Improved Mathnasium ® Database Management System

Lindsay Saxman

13 May 2022

1 Introduction

In 2002, Peter Markovitz and David Ullendorff founded Mathnasium®, a math-only learning center designed to make kids fall in love with math. Today, there are more than 1,100 Mathnasium locations all over the world making a positive impact on mathematics education.[1] I have worked as the assistant center director at a Mathnasium for three years, where I currently oversee more than 300 students. We tend to have issues with missing data, somewhat due to Mathnasium's online system (called Radius), but generally due to poor management. When inputting data, if a field is not required on Radius, the franchise owner sometimes does not bother entering the data, or parents choose to only fill out required fields on enrollment forms. While an improved DBMS would not affect this human behavior, it would greatly facilitate the identification of missing data.

My inspiration for the DBMS was our student birthday data, which I use to send out birthday cards at the start of each month. On Radius, there is only one method to access student birthdays without individually searching each student's records, which only allows the user to see student birthdays from one month at a time. "Birthday" isn't a required field on the enrollment forms, so there are many students with missing birthday data that will never receive a birthday card, because Radius will never output students with no birthday listed on the student birthday report. Currently, the only way for me to figure out which students are missing birthday data is to generate the report for each of the 12 months of the year, compile all of these names, then take the set difference between these names and the set of all current students.

This process wouldn't be difficult, but it would be time consuming and difficult to replicate (since we constantly have new enrollments, I would have to repeat this process about once a month). An improved DBMS that stores birthdays along with the rest of the student data would solve this issue.

One question I would like this database to address is *What data is missing?* I want all of our data to be accurate and complete, but I can't begin obtaining and filling in this data if I can't figure out exactly where I am missing data. A DBMS would help me quickly determine what student is missing what data, which Mathnasium's current system is not capable of.
Another question this database will address is *Which students have birthdays this coming month?* that I have to answer on a monthly basis to send out birthday cards.

The main objective of this project is to integrate all of the essential data for a Mathnasium learning center relating to students in a DBMS.

2 Data and Database Schema

Intended users for this database are Mathnasium employees, such as myself. Although it could be adapted for other businesses, this database is specific to and designed for Mathnasium. There should not be any users outside of the

Mathnasium network.

The proposed DBMS will integrate all of the essential data for Mathnasium relating to our students. The database will include seven main tables: STUDENT, ACCOUNT, GUARDIAN, SCHOOL, ATTENDANCE, ASSESSMENT, and CENTER. There are two additional tables called ASSESSMENT TRACKER and ATTENDANCE TRACKER which represent the relation between the STUDENT table and the ASSESSMENT and ATTENDANCE tables, respectively.

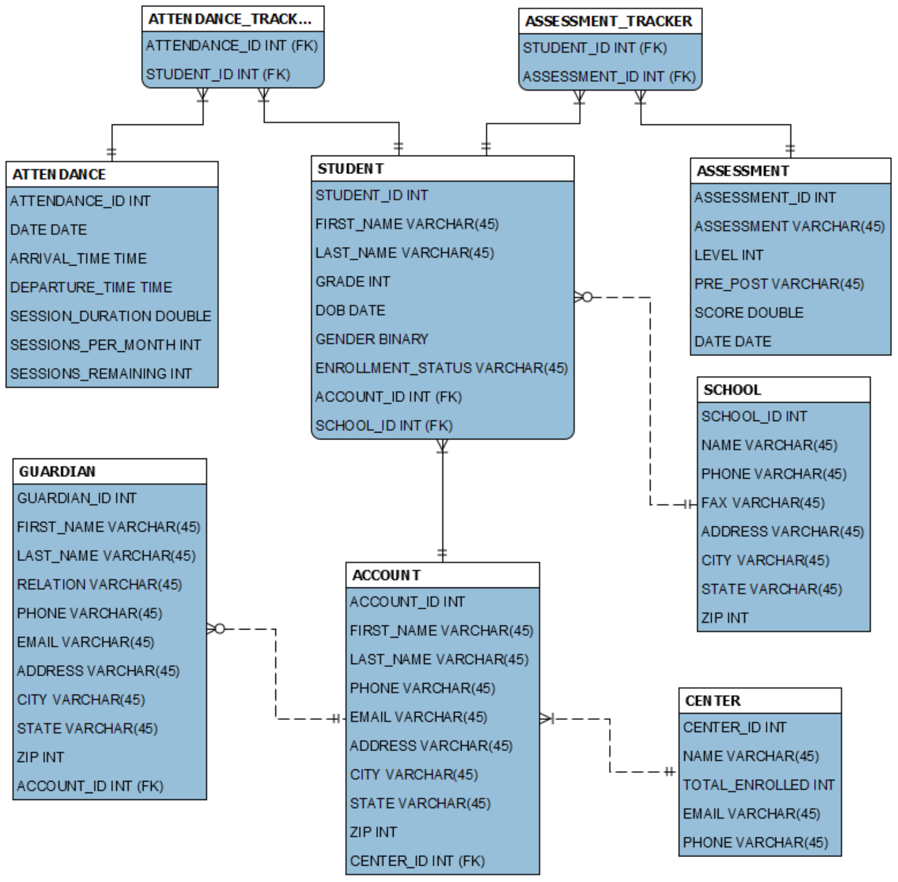

Figure 1: Mathnasium entity-relationship model

The primary entities in this DBMS are students, accounts, schools, guardians, Mathnasium centers, and assessments. Some of the domains include name, phone number, email address, mailing/street address, gender, date, and assessment score.

The STUDENT table has primary key STUDENT ID. STUDENT attributes include first and last name, grade, date of birth, gender, and enrollment status.

The ACCOUNT table has primary key ACCOUNT ID. ACCOUNT attributes include first and last name, phone number, email address, street address, city, state, and ZIP code. There is a mandatory, one-to-many relationship between the ACCOUNT and STUDENT tables, as more than one student can be enrolled under one account (i.e. parents with more than one child enrolled). Each record in the STUDENT table must reference an account in the ACCOUNT table (i.e. foreign key ACCOUNT ID cannot be null in the STUDENT table).

The GUARDIAN table has primary key GUARDIAN ID. GUARDIAN attributes include the same attributes as the ACCOUNT table, but with an additional "Relation" attribute, which indicates the relation of the guardian to the child (mother, father, brother, friend, aunt, etc.). There is a non-mandatory, one-to-many relationship between the ACCOUNT and GUARDIAN tables. For each account, there can be multiple guardians listed (or no guardians at all), but a guardian can only be linked to one account.

The SCHOOL table has primary key SCHOOL ID. SCHOOL attributes include school name, phone number, fax number, email address, and mailing address. There is a many-to-one relationship between STUDENT and SCHOOL. Many students share the same school, but a student can only be enrolled in one school.

The CENTER table has primary key CENTER ID. CENTER attributes include name, total enrollment count, phone number, and email address. There is a mandatory, one-to-many relationship between CENTER and ACCOUNT, as each Mathnasium center has many, many accounts, as long as each account in the ACCOUNT table does reference a center in the CENTER table (i.e. foreign key CENTER ID in the ACCOUNT table cannot be null).

The ATTENDANCE table has primary key ATTENDANCE ID. ATTENDANCE attributes include date, arrival and departure times, session duration, total number of monthly sessions (default is 10 sessions per month), and number of remaining monthly sessions. There is a many-to-one relationship between ATTENDANCE and STUDENT. Any record in the ATTENDANCE table references only one STUDENT ID, but one student will have many records in the ATTENDANCE table. The table ATTENDANCE TRACKER shows this mandatory relationship, in which the foreign keys ATTENDANCE ID and STUDENT ID cannot be null.

The ASSESSMENT table has primary key ASSESSMENT ID. ASSESSMENT attributes include assessment name, level, pre/post (was this a pre assessment or a post assessment?) score, and date taken. There is a many-to-one relationship between ASSESSMENT and STUDENT. Any record in the ASSESSMENT table can only be associated with one student (students take assessments individually), but one student can be associated with multiple records in the ASSESSMENT table. The table ASSESSMENT TRACKER shows this mandatory relationship, in which the foreign keys ASSESSMENT ID and STUDENT ID cannot be null.

3 Relevant Operations

The entity set of interest most often used in this DBMS is the set of all students currently enrolled. Using this DBMS, we can perform a query to obtain the total number of enrolled students.

SELECT COUNT(STUDENT_ID) FROM STUDENT WHERE ENROLLMENT_STATUS = 'ENROLLED';

As stated in the introduction, the main inspiration behind this DBMS was the student birthday data. In order to give out birthday cards, I need to find all currently enrolled students that have birthdays this coming month (June).

SELECT * FROM STUDENT WHERE MONTH(DOB) = 06;

Attendance is something we have to track very closely in order to properly staff the center, and to ensure students are not using more than their 10 monthly sessions. Around the middle of each month, the franchise owner likes to check out which students have not been attending very much (less than three times this month), plus which students have used almost all of their sessions.

SELECT * FROM ATTENDANCE WHERE (SESSIONS_REMAINING ¡ 3) OR (SESSIONS_REMAINING ¿ 7);

Suppose I'd like to reward the student with the best assessment score on our Computation Checkup, which many students tend to make careless errors on (meaning the scores are often lower than expected).

SELECT MAX(SCORE) FROM ASSESSMENT WHERE ASSESSMENT = 'Computation Checkup';

4 Significance of the Operations

Managing accounts and enrollments will be much easier for the franchise owner using the DBMS. He will be more efficient obtaining answers to questions such as:

What is the billing address associated with this account?

What is the total number of currently enrolled students? How many are currently enrolled at the Fairfax center, and how many are currently enrolled at the Herndon center?

If I haven't received an email response from this person, is there another email address associated with the account or its guardians?

As stated above, this DBMS would help the franchise owner manage and query account and enrollment data. Our enrollment contract is month-to-month, so the franchise owner must update this data on a monthly basis to ensure accounts that discontinue or disenroll do not get charged the following month. Thus, it is essential that this data is correct and current, which is difficult to achieve for the franchise owner.

Handling null/missing data is never fun, and I want to know what data is missing from my records. I need all of our data to be accurate and complete, but I can't begin obtaining and filling in this data if I can't figure out exactly where I am missing data. Using this DBMS will identify what data is missing without me having to go hunting for it.

In conclusion, the proposed Mathnasium DBMS can address countless questions about students, accounts, schools, enrollments, and more. Although there are still improvements that could be made, I am more than satisfied with this shiny, new DBMS I can use to save myself a few headaches at my job.

Notes

1. The Mathnasium Franchise Story. Mathnasium. Retrieved May 13, 2022, from https://mathnasiumfranchise.com/the-mathnasium-franchise-story/