Elegant Occasions with Alyssa – A Wedding-Planning Database

Abstract

The aim of this document is to formally present the environment and constraints around the database which supports an Asheville-based wedding-planning business, Elegant Occasions with Alyssa. The development of this database was completed using https://dbdiagram.io (for ERD design), Visual Studio Code, an azure based MySQL server through John Locklear's DBA120 class at AB-Tech, and the command line. The result is a functional database that is production-ready and equipped for connection to a user application interface.

Introduction

The goal of this project is the development of a wedding-planning tool, a database, to keep track of wedding details. The business it is centered around, "Elegant Occasions with Alyssa," specializes in organizing and coordinating weddings for clients. This agency offers a range of services, including venue selection, catering, floral arrangements, photography, and entertainment. The objective of this project is to create and document a database system to manage client appointments and wedding details efficiently.

Business Requirements

The database itself is designed around several business requirements provided by business-owner, Alyssa:

1. Clients cannot have overlapping appointments.
   This requirement is to be enforced via the user application when scheduling with their wedding-planning app. The application would query for existing appointments and check for overlaps using date/time comparison operators in regards to the **APPOINTMENTS** table's **start_time** and **end_time**(s). The application would only insert the new appointment if no overlaps exist.

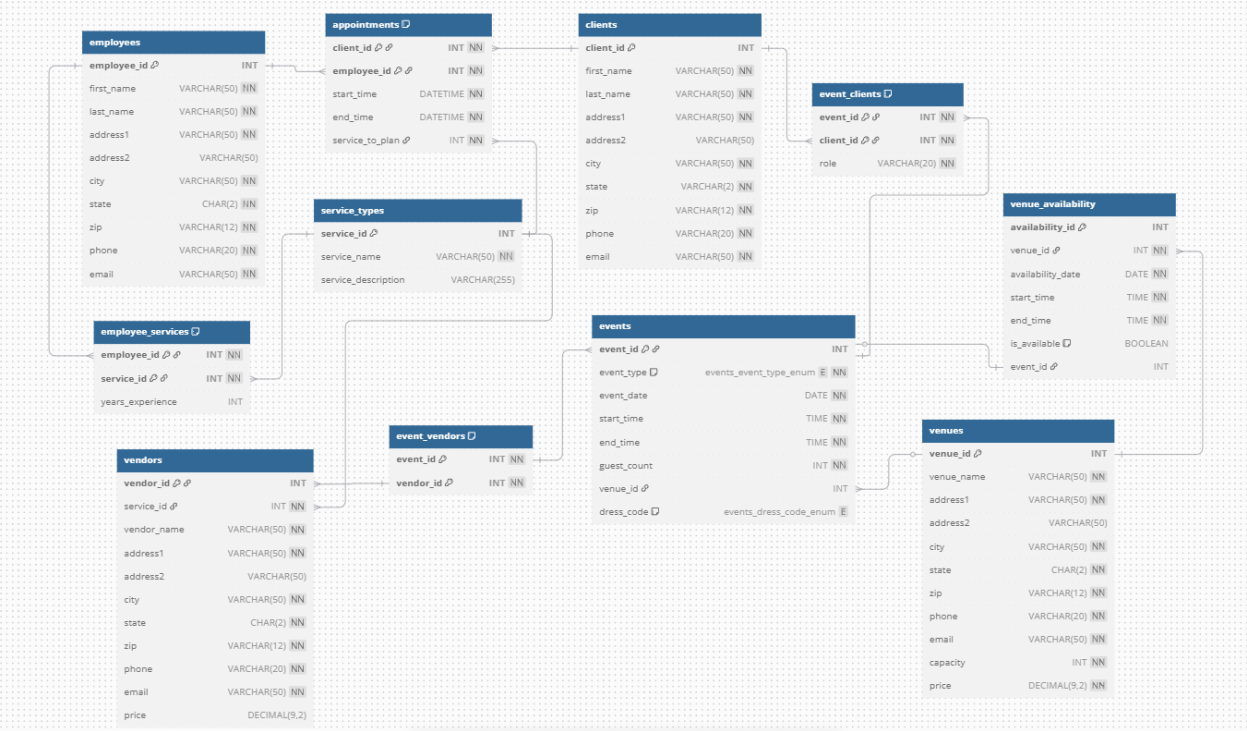
2. Each appointment must have at least one service associated with it.
   The **APPOINTMENTS** table requires that a primary service be selected when making an appointment. The primary service is selected within the clients' scheduling

application; a drop-down list is provided (and required) for making appointments. In the database creation script, the **service_to_plan** column is designated as NOT NULL when creating the **APPOINTMENTS** table.

3. An appointment cannot be scheduled without specifying an employee.
   This requirement is handled in the users' scheduling application; from a database creation standpoint, an **employee_id** is one part of a composite primary key for the **APPOINTMENTS** table, so an appointment simply cannot exist without specifying an employee.

4. Venues should have available capacity for the expected number of guests.
   The ***set_venue_id()*** procedure checks against a scenario in which the guest count for an event would exceed the capacity of a given venue. This prevents overbooking issues when clients choose their own venue through the user's wedding-planning application.

ERD



The above entity relationship diagram was designed with https://dbdiagram.io based on the following conditions:

1. Clients can have multiple appointments but only one wedding.

The **CLIENTS** table has a one-to-many relationship with the **APPOINTMENTS** table; clients are able to have as many appointments as desired. The **CLIENTS** table has a many-to-many relationship with the **EVENTS** table, allowed by a junction table of **EVENT_CLIENTS**. The UNIQUE constraint prevents clients from having the multiple roles.

2. Appointments are scheduled with clients and may involve multiple services.
   An appointment has a composite primary key of **client_id** and **employee_id**, ensuring that there are both a wedding planner (employee) and a client present for the appointment.
   The many-to-many relationship between the **APPOINTMENTS service_to_plan** and **SERVICE_TYPES service_name** fields indicate that multiple appointments may be used to determine multiple services provided by vendors (and the SERVICE_TYPES junction table allows for the possibility of an employee also having a role as a vendor via service_id).

3. Services are offered by employees and can be associated with multiple appointments.
   Employees are trained in planning for multiple services. The main service the appointment is themed around will be matched with an employee who is trained in that service. The user application for creating appointments allows clients to choose either a service or an employee first, then provides the available options for the other based on the client's selection.
   Clients are able to have as many appointments as desired to decide upon a service vendor(s), as is demonstrated by the many-to-many relationship between the **APPOINTMENTS** table's **service_to_plan** field and the **SERVICE_TYPES service_name** field. This relationship demonstrates that an appointment can be used to plan for multiple services.
   The one-to-many relationship between the **EMPLOYEES** table and **EMPLOYEE_SERVICES** table illustrates that an employee (one of our wedding-planners) is able to plan multiple services with the client at a time, per the client's wishes, the employee's specialties, and the allowed-for time within the appointment.

4. Venues are where weddings take place and can host multiple weddings.
   The one-to-many relationship between **VENUE's venue_id** and **EVENTS venue_id** indicates that a venue can host multiple weddings.
   The **VENUE_AVAILABILITY** table tracks whether a specified venue (**venue_id**) is available during a given **start_time** and **end_time**. The **VENUE_AVAILABILITY** table

can be queried directly via the ***set_venue_id()*** procedure to determine availability, and if **is_available** returns true, a client can book the venue. Once booked, **is_available** would be set to FALSE and the **event_id** would be set to match the corresponding rehearsal dinner, wedding, or reception.

Table Definitions

1. **employees**

   Stores information about employees of the wedding planning business, Elegant Occasions with Alyssa

Field	**Type**	**Constraints**	**Description**
employee_id	INT	PRIMARY_KEY, AUTO_INCREMENT	Unique identifier for each employee
first_name	VARCHAR(50)	NOT NULL	Employee's first name
last_name	VARCHAR(50)	NOT NULL	Employee's last name
address_1	VARCHAR(50)	NOT NULL	Primary address line
address_2	VARCHAR(50)		Secondary address line
city	VARCHAR(50)	NOT NULL	City
state	CHAR(2)	NOT NULL	State (2-letter code)
zip	VARCHAR(12)	NOT NULL	ZIP/Postal Code
phone	VARCHAR(20)	NOT NULL	Contact phone number
email	VARCHAR(50)	NOT NULL, UNIQUE	Email address (must be unique from other employees)

2. **service_types**

   Defines the types of services offered by the wedding planning business

Field	**Type**	**Constraints**	**Description**
service_id	INT	PRIMARY_KEY, AUTO_INCREMENT	Unique identifier for

			each service type
service_name	VARCHAR(50)	NOT NULL, UNIQUE	Name of the service (e.g., photography, catering)
service_description	VARHAR(255)		Detailed description of the service

3. **employee_services**

Junction table linking employees to the services they can provide, including their experience level.

Field	**Type**	**Constraints**	**Description**
employee_id	INT	PRIMARY KEY, FOREIGN KEY, NOT NULL	References employee_id of employees table
service_id	INT	PRIMARY KEY, FOREIGN KEY, NOT NULL	References service_id of service_types table
years_experience	INT		Number of years of experience in this service

4. **clients**

Stores information about clients using the wedding planning services.

Field	**Type**	**Constraints**	**Description**
client_id	INT	PRIMARY_KEY, AUTO_INCREMENT	Unique identifier for each client
first_name	VARCHAR(50)	NOT NULL	Client's first name
last_name	VARCHAR(50)	NOT NULL	Client's last name
address_1	VARCHAR(50)	NOT NULL	Primary address line
address_2	VARCHAR(50)		Secondary address line
city	VARCHAR(50)	NOT NULL	City
state	CHAR(2)	NOT NULL	State (2-letter code)
zip	VARCHAR(12)	NOT NULL	ZIP/Postal Code

phone	VARCHAR(20)	NOT NULL	Contact phone number
email	VARCHAR(50)	NOT NULL, UNIQUE	Email address (must be unique from other clients)

5. **appointments**

Tracks meetings between clients and employees planning specific services

Field	**Type**	**Constraints**	**Description**
client_id	INT	PRIMARY KEY, NOT NULL, FOREIGN KEY	References the client_id of the clients table
employee_id	INT	PRIMARY KEY, FOREIGN KEY NOT NULL	References the employee_id of the employees table
start_time	DATETIME	NOT NULL	Start time of the appointment
end_time	DATETIME	NOT NULL	End time of the appointment
service_to_plan	INT	NOT NULL	References service_id of the service_types table. The selected service to plan for during the appointment.

6. **vendors**

Stores information about external service providers (photographers, caterers, etc.).

Field	**Type**	**Constraints**	**Description**
vendor_id	INT	PRIMARY KEY, AUTO_INCREMENT	A unique identifier for each vendor
service_id	INT	NOT NULL, FOREIGN KEY	References the service_id of the service_types table. The service provided by the vendor.
vendor_name	VARCHAR(50)	NOT NULL, UNIQUE	Vendor's last name

address_1	VARCHAR(50)	NOT NULL	Primary address line
address_2	VARCHAR(50)		Secondary address line
city	VARCHAR(50)	NOT NULL	City
state	CHAR(2)	NOT NULL	State (2-letter code)
zip	VARCHAR(12)	NOT NULL	ZIP/Postal Code
phone	VARCHAR(20)	NOT NULL	Contact phone number
email	VARCHAR(50)	NOT NULL, UNIQUE	Email address (must be unique from other clients)
price	DECIMAL (9,2)		The price the vendor requires for his/her services

7. **venues**

Stores information about locations where events can be held.

Field	**Type**	**Constraints**	**Description**
venue_id	INT	PRIMARY KEY, AUTO_INCREMENT	A unique identifier for each venue
venue_name	VARCHAR(50)	NOT NULL, UNIQUE	Name of the venue
address_1	VARCHAR(50)	NOT NULL	Primary address line
address_2	VARCHAR(50)		Secondary address line
city	VARCHAR(50)	NOT NULL	City
state	CHAR(2)	NOT NULL	State (2-letter code)
zip	VARCHAR(12)	NOT NULL	ZIP/Postal Code
phone	VARCHAR(20)	NOT NULL	Contact phone number
email	VARCHAR(50)	NOT NULL, UNIQUE	Email address (must be unique from other clients)
capacity	INT	NOT NULL	Maximum number of guests the venue can accommodate

price	DECIMAL (9,2)		The price the vendor requires for his/her services

8. **events**

Central table tracking all events (weddings, receptions, dress rehearsals).

Field	Type	Constraints	Description
event_id	INT	PRIMARY KEY, AUTO_INCREMENT	A unique identifier for each event
event_type	ENUM	NOT NULL	Type of event (‘wedding’, ‘reception’, ‘dress rehearsal’)
event_date	DATE	NOT NULL	Date of the event
start_time	TIME	NOT NULL	Start time of the event
end_time	TIME	NOT NULL	End time of the event
guest_count	INT	NOT NULL	Expected number of guests
venue_id	INT	FOREIGN KEY	References venue_id of the venues table, specifies the location of the event
dress_code	ENUM		Specifies the dress code for the event ('white tie', 'black tie', 'black tie optional', 'semi-formal', 'casual')

9. **event_clients**

Junction table linking clients to events, specifying their role in the event.

Field	Type	Constraints	Description
event_id	INT	PRIMARY KEY, FOREIGN KEY, NOT NULL	References the event_id of the events table

client_id	INT	PRIMARY KEY, FOREIGN KEY, NOT NULL, UNIQUE	References the client_id of the clients table
role	VARCHAR(20)	NOT NULL, UNIQUE	Client's role in the event (e.g., 'bride', 'groom', 'partner', 'parent')

10. **event_vendors**

Junction table linking vendors to events they are providing services for.

Field	Type	Constraints	Description
event_id	INT	PRIMARY KEY, FOREIGN KEY, NOT NULL	References the event_id of the events table
vendor_id	INT	PRIMARY KEY, FOREIGN KEY, NOT NULL	References the vendor_id of the vendors table

11. **venue_availability**

Tracks when venues are available or booked.

Field	Type	Constraints	Description
availability_id	INT	PRIMARY KEY, AUTO_INCREMENT	Unique identifier for each booking event
venue_id	INT	FOREIGN KEY, NOT NULL	References the venue_id of the venues table
availability_date	DATE	NOT NULL	Date for availability
start_time	TIME	NOT NULL	Start time of availability window
end_time	TIME	NOT NULL	End time of availability window
is_available	BOOLEAN	DEFAULT TRUE	Whether the venue is available during this time

event_id	INT	FOREIGN KEY, UNIQUE	References the event_id of the events table

Key Constraints

Primary Keys

- Each table has an appropriate primary key, either a single field or a composite key
- Composite primary keys are used in junction tables like **EVENT_CLIENTS**and **EVENT_VENDORS**

Foreign Keys

- Proper referential integrity is maintained through foreign key constraints
- All relationships are properly defined between tables

Unique Constraints

- Email addresses must be unique in both **CLIENTS** and **EMPLOYEES** tables
- **Vendor** and **venue names** must be unique
- **event_id** in **VENUE_AVAILABILITY** ensures a venue slot can only be booked for one event

Business Rules

- The **EVENT_CLIENTS** table can be configured with a UNIQUE constraint on (client_id, role) to enforce that a client can only have one role of a specific type (e.g., only be a "bride" once)
- A limitation of the above rule is that a bride cannot be a “bride” at both a wedding and reception. A trigger or CHECK constraint may be a better way to implement this rule.

Data Flow

1. **Client Registration**: New clients are added to the clients table
2. **Appointment Scheduling**: Appointments are scheduled between clients and employees
3. **Event Creation**: Events (weddings, receptions) are created in the events table

4. **Client Role Assignment**: Clients are assigned roles in events via the event_clients table
5. **Venue Booking**: Venues are checked for availability and booked for events
6. **Vendor Selection**: Vendors are selected for events based on the services needed

Data Normalization

Every non-key column depends ONLY on the primary key(s). This ensures that the database is maintained to the industry standard of 3rd Normal Form.

Conclusion

The Elegant Occasions with Alyssa database provides a robust foundation for managing all aspects of the wedding planning business. This carefully designed relational database system ensures data integrity while supporting the complex relationships between clients, events, venues, vendors, and services.

This database design utilizes both normalization principles and practical business needs, resulting in a system that is operationally effective. The clear table structure with appropriate constraints ensures referential data integrity while still providing the flexibility needed to accommodate the unique nature of wedding planning services.

As the business grows, this database can be extended with additional features such as payment tracking, communication logs, or inventory management while maintaining its solid foundational structure. The documentation provided here serves as a reference and a guide to the business logic embedded in the database design.