Semesterprojekt Datenbanken:

Modellierung eines Ticketsystems

Name: Emil Schläger

Kurs: INF23B

Contents

List of Figures

1 Anforderungsanalyse

Es soll eine Ticketplattform entstehen, die es Veranstaltern und Künstlern ermöglicht, Events zu planen und Tickets an Kunden zu verkaufen.
Um die Plattform nutzen zu können, muss man sich als User anmelden. Für eine Konto müssen der volle Name, eine Email-Adresse, ein Passwort und eine Kontonummer hinterlegt werden. Angabe einer Telefonnummer für Zwei-Faktor Authentifizierung ist optional.
Als User kann man nun Tickets zu Events kaufen. Jedes Ticket besitzt einen Preis, sowie eine Kategorisierung, ob es ein Sitz- oder Stehplatzticket ist. Ist es ein Sitzplatzticket, muss noch die Sitznummer hinterlegt werden. Dabei darf für ein Event selbstverständlich nicht mehrfach derselbe Sitz gebucht werden. Für das jeweilige Event muss gespeichert werden, an welchem Tag es stattfindet, sowie um welche Uhrzeit es anfängt. Es kann auch hinterlegt werden, um wie viel Uhr Einlass ist. Auch findet jedes Event in genau einer Venue statt. Für diese ist jeweils ihr Name, ihre Adresse und ihre Kapazität hinterlegt. Die Kapazität ist hierbei aufgeteilt in die Anzahl an möglichen Steh- und möglichen Sitzplätzen.
Ein Event ist immer ein Teil einer Tour. Jede Tour ist von einem Veranstalter organisiert, und dementsprechend auch auf der Plattform verwaltet. Z.B. können einzelne Events in größere Venues verlegt werden, wenn das Event ausverkauft ist, aber noch Nachfrage existiert. Zu jeder Tour muss nur der Name gespeichert werden; der Zeitrahmen der Tour errechnet sich durch die Daten des frühesten und spätesten Events der Tour.
Auf einer Tour spielt immer mindestens ein Künstler. Oft bringt der Künstler aber Vorbands mit. In diesem Fall ist wichtig zu speichern, in welcher Reihenfolge die Künstler in der Tour spielen. Der Hauptkünstler spielt dabei stets als letztes. Jeder Künstler hat einen Namen und gehört einem Genre an. Ein Künstler spielt auf jedem Event einer Tour immer dieselbe Setlist. Die Plattform gibt Künstlern die Möglichkeit, diese Setlist auf der Plattform für ihre Fans zu veröffentlichen. Eine Setlist besteht aus mehreren Songs, die wiederum Teil eines Albums sind. Auch diese sind für User der Plattform auf den Profilen der Künstler einzusehen. Ein Album wird immer von einem Label veröffentlicht, auch Künstler sind immer bei einem Plattenlabel unter Vertrag. Für jedes Label muss sein Name, sowie die Adresse seines Hauptsitzes gespeichert werden. Für den Vertrag zwischen Label und Künstler muss auch ein Ablaufdatum gespeichert werden.
Die Plattform soll auch die Finanzen einer Tour verwalten können, da die Bezahlung der

Tickets bei der Plattform landet. Organisiert also ein Veranstalter eine Tour für einen Künstler, entsteht dabei ein weiterer Vertrag, der vereinbart, wie viel Prozent der Umsätze der Tour an den Veranstalter gehen. Der Umsatz einer Tour errechnet sich ganz einfach aus der Summe der Preise aller verkauften Tickets. Auch die Plattenlabel verdienen an den Tickets mit. Dieser Festbetrag errechnet sich wie folgt: Bei Veröffentlichung eines Albums A über Label L wird ein Betrag festgelegt, der an das Label gezahlt wird, jedes mal wenn der Künstler einen Song $s \in A$ spielt. Am Ende einer Tour kann also mithilfe der Setlist der Künstler und der Anzahl an Events ausgerechnet werden, wie viel an das Label gezahlt werden muss.

Da Künstler, Veranstalter und Label jeweils Zugang auf die Plattform haben müssen, um die für sie relevanten Daten zu verwalten, muss jedem indivuduellen Künstler, Veranstalter oder label ein User Account zugeordnet werden. Damit die Plattform entscheiden kann, welche Zugriffsrechte ein einzelner User hat, wird jedem Account eine entsprechende Rolle zugeordnet.

2 Konzeptioneller Entwurf

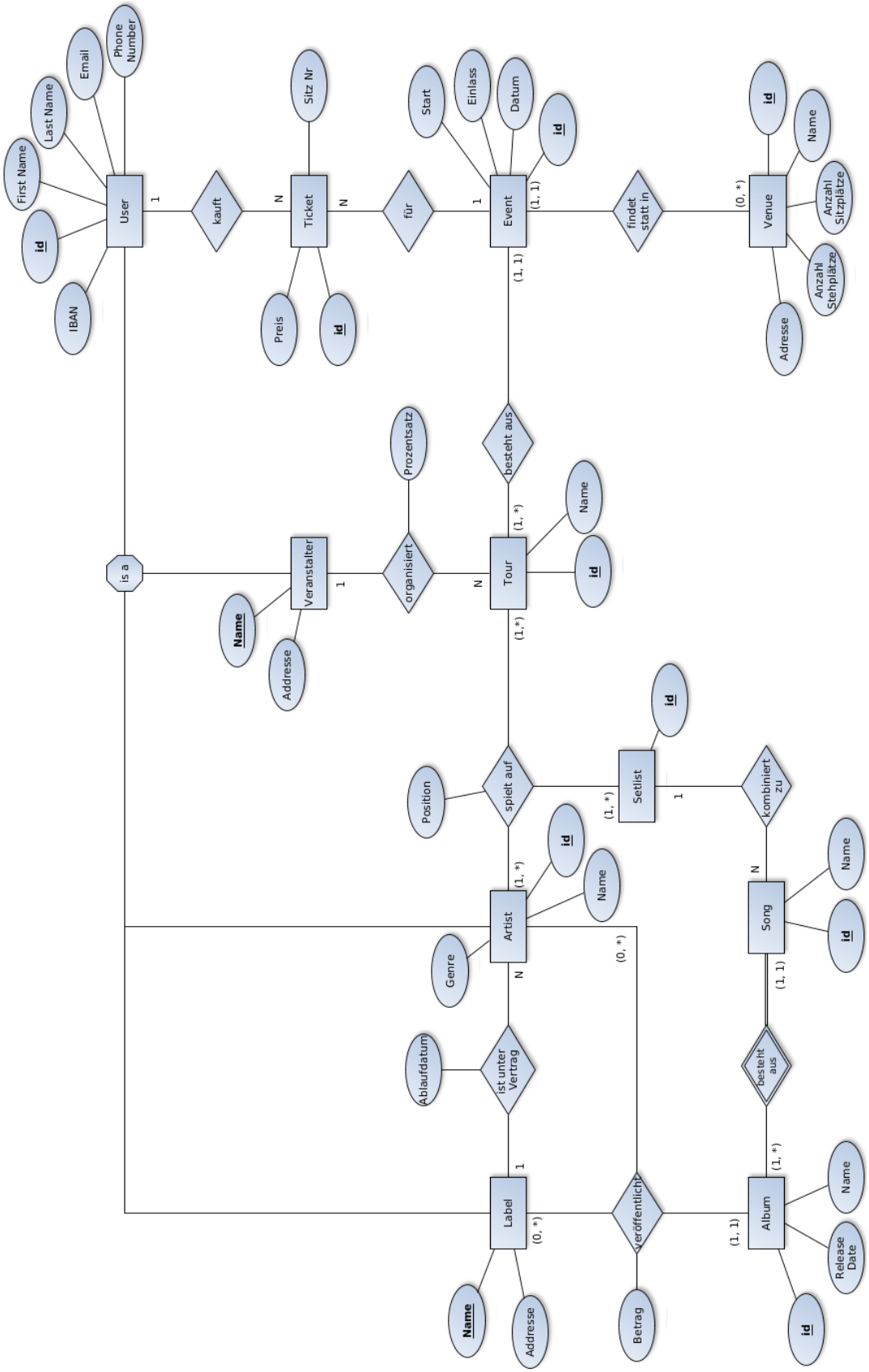

Figure 1: Entity Relationship Model

3 Logischer Entwurf

3.1 Relationales Modell

User: {[$\underline{User\_ID}$, $First\_Name$, $Last\_Name$, $Email$, $Phone\_Number$]}

Ticket: {[$\underline{Ticket\_Nr}$, $Price$, $Seat\_Nr$, $\underline{User\_ID}$, $\underline{Event\_ID}$]}

Event: {[$\underline{Event\_ID}$, $Begin$, $Doors$, $Date$, $\underline{Venue\_ID}$, $\underline{Tour\_ID}$]}

Venue: {[$\underline{Venue\_ID}$, $Name$, $\underline{Address}$, $Capacity\_Standing$, $Capacity\_Seated$]}

Tour: {[$\underline{Tour\_ID}$, $Name$, $\underline{Organizer\_ID}$, $Percentage$]}

Artist: {[$\underline{Artist\_ID}$, $Name$, $Genre$, $\underline{Label\_Name}$, $Contract\_End$, $\underline{User\_ID}$]}

Setlist: {[$\underline{Setlist\_ID}$]}

Plays_In: {[$\underline{Artist\_ID}$, $\underline{Tour\_ID}$, $\underline{Setlist\_ID}$, $Position$]}

Song: {[$\underline{Song\_ID}$, $\underline{Album\_ID}$, $Name$, $\underline{Setlist\_ID}$]}

Album: {[$\underline{Album\_ID}$, $Name$, $Released$, $\underline{Label\_Name}$]}

Label: {[$\underline{Name}$, $\underline{Address}$, $\underline{User\_ID}$]}

Releases: {[$\underline{Artist\_ID}$, $\underline{Album\_ID}$, $\underline{Label\_Name}$, $Label\_Fee$]}

Organizer: {[$\underline{Name}$, $\underline{Address}$, $\underline{User\_ID}$]}

Address: {[$\underline{Address\_ID}$, $Country$, $City$, $Postal\_Code$, $Street$, $House\_Number$]}

3.2 Exemplarisches Prüfen auf 3NF

Hier wird am Beispiel der Relation *Venue* gezeigt, dass diese in der 3. Normalform ist. Hierfür müssen zunächst 1NF und 2NF geprüft werden:

- 1NF: Alle Attribute sind so konzipiert, dass sie nur einen wert enthalten, also atomar sind. Dadurch ist die Relation in 1NF.
- 2NF: Der Primärschlüssel der Relation, $\{Venue\_ID\}$, enthält nur ein Element. Dadurch kann es keine funktionalen Abhängigkeiten von einem Teil des Schlüssels geben. Dadurch ist die Relation in 2NF.

Um zu prüfen, ob die Relation in 3NF ist, müssen zuerst alle existierenden funktionalen Abhängigkeiten innerhalb der Relation bestimmt werden:

$$F = \begin{cases} Venue\_ID \rightarrow [Venue] \\ Name,\ Address \rightarrow [Venue] \end{cases}$$

Dadurch ergeben sich zwei Schlüsselkandidaten: $\{Venue\_ID\}$ und $\{Name,\ Address\}$. Anhand der folgenden Tabelle werden alle funktionalen Abhängigkeiten der Form

$$X \rightarrow \alpha$$

untersucht:

X	α	X ist Superschlüssel	α ist prim
Venue_ID	Name	✓	X
	Address	✓	✓
	Name	✓	✓
	Capacity_Seated	✓	X
	Capacity_Standing	✓	X
Name, Address	Venue_ID	✓	✓
	Capacity_Seated	✓	X
	Capacity_Standing	✓	X

Table 1: 3NF Prüfung für Ticket

Da für jede Funktionale Abhhängigkeit $f \in F$ gilt, dass die Domäne von f ein Superschlüssel ist, oder die Bildmenge ein Primattribut, ist die Relation in 3NF.

4 Datenbankschema

Korrekturen, die für diesen Teil an der Modellierung vorgenommen wurden:

- Beziehung "Song kombiniert zu Setlist" wurde von einer 1-N Beziehung in eine N-M Beziehung transformiert. Dies ist ein Ergebnis dessen, dass Löschregeln sonst semantisch nicht sinnvol definiert werden könnten.

4.1 Keine Cascade Zyklen

Hier wird gezeigt, dass es keine Zyklen aus Löschregeln gibt, die zu ungewolltem Datenverlust führen könnten. Hierfür wird in einer Tiefensuche jeder `on delete cascade` foreign key constraints überprüft, was passieren würde, wenn er ausgelöst würde. Die benutzte Darstellung ist wie folgt zu lesen: "Tabelle, in der ursprünglich gelöscht wird" → "Tabelle, in der dies zu einer Cascade führt".

1. `event` → `ticket`: `ticket` wird von keiner weiteren Tabelle referenziert.
2. `artist, tour, setlist` → `plays_in`: `plays_in` wird von keiner weiteren Tabelle referenziert.
3. `setlist, song` →`part_of`: `part_of` wird von keiner weiteren Tabele referenziert.
4. `album` → `song`: Löschen in `song` löst ein Löschen in `part_of` auf, welches von keiner weiteren Tabelle referenziert wird.
5. `artist, album, music_label` → `releases`: `releases` wird von keiner weiteren Tabelle referenziert.

Somit existiert nirgends ein Zyklus aus Cascades, ungewolltes Verhalten kann also in dieser Hinsicht ausgeschlossen werden.