Database design document

Groub members:

Rawad abdalla: 315281964

Mohamad Abed Aljwad: 206954950

Maher Bathish : 209417112

Date :

30/12/2024

Overview

This document outlines the database design for the application in our project, covering its schema, relationships, constraints, and operational considerations. The goal is to create a scalable, normalized, and efficient database system that supports the core functionalities of the application, including managing parking events, customers, vehicles, parking spaces, Citations and transaction logging.

Database Objectives

- Consistency: Ensure referential and transactional integrity.
- Scalability: Support future growth in data and queries.
- Performance: Optimize read and write operations.
- Security: Restrict access based on user roles and enforce data privacy.
- Reliability: ensures consistent working at all hours of the day.

Schema Design

1. Entities and Relationships

The database schema is designed around the following core entities:

- Customers: Stores customer information.
- Vehicles: Stores vehicle information and links vehicles to customers.
- ParkingSpaces: Stores all parking space details and tracks parking space status.

- Zones: Allocates parking spaces and sets rates.
- ParkingEvents: Records parking events for vehicles.
- Citations: Tracks citations issued for parking violations.
- SystemLog: Records PEO actions.

2. Logical Data Model

- **Customers Table**

Stores customer details:

Column Name	Data Type	Constraints	Description
CustomerID	INT	PRIMARY KEY	Unique identifier for a customer.
Name	VARCHAR(255)	NOT NULL	Customer's name.
Phone	VARCHAR(15)	NOT NULL	Customer's phone number.
Email	VARCHAR(100)	UNIQUE, NOT NULL	Customer's email address.
TotalPaid	DECIMAL(10, 2)	default 0	How mush the customer paid
password	varchar(255)	NOT NULL	Customer password

- **Vehicles Table**

Links vehicles to customers:

Column Name	Data Type	Constraints	Description
VehicleID	INT	PRIMARY KEY	Unique identifier for a vehicle.
CustomerID	INT	FOREIGN KEY REFERENCES Customers(CustomerID)	Associated customer.

Column Name	Data Type	Constraints	Description
LicensePlate	VARCHAR(20)	UNIQUE, NOT NULL	Vehicle license plate.

- **ParkingSpaces Table**

Tracks the state of parking spaces:

Column Name	Data Type	Constraints	Description
SpaceID	INT	PRIMARY KEY	Unique identifier for a parking space.
ZoneID	INT	FOREIGN KEY REFERENCES Zones(ZoneID)	Associated zone.
Occupied	BOOLEAN	DEFAULT FALSE	Indicates if the space is occupied.
MaxTime	INT	NOT NULL	Maximum parking time in minutes.

- **Zones Table**

Defines parking zones and rates:

Column Name	Data Type	Constraints	Description
ZoneID	INT	PRIMARY KEY	Unique identifier for a zone.
ZoneName	VARCHAR(100)	NOT NULL, unique	Zone name.
HourlyRate	DECIMAL(10,2)	NOT NULL	Cost per hour.

- **ParkingEvents Table**

Logs parking events for vehicles:

Column Name	Data Type	Constraints	Description
EventID	INT	PRIMARY KEY, AUTO_INCREMENT	Unique identifier for a parking event.

Column Name	Data Type	Constraints	Description
VehicleID	INT	NOT NULL, FOREIGN KEY REFERENCES Vehicles(VehicleID)	Associated vehicle.
SpaceID	INT	NOT NULL, FOREIGN KEY REFERENCES ParkingSpaces(SpaceID)	Associated parking space.
StartTime	DATETIME	NOT NULL	Parking start time.
EndTime	DATETIME	NULLABLE	Parking end time.
TotalCost	DECIMAL(10,2)	DEFULT 0	Total cost for parking.
MaxTime	INT	DEFULT NULL	Maximum time allowed for the car.

Citations Table

Tracks citations issued for parking violations:

Column Name	Data Type	Constraints	Description
CitationID	INT	PRIMARY KEY	Unique identifier for a citation.
VehicleID	INT	NOT NULL , FOREIGN KEY REFERENCES Vehicles(VehicleID)	Associated vehicle.
SpaceID	INT	NOT NULL, FOREIGN KEY REFERENCES ParkingSpaces(SpaceID)	Associated parking space.
ZoneID	INT	NOT NULL, FOREIGN KEY REFERENCES Zones(ZoneID)	
InspectionTime	DATETIME	NOT NULL	
CitationCost	DECIMAL(10, 2)	not null	CitationCost

- **SystemLog Table**

Records SystemLog:

Column Name	Data Type	Constraints	Description
LogID	INTEGER	primary key autoincrement	A unique identifier for each log entry. Automatically increments for each new record.
QueryTime	DATETIME	NOT NULL	The timestamp indicating when the query was made.
VehicleNumber	TEXT	NOT NULL	The unique identifier (e.g., license plate) of the vehicle involved in the query.
ParkingSpaceId	TEXT	NOT NULL	The identifier of the parking space associated with the query.
Response	TEXT	NOT NULL	The result or feedback generated by the system in response to the query (e.g., "Parking OK)

Relationships

- **Customers → Vehicles**: One-to-Many (A customer can own multiple vehicles).
- **ParkingSpaces → Zones**: Many-to-One (Multiple spaces belong to a zone).
- **ParkingEvents → Vehicles**: Many-to-One (Multiple events can be associated with one vehicle).
- **ParkingEvents → ParkingSpaces**: Many-to-One (Multiple events can be associated with one space).
- **Citations → Vehicles**: Many-to-One (Multiple citations can be associated with one vehicle).

Key Updates in Database Tables:

- A password field has been added for each customer to facilitate login functionality, allowing for easier access and modifications to customer data via the new Login UI.
- The structure of all tables has been updated to include the `ENGINE=ndbcluster` setting. This change optimizes the database for clustering, enhancing performance and scalability.

Working with Clustering:

To set up the clustering for our MySQL database, we followed a series of structured steps to ensure high availability and distributed performance. Here's how we worked through it:

Steps Taken:

1. **Download the Same Version of MySQL Server**: We ensured that all nodes in the cluster run the same version of MySQL to maintain consistency and compatibility across the system.
2. **Configuration of the my.ini File on Each Node**: Each node in the cluster was configured with the appropriate my.ini file. This configuration ensures that all nodes can communicate with each other within the cluster.
3. **Configure the Management Node**: On the management node, we configured the config.ini file. This file specifies the nodes in the cluster, including data nodes and MySQL server nodes.
4. **Create the Database in MySQL Workbench**: In MySQL Workbench, each node opened a new database based on its IP address. The database was created from scratch with all necessary changes implemented to align with the cluster configuration.

Advantages of using clustering in a database system:

1. High Availability

- **Fault Tolerance**: Clustering ensures that if one node (server) fails, others in the cluster can continue operating without downtime. This is achieved through data replication and redundancy.
- **Automatic Failover**: In case of a node failure, the cluster can automatically switch to a backup node, ensuring the system remains available and operational.

2. Scalability

- **Horizontal Scaling**: Clustering allows you to scale your database horizontally by adding more nodes to the cluster. As the system grows and traffic increases, you can easily add nodes to distribute the load without significant reconfiguration.
- **Load Balancing**: Requests can be distributed across multiple nodes, which optimizes resource utilization and reduces the load on any single node. This helps maintain high performance even under heavy workloads.

3. Performance Optimization

- **Data Distribution**: Clustering enables the distribution of data across multiple nodes, improving read and write performance. Each node can handle a portion of the data, reducing bottlenecks.
- **Faster Queries**: By storing data on multiple nodes, query performance can improve, especially in read-heavy environments, since multiple nodes can respond to queries in parallel.

4. Data Redundancy and Backup

- **Replication**: In a cluster, data is typically replicated across multiple nodes, providing redundancy. This ensures that there are multiple copies of the data, reducing the risk of data loss.
- **Automatic Data Recovery**: When a node fails, data can be recovered from replicated nodes, and recovery is usually automatic, minimizing the risk of data corruption.

5. Improved Fault Isolation

- **Node Independence**: Since each node in the cluster operates independently, a failure in one node typically doesn't affect the other nodes. This means that failures are isolated, and you can address issues without taking down the entire system.
- **Reduced Downtime**: In case of hardware or software issues, clusters can continue functioning while maintenance is performed on the affected node, leading to less downtime.

6. Flexible Data Management

- **Geographic Distribution**: Clusters can be set up to span multiple geographic locations, improving performance by reducing latency for users in different regions.
- **Data Locality**: By placing related data on the same node or group of nodes, you can reduce the need for cross-node communication, further improving performance.

7. Centralized Management

- **Single Point of Control**: Cluster management tools allow for centralized monitoring and management of the entire system, making it easier to oversee and optimize the cluster as a whole.

- **Simplified Backup and Recovery**: Managing backups and recovery processes becomes easier with a clustered system, as data replication and redundancy allow for efficient and reliable backup strategies.

8. Cost Efficiency

- **Cost-Effective Scaling**: Instead of purchasing expensive high-end hardware for scaling, clusters allow you to scale by adding additional low-cost nodes. This makes it easier to scale the system economically.
- **Efficient Resource Use**: Clusters can efficiently allocate resources across nodes, ensuring that hardware is utilized optimally.

9. Maintenance Without Downtime

- **Rolling Updates**: You can perform maintenance tasks (e.g., software updates, hardware upgrades) on one node at a time without taking down the entire system. This means you can keep the system operational while performing necessary tasks.

Configuration Files:

```
[ndbd default]
NoOfReplicas=3
DataMemory=256M

[ndb_mgmd]
NodeId=1
HostName=100.76.110.20   # Management Node
PortNumber=1186

[ndbd]
NodeId=2
HostName=100.76.110.20   # Data Node 1
DataDir=C:\ProgramData\MySQL\MySQL Server 9.1\Data

[ndbd]
NodeId=3
HostName=100.85.154.51   # Data Node 2
DataDir=C:\ProgramData\MySQL\MySQL Server 9.1\Data

[ndbd]
NodeId=4
HostName=100.78.144.87   # Data Node 2
DataDir=D:\ProgramData\MySQL\MySQL Server 9.1\Data

[mysqld]
NodeId=5
HostName=100.76.110.20

[mysqld]
NodeId=6
HostName=100.85.154.51

[mysqld]
NodeId=7
HostName=100.78.144.87
```

MySQL Server Instance Configuration (my.ini): example for one node

```
# MySQL Server Instance Configuration File

[mysqld]

# Existing configurations
port=3306
datadir=C:/ProgramData/MySQL/MySQL Server 9.1/Data

# MySQL Cluster settings
ndbcluster                              # Enable NDB Cluster
ndb-connectstring=100.76.110.20:1186  # Management Node address

# General settings
bind-address=0.0.0.0                    # Allow connections from any IP address
log-bin="cluster-bin"                   # Enable binary log for replication
server-id=1                             # Unique server ID for replication

ndbcluster
#ndb-connectstring=100.76.110.20:1186
default-storage-engine=InnoDB

# Logging settings
log-output=FILE
general-log=0
general_log_file="MOHAMMAD.log"
slow-query-log=1
slow_query_log_file="MOHAMMAD-slow.log"
long_query_time=10
log-error="MOHAMMAD.err"

# Case sensitivity
lower_case_table_names=1

# Secure file privilege
secure-file-priv="C:/ProgramData/MySQL/MySQL Server 9.1/Uploads"

# Connection settings
max_connections=151
max_allowed_packet=64M
max_connect_errors=100
```


```
# Secure file privilege
secure-file-priv="C:/ProgramData/MySQL/MySQL Server 9.1/Uploads"

# Connection settings
max_connections=151
max_allowed_packet=64M
max_connect_errors=100
```


```
# Cache settings
table_open_cache=4000
tmp_table_size=39M

# MyISAM settings
myisam_max_sort_file_size=2146435072
myisam_sort_buffer_size=70M
key_buffer_size=8M
read_buffer_size=128K
read_rnd_buffer_size=256K

# InnoDB settings
innodb_flush_log_at_trx_commit=1
innodb_log_buffer_size=16M
innodb_buffer_pool_size=128M
innodb_autoextend_increment=64
innodb_buffer_pool_instances=8
innodb_file_per_table=1
innodb_checksum_algorithm=0

# Sorting and join settings
join_buffer_size=256K
sort_buffer_size=256K

# Binary Log settings
binlog_row_event_max_size=8K
sync_source_info=10000
sync_relay_log=10000

# MySQL Cluster section
[mysql_cluster]
ndb-connectstring=100.76.110.20:1186
```

Database design