Database Physical Data Model Specification

Created on 05/10/2024, 12:37:14

1 Introduction

- This document describes the physical data model of the database that supports the business requirements and the logical data model.
- The physical data model defines the tables, columns, data types, constraints, indexes, and relationships of the database.
- The physical data model also specifies the storage parameters, performance tuning, backup and recovery, and security aspects of the database.

2 Physical Data Model Diagram

- The physical data model diagram shows the graphical representation of the database structure and design.
- The diagram uses the following notation:
- Rectangles represent tables.
- Ellipses represent columns.
- Lines represent relationships between tables.
- Diamonds represent cardinality and optionality of the relationships.
- Crow's feet represent many-to-many relationships.
- The physical data model diagram is attached as an appendix to this document.

3 Physical Data Model Description

- This section provides a detailed description of each table, column, data type, constraint, index, and relationship in the physical data model.
- The tables are listed in alphabetical order.
- For each table, the following information is provided:
- Table name: The name of the table.
- Table description: A brief description of the purpose and function of the table.
- Table columns: A list of the columns in the table, along with their data types, lengths, precisions, scales, nullability, default values, and descriptions.
- Table constraints: A list of the constraints on the table, such as primary keys, foreign keys, unique keys, check constraints, and referential integrity constraints.
- Table indexes: A list of the indexes on the table, along with their names, types, columns, and descriptions.
- Table relationships: A list of the relationships between the table and other tables, along with their names, types, cardinality, optionality, and descriptions.

3.1 Students Table

This table stores the information about students enrolled in the school.

student_id: An integer that uniquely identifies a student.

first_name: A string that stores the first name of the student.

last_name: A string that stores the last name of the student.

email: A string that stores the email address of the student.

phone: A string that stores the phone number of the student.

major: A string that stores the name of the major that the student is pursuing.

minor: A string that stores the name of the minor that the student is pursuing, if any.

advisor: A foreign key that references the professor_id column of the Professors table, indicating the academic advisor of the student.

3.1.1 Table constraints:

The student_id column is the primary key of the table.

The email column is unique, meaning that no two students can have the same email address.

The phone column is unique, meaning that no two students can have the same phone number.

The advisor column is a foreign key that references the professor_id column of the Professors table, enforcing the referential integrity constraint that the advisor of a student must be a valid professor.

3.1.2 Table indexes:

An index on the student_id column to speed up the queries that search by student_id.

An index on the last_name column to speed up the queries that search by last name.

An index on the major column to speed up the queries that search by major.

An index on the minor column to speed up the queries that search by minor.

An index on the advisor column to speed up the queries that search by advisor.

3.1.3 Table relationships:

The Students table has a one-to-many relationship with the Classes table, meaning that a student can enroll in many classes, but a class can only have one student.

The Students table has a one-to-one relationship with the Grades table, meaning that a student can have only one grade for each class.

The Students table has a many-to-many relationship with the Exams table, meaning that a student can take many exams, and an exam can be taken by many students.

The Students table has a one-to-many relationship with the Professors table, meaning that a student can have only one advisor, but a professor can advise many students.

3.2 Classes Table

This table stores the information about the classes offered by the school.

class_id: An integer that uniquely identifies a class.

course_id: A foreign key that references the course_id column of the Courses table, indicating the course that the class belongs to.

professor_id: A foreign key that references the professor_id column of the Professors table, indicating the instructor of the class.

semester: A string that stores the semester when the class is offered, such as "Fall 2020".

time: A string that stores the time when the class is held, such as "Monday 10:00-11:50".

location: A string that stores the location where the class is held, such as "Room 101".

capacity: An integer that stores the maximum number of students that can enroll in the class.

enrollment: An integer that stores the current number of students enrolled in the class.

3.2.1 Table constraints:

The class_id column is the primary key of the table.

The course_id column is a foreign key that references the course_id column of the Courses table, enforcing the referential integrity constraint that the course of a class must be a valid course.

The professor_id column is a foreign key that references the professor_id column of the Professors table, enforcing the referential integrity constraint that the instructor of a class must be a valid professor.

The enrollment column must be less than or equal to the capacity column, enforcing the check constraint that the class cannot be over-enrolled.

3.2.2 Table indexes:

An index on the class_id column to speed up the queries that search by class_id.

An index on the course_id column to speed up the queries that search by course_id.

An index on the professor_id column to speed up the queries that search by professor_id.

An index on the semester column to speed up the queries that search by semester.

An index on the time column to speed up the queries that search by time.

An index on the location column to speed up the queries that search by location.

An index on the enrollment column to speed up the queries that search by enrollment.

3.2.3 Table relationships:

The Classes table has a many-to-one relationship with the Courses table, meaning that a class can belong to only one course, but a course can have many classes.

The Classes table has a many-to-one relationship with the Professors table, meaning that a class can have only one instructor, but a professor can teach many classes.

The Classes table has a one-to-many relationship with the Students table, meaning that a class can have many students, but a student can enroll in only one class.

The Classes table has a one-to-many relationship with the Exams table, meaning that a class can have many exams, but an exam can belong to only one class.

3.3 Grades Table

This table stores the information about the grades of the students for each class.

grade_id: An integer that uniquely identifies a grade record.

student_id: A foreign key that references the student_id column of the Students table, indicating the student who received the grade.

class_id: A foreign key that references the class_id column of the Classes table, indicating the class for which the grade was given.

grade: A string that stores the letter grade of the student, such as "A" or "B+".

3.3.1 Table constraints:

The grade_id column is the primary key of the table.

The student_id column is a foreign key that references the student_id column of the Students table, enforcing the referential integrity constraint that the student of a grade must be a valid student.

The class_id column is a foreign key that references the class_id column of the Classes table, enforcing the referential integrity constraint that the class of a grade must be a valid class.

The grade column must be one of the following values: "A", "A-", "B+", "B", "B-", "C+", "C", "C-", "D+", "D", "D-", or "F", enforcing the check constraint that the grade must be a valid letter grade.

The combination of student_id and class_id columns is unique, meaning that no two grades can have the same student and class.

3.3.2 Table indexes:

An index on the grade_id column to speed up the queries that search by grade_id.

An index on the student_id column to speed up the queries that search by student_id.

An index on the class_id column to speed up the queries that search by class_id.

An index on the grade column to speed up the queries that search by grade.

3.3.3 Table relationships:

The Grades table has a one-to-one relationship with the Students table, meaning that a grade can belong to only one student, and a student can have only one grade for each class.

The Grades table has a one-to-one relationship with the Classes table, meaning that a grade can belong to only one class, and a class can have only one grade for each student.

3.4 Exams Table

This table stores the information about the exams given in each class.

exam_id: An integer that uniquely identifies an exam.

class_id: A foreign key that references the class_id column of the Classes table, indicating the class that the exam belongs to.

name: A string that stores the name of the exam, such as "Midterm" or "Final".

date: A date that stores the date when the exam is given.

weight: A decimal that stores the percentage of the final grade that the exam contributes to, such as 0.3 or 0.5.

3.4.1 Table constraints:

The exam_id column is the primary key of the table.

The class_id column is a foreign key that references the class_id column of the Classes table, enforcing the referential integrity constraint that the class of an exam must be a valid class.

The weight column must be between 0 and 1, enforcing the check constraint that the weight must be a valid percentage.

The combination of class_id and name columns is unique, meaning that no two exams can have the same name and class.

3.4.2 Table indexes:

An index on the exam_id column to speed up the queries that search by exam_id.

An index on the class_id column to speed up the queries that search by class_id.

An index on the name column to speed up the queries that search by name.

An index on the date column to speed up the queries that search by date.

An index on the weight column to speed up the queries that search by weight.

3.4.3 Table relationships:

The Exams table has a many-to-one relationship with the Classes table, meaning that an exam can belong to only one class, but a class can have many exams.

The Exams table has a many-to-many relationship with the Students table, meaning that an exam can be taken by many students, and a student can take many exams.

3.5 Professors Table

This table stores the information about the professors who work at the school.

professor_id: An integer that uniquely identifies a professor.

first_name: A string that stores the first name of the professor.

last_name: A string that stores the last name of the professor.

email: A string that stores the email address of the professor.

phone: A string that stores the phone number of the professor.

department: A string that stores the name of the department that the professor belongs to.

title: A string that stores the title of the professor, such as "Assistant Professor" or "Professor".

salary: A decimal that stores the annual salary of the professor.

3.5.1 Table constraints:

The professor_id column is the primary key of the table.

The email column is unique, meaning that no two professors can have the same email address.

The phone column is unique, meaning that no two professors can have the same phone number.

The salary column must be greater than zero, enforcing the check constraint that the salary must be a positive value.

3.5.2 Table indexes:

An index on the professor_id column to speed up the queries that search by professor_id.

An index on the last_name column to speed up the queries that search by last name.

An index on the department column to speed up the queries that search by department.

An index on the title column to speed up the queries that search by title.

An index on the salary column to speed up the queries that search by salary.

3.5.3 Table relationships:

The Professors table has a one-to-many relationship with the Classes table, meaning that a professor can teach many classes, but a class can have only one instructor.

The Professors table has a one-to-many relationship with the Students table, meaning that a professor can advise many students, but a student can have only one advisor.

3.6 Courses Table

This table stores the information about the courses offered by the school.

course_id: An integer that uniquely identifies a course.

name: A string that stores the name of the course, such as "Database Systems" or "Artificial Intelligence".

description: A string that stores the description of the course, such as "An introduction to the design and implementation of database systems" or "A survey of the methods and applications of artificial intelligence".

credits: An integer that stores the number of credits that the course is worth, such as 3 or 4.

curriculum: A string that stores the name of the curriculum that the course belongs to, such as "Computer Science" or "Mathematics".

prerequisites: A string that stores the list of courses that must be taken before taking the course, separated by commas, such as "CS101, CS102" or "MATH201, MATH202".

3.6.1 Table constraints:

The course_id column is the primary key of the table.

The name column is unique, meaning that no two courses can have the same name.

The credits column must be between 1 and 6, enforcing the check constraint that the credits must be a valid value.

3.6.2 Table indexes:

An index on the course_id column to speed up the queries that search by course_id.

An index on the name column to speed up the queries that search by name.

An index on the credits column to speed up the queries that search by credits.

An index on the curriculum column to speed up the queries that search by curriculum.

An index on the prerequisites column to speed up the queries that search by prerequisites.

3.6.3 Table relationships:

The Courses table has a one-to-many relationship with the Classes table, meaning that a course can have many classes, but a class can belong to only one course.