Edge Hill University

The Department of Computer Science

CIS4503 Databases

Coursework 2 Part A
2024/2025

Sunbeat Music Festival Database Design Report

Student Name: Sunday Idika
Student Number: 25978497

1. Introduction

1.1 Overview of the Task

This report's objective is to design a relational database for managing data from the Music Festival Event Management System. The database should store information about attendees, tickets, performances, performers, stages, camping reservations, non-musical entertainment events, and the overall festival schedule. The design must ensure that the database is well-organized, minimizes redundancy, and can handle complex relationships between the different entities (Yao et al., 2022).

1.2 Scenario Description

The scenario involves a festival where attendees can buy tickets for musical performances, book camping spots, and attend non-musical entertainment events. The data that needs management includes personal details of attendees, ticket information, performance details, performer information, stage locations, camping reservations, and the schedule for non-musical events. The challenge is designing a database that can efficiently store this data while maintaining relationships between these entities. (Al-Ghifari and Azizah, 2022).

This report outlines the database design process, from the creation of an Entity-Relationship Diagram (ERD), and normalization to the development of a data dictionary.

2. Ambiguities and Assumptions

2.1 Identified Ambiguities

There were a few areas in the scenario where the requirements were unclear:
In the process of designing the ERD for the festival database, several ambiguities arose due to overlapping or unclear attributes and relationships.

- One major ambiguity involved the **relationship between Attendees and Performances**: in the initial design, attributes such as TicketType, PerformanceID, PerformerName, and StageLocation were all included in the Attendee entity, creating confusion. This setup implied that an attendee could be directly associated with specific performers and stage locations, which isn't accurate since these details should instead belong to performances and tickets. To resolve this, I separated these attributes into their appropriate entities—linking Attendee to Ticket and Ticket to Performance for a clearer, normalized structure. For example, separating TicketType into its own entity avoids situations where an attendee could accidentally be linked with the wrong performance or ticket type.

- **Redundant Performer Information**: In the unnormalized form, Performance included PerformerName and StageLocation, which implied each performance might have a direct link to these details. However, a performer should ideally be represented by a PerformerID in Performance, which links back to the Performer entity containing PerformerName and Genre. By centralizing performer data in the Performer table and referencing it through PerformerID in Performance, data redundancy is reduced, and clarity is improved.

- Another area of ambiguity was with the **Festival Schedule**. The unnormalized form listed a ScheduleID and attributes like EventID, StageID or LocationID, Date, and Time, suggesting it could include either a performance or entertainment event, but without clear distinction. To clarify, I introduced an EventType attribute in the final ERD, specifying whether the scheduled event is a musical performance or a non-musical entertainment activity. This attribute

reduces ambiguity and improves database querying by allowing explicit identification of event types, enhancing the schema's clarity.

2.2 Assumptions in the Database Model

Here are the assumptions made in designing the ERD:

- Unique Identification: Each entity requires a unique identifier. AttendeeID, TicketID, PerformanceID, PerformerID, StageID, EntertainmentID, and ScheduleID serve as primary keys, ensuring each record is uniquely identifiable.

- Performance Structure: Each performance occurs on a single stage at a specific date and time. A performer can have multiple performances, but each is tied to a unique PerformanceID, Date, Time, and StageID.

- Tickets: Each attendee can purchase multiple tickets, each linked to a specific performance. The TicketType indicates ticket categories (e.g., General Admission, VIP).

- Camping Reservations: Attendees can reserve a camping space, and a single reservation includes details like CampingType, Price, CheckInDate, and CheckOutDate.

- Festival Schedule Representation: The Festival Schedule contains both performances and non-musical entertainment events, differentiated by an EventType attribute that clarifies the type of event.

- Stages and Locations: Each stage has a specific location and capacity. Non-musical entertainment events are either at a particular Location or a StageID if performed on a specific stage.

- Single Role of Attendees: Each attendee is represented as a single individual entity with no differentiation between roles (e.g., attendees are not staff or performers).

- Data Completeness: It’s assumed that all fields (such as Date, Time, Location) are consistently provided for each event to maintain schedule accuracy.

3. Database Design

This section explores the database design process for managing a festival's attendee, performance, and scheduling data. It includes the initial ERD, key concepts, final ERD, and a justification of the final design choices.

3.1 Entity-Relationship Diagram (ERD) - Initial Design

An Entity-Relationship Diagram (ERD) is a diagram that visually represents the structure of a database. It shows the entities involved, the relationships between them, and their attributes (Afiifah, Azzahra and Anggoro, 2022). The first draft of the ERD, as shown in Figure 1, includes the following entities: **Attendee**, **Ticket**, **Performance**, **Performer**, **Stage**, **Camping Reservation**, **Non-Musical Entertainment**, and **Festival Schedule**.

Initial Entity Relationship Diagram (ERD)

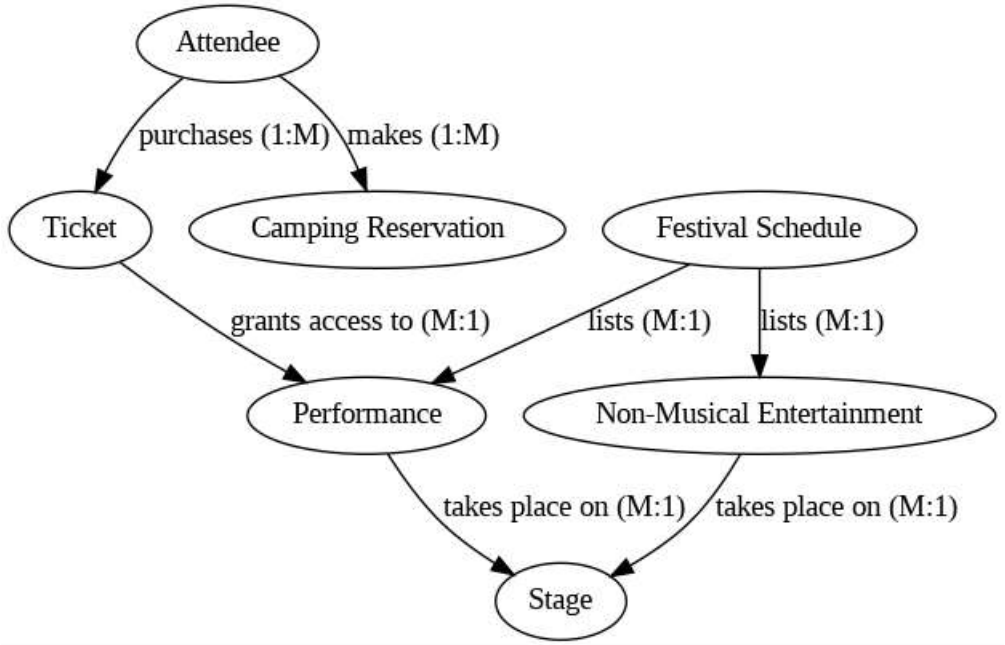

Figure 1 (designed using Python)

The initial design of the entity-relationship diagram (ERD) captures all essential data points identified in the unnormalized form of the dataset. The following entities are established in the initial ERD based on attributes and preliminary relationships identified in the initial analysis:

1. **Attendee**: Contains attributes such as AttendeeID, Name, and Email. Initially, TicketType, PerformanceID, and other related details were also associated with this entity.
2. **Ticket**: Represents a relationship between attendees and performances, with attributes TicketID, AttendeeID, PerformanceID, and TicketType (such as General Admission or VIP).
3. **Performance**: This entity includes information about each specific performance, such as PerformanceID, PerformerName, StageLocation, PerformanceTime, Genre, Location, and Capacity.
4. **Performer**: Contains details unique to performers, such as PerformerID, Name, Genre, and the performances they're associated with through PerformanceID.
5. **Stage**: Represents physical venues within the festival, identified by StageID, with additional attributes for Location and Capacity.
6. **Camping Reservation**: Tracks attendee reservations for camping with attributes ReservationID, AttendeeID, CampingType, Price, CheckInDate, and CheckOutDate.
7. **Non-Musical Entertainment**: Captures other entertainment options provided at the festival, identified by EntertainmentID, and includes attributes like Name, Description, Date, Time, and Location.
8. **Festival Schedule**: Represents the event schedule with attributes ScheduleID, EventID, EventType (to distinguish between a performance or non-musical entertainment), StageID or LocationID, Date, and Time.

3.2 Explanation of ERD Concepts

The key concepts that inform the ERD design, including the nature of entities, relationships, primary and foreign keys, and the normal forms applied to the database schema were discussed in this section (Agrawal et al., 2009).

- **Entities**: Each entity in the ERD represents a table in the database. For instance, the **Attendee** entity represents a table that stores details about festival attendees (such as name and email).
- **Attributes**: These are the data fields that belong to each entity. For example, the **Attendee** entity has attributes like AttendeeID, Name, and Email.
- **Relationships**: This shows how entities are connected. For example, **Attendee** and **Ticket** have a one-to-many relationship (an attendee can buy multiple tickets).
- **Cardinality**: This shows the number of instances of one entity related to another. For example, Each **Attendee** can have multiple **Tickets** (one-to-many), but each **Ticket** is linked to only one **Attendee**.
- **Optionality**: This defines whether a relationship is mandatory or optional. For example, Each **Camping Reservation** is associated with a single **Attendee**, but an **Attendee** may have multiple **Camping Reservations** (one-to-many).

See Appendix B for the summary of Relationships with Cardinality and Optionality.

- **Primary and Foreign Keys**:
  - **Primary keys** uniquely identify each record in a table (e.g., AttendeeID in Attendee, TicketID in Ticket).
  - **Foreign keys** are references to primary keys in other tables. For instance, AttendeeID in Ticket is a foreign key linking to AttendeeID in the Attendee table, establishing a relational link.
- **Normalization**: The ERD undergoes normalization to ensure the database structure is efficient and minimizes redundancy. The schema follows these normal forms:
  - **1NF**: Removes repeating groups by ensuring each attribute is atomic.
  - **2NF**: Ensures all non-key attributes are fully functionally dependent on the primary key.
  - **3NF**: Eliminates transitive dependencies so that each non-key attribute depends only on the primary key

3.3 Final Entity-Relationship Diagram (ERD)

The final ERD (See Figure 2) includes a refined structure, incorporating adjustments from the initial design to reduce redundancies, eliminate ambiguities, and improve clarity. Key changes made in the final ERD include:

1. **Attendee**:
   - Attendee is simplified to include only AttendeeID, Name, and Email, removing performance-specific data that was previously attached.
2. **Ticket**:
   - Ticket holds TicketID, AttendeeID (FK to Attendee), PerformanceID (FK to Performance), and TicketType. By linking each ticket to a specific performance, this entity serves as an associative table for attendees and performances.
3. **Performance** and **Performer**:
   - The Performance entity now includes PerformanceID, PerformerID (FK to Performer), StageID (FK to Stage), Date, and Time, linking each performance to a unique performer and stage.
   - Performer remains unchanged, containing PerformerID, Name, and Genre.
4. **Stage**:

- Stage retains StageID, Location, and Capacity, representing the physical setup for performances.

5. **Camping Reservation**:
   - This entity includes ReservationID, AttendeeID (FK to Attendee), CampingType, Price, CheckInDate, and CheckOutDate, capturing essential details for camping.
6. **Non-Musical Entertainment**:
   - Non-Musical Entertainment includes EntertainmentID, Name, Description, Date, Time, and Location.
7. **Festival Schedule**:
   - The final Festival Schedule entity includes ScheduleID, EventID (FK to Performance or Non-Musical Entertainment), EventType, StageID or Location, Date, and Time, which helps schedule and differentiate musical and non-musical events.
   -

Final ERD Diagram

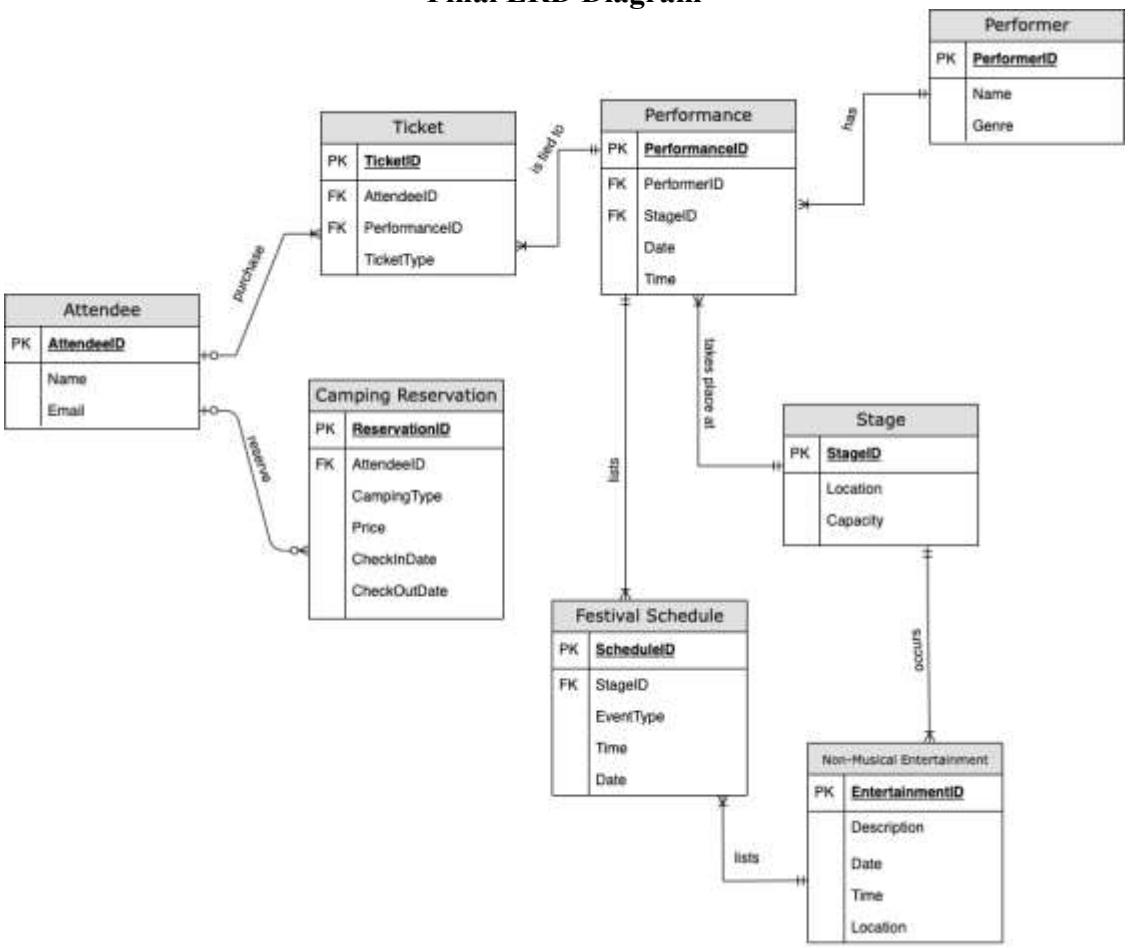

Figure 2 (designed using diagrams.net)

Bidirectional Relationships in the Final ERD: Entity Connections and Data Flow

1. **Attendee and Ticket**
   - Each **Attendee** can hold multiple **Tickets** for different performances.
   - Each **Ticket** is associated with one **Attendee**, representing who purchased or holds it.
2. **Ticket and Performance**
   - Each **Ticket** is linked to one specific **Performance**, indicating which event it grants access to.
   - Each **Performance** can have multiple **Tickets** issued for it, corresponding to various attendees who will attend that performance.
3. **Performance and Performer**

- Each **Performance** is associated with one **Performer** who is the primary act for that performance.
- Each **Performer** can have multiple **Performances**, allowing the same performer to appear in different slots or on different stages during the festival.

4. **Performance and Stage**
    - Each **Performance** takes place on one **Stage**, representing the physical location of the performance.
    - Each **Stage** can host multiple **Performances** over time, as different performers or events are scheduled there.
5. **Attendee and Camping Reservation**
    - Each **Attendee** can make multiple **Camping Reservations** if they book different types of camping arrangements.
    - Each **Camping Reservation** is linked to one **Attendee**, indicating who made the reservation.
6. **Non-Musical Entertainment and Festival Schedule**
    - Each **Non-Musical Entertainment** event can be scheduled once or multiple times in the **Festival Schedule**.
    - Each entry in the **Festival Schedule** can represent a **Non-Musical Entertainment** event, if the EventType specifies this kind of event.
7. **Performance and Festival Schedule**
    - Each **Performance** can appear once or multiple times in the **Festival Schedule** (e.g., if a performance is rescheduled or has an encore).
    - Each entry in the **Festival Schedule** can represent a **Performance** event, if the EventType specifies this type of event.
8. **Stage and Festival Schedule**
    - Each **Stage** can be associated with multiple entries in the **Festival Schedule**, indicating different events happening on that stage.
    - Each entry in the **Festival Schedule** can reference a **Stage**, showing where an event will take place if it's a performance scheduled at a specific stage.

3.4 Justification of Final ERD Choices

The final ERD reflects a careful balance of normalization, usability, and flexibility in the festival database design. Here's the rationale behind the key changes:

1. **Data Integrity and Reduction of Redundancy**:
    - By separating Attendee, Ticket, and Performance, the database ensures that each ticket is associated with a specific performance without duplicating attendee or performance data. The ERD's normalization up to 3NF reduces redundancy, particularly by creating separate Performer and Stage entities.
2. **Clear Relationships and Distinct Entity Roles**:
    - The final ERD avoids redundant fields by giving each entity a focused role. For example, Attendee no longer contains PerformanceID or StageLocation; instead, Ticket serves as the bridge linking attendees to performances, clarifying the relationship between entities.
3. **Flexibility in Scheduling**:
    - With Festival Schedule, the database can handle both musical performances and non-musical events. By including an EventType attribute, the schedule can efficiently differentiate between types of events, facilitating better data retrieval for event planning.
4. **Enhanced Query Efficiency**:
    - The use of foreign keys (e.g., linking Performance to Performer through PerformerID and to Stage through StageID) enhances query efficiency

by centralizing performer and stage data. This relational setup allows for simpler, more focused queries, improving database performance.

5. **Consistency in Event Management**:
    - The addition of Date and Time attributes to entities such as Performance, Non-Musical Entertainment, and Festival Schedule ensures that all scheduled events have clearly defined temporal attributes, enhancing schedule management.
6. **Scalability and Adaptability**:
    - The design is scalable and adaptable for potential festival growth. If the festival introduces more stages or additional entertainment types, the structure can be easily extended by adding new records or attributes without altering the core schema.

4. Normalization Process

4.1 Explanation of Normalization

Normalization is the process of organizing a database to reduce redundancy and dependency by dividing large tables into smaller, more manageable ones. The goal is to eliminate common data issues, such as update anomalies, insertion anomalies, and deletion anomalies (Huang et al., 2020). The process involves several stages, see Appendix A:

- **First Normal Form (1NF)**: Ensures that all attributes contain atomic values (no repeating groups) and that each record is unique. For example, attributes like StageLocation and PerformerName in the Performance entity contain repeating groups. By normalizing to 1NF, these are moved to distinct entities, preventing redundancy.
- **Second Normal Form (2NF)**: Removes partial dependencies, ensuring that all non-key attributes are fully dependent on the primary key. For example, partially dependent attributes like StageLocation (linked to StageID) are removed from Performance to prevent updates from affecting unrelated data
- **Third Normal Form (3NF)**: Eliminates transitive dependencies, ensuring that non-key attributes depend only on the primary key.

4.2 Table of Normalization Process

Stage	Description	Entity Changes
UNF	Unnormalized data with redundancy	Initially, multiple entities had redundant information, especially between performances and stages.
1NF	Removed repeating groups	Created separate tables for each entity and eliminated multi-valued attributes.
2NF	Removed partial dependencies	Split attributes such as **Performer** and **Stage** into separate tables to avoid partial dependencies on the primary key.
3NF	Removed transitive dependencies	Ensured that attributes like **Location** and **Date**

		are directly related to their respective primary keys, not other attributes.

4.3 Discussion of Entities in Third Normal Form (3NF)

- **Attendee Table**: All attributes in the **Attendee** table are fully dependent on the **AttendeeID**, which is the primary key. This ensures the table is in 3NF.
- **Ticket Table**: The **TicketType** attribute depends only on the **TicketID**, and not on other attributes, making this table compliant with 3NF.
- **Performance Table**: Attributes like **Date** and **Time** depend only on the **PerformanceID** and not on any other non-key attributes.
- **Performer Table**: The **Name** and **Genre** attributes depend only on the **PerformerID**, which is the primary key of this table.

5. Data Dictionary

5.1 Definition and Purpose of a Data Dictionary

A data dictionary is a centralized repository that defines the structure of the database. It provides detailed descriptions of each table, its attributes, and their relationships to other tables. The purpose of a data dictionary is to standardize the way data is stored and to make the database more accessible and easier to understand for developers and users (Rashid et al., 2020).

5.2 Data Dictionary for Each Entity

- **Attendee Table**

Name	**Optional**	**Format**	**Length**	**Description**
AttendeeID	No	INT	10	Primary key, unique attendee ID
Name	No	VARCHAR	100	Full name of the attendee
Email	No	VARCHAR	100	Email address of the attendee

- Ticket Table

Name	Optional	Format	Length	Description
TicketID	No	INT	10	Primary key, unique ticket ID
AttendeeID	No	INT	10	Foreign key to Attendee table
PerformanceID	No	INT	10	Foreign key to Performance table
TicketType	No	VARCHAR	50	Type of ticket (e.g., general, VIP)

- **Performance Table**

Name	Optional	Format	Length	Decription
PerformanceID	No	INT	10	Primary key, unique performance ID
PerformerID	No	INT	10	Foreign key to Performer table
StageID	No	INT	10	Foreign key to Stage table
Date	No	DATE	10	Date of the performance
Time	No	TIME	8	Time of the performance

- **Performer (Artist) Table**

Name	Optional	Format	Length	Decription
PerformerID	No	INT	10	Primary key, unique performer ID
Name	No	VARCHAR	100	Performer's name
Genre	No	VARCHAR	50	Performer's genre (e.g., Rock, Jazz, etc.)

- **Stage Table**

Name	Optional	Format	Length	Description
StageID	No	INT	10	Primary key, unique reservation ID
Location	No	VARCHAR	100	Location of the stage
Capacity	No	INT	10	Maximum capacity of the stage

- **Camping Reservation Table**

Name	Optional	Format	Length	Description

ReservationID	No	INT	10	Primary key, unique reservation ID
AttendeeID	No	INT	10	Foreign key to Attendee table
CampingType	No	VARCHAR	50	Type of camping spot reserved

- **Non-Musical Entertainment Table**

Name	Optional	Format	Length	Description
EntertainmentID	No	INT	10	Primary key, unique entertainment ID
Name	No	VARCHAR	100	Name of the entertainment event
Description	No	TEXT		A brief description of the event
Date	No	DATE	10	Date of the event
Time	No	TIME	8	Time of the event
Location	No	VARCHAR	100	Location of the event

- **Festival Schedule Table**

Name	Optional	Format	Length	Description
ScheduleID	No	INT	10	Primary key, unique schedule ID
EventID	No	INT	10	Foreign key to either Performance or Entertainment table
EventType	No	VARCHAR	50	Type of event (e.g., performance, entertainment)

StageID	Yes	INT	10	Foreign key to Stage table (nullable, used only for performances)
LocationID	Yes	INT	10	Foreign key to Location (used for non-musical entertainment events)
Date	No	DATE	10	Date of the event
Time	No	TIME	8	Time of the event

6. References

1. AFIIFAH, K., AZZAHRA, Z.F., and ANGGORO, A.D., 2022. Analisis Teknik Entity-Relationship Diagram dalam Perancangan Database Sebuah Literature Review. *INTECH*. 3 (1), pp. 8–11.
2. AGRAWAL, D., ABBADI, A.E., EMEKCI, F., and METWALLY, A., 2009. Database Management as a Service: Challenges and Opportunities. *IEEE Xplore* [online]. Available from: https://ieeexplore.ieee.org/document/4812596?arnumber=4812596 [Accessed 30 Apr 2022].
3. AL-GHIFARI, M.F. and AZIZAH, F.N., 2022. Development of Application for Entity-Relationship Diagram Conversion to Logical Schema of NoSQL Column Oriented. *IEEE Xplore* [online]. Available from: https://ieeexplore.ieee.org/document/9972074?arnumber=9972074 [Accessed 19 Jan 2023].
4. DRAW.IO, 2024. Flowchart Maker & Online Diagram Software. *app.diagrams.net* [online]. Available from: https://app.diagrams.net/.
5. HUANG, L., QIN, J., ZHOU, Y., FANG, Y., LIU, L., and SHAO, L., 2020. Normalization Techniques in Training DNNs: Methodology, Analysis and Application. *IEEE transactions on pattern analysis and machine intelligence*. 45 (8), pp. 1–20.
6. RASHID, S.M., MCCUSKER, J.P., PINHEIRO, P., BAX, M.P., SANTOS, H.O., STINGONE, J.A., DAS, A.K., and MCGUINNESS, D.L., 2020. The Semantic Data Dictionary – An Approach for Describing and Annotating Data. *Data Intelligence* [online]. 2 (4), pp. 443–486. Available from: https://direct.mit.edu/dint/article/2/4/443/94892/The-Semantic-Data-Dictionary-An-Approach-for.
7. YAO, X., LI, J., TAO, Y., and JI, S., 2022. Relational Database Query Optimization Strategy Based on Industrial Internet Situation Awareness System. *2022 7th International Conference on Computer and Communication Systems (ICCCS)* [online]. pp. 152–155. Available from: https://ieeexplore.ieee.org/abstract/document/9846105 [Accessed 10 Nov 2024].

7. Appendices

Appendix A: Normalization Table

UNF	1NF	2NF	3NF
The unnormalized form lists all the data attributes that the database may contain: **Attendee**: AttendeeID, Name, Email, TicketType, PerformanceID, PerformerName, StageLocation, PerformanceTime, Location, Capacity, CampingType, CheckInDate, CheckOutDate **Ticket**: TicketID, AttendeeID, PerformanceID, TicketType **Performance**: PerformanceID, PerformerName, StageLocation, PerformanceTime, Genre, Location, Capacity **Performer**: PerformerID, Name, Genre, PerformanceID, StageLocation, PerformanceTime **Stage**: StageID, Location, Capacity, PerformanceID, PerformerName, EntertainmentID,	1NF removes repeating groups by separating attributes into distinct columns, assigning each row a unique primary key. Key attributes are identified: **Attendee**: • Attributes: AttendeeID (PK), Name, Email • Notes: Each attendee is uniquely identified by the AttendeeID. **Ticket**: • Attributes: TicketID (PK), AttendeeID (FK to Attendee), PerformanceID (FK to Performance), TicketType • Notes: TicketType specifies general admission, VIP, or other types. **Performance**: • Attributes: PerformanceID (PK), PerformerI	In 2NF, all attributes are fully functionally dependent on the primary key, eliminating partial dependencies: **Attendee**: • Attributes: AttendeeID (PK), Name, Email **Ticket**: • Attributes: TicketID (PK), AttendeeID (FK to Attendee), PerformanceID (FK to Performance), TicketType **Performance**: • Attributes: PerformanceID (PK), PerformerID (FK to Performer), StageID (FK to Stage), Date, Time • Notes: Date and Time depend on the unique PerformanceID. **Performer**: • Attributes: PerformerID (PK), Name, Genre	In 3NF, each non-key attribute depends only on the primary key, removing any transitive dependencies: **Attendee**: • Attributes: AttendeeID (PK), Name, Email **Ticket**: • Attributes: TicketID (PK), AttendeeID (FK), PerformanceID (FK), TicketType • Notes: TicketType is fully dependent on TicketID, ensuring data inte **Performance**: • Attributes: PerformanceID (PK), PerformerID (FK), StageID (FK), Date, Time **Performer**: • Attributes: PerformerID (PK), Name, Genre • Notes: Name and Genre depend

EntertainmentLocation **Camping Reservation**: ReservationID, AttendeeID, CampingType, Price, CheckInDate, CheckOutDate **Non-Musical Entertainment**: EntertainmentID, Name, Description, Date, Time, Location **Festival Schedule**: ScheduleID, EventID (could represent either a Performance or Entertainment), StageID or Location, Date, Time	D, StageID, Date, Time • Notes: Each performance has a unique PerformanceID. **Stage**: • Attributes: StageID (PK), Location, Capacity • Notes: Each stage is identified by a unique StageID. **Camping Reservation**: • Attributes: Reservation ID (PK), AttendeeID (FK to Attendee), CampingType, Price, CheckInDate, CheckOutDate **Non-Musical Entertainment**: • Attributes: EntertainmentID (PK), Name, Description, Date, Time, Location **Festival Schedule**: • Attributes: ScheduleID (PK), EventID, EventType, StageID or LocationID, Date, Time	**Stage**: • Attributes: StageID (PK), Location, Capacity **Camping Reservation**: • Attributes: Reservation ID (PK), AttendeeID (FK to Attendee), CampingType, Price, CheckInDate, CheckOutDate **Non-Musical Entertainment**: • Attributes: EntertainmentID (PK), Name, Description, Date, Time, Location • Notes: Location details are now fully dependent on EntertainmentID. **Festival Schedule**: • Attributes: ScheduleID (PK), EventID (FK to Performance or Entertainment), EventType, StageID or LocationID, Date, Time	solely on PerformerID. **Stage**: • Attributes: StageID (PK), Location, Capacity **Camping Reservation**: • Attributes: Reservation ID (PK), AttendeeID (FK), CampingType, Price, CheckInDate, CheckOutDate • Notes: Each attribute is fully dependent on Reservation ID. **Non-Musical Entertainment**: • Attributes: EntertainmentID (PK), Name, Description, Date, Time, Location • Notes: Each attribute depends only on EntertainmentID, preventing redundant location data. **Festival Schedule**: • Attributes: ScheduleID (PK), EventID (FK),

			EventType, StageID or LocationID, Date, Time • Notes: All attributes depend on ScheduleID, and EventID links either a Performance or Entertainment event.

7.2 Summary of Relationships with Cardinality and Optionality Table

Relationship	Cardinality	Optionality (Attendee / Ticket End)	Explanation
Attendee - Ticket	1	Optional / Mandatory	An Attendee can purchase multiple Tickets, but not all Attendees buy tickets. Each Ticket belongs to one Attendee.
Ticket - Performance	M:1	Mandatory / Mandatory	Each Ticket is tied to one Performance; a Performance can have many Tickets sold for it.
Performance - Performer	M:1	Mandatory / Mandatory	Each Performance has at least one Performer; a Performer can participate in multiple Performances.

Performance - Stage	M:1	Mandatory / Mandatory	Each Performance takes place at one Stage; a Stage can host many Performances.
Attendee - Camping Reservation	1	Optional / Optional	An Attendee may reserve multiple Camping Reservations, but Camping Reservations are optional for the Attendee.
Non-Musical Entertainment - Stage	M:1	Mandatory / Mandatory	Each Non-Musical Entertainment event occurs at a Stage; a Stage can host multiple Non-Musical Entertainment events.
Festival Schedule - Performance	M:1	Mandatory / Mandatory	Every Festival Schedule entry lists a Performance with specific timing and location, and each Performance must appear in the Festival Schedule.

Festival Schedule - Non-Musical Entertainment	M:1	Mandatory / Mandatory	Every Festival Schedule entry lists a Non-Musical Entertainment event, and each Non-Musical Entertainment must be scheduled in the Festival Schedule.