Importance of Having a Schema in Datasets

(Primary Key and Foreign Key)

Project Assignment - 02

Aurabrato Ghosh

Azure Data Engineering

Table of Contents

Introduction

Overview of Structured Data

Structured data refers to data that is organized in a predefined schema, typically stored in relational databases using tables with columns and data types. Each row in a table represents a record, and each column represents a specific attribute of that record.

What Is a Schema?

A schema is the blueprint or structure of a database. It defines:

- Table structures
- Data types
- Constraints like primary keys (PK), foreign keys (FK), and unique constraints
- Relationships between different entities

In SQL, the schema ensures that the data inserted, queried, and maintained follows strict rules — reducing redundancy, preventing data corruption, and enabling reliable joins between tables.

Importance of Schema

- **Data Integrity:** Primary and foreign keys enforce valid relationships, ensuring that records are not duplicated or mis-linked.
- **Consistency & Accuracy:** Schemas enforce consistent data entry (e.g., you can't enter a string into a column defined as INT).
- **Referential Integrity:** A student can't be enrolled in a course unless their Student_ID exists assured through foreign keys.
- **Query Optimization:** Indexes on primary keys enhance the performance of searches, sorts, and joins.
- **Prevention of Anomalies:** Schema constraints prevent insertion, update, or deletion anomalies that could break business logic.

Dataset Analysis — Schema-Based Implementation

Dataset Overview

For this project, a structured dataset was created using five relational tables. All tables use appropriate data types and constraints including:

- **PRIMARY KEY** – to ensure uniqueness of rows
- **FOREIGN KEY** – to enforce referential integrity between entities

Table Descriptions

Project2_Students - Stores student information

Column Name	Data Type	Constraint
Student_ID	INT	PRIMARY KEY
Student_Name	VARCHAR(100)	NOT NULL
City	VARCHAR(100)	NULLABLE

Project2_Courses - Stores course details

Column Name	Data Type	Constraint
Course_ID	INT	PRIMARY KEY
Course_Name	VARCHAR(100)	NOT NULL

Project2_Enrollments - Represents which student enrolled in which course

Column Name	Data Type	Constraint
Enrollment_ID	INT	PRIMARY KEY
Student_ID	INT	FOREIGN KEY → Students
Course_ID	INT	FOREIGN KEY → Courses
Enrollment_Date	DATE	NOT NULL

Project2_Payments - Tracks payments made by students

Column Name	Data Type	Constraint
Payment_ID	INT	PRIMARY KEY
Student_ID	INT	FOREIGN KEY → Students
Amount	DECIMAL(10,2)	NOT NULL
Payment_Date	DATE	NOT NULL

Project2_Tickets - Logs support tickets raised by students

Column Name	Data Type	Constraint
Ticket_ID	INT	PRIMARY KEY
Student_ID	INT	FOREIGN KEY → Students
Issue_Desc	VARCHAR(255)	NULLABLE
Created_Date	DATE	DEFAULT GETDATE()

```sql
-- Students Table
CREATE TABLE Project2_Students (
    Student_ID INT PRIMARY KEY,
    FullName VARCHAR(100),
    City VARCHAR(100)
);

-- Courses Table
CREATE TABLE Project2_Courses (
    Course_ID INT PRIMARY KEY,
    CourseName VARCHAR(100)
);

-- Enrollments Table
CREATE TABLE Project2_Enrollments (
    Enrollment_ID INT PRIMARY KEY,
    Student_ID INT FOREIGN KEY REFERENCES Project2_Students(Student_ID),
    Course_ID INT FOREIGN KEY REFERENCES Project2_Courses(Course_ID),
    EnrollmentDate DATE
);

-- Payments Table
CREATE TABLE Project2_Payments (
    Payment_ID INT PRIMARY KEY,
    Student_ID INT FOREIGN KEY REFERENCES Project2_Students(Student_ID),
    Amount DECIMAL(10,2),
    PaymentDate DATE
);

-- Support Tickets Table
CREATE TABLE Project2_Tickets (
    Ticket_ID INT PRIMARY KEY,
    Student_ID INT FOREIGN KEY REFERENCES Project2_Students(Student_ID),
    Issue VARCHAR(255),
    CreatedAt DATETIME
);
```

Image 1: Table creation SQL queries

```sql
-- Insert Customers
INSERT INTO DEMO_NCPL VALUES
(1, 'Alice', 'Toronto'),
(2, 'Bob', 'Montreal'),
(3, 'Charlie', 'Calgary'),
(4, 'Diana', 'Vancouver');

-- Students
INSERT INTO Project2_Students VALUES
(1, 'Arjun Sharma', 'Delhi'),
(2, 'Meera Kaur', 'Chandigarh'),
(3, 'Ravi Nair', 'Mumbai');

-- Courses
INSERT INTO Project2_Courses VALUES
(101, 'Azure Data Fundamentals'),
(102, 'SQL for Data Analysis');

-- Enrollments
INSERT INTO Project2_Enrollments VALUES
(1001, 1, 101, '2024-05-01'),
(1002, 2, 102, '2024-06-15');

-- Payments
INSERT INTO Project2_Payments VALUES
(5001, 1, 299.00, '2024-05-05'),
(5002, 2, 399.00, '2024-06-20');

-- Tickets
INSERT INTO Project2_Tickets VALUES
(7001, 1, 'Login issue', GETDATE()),
(7002, 2, 'Course access problem', GETDATE());
```

Image 2: Data Insertion SQL queries

Entity Relationship (ER) Overview

- One student → many enrollments, payments, and tickets.
- One course → many enrollments.
- Enrollments → junction table between students and courses.

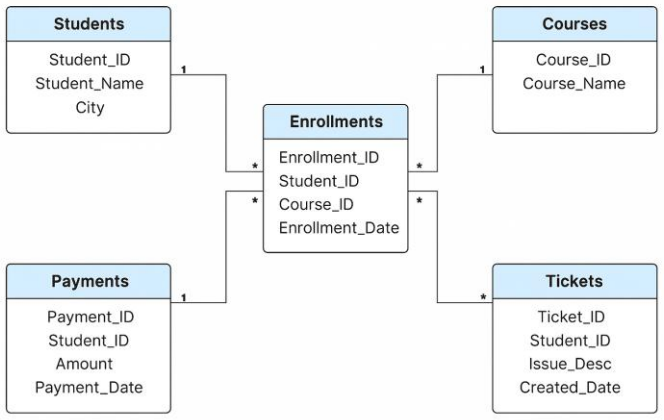

Image 3: ER relationship Diagram

Schema Integrity Before Alteration

Initially, the dataset was strictly regulated by:

- Primary keys that prevented duplicate entries.
- Foreign keys that ensured each related record had a valid parent (e.g., no orphan enrollments).

Result: Only valid, clean, and logically connected data could be inserted.

```
-- Duplicate primary key
INSERT INTO Project2_Students VALUES (1, 'Duplicate Arjun', 'Bangalore');

-- Insert NULL into primary key field
INSERT INTO Project2_Tickets VALUES (NULL, 1, 'NULL Ticket ID Test', GETDATE());

-- Insert enrollment with non-existent student
INSERT INTO Project2_Enrollments VALUES (1003, 999, 101, '2024-07-01');

-- Insert payment with non-existent student
INSERT INTO Project2_Payments VALUES (5003, 888, 189.00, '2024-07-25');

-- Insert Tickets for a non-existent student
INSERT INTO Project2_Tickets VALUES (7003, 999, 'Fake Student ID', GETDATE());
```

Image 4: Bad data insertion attempts

```
9:12:55 PM    Started executing query at Line 89
              Msg 2627, Level 14, State 1, Line 1
              Violation of PRIMARY KEY constraint 'PK__Project2__A2F4E9ACF2771454'. Cannot insert duplicate key in object 'dbo.Project2_Students'. The duplicate key v
              alue is (1).
              The statement has been terminated.
              Total execution time: 00:00:00.098
```

Image 5: Duplicate Primary Key violation

```
9:13:44 PM    Started executing query at Line 92
              Msg 515, Level 16, State 2, Line 1
              Cannot insert the value NULL into column 'Ticket_ID', table 'AG_SQL_DB.dbo.Project2_Tickets'; column does not allow nulls. INSERT fails.
              The statement has been terminated.
              Total execution time: 00:00:00.068
```

Image 6: Insertion of NULL values in Primary Key violation

```
9:15:49 PM    Started executing query at Line 95
              Msg 547, Level 16, State 0, Line 1
              The INSERT statement conflicted with the FOREIGN KEY constraint "FK__Project2___Stude__59C55456". The conflict occurred in database "AG_SQL_DB", table
              "dbo.Project2_Students", column 'Student_ID'.
              The statement has been terminated.
              Total execution time: 00:00:00.092
```

Image 7: Insertion of data into non-existent Primary Key violation

```
9:16:20 PM    Started executing query at Line 98
              Msg 547, Level 16, State 0, Line 1
              The INSERT statement conflicted with the FOREIGN KEY constraint "FK__Project2___Stude__5D95E53A". The conflict occurred in database "AG_SQL_DB", table
              "dbo.Project2_Students", column 'Student_ID'.
              The statement has been terminated.
              Total execution time: 00:00:00.083
```

Image 8: Insertion of data into non-existent Foreign Key violation

Schema-less Behavior and Data Inconsistencies

After designing and validating the structured database with schema enforcement, we deliberately removed:

- All primary key constraints
- All foreign key constraints

Once constraints were removed, all the invalid records were allowed.

```
-- Drop Foreign Keys
ALTER TABLE Project2_Enrollments DROP CONSTRAINT FK__Project2___Stude__59C55456;
ALTER TABLE Project2_Enrollments DROP CONSTRAINT FK__Project2___Cours__5AB9788F;

ALTER TABLE Project2_Payments DROP CONSTRAINT FK__Project2___Stude__5D95E53A;

ALTER TABLE Project2_Tickets DROP CONSTRAINT FK__Project2___Stude__607251E5;

-- Drop Primary Keys
ALTER TABLE Project2_Students DROP CONSTRAINT PK__Project2__A2F4E9ACF2771454;
ALTER TABLE Project2_Courses DROP CONSTRAINT PK__Project2__37E005FBA5124E6D;
ALTER TABLE Project2_Enrollments DROP CONSTRAINT PK__Project2__4365BD6AAA78E87A;
ALTER TABLE Project2_Payments DROP CONSTRAINT PK__Project2__DA6C7FE1F5983115;
ALTER TABLE Project2_Tickets DROP CONSTRAINT PK__Project2__ED7260D9EAE7BCA5;
```

Image 9: Removal of Key constraints SQL queries

Observed Issues After Dropping Schema Constraints

- **Duplicate Primary Keys Allowed:** The database accepted multiple rows with the same Student_ID (e.g., duplicate records for ID = 1), normally blocked by a PRIMARY KEY constraint.
- **NULL Values Inserted in Primary Key Columns:** Records with NULL in Ticket_ID were inserted, this breaks the uniqueness and identity of records.
- **Invalid Foreign Key References Accepted:** Enrollments and payments were created for Student_ID values that do not exist in the Students table (e.g., 888, 999); In a schema-enforced system, these would be rejected.
- **Orphan Records in Child Tables:** Orphan rows exist in Project2_Enrollments, Project2_Payments, and Project2_Tickets with no matching parent record in Project2_Students.
- **JOIN Operations Became Unreliable:** Queries involving joins between child and parent tables return NULL values, causing data loss in reporting or analysis.
- **Lack of Error Detection at Insert Time:** SQL accepted bad data silently — without any warning or rejection — since there were no constraints to validate it.
- **Compromised Data Integrity:** The overall trustworthiness of the dataset decreased due to potential for duplicates, unlinked data, and incomplete records.
- **Manual Data Validation Becomes Necessary:** Without schema-level checks, all validation must be handled manually through queries and scripts, increasing complexity.

```
SELECT *
FROM Project2_Enrollments e
LEFT JOIN Project2_Students s ON e.Student_ID = s.Student_ID
WHERE s.Student_ID IS NULL;

SELECT Student_ID, COUNT(*)
FROM Project2_Students
GROUP BY Student_ID
HAVING COUNT(*) > 1;
```

Image 10: Validation SQL queries

Comparative Analysis

Advantages of Having a Schema

- Enforces data integrity through primary and foreign key constraints.
- Prevents duplicate, null, or invalid entries in key columns.
- Enables accurate and reliable JOIN operations across tables.
- Improves query performance through indexing on key fields.
- Ensures referential integrity between related entities.
- Reduces chances of logical anomalies in insert, update, or delete operations.
- Facilitates easier debugging and auditing of data issues.
- Makes databases more predictable and maintainable.
- Promotes adherence to business rules and relationships.
- Enhance data trustworthiness and compliance in analytics.

Disadvantages of Having a Schema

- Schema enforcement may reject data that doesn't conform, requiring pre-validation.
- Inflexibility can slow down development when frequent schema changes are needed.
- Schema migrations or updates can be complex in large systems.
- Overhead of defining and maintaining constraints across all tables.
- Increased initial setup time for designing relationships and keys.

Advantages of Not Having a Schema

- More flexibility to insert any kind of data quickly.
- Suitable for experimentation or handling semi-structured/unstructured sources.
- Faster to set up initially, especially in prototyping stages.
- Schema-less systems can adapt quickly to changes in data structure.

Disadvantages of Not Having a Schema

- High risk of data inconsistencies and duplication.
- Manual validation required to ensure relational integrity.
- JOINs may fail or produce nulls due to missing relationships.
- Debugging and tracing errors become more difficult.
- Increases the likelihood of orphaned or corrupt records.
- Weak data governance leads to unreliable reporting.
- Difficult to scale for enterprise-level applications without structure.

References

- Microsoft Learn – SQL Constraints: https://learn.microsoft.com/en-us/sql/relational-databases/tables/table-constraints
- W3Schools – SQL PRIMARY KEY Constraint: https://www.w3schools.com/sql/sql_primarykey.asp
- W3Schools – SQL FOREIGN KEY Constraint: https://www.w3schools.com/sql/sql_foreignkey.asp
- GeeksforGeeks – Advantages and Disadvantages of DBMS Schema: https://www.geeksforgeeks.org/schema-in-dbms/
- IBM Documentation – Schema Design Best Practices: https://www.ibm.com/docs/en/cloud-paks/cp-data/4.0?topic=schema-schema-design-best-practices
- Stack Overflow – Common Schema Design Mistakes: https://stackoverflow.blog/2020/11/23/a-primer-on-database-schema-design/