.

Similarly, "rentingperiod" is a weak entity in relation to "student," resulting in the inclusion of another foreign key, "studentid." Notably, a deliberate decision was made to designate the "studentid" foreign key as the primary key in the "rentingperiod" table. This choice is deemed crucial as it reflects the significance of each student having a distinct renting period for the purpose of instrument rental.

In the logical model, multivalued attributes from various entities are now represented as individual tables, establishing a relational connection with their respective original entities. The table "instrumenttype," is derived from the "lesson" entity, demonstrating a 1-to-many relationship with "lesson." Each of these multivalued tables incorporates "lessonid" as a foreign key and possesses an additional primary key to ensure uniqueness.

Similarly, the attributes "phone" and "email" have been separated into distinct tables due to their multivalued nature within the context of a person. To guarantee uniqueness in these tables, "phone" includes "personid" as a foreign key, and the table features a column dedicated to storing phone numbers. Both "personid" and the phone numbers column are designated as primary keys for the "phone" table, ensuring distinct and unique identification.

It should be noted that, the coloumns inside the tables representing multivalued attributes have a foreign key contraint set to "ON DELETE CASCADE". This is to ensure when a record in when a record in the referenced (parent) table: "person" is deleted, all corresponding records in the multivalued attribute table: "phone" and "emails" will also be automatically deleted.

In relationships where two entities have a many-to-many connection, a solution is found by introducing a new table, often called a cross-reference table, which acts as a bridge connecting the two entities. This can be seen in two different places in the logical diagram.

Initially, a cross-reference table named "personaddress" is generated to capture the relationship between a person and an address. This illustrates that one person can be associated with multiple addresses, and conversely, multiple persons can reside at a single address, establishing a many-to-many relationship between the two entities. The "personaddress" table contains two foreign keys, personid and addressid, both designated as the primary key to guarantee uniqueness and facilitate access to the associated entities.

Likewise, an additional cross-reference table named "studentlesson" is established to emphasize the connection between a student and a lesson. This table serves as a valuable tool for tracking the lessons undertaken by a specific student. The "studentlesson" table comprises two primary keys, "lessonid" and "studentid," both functioning as foreign keys derived from the corresponding "student" and "lesson" classes. It's noteworthy that the "lesson" class is associated with the "pricingscheme" and possesses the "priceid" as a foreign key, thereby gaining access to the various columns responsible for price calculations.

It's important to highlight that the foreign key columns within the cross-reference table are configured with a foreign key constraint specified as "ON DELETE CASCADE." Consequently, in the event that a record in the referenced (parent) table is deleted, all associated records in the referencing (child) table will be automatically deleted as well. This means that if a student or lesson record is deleted, the corresponding tuple in the "studentlesson" table will also be deleted, thereby ensuring the maintenance of referential integrity.

Moreover, the logical model enhances the visibility of sibling relationships among students through the establishment of specific sibling pairings. This is achieved by introducing a cross-reference table between students, creating the "siblingstudent" table. This table has two primary keys, "studentid" and "siblingid," both serving as foreign keys derived from the parent entity, student. One coloumn presented the "studentid" and another student which is represented by "siblingid".This structured approach enables the system to clearly indicate which students are siblings.

Furthermore, the "availableschdule" entity (weak) is related to instructor (strong), thereby possessing "instructorid" as a foreign key. Since each instructor has one or many available schedules, available schedule cannot exist without the instructor. Thereby, "instructorid" is made as an primary key in the available schedule and the foreign key constraint is set to "ON DELETE CASCADE".

5 Discussion

An effective logical model should encompass all the the necessary information specific to the the business description of Soundgood Music School. The logical model crafted for this assignment adeptly converts the conceptual model into a logical one, encompassing essential physical elements. This procedure is marked by the development of the logical model in Astah, involving a series of steps to transform distinct entities into respective tables and populating columns as respective attributes. Thereby, the model is carefully aligned with the specified data in the requirements and descriptions, offering insights into how Soundgood Music School manages its operations. Additonally, the model is characterized by efficiency and organization, this can be notes with IE notation guidelines and the utilization of conventional naming practices, exemplified by the inclusion of designations like "lesson id."

Moreover, the necessity of normalization is evident in order to provide a structured data representation. With a methodical effort that closely follows the instructions described in the method chapter, there is reason for confidence that the third normal form (3NF) has been attained. This level of standardization provides protection against anomalies and the possibility of duplicate data in addition to facilitating effective data administration.

Making linkages between the tables related to "available schedule" and "instrument type" and the main "instrument" table is a strategic way to deal with the problem of duplicate instrument names across different tables. The removal of the "lesson type" and "skill level" elements from the "lesson" database, which acknowledged their derived from the pricing id, was another noteworthy improvement which was made inorder to avoid duplicates.

The "pricing scheme" table contains information about costs associated with different lesson types and skill levels. Instead of representing this information as free-text constants, it is organized and structured within the "pricing scheme" table. Furthermore, the representation of this data is done using ENUMs (enumerated types) through lookup tables, which enhances efficiency and clarity in handling and referencing the cost-related information in the context of lessons. ENUMs provide a structured way to define and manage a set of constant values for specific attributes, in this case, the cost-related details for different types of lessons and skill levels.

It can also be noted that the addition of columns labeled "min number" and "max number" to the "lesson" table creates a conflicting environment, especially when it comes to "individual" lessons. Maintaining homogeneity across all lesson kinds is the fundamental rationale, despite the data appearing to be redundant. Thus both minimum and maximum numbers are set to one, guaranteeing that even individual lessons follow a consistent framework.

The creation of primary keys for strong entities is emphasized in the Method and Results section. The effective completion of the tasks listed in the project description is made possible by this strategic configuration, which makes it easier to link with weaker components. Additionally, this crucial configuration is essential to guaranteeing that no derived data for the students monthly payment.

Additionally, the student lesson table is used to calculate the total monthly fee per student. The student ID and lesson ID are the primary keys used in this table to track each student's attendance for a particular lesson. By referencing the lesson ID, the corresponding price ID can be retrieved, making it easier for students to retrieve the cost associated with each lesson they complete. As such, the choice of primary keys is made on purpose in order to guarantee clear understanding and accurate data retrieval.

Interconnecting tables through primary and foreign keys, the tables are equipped with foreign key constraints, including ON DELETE SET NULL, ON DELETE CASCADE, or ON DELETE DEFAULT attributes. A specific illustration involves linking the "Instructor" table to the "available schedule" by incorporating the "instructor id" primary key as a foreign key in the "available schedule." This is exemplified in for the primary key of "available schedule" is the "instructor id" foreign key, as each schedule is dependent on the instructor for whom it is intended. As a result, ON DELETE CASCADE constraints is set to the foreign key, guaranteeing that the deletion of an instructor thus removes the schedule that goes along with it.

Furthermore, another fundamental functionality of the database is its need to adhere to the specific guidelines, including the requirement for students under the age of 18 to have a designated point of contact and the limitation allowing each student to rent a maximum of two instruments for a maximum of 12 months. To ensure adherence to these regulations, a dedicated table called "school rules" has been created. This table systematically lists the descriptions of the rules along with their corresponding values, providing a structured means of upholding and monitoring compliance.

It should also be noted that connecting rules directly to tables like "student" would imply that all students, regardless of age, must adhere to these restrictions. Such an approach would deviate significantly from normalization principles, particularly for students aged 18 and above, who are not subject to these rules. Consequently,the utilization of triggers was contemplated, specifically 'CHECK' constraints, during the initial stages of database and table creation instead of establishing a direct link between this table and others. It's noteworthy that these triggers were not incorporated into this seminar due to uncertainty about their necessity.

An extra condition required that the database be flexible enough to adjust to changing lesson prices so that students could accurately retrieve costs based on enrollment periods. For instance, the system must retrieve the cost in effect at the time of enrollment, not just the current rate, if a student signs up for a particular lesson and the lesson's cost changes before the student finishes paying.

The 'pricing scheme' table now has a 'price date' attribute in order to fulfill this requirement. This feature allows the timing of each price's activation to be recorded. One benefit of implementing multiple price versions is that historical accuracy is maintained, which is important for guaranteeing that students are billed the appropriate amount for the duration of their enrollment. This approach contributes to financial transparency and upholds agreed-upon rates.

Nevertheless, there are disadvantages to this strategy. Keeping several versions up to date may cause the database to grow larger, which could have an impact on system performance and storage capacity. Efficient timestamp-based querying is required to retrieve specific price data, which may pose certain challenges. Furthermore, out-of-date records may accumulate in the absence of proper database maintenance, clogging the system and making data retrieval more difficult. Errors or confusion may arise from this circumstance.