Contents

Introduction

Research Report: Database Design for Training Management Systems

Training Management Systems (TMS) have become essential tools for organizations seeking to efficiently manage their workforce development programs. These systems serve as centralized platforms for coordinating training activities, tracking employee progress, and optimizing resource allocation across multiple training initiatives.

According to research by Vertabelo (2024), effective database design for learning management systems must efficiently handle complex relationships between courses, students, instructors, schedules, and enrollment data. The design must support various analytics capabilities including attendance tracking, performance monitoring, and resource utilization analysis. The normalized database structure ensures data integrity while supporting scalable operations as organizations grow their training programs.

A study on enterprise training management systems published in Applied Mechanics and Materials (Scientific.Net, 2012) emphasizes the importance of proper entity relationship modeling in training databases. The research highlights that successful training management systems require careful consideration of temporal relationships, particularly in scheduling and enrollment tracking, where time-based constraints and dependencies must be maintained across multiple entities.

Industry case studies, such as the SumTotal Systems optimization project documented by Percona (2023), demonstrate that training management databases often handle high-volume concurrent operations. The case study reveals that proper indexing strategies and relationship design are crucial for maintaining performance when managing thousands of simultaneous user sessions across multiple training programs.

Key Design Principles:

- Maintaining third normal form to eliminate data redundancy
- Implementing proper temporal data handling for scheduling conflicts
- Designing flexible enrollment systems for various training formats
- Ensuring audit trails for compliance and reporting requirements

Assumptions and Constraints:

- Each trainer can be assigned to multiple schedules, but each schedule has only one trainer
- Trainees can enroll in multiple courses, and courses can have multiple trainees
- Email addresses are unique across the system for both trainees and trainers
- Each course can have multiple scheduled sessions with different trainers
- Cascade deletions are implemented where logically appropriate

The Importance of Data Modeling and SQL in Data Science

Key Insights from Research

1. Why Structured Data is Important in Data Science Pipelines

Structured data (organized in tables with defined schemas) is crucial because it ensures consistency, reduces errors, and enables efficient querying. Unstructured or poorly structured data can lead to inaccurate analyses and require extensive cleaning. According to IBM, structured data is essential for reliable analytics and machine learning (IBM, "The Importance of Data Structures in Data Science").

2. The Role of Data Modeling in Preparing Data for Analysis or Machine Learning

Data modeling defines how data is stored, related, and retrieved. A well-designed data model ensures:

- Efficient querying (optimized for performance)
- Data integrity (avoiding duplication and inconsistencies)
- Scalability (handling large datasets without performance loss)

For example, Netflix uses data modeling to organize user viewing habits, enabling personalized recommendations (Netflix Tech Blog, "How Netflix Uses Data Modeling").

3. How Relational Databases Support Scalable and Clean Data Practices

Relational databases (e.g., PostgreSQL, MySQL) enforce structured storage with:

- ACID compliance (ensuring transactional reliability)
- Normalization (reducing redundancy)
- Indexing (speeding up queries)

Uber relies on relational databases to manage ride data efficiently, ensuring real-time analytics (Uber Engineering, "Scalable Data Warehousing at Uber").

4. Why SQL is Still Foundational Despite Python and Pandas

SQL remains essential because:

- It's optimized for data retrieval (faster than Pandas for large datasets)
- It's the standard for database interactions (most companies store data in SQL databases)
- It integrates with Python (libraries like SQLAlchemy allow direct queries)

Google BigQuery and Snowflake rely on SQL for large-scale analytics, proving its continued relevance (Google Cloud Blog, "Why SQL Still Matters in Big Data").

5. Example of SQL for Insights Before Machine Learning

Before training a churn prediction model, a data scientist might use SQL to:

```
SELECT customer_id, COUNT(*) AS transactions, AVG(purchase_amount) AS avg_spend
FROM sales
GROUP BY customer_id
HAVING COUNT(*) < 3;
```

This identifies low-engagement customers, helping prioritize ML efforts. Spotify uses similar SQL queries to analyze user behavior before applying ML (Spotify Engineering, "Data Pipelines at Spotify").

Reflection on Course Connections

This research reinforces why our course emphasizes SQL and data modeling. Understanding these concepts ensures we can:

- Extract data efficiently (SQL)
- Design databases that avoid errors (data modeling)
- Work with real-world systems (relational databases)

Requirements Analysis for a Course Management System

1. Identifying User Needs

Trainee Requirements

- **View** Browse course catalog with details
- **Track** See enrolled and completed courses
- **Access** Check schedule and trainer details

Trainer Requirements

- **View** See assigned courses
- **Check** Access scheduled sessions
- **Track** View enrolled trainees list

Admin Requirements

- **Manage** Add and update courses
- **Schedule** Set up class timings
- **Enroll** Register trainees into courses
- **Monitor** Track enrollments & conflicts

2. Entity-Relationship Modeling

Key Entities & Relationships

Entity	Description	Key Attributes
Trainee	Represents a learner	trainee_id, name, email
Trainer	Represents an instructor	trainer_id, name, expertise
Course	Training programs offered	course_id, title, description, duration
Enrollment	Junction table linking Trainees and Courses	enrollment_id, trainee_id, course_id, enrollment_date, status
Schedule	Defines session timings	schedule_id, course_id, trainer_id, date, start_time, end_time

Relationship Diagram

Trainee → Course (Many-to-Many via Enrollment)

- A trainee can enroll in multiple courses
- A course can have multiple trainees

Trainer → Course (One-to-Many)

- A trainer can teach multiple courses
- A course is assigned to one trainer

Schedule → Course (One-to-Many)

- A course can have multiple scheduled sessions

3. System Workflow

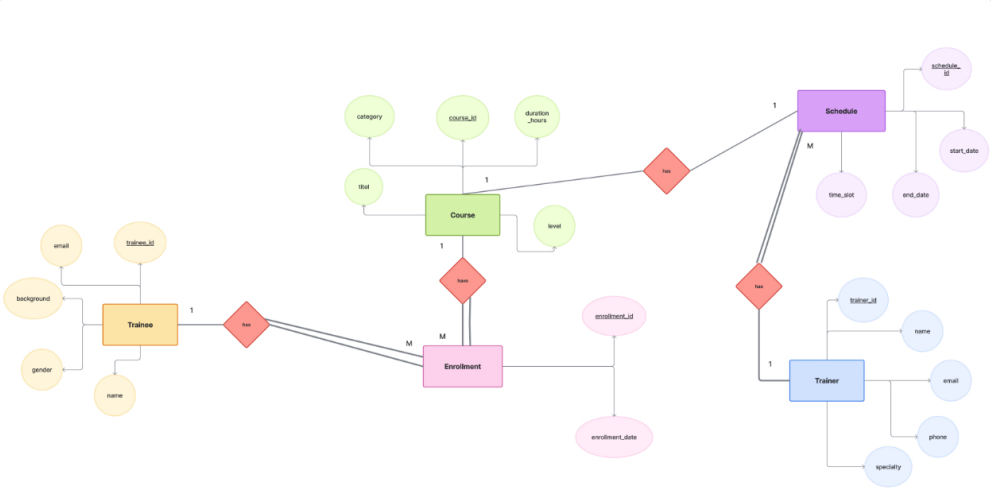
1. Admin creates a course → Assigns a trainer → Sets schedule
2. Trainee enrolls → Enrollment record is created
3. Trainer views assigned courses → Checks trainee list and schedule
4. Trainee accesses course details → Views sessions and progress

Entity Relationship Diagram

ERD Notation Guide

- Rectangle: Entity Type
- Double Rectangle: Weak Entity
- Diamond: Relationship Type
- Double Diamond: Identifying Relationship
- Oval: Attribute
- Underlined Oval: Key Attribute
- Double Oval: Multivalued Attribute
- Dashed Oval: Derived Attribute

Training Institute Database ERD



Relational Schema Mapping

The relational schema mapping shows how the conceptual ER model is translated into actual database tables with their attributes, data types, and relationships.

Schema Components:

Primary Tables:

- **Trainee** - Stores learner information
- **Trainer** - Stores instructor details
- **Course** - Contains course catalog

Junction & Schedule Tables:

- **Enrollment** - Links trainees to courses
- **Schedule** - Manages session timings

SQL Data Definition Language (DDL)

```
CREATE DATABASE CourseManagementSystem;
USE CourseManagementSystem;
CREATE TABLE Trainee (
    trainee_id INT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    gender CHAR(1) CHECK (gender IN ('M', 'F')),
    email VARCHAR(100) UNIQUE NOT NULL,
    background VARCHAR(200)
);

CREATE TABLE Trainer (
    trainer_id INT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    specialty VARCHAR(100),
    phone VARCHAR(20),
    email VARCHAR(100) UNIQUE NOT NULL
);

CREATE TABLE Course (
    course_id INT PRIMARY KEY,
    title VARCHAR(100) NOT NULL,
    category VARCHAR(50),
    duration_hours INT CHECK (duration_hours > 0)
);

CREATE TABLE Schedule (
    schedule_id INT PRIMARY KEY,
    course_id INT,
    trainer_id INT,
    start_date DATE NOT NULL,
    end_date DATE NOT NULL,
    time_slot VARCHAR(20) CHECK (time_slot IN ('Morning', 'Evening')),

    FOREIGN KEY (course_id) REFERENCES Course(course_id),
    FOREIGN KEY (trainer_id) REFERENCES Trainer(trainer_id),
    CONSTRAINT valid_dates CHECK (end_date >= start_date)
);
```

```sql
CREATE TABLE Enrollment (
    enrollment_id INT PRIMARY KEY,
    trainee_id INT,
    course_id INT,
    enrollment_date DATE DEFAULT (CURRENT_DATE),

    FOREIGN KEY (trainee_id) REFERENCES Trainee(trainee_id),
    FOREIGN KEY (course_id) REFERENCES Course(course_id),
    CONSTRAINT unique_enrollment UNIQUE (trainee_id, course_id)
);

-- Insert into Trainee Table
INSERT INTO Trainee (trainee_id, name, gender, email, background) VALUES
(1, 'Aisha Al-Harthy', 'F', 'aisha@example.com', 'Engineering'),
(2, 'Sultan Al-Farsi', 'M', 'sultan@example.com', 'Business'),
(3, 'Mariam Al-Saadi', 'F', 'mariam@example.com', 'Marketing'),
(4, 'Omar Al-Balushi', 'M', 'omar@example.com', 'Computer Science'),
(5, 'Fatma Al-Hinai', 'F', 'fatma@example.com', 'Data Science');

-- Insert into Trainer Table
INSERT INTO Trainer (trainer_id, name, specialty, phone, email) VALUES
(1, 'Khalid Al-Maawali', 'Databases', '96891234567', 'khalid@example.com'),
(2, 'Noura Al-Kindi', 'Web Development', '96892345678', 'noura@example.com'),
(3, 'Salim Al-Harthy', 'Data Science', '96893456789', 'salim@example.com');

-- Insert into Course Table
INSERT INTO Course (course_id, title, category, duration_hours) VALUES
(1, 'Database Fundamentals', 'Databases', 20),
(2, 'Web Development Basics', 'Web', 30),
(3, 'Data Science Introduction', 'Data Science', 25),
(4, 'Advanced SQL Queries', 'Databases', 15);

-- Insert into Schedule Table
INSERT INTO Schedule (schedule_id, course_id, trainer_id, start_date, end_date, time_slot) VALUES
(1, 1, 1, '2025-07-01', '2025-07-10', 'Morning'),
(2, 2, 2, '2025-07-05', '2025-07-20', 'Evening'),
(3, 3, 3, '2025-07-10', '2025-07-25', 'Evening'),
(4, 4, 1, '2025-07-15', '2025-07-22', 'Morning');

-- Insert into Enrollment Table
INSERT INTO Enrollment (enrollment_id, trainee_id, course_id, enrollment_date) VALUES
(1, 1, 1, '2025-06-01'),
(2, 2, 1, '2025-06-02'),
(3, 3, 2, '2025-06-03'),
(4, 4, 3, '2025-06-04'),
(5, 5, 3, '2025-06-05'),
(6, 1, 4, '2025-06-06');

-- Check all trainees
SELECT * FROM Trainee;
-- Check all trainers
SELECT * FROM Trainer;
-- Check all courses
SELECT * FROM Course;
-- Check all schedules
SELECT * FROM Schedule;
-- Check all enrollments
SELECT * FROM Enrollment;

--  Add the level column to Course table with default 'Beginner' value
ALTER TABLE Course
ADD COLUMN level VARCHAR(20) CHECK (level IN ('Beginner', 'Intermediate', 'Advanced')) DEFAULT 'Beginner';

-- Update existing courses with their correct levels
UPDATE Course SET level = 'Beginner' WHERE course_id IN (1, 2);
UPDATE Course SET level = 'Intermediate' WHERE course_id = 3;
UPDATE Course SET level = 'Advanced' WHERE course_id = 4;

--  Modify the Schedule table to allow 'Weekend' time_slot

ALTER TABLE Schedule
MODIFY COLUMN time_slot VARCHAR(20);

-- add the new CHECK constraint
ALTER TABLE Schedule
ADD CONSTRAINT chk_time_slot
CHECK (time_slot IN ('Morning', 'Evening', 'Weekend'));

-- Update the schedule for course_id 3 to be 'Weekend' instead of 'Evening'
UPDATE Schedule SET time_slot = 'Weekend' WHERE schedule_id = 3;

-- Verify the changes
SELECT * FROM Course;
SELECT * FROM Schedule;
```

Sample Data Population

Trainee Table

ID	Name	Gender	Email	Background
1	Aisha Al-Harthy	F	aisha@example.com	Engineering
2	Sultan Al-Farsi	M	sultan@example.com	Business
3	Mariam Al-Saadi	F	mariam@example.com	Marketing
4	Omar Al-Balushi	M	omar@example.com	Computer Science
5	Fatma Al-Hinai	F	fatma@example.com	Data Science

Trainer Table

ID	Name	Specialty	Phone	Email
1	Khalid Al-Maawali	Databases	96891234567	khalid@example.com
2	Noura Al-Kindi	Web Development	96892345678	noura@example.com
3	Salim Al-Harthy	Data Science	96893456789	salim@example.com

Course Table

ID	Title	Category	Duration (Hours)	Level
1	Database Fundamentals	Databases	20	Beginner
2	Web Development Basics	Web	30	Beginner
3	Data Science Introduction	Data Science	25	Intermediate
4	Advanced SQL Queries	Databases	15	Advanced

Schedule Table

ID	Course ID	Trainer ID	Start Date	End Date	Time Slot
1	1	1	2025-07-01	2025-07-10	Morning
2	2	2	2025-07-05	2025-07-20	Evening
3	3	3	2025-07-10	2025-07-25	Weekend
4	4	1	2025-07-15	2025-07-22	Morning

Enrollment Table

ID	Trainee ID	Course ID	Enrollment Date
1	1	1	2025-06-01
2	2	1	2025-06-02
3	3	2	2025-06-03

4	4	3	2025-06-04
5	5	3	2025-06-05
6	1	4	2025-06-06

Trainee Perspective SQL Query Solutions

1. Show all available courses (title, level, category)

```
-- Retrieves all courses with their title, difficulty level, and category
SELECT title, level, category
FROM Course;
```

2. View beginner-level Data Science courses

```
-- Filters courses to only show beginner-level Data Science courses
SELECT title, level, category
FROM Course
WHERE category = 'Data Science' AND level = 'Beginner';
```

3. Show courses this trainee is enrolled in (using trainee_id = 1)

```
-- Displays all courses that trainee_id 1 is enrolled in
SELECT c.title
FROM Course c
JOIN Enrollment e ON c.course_id = e.course_id
WHERE e.trainee_id = 1;
```

4. View the schedule for the trainee's enrolled courses (trainee_id = 1)

```
-- Shows start dates and time slots for all courses trainee_id 1 is taking
SELECT s.start_date, s.time_slot
FROM Schedule s
JOIN Enrollment e ON s.course_id = e.course_id
WHERE e.trainee_id = 1;
```

5. Count how many courses the trainee is enrolled in (trainee_id = 1)

```
-- Returns the total number of courses trainee_id 1 is enrolled in
SELECT COUNT(*) AS enrolled_courses_count
FROM Enrollment
WHERE trainee_id = 1;
```

6. Show course titles, trainer names, and time slots for trainee (trainee_id = 1)

```
-- Displays a complete schedule showing course, instructor, and session time
SELECT
    c.title AS course_title,
    t.name AS trainer_name,
    s.time_slot
FROM Enrollment e
JOIN Course c ON e.course_id = c.course_id
```

```
JOIN Schedule s ON e.course_id = s.course_id
JOIN Trainer t ON s.trainer_id = t.trainer_id
WHERE e.trainee_id = 1;
```

Trainer Perspective SQL Query Solutions

1. List all courses the trainer is assigned to (trainer_id = 1)

```
-- Retrieves all courses assigned to a specific trainer
SELECT c.title
FROM Course c
JOIN Schedule s ON c.course_id = s.course_id
WHERE s.trainer_id = 1;
```

---

2. Show upcoming sessions (with dates and time slots) (trainer_id = 1)

```
-- Displays future sessions for a trainer including schedule details
SELECT
    c.title AS course_title,
    s.start_date,
    s.end_date,
    s.time_slot
FROM Schedule s
JOIN Course c ON s.course_id = c.course_id
WHERE s.trainer_id = 1
AND s.start_date >= CURRENT_DATE
ORDER BY s.start_date;
```

---

3. Count trainees enrolled in each of your courses (trainer_id = 1)

```
-- Shows enrollment counts per course for a trainer
SELECT
    c.title AS course_title,
    COUNT(e.trainee_id) AS enrolled_trainees
FROM Course c
JOIN Schedule s ON c.course_id = s.course_id
LEFT JOIN Enrollment e ON c.course_id = e.course_id
WHERE s.trainer_id = 1
GROUP BY c.title;
```

---

4. List names and emails of trainees in each course (trainer_id = 1)

```
-- Displays trainee details for each course taught by the trainer
SELECT
    c.title AS course_title,
    t.name AS trainee_name,
    t.email AS trainee_email
FROM Trainee t
JOIN Enrollment e ON t.trainee_id = e.trainee_id
JOIN Course c ON e.course_id = c.course_id
JOIN Schedule s ON c.course_id = s.course_id
WHERE s.trainer_id = 1
ORDER BY c.title, t.name;
```

---

5. Show trainer's contact info and assigned courses (trainer_id = 1)

```
-- Returns trainer contact information with their course assignments
SELECT
    tr.name AS trainer_name,
    tr.phone,
```

```
    tr.email,
    GROUP_CONCAT(c.title SEPARATOR ', ') AS assigned_courses
FROM Trainer tr
JOIN Schedule s ON tr.trainer_id = s.trainer_id
JOIN Course c ON s.course_id = c.course_id
WHERE tr.trainer_id = 1
GROUP BY tr.trainer_id;
```

---

6. Count number of courses the trainer teaches (trainer_id = 1)

```
-- Calculates how many distinct courses a trainer is teaching
SELECT
    COUNT(DISTINCT s.course_id) AS total_courses_teaching
FROM Schedule s
WHERE s.trainer_id = 1;
```

Admin Perspective SQL Query Solutions

1. Add a new course (INSERT statement)

```
-- Inserts a new course into the Course table
INSERT INTO Course (course_id, title, category, duration_hours, level)
VALUES (5, 'Python Programming', 'Programming', 40, 'Intermediate');
```

---

2. Create a new schedule for a trainer

```
-- Schedules a new course session with trainer assignment
INSERT INTO Schedule (schedule_id, course_id, trainer_id, start_date, end_date, time_slot)
VALUES (5, 5, 2, '2025-08-01', '2025-08-15', 'Evening');
```

---

3. View all trainee enrollments with course title and schedule info

```
-- Comprehensive enrollment report with course and schedule details
SELECT
    t.name AS trainee_name,
    c.title AS course_title,
    c.level AS course_level,
    s.start_date,
    s.end_date,
    s.time_slot,
    tr.name AS trainer_name
FROM Enrollment e
JOIN Trainee t ON e.trainee_id = t.trainee_id
JOIN Course c ON e.course_id = c.course_id
JOIN Schedule s ON c.course_id = s.course_id
JOIN Trainer tr ON s.trainer_id = tr.trainer_id
ORDER BY c.title, t.name;
```

---

4. Show how many courses each trainer is assigned to

```
-- Counts course assignments per trainer
SELECT
    tr.name AS trainer_name,
    COUNT(DISTINCT s.course_id) AS assigned_courses
FROM Trainer tr
LEFT JOIN Schedule s ON tr.trainer_id = s.trainer_id
GROUP BY tr.trainer_id
ORDER BY assigned_courses DESC;
```

5. List all trainees enrolled in "Database Fundamentals"

```
-- Retrieves trainee details for a specific course
SELECT
    t.name AS trainee_name,
    t.email,
    t.background
FROM Trainee t
JOIN Enrollment e ON t.trainee_id = e.trainee_id
JOIN Course c ON e.course_id = c.course_id
WHERE c.title = 'Database Fundamentals';
```

6. Identify the course with the highest number of enrollments

```
-- Finds the most popular course by enrollment count
SELECT
    c.title AS course_title,
    COUNT(e.enrollment_id) AS enrollment_count
FROM Course c
LEFT JOIN Enrollment e ON c.course_id = e.course_id
GROUP BY c.course_id
ORDER BY enrollment_count DESC
LIMIT 1;
```

7. Display all schedules sorted by start date

```
-- Shows all schedules in chronological order
SELECT
    c.title AS course_title,
    tr.name AS trainer_name,
    s.start_date,
    s.end_date,
    s.time_slot
FROM Schedule s
JOIN Course c ON s.course_id = c.course_id
JOIN Trainer tr ON s.trainer_id = tr.trainer_id
ORDER BY s.start_date ASC;
```

Database Error Log

Timestamp	Error Code	Description	Solution
2025-07-05 09:30	1064	Syntax error near 'CURRENT_DATE' in Enrollment table creation	Changed DEFAULT CURRENT_DATE to DEFAULT (CURRENT_DATE) for MySQL
2025-07-05 10:15	1054	Unknown column 'duration_hours' in Course table	Fixed typo from 'duration_hours' to 'duration_hours'
2025-07-06 11:05	1064	Syntax error in CHECK constraint for time_slot	Modified CHECK constraint to include 'Weekend' option
2025-07-06 14:20	1146	Table 'CourseManagementSystem.Level' doesn't exist	Added 'level' column to Course table instead of separate table
2025-07-06 15:45	1452	Foreign key constraint fails (invalid trainer_id in Schedule)	Verified all trainer_id values exist in Trainer table before inserting
2025-07-07 08:10	1364	Field 'level' doesn't have a default value	Added DEFAULT 'Beginner' to level column
2025-07-07 10:30	1062	Duplicate entry '1' for key 'PRIMARY' in Enrollment	Ensured enrollment_id values are unique across all inserts

References

IBM. "The Importance of Data Structures in Data Science."
https://www.ibm.com/topics/data-structures

Mohimen, M. (2024). Learning Management System: An Operational Database Design. Medium.
https://medium.com/@mgbrmohimen/learning-management-system-an-operational-database-design-4dc04c2c863b

Netflix Tech Blog. "How Netflix Uses Data Modeling."
https://netflixtechblog.com/

Percona. (2023). Percona Helps SumTotal Systems Optimize a Busy MySQL Database to Ensure Uptime and Maintain Performance.
https://www.percona.com/about-percona/case-studies/percona-helps-sumtotal-systems-optimize-busy-mysql-database-ensure-uptime

Scientific.Net. (2012). The Design and Implementation of Enterprise Training Management System Based on .NET. Applied Mechanics and Materials, 644-650, 6056.
https://www.scientific.net/AMM.644-650.6056

Spotify Engineering. "Data Pipelines at Spotify."
https://engineering.atspotify.com/

Uber Engineering. "Scalable Data Warehousing at Uber."
https://eng.uber.com/

Vertabelo. (2024). Database Design for a Learning Management System.
https://vertabelo.com/blog/database-design-management-system/