Friedrich-Schiller-Universität Jena
Wirtschaftswissenschaftliche Fakultät
Lehrstuhl für Wirtschaftsinformatik
Prof. Dr. J.Ruhland

Projekt III: Verbesserung des Datenmanagements der EVENTUS GmbH

-

Projektbericht

Eingereicht von:

Olha Omelianchuk	Marcel Neff	Bohdan Biliavskyi	Paul Passek

Jena, den 31.01.2017

Inhaltsverzeichn

I AbbildungsverzeichnisY

II Abkürzungsverzeichnis

DBMS – Datenbankmanagementsystem

RDBMS – Relationales Datenbankmanagementsystem

DB – Datenbank

DML – Data Manipulation Language

DDL – Data Definition Language

rDB – relationale Datenbank

1 Inhaltstext

Kleine und mittelständische Unternehmen stehen mit zunehmender Größe und Komplexität der Geschäftsprozesse früher oder später vor der Entscheidung ein effizientes und konsistentes Datenmanagement zu etablieren.

Am fiktiven Beispiel der EVENTUS GmbH, ein Veranstaltungsdienstleister mit Spezialisierung auf Organisation von Kongressen, Konzerten und Events, wird gezeigt wie ein beispielhafter DB-Entwurf aussehen könnte. Das Beispielunternehmen verwendet aktuell eine Excel-Datei für die Verwaltung sämtlich relevanter Daten. Durch zunehmende Veranstaltungsaufträge ist die Verwaltung der Veranstaltungsinformationen unübersichtlich und problematisch geworden. Daher strebt das Unternehmen im Rahmen eines effizienteren Datenmanagements die Einführung eines DBMS an.

Dieses Projekt beschäftigt sich basierend auf dieser Problemstellung mit der Datenstrukturierung, der Datenbankentwicklung, der Datenabfrage, der Rechte- und Rollenverwaltung und der XML-Schemaentwicklung.
Dabei soll zunächst die Analyse und Strukturierung der vorliegenden Daten erfolgen, woraufhin den späteren Attributen der DB Datentypen zugeordnet werden. Im Anschluss wird ein relationales Schema durch festlegen von Primär- und Fremdschlüsseleigenschaften gebildet. Das Ziel soll darin bestehen die Daten so effizient wie möglich in Tabellen der späteren DB zu organisieren. Deshalb wird versucht, durch Normalisierungsvorgänge, einen optimalen Trade-Off zwischen Effizienz der Datenhaltung und Abfrageeffizienz und -geschwindigkeit zu erreichen.
Als nächstes folgt die Implementierung der zuvor spezifizierten Tabellen im DBMS mittels DDL-Befehlen. Diese werden vereinzelt noch durch Trigger unterstützt, welche der DB helfen sollen konsistente Daten in den Tabellen zu halten. Im Anschluss an die Tabellenerstellung kann das Grundgerüst mit Daten gefüllt werden.
Daraufhin wird der DB eine Rechte- und Rollenverwaltung aufgesetzt, welche es ermöglichen soll, dass nicht alle Mitarbeiter alle Daten lesen und oder verändern können. Damit ausgestattet, werden der Datenbank Standardabfragen hinterlegt, die auch an die Rechte- und Rollenzuweisung gebunden sind.

Es soll weiterhin möglich sein Veranstaltungsprogramme in einzelnen Zellen der DB hinterlegen zu können. Dafür wird ein XML-Schema vorbereitet, welches als Validierung-Schema für spätere XML-Instanzen genutzt werden kann. Denkbar ist ein Einsatz von XML-Dokumenten für den Bereich Onlinemarketing oder Flyer Gestaltung.

Als zusätzliche Funktion wird der EVENTUS GmbH ein Programm zur Verfügung gestellt, welche das Anlegen, Ändern und Löschen von Datensätzen vereinfachen und übersichtlich machen soll. Dabei kommuniziert das Programm mit dem SQL-Server und soll die Daten nutzerfreundlich integrieren, damit der Umgang mit dem System vereinfacht wird.

2 Datenanalyse und Datenstrukturierung in Tabellenform

Um ein Datenbankschema zu entwickeln, welches die verfügbaren Daten so gut wie möglich darstellt, bedarf es zuerst einer genauen Analyse der Ausgangsdaten. Im Falle der EVENTUS GmbH liegt eine Excel-Datei vor, die sämtliche Informationen über Veranstaltungen, Showacts, Lieferanten, Gäste, Mitarbeiter, Kosten und Erlöse, aufgeteilt auf vier Arbeitsblätter, enthält. In einem ersten Schritt sollen alle erfassten Datensätze gemäß ihres Informations – und Aussagegehalts in verschiedene Gruppen unterteilt werden, die später die Tabellen innerhalb der Datenbank darstellen. Dabei orientiert sich die Grobgliederung der Daten an der bisherigen Gliederung gemäß der Excel-Datei.

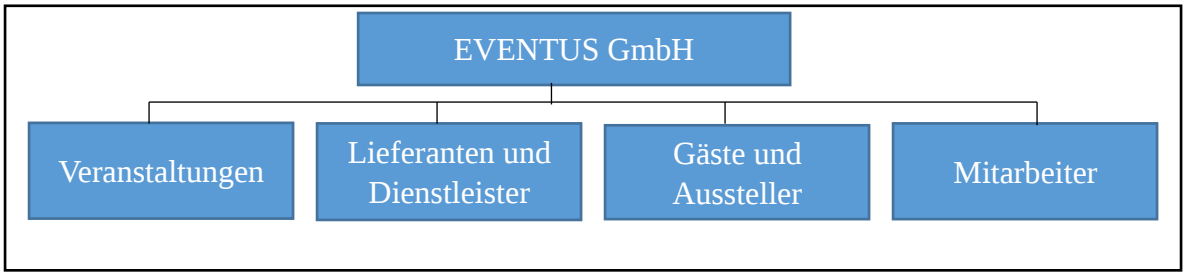

Abbildung 1: Grobgliederung der Daten

Innerhalb dieser Grobgliederung werden den vorliegenden Daten nun Datentypen zugewiesen, die die Attribute der späteren DB spezifizieren. Darüber hinaus werden Primär- und Fremdschlüsseleigenschaften festgelegt, damit Beziehungen zwischen verschiedenen Tabellen hergestellt werden können. Im Folgenden ist ein Ausschnitt aus der Datentyp-Definition für die Veranstaltungsstammdaten dargestellt. Dabei stellen unterstrichene und normal gedruckte Elemente Primärschlüssel-Eigenschaften dar und unterstrichene kursiv gedruckte stellen *Fremdschlüsssel-Eigenschaften* dar. Für eine gesamte Übersicht über alle Datentypen und Schlüsselbeziehungen siehe **Kapitel 3.1**.

Attribut	Datentyp
Veranstaltungs-ID	INTEGER
Veranstaltungsname	VARCHAR(100)
Veranstaltungsinformationen	VARCHAR(100)
Veranstaltungsstartdatum	DATE
Veranstaltungsauftraggeber-ID	INTEGER

Abbildung 2: Ausschnitt Datentyp-Definition

Nachdem für die Zuordnung spezifizierender Datentypen abgesch
begonnen werden, die Attribute in einer Tabellenstruktur zu organisieren
Abbildung 3 zeigt alle Tabellen des Grobgliederungselements Ver

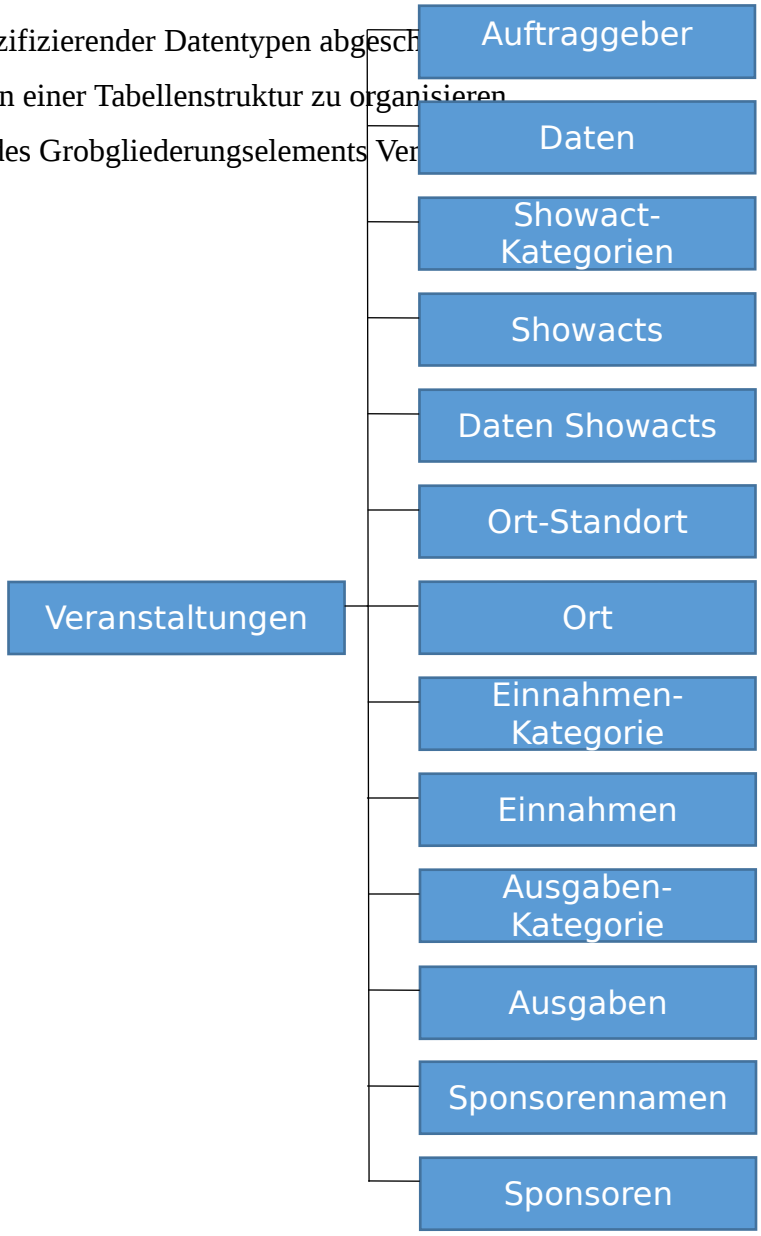

Abbildung 3: Datenbanktabellen des Grobgliederungselements Veranstaltungen

Die Beziehungen der einzelnen Tabellen zueinander stellt Abbildung 4 dar.

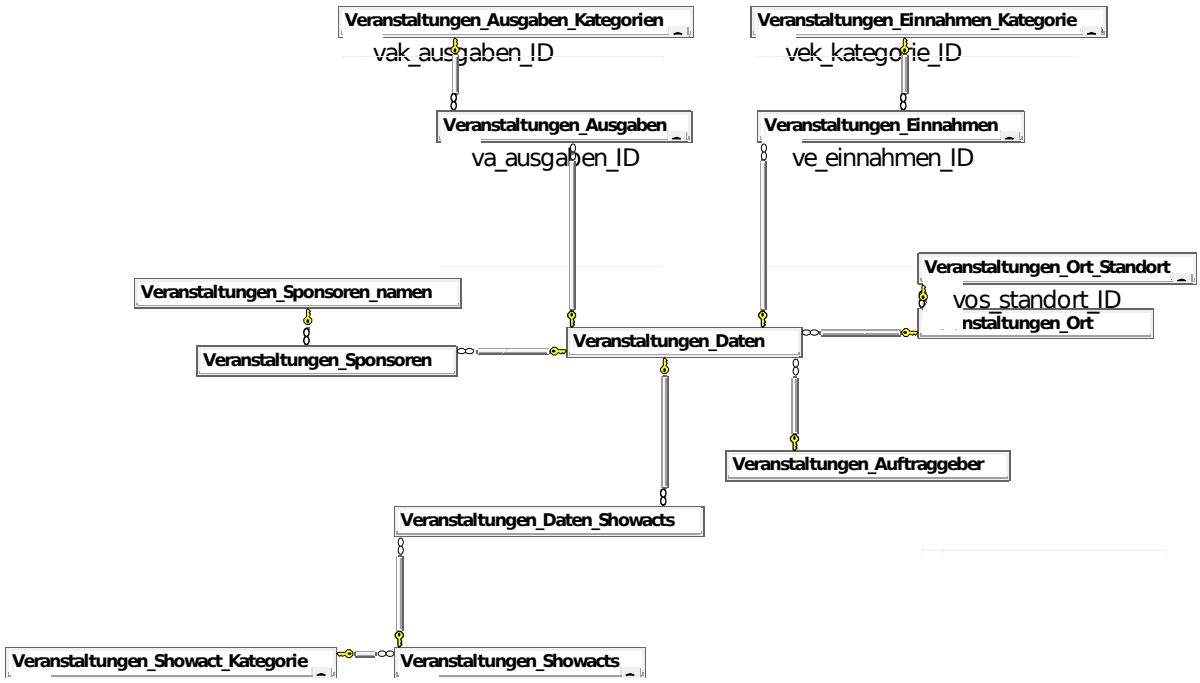

Abbildung 4: Beziehungsdiagramm Grobgliederungselement Veranstaltungen

Das Grobgliederungselement Veranstaltungen umfasst die meisten Tabellen, da hier das gesamte Auftragsmanagement, von der Showact-Kategorie bis hin zur Auftragsbuchung integriert ist.

Die folgende Abbildung zeigt die restlichen drei Grobgliederungselemente, mit deren Beziehungen zur Tabelle Veranstaltungen-Daten, zusammengefasst in einer Grafik.

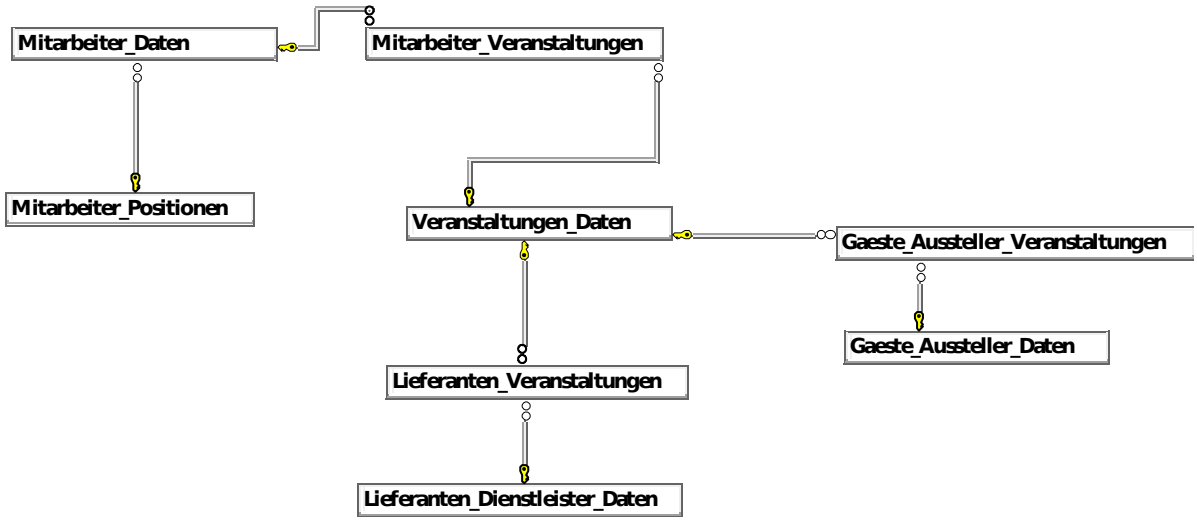

Abbildung 5: Grobgliederungselemente Mitarbeiter, Lieferanten und Dienstleister , Gäste und Aussteller

3 DDL – Trigger – View

3.1 DDL – Aufbau Grundstruktur

Aus den Anforderungen und dem erstellten Grundschema unter Kapitel 2 ergeben sich für die rDB EVENTUS die folgenden DDL-Befehle.

```
create database EVENTUS
--Tabellenkonfiguration
use EVENTUS
create table Veranstaltungen_Auftraggeber(
        vaa_auftraggeber_ID int PRIMARY KEY,
        vaa_auftraggeber_name varchar(50) not null)

create table Veranstaltungen_Daten (
        vd_veranst_ID int PRIMARY KEY,
        vd_veranst_name varchar (100) NOT NULL,
        vd_veranst_info varchar(50),
        vd_veranst_von date NOT NULL,
        vd_veranst_bis date,
        vd_veranst_ort_ID int FOREIGN KEY REFERENCES Veranstaltungen_Ort,
        vd_veranst_auftraggeber int FOREIGN KEY REFERENCES
        Veranstaltungen_Auftraggeber)

create table Veranstaltungen_Showact_Kategorie (
        vsk_kategorie_ID int PRIMARY KEY,
        vsk_kategorie_name varchar(50) not null)


create table Veranstaltungen_Showacts (
        vs_showact_ID int PRIMARY KEY,
        vs_titel varchar(8),
        vs_vorname varchar(50),
        vs_name varchar(100),
        vs_firma varchar(100),
        vs_kategorie_ID int FOREIGN KEY REFERENCES Veranstaltungen_Showact_Kategorie)

create table Veranstaltungen_Daten_Showacts (
        vds_veranst_ID int FOREIGN KEY REFERENCES Veranstaltungen_Daten,
        vds_showact_ID int FOREIGN KEY REFERENCES Veranstaltungen_Showacts)

create table Veranstaltungen_Ort_Standort (
        vos_standort_ID int PRIMARY KEY,
        vos_standort_name varchar(100)UNIQUE not null)

create table Veranstaltungen_Ort (
        vo_ort_ID int PRIMARY KEY,
        vo_standort_ID int FOREIGN KEY REFERENCES Veranstaltungen_Ort_Standort,
        vo_ort_name varchar(100),
        vo_ort_info      varchar(100))

create table Veranstaltungen_Einnahmen_Kategorie (
        vek_kategorie_ID int PRIMARY KEY,
        vek_kategorie_name varchar(50) UNIQUE not null)
```

```
create table Veranstaltungen_Einnahmen (
        ve_einnahmen_ID int FOREIGN KEY REFERENCES
        Veranstaltungen_Einnahmen_Kategorie,
        ve_veranst_ID int FOREIGN KEY REFERENCES Veranstaltungen_Daten,
        ve_einnahmen_hoehe money)

create table Veranstaltungen_Ausgaben_Kategorien (
        vak_ausgaben_ID int PRIMARY KEY,
        vak_ausgaben_name varchar(50) not null)

create table Veranstaltungen_Ausgaben (
        va_ausgaben_ID int FOREIGN KEY REFERENCES
        Veranstaltungen_Ausgaben_Kategorien,
        va_veranst_ID int FOREIGN KEY REFERENCES Veranstaltungen_Daten,
        va_ausgaben_betrag money )

create table Veranstaltungen_Sponsoren_namen (
        vsn_sponsoren_ID int PRIMARY KEY,
        vsn_sponsoren_name varchar(100) UNIQUE not null)

create table Veranstaltungen_Sponsoren (
        vsp_sponsoren_ID int FOREIGN KEY REFERENCES
Veranstaltungen_Sponsoren_namen,
        vsp_veranst_ID int FOREIGN KEY REFERENCES Veranstaltungen_Daten,
        vsp_sponsoren_betrag money DEFAULT(0))
```

Nachdem die wichtigsten Tabellen für das Veranstaltungsmanagement implementiert sind, folgt der zweite Block, welcher aus Tabellen von Lieferanten und Dienstleistern besteht.

```
--Block2: Lieferanten und Dienstleister

create table Lieferanten_Dienstleister_Daten (
        lfdd_lieferanten_ID int PRIMARY KEY,
        lfdd_lieferanten_name varchar(100) not null,
        lfdd_lieferanten_straße varchar(50) not null,
        lfdd_lieferanten_hsnr varchar(10) not null,
        lfdd_lieferanten_plz varchar(5),
        lfdd_lieferanten_ort int FOREIGN KEY REFERENCES Veranstaltungen_Ort_Standort,
        lfdd_lieferanten_bereich int FOREIGN KEY REFERENCES
        Veranstaltungen_Ausgaben_Kategorien)

create table Lieferanten_Veranstaltungen (
        lv_lieferanten_ID int FOREIGN KEY REFERENCES Lieferanten_Dienstleister_Daten,
        lv_veranst_ID int FOREIGN KEY REFERENCES Veranstaltungen_Daten)
```

Der dritte Block besteht aus Stammdaten über die Gäste und Aussteller die jeweils an den Veranstaltungen teilnehmen. Die Spalte gad_gaeste_email wird mithilfe eines Check-Befehls bei der Befüllung mit Daten auf ein gültiges E-Mail Format überprüft. Dabei befindet sich im auskommentierten Teil eine mögliche Alternative, die mithilfe von Regular Expressions die E-Mail Adressen prüfen würde.

```
--Block3: Gäste und Aussteller

create table Gaeste_Aussteller_Daten (
        gad_gaeste_ID int PRIMARY KEY,
        gad_gaeste_vname varchar(50),
        gad_gaeste_nname varchar(100) not null,
        gad_gaeste_str varchar(100),
        gad_gaeste_hsnr varchar(10),
        gad_gaeste_plz varchar(5) CHECK(gad_gaeste_plz like '[0-9][0-9][0-9][0-9]
        [0-9]' Or gad_gaeste_plz like ''),
        gad_gaeste_ort_ID int FOREIGN KEY REFERENCES Veranstaltungen_Ort_Standort ,
        gad_gaeste_email varchar(50)  CHECK(gad_gaeste_email LIKE '%__@%_.__' Or
        gad_gaeste_email LIKE '%__@%_.___' Or gad_gaeste_email LIKE '%__@%_.____' Or
        gad_gaeste_email like '' ), --'^[A-Z,a-z]*[0-9]*[.]*[@][A-Z,a-z]{1,*}[-]{0,*}[.]{0,*}[.]
        [a-z){0,3}$'
        gad_gaeste_telnr varchar(15)  CHECK(gad_gaeste_telnr LIKE '[0-9][0-9][0-9]
        [0-9][0-9]/[0-9][0-9][0-9][0-9][0-9][0-9]' OR gad_gaeste_telnr LIKE '[0-9]
        [0-9][0-9][0-9]/[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]' OR gad_gaeste_telnr like
        ''  ),
        gad_gaeste_vip varchar(4) CHECK(gad_gaeste_vip LIKE 'Ja' OR gad_gaeste_vip LIKE
        'NEIN'))

create table Gaeste_Aussteller_Veranstaltungen (
        gav_gaeste_ID int FOREIGN KEY REFERENCES Gaeste_Aussteller_Daten,
        gav_veranst_ID int FOREIGN KEY REFERENCES Veranstaltungen_Daten)
```

Der vierte Bestandteil der rDB EVENTUS sind die Informationen über die Mitarbeiter im Unternehmen. Diese Tabellen stellen dann auch die Grundlage für ein Rollen- und Sicherheitskonzept dar, welches später innerhalb des RDBMS implementiert wird.

```
--Block 4: Mitarbeiter

create table Mitarbeiter_Positionen (
        map_position_ID int PRIMARY KEY,
        map_position_name varchar(50) not null)

create table Mitarbeiter_Daten (
        mad_mitarbeiter_ID int PRIMARY KEY,
        mad_mitarbeiter_vname varchar(30)not null,
        mad_mitarbeiter_nname varchar(30)not null,
        mad_position_ID int FOREIGN KEY REFERENCES Mitarbeiter_Positionen,
        mad_mitarbeiter_stunden varchar (15) not null,
        mad_mitarbeiter_von date not null,
        mad_mitarbeiter_bis date,
        mad_mitarbeiter_jahresgehalt money,
        mad_mitarbeiter_monatsgehalt money)

create table Mitarbeiter_Veranstaltungen (
        mav_veranst_ID int FOREIGN KEY REFERENCES Veranstaltungen_Daten,
        mav_mitarbeiter_ID int FOREIGN KEY REFERENCES Mitarbeiter_Daten)
```

Die nun fertig erstellten Tabellen dienen als Grundlage für eine effizientere Datenhaltung. Durch das implementierte relationale Schema, wird außerdem eine schnelle Abfrage der Daten ermöglicht.

3.2 Trigger-Steuerung für eine effizientere Datenhaltung und -pflege

Im Falle der EVENTUS GmbH reicht es aber nicht aus nur die Tabellen zu erstellen, es gab in der Vergangenheit auch Aspekte bei der Datenpflege die nicht beachtet wurden. Damit eine konsistente und vollständige Datenbank entsteht, werden im Folgenden Trigger implementiert, die einerseits die Aktualisierung von Views übernehmen und andererseits die Datenhaltung steuern.

```
--Trigger für DB
use EVENTUS
--**
--Import-Trigger -> Datumseinstellung, Viewaktualisierung

CREATE TRIGGER Veranstaltung_Daten_zuk_Datum on dbo.Veranstaltungen_Daten
INSTEAD OF INSERT,UPDATE, DELETE
AS

SET NOCOUNT ON
	Declare @vorgang char(1)

	If (Select Count(*) from deleted) = 0
		Set @vorgang = 'I'
	Else
		If (Select COUNT(*) from inserted) = 0
		Set @vorgang = 'D'
		Else
		Set @vorgang = 'U'

If @vorgang = 'I'
Begin
	Declare @vd_veranst_ID int , @vd_veranst_name varchar(100) , @vd_veranst_info
	varchar(50),@vd_veranst_von date,@vd_veranst_bis date,@vd_veranst_ort_ID int,
	@vd_veranst_auftraggeber int


	Declare Import_Cursor_Datum Cursor For
	Select i.vd_veranst_ID, i.vd_veranst_name, i.vd_veranst_info, i.vd_veranst_von,
	i.vd_veranst_bis,i.vd_veranst_ort_ID, i.vd_veranst_auftraggeber FROM inserted i

	Open Import_Cursor_Datum

	Fetch next from Import_Cursor_Datum into @vd_veranst_ID, @vd_veranst_name,
	@vd_veranst_info,@vd_veranst_von, @vd_veranst_bis,@vd_veranst_ort_ID,
	@vd_veranst_auftraggeber

	While @@FETCH_STATUS = 0

		BEGIN

		If(@vd_veranst_bis = '1900-01-01')
```

```
                BEGIN
                INSERT INTO Veranstaltungen_Daten (vd_veranst_ID, vd_veranst_name,
                vd_veranst_info, vd_veranst_von, vd_veranst_bis, vd_veranst_ort_ID,
                vd_veranst_auftraggeber)
                VALUES (@vd_veranst_ID, @vd_veranst_name, @vd_veranst_info,
                @vd_veranst_von,@vd_veranst_von, @vd_veranst_ort_ID,
                @vd_veranst_auftraggeber)

                END

                ELSE

                BEGIN
                INSERT INTO Veranstaltungen_Daten (vd_veranst_ID, vd_veranst_name,
                vd_veranst_info, vd_veranst_von, vd_veranst_bis, vd_veranst_ort_ID,
                vd_veranst_auftraggeber)
                VALUES (@vd_veranst_ID, @vd_veranst_name, @vd_veranst_info,
                @vd_veranst_von, @vd_veranst_bis, @vd_veranst_ort_ID,
                @vd_veranst_auftraggeber)
                END
                If (@vd_veranst_bis <> '1900-01-01')
                                BEGIN
                                If (@vd_veranst_bis < @vd_veranst_von)
                                RAISERROR('Das Enddatum darf nicht vor dem
                                Startdatum liegen',11,1)
                                Rollback transaction
                                END

        Fetch next from Import_Cursor_Datum into @vd_veranst_ID, @vd_veranst_name,
        @vd_veranst_info,@vd_veranst_von, @vd_veranst_bis, @vd_veranst_ort_ID,
        @vd_veranst_auftraggeber

                END

        Close Import_Cursor_Datum
        Deallocate Import_Cursor_Datum
        EXECUTE sp_refreshview N'nächste_Veranstaltung'

End

If  @vorgang = 'D' --or @vorgang = 'U'
Begin
        Declare Export_Cursor_Datum Cursor For
        Select d.vd_veranst_ID FROM deleted d

                Open Export_Cursor_Datum
                        Fetch next from Export_Cursor_Datum into @vd_veranst_ID
                        While @@FETCH_STATUS = 0
                        BEGIN
                        Delete from Veranstaltungen_Daten
                        where vd_veranst_ID = @vd_veranst_ID

                        Fetch next from Export_Cursor_Datum into @vd_veranst_ID
                        END
                Close Export_Cursor_Datum
        Deallocate Export_Cursor_Datum
        EXECUTE sp_refreshview N'nächste_Veranstaltung'
```

End

```
If @vorgang = 'U'

        Begin
                Declare Update_Cursor_Datum Cursor For
                Select d.vd_veranst_ID FROM deleted d

                Open Update_Cursor_Datum
                Fetch next from Update_Cursor_Datum into @vd_veranst_ID
                While @@FETCH_STATUS = 0
                BEGIN
                Delete from Veranstaltungen_Daten
                where vd_veranst_ID = @vd_veranst_ID

                Fetch next from Update_Cursor_Datum into @vd_veranst_ID
        END

        Close Update_Cursor_Datum
        Deallocate Update_Cursor_Datum

        Declare Import_Cursor_Datum Cursor For
        Select i.vd_veranst_ID, i.vd_veranst_name, i.vd_veranst_info, i.vd_veranst_von,
        i.vd_veranst_bis, i.vd_veranst_ort_ID , i.vd_veranst_auftraggeber FROM inserted i

        Open Import_Cursor_Datum
        Fetch next from Import_Cursor_Datum into @vd_veranst_ID, @vd_veranst_name,
        @vd_veranst_info,@vd_veranst_von, @vd_veranst_bis,@vd_veranst_ort_ID,
        @vd_veranst_auftraggeber


        While @@FETCH_STATUS = 0

                BEGIN

                If(@vd_veranst_bis = '1900-01-01')
                BEGIN
                        INSERT INTO Veranstaltungen_Daten (vd_veranst_ID,
                        vd_veranst_name, vd_veranst_info, vd_veranst_von,
                        vd_veranst_bis,vd_veranst_ort_ID,vd_veranst_auftraggeber)
                        VALUES (@vd_veranst_ID, @vd_veranst_name, @vd_veranst_info,
                        @vd_veranst_von, @vd_veranst_von, @vd_veranst_ort_ID,
                        @vd_veranst_auftraggeber)

                END

                ELSE

                BEGIN
                        INSERT INTO Veranstaltungen_Daten (vd_veranst_ID,
                        vd_veranst_name, vd_veranst_info, vd_veranst_von,
                        vd_veranst_bis,vd_veranst_ort_ID,vd_veranst_auftraggeber)
                        VALUES (@vd_veranst_ID, @vd_veranst_name, @vd_veranst_info,
                        @vd_veranst_von, @vd_veranst_bis, @vd_veranst_ort_ID,
                        @vd_veranst_auftraggeber)

                END
```

```
                If (@vd_veranst_bis <> '1900-01-01')
                                        BEGIN
                                        If (@vd_veranst_bis < @vd_veranst_von)
                                        RAISERROR('Das Enddatum darf nicht vor dem
                                        Startdatum liegen',11,1)
                                        Rollback transaction
                                        END
        Fetch next from Import_Cursor_Datum into @vd_veranst_ID, @vd_veranst_name,
        @vd_veranst_info,@vd_veranst_von, @vd_veranst_bis, @vd_veranst_ort_ID,
        @vd_veranst_auftraggeber
                END
                Close Import_Cursor_Datum
                Deallocate Import_Cursor_Datum
                EXECUTE sp_refreshview N'nächste_Veranstaltung'
        End
--###
```

Dieser Trigger übernimmt die automatische Datumssetzung innerhalb der Tabelle Veranstaltungen_Daten, falls in der Spalte vd_veranst_bis kein Veranstaltungsenddatum beim Insert oder Update Befehl mitgegeben wird. Für diesen Fall wird in die Spalte vd_veranst_bis der Wert aus der Spalte vd_veranst_von eingetragen und es wird umgangen, dass der Standardwert ‘1900-01-01‘ für null im date-Format auftaucht. Des Weiteren wird beim Insert- und Update Befehl überprüft wenn ein Enddatum mitgegeben wurde, ob das Datum gleich oder größer dem Startdatum ist. Falls Werte gelöscht werden übernimmt auch dies der Trigger.

Die folgenden zwei Trigger sorgen wiederum dafür, dass die Daten der Views nach dem Insert-, Update- oder Delete Befehl aktualisiert werden.

```
--Trigger für Einnahmenaktualisierung View
--**
Create Trigger Einnahmenaktualisiserung on dbo.Veranstaltungen_Einnahmen
After Insert,Update,Delete
As
Begin
Execute sp_refreshview N'Einnahmen'
End
--###
--Trigger für Ausgabenaktualisierung View
--**
Create Trigger Ausgabenaktualisierung on dbo.Veranstaltungen_Ausgaben
After Insert,Update,Delete
As
Begin
EXECUTE sp_refreshview N'Ausgaben'
End
```

```
--###

--Besucheranzahl-> Viewaktualisierung & Import
--**
CREATE TRIGGER Gäste_View_Aktualisiserung ON dbo.Gaeste_Aussteller_Veranstaltungen
After INSERT,Update,Delete
AS
BEGIN
EXECUTE sp_refreshview N'Besucheranzahl'

END
--##
```

Der nächste Trigger wird besonders bei der Auftragsabrechnung benötigt. Bisher ist es häufig dazu gekommen, dass Einnahmenbuchungen insbesondere von Sponsorenbeträgen gefehlt haben. Um zu vermeiden, dass in der Sponsorentabelle Zahlungen gepflegt werden, die später aber nicht in den Einnahmen auftauchen sorgt der Trigger dafür, dass bevor Werte in der Tabelle Veranstaltungen_Sponsoren eingetragen werden können geprüft wird ob bereits Buchungen in der Einnahmentabelle vorliegt. Falls dies nicht der Fall ist, wird der Insert-Befehl per Rollback zurückgesetzt und ein Fehler ausgegeben, der den Systemnutzer darauf hinweisen soll, dass erst Einnahmenbuchungen ausgeführt werde müssen bevor ein Sponsorenbetrag gepflegt werden kann. Der Trigger sorgt dafür, falls in den Einnahmen eine Sponsorenzahlung fehlt, diese aber z.B. nachträglich in der Sponsorentabelle gepflegt wird, dann wird der entsprechende Wert in den Einnahmen hinzugefügt, geändert oder gelöscht. Damit der Trigger auch bei einem ausgeführten Bulk-Insert einwandfrei arbeitet, werden alle Spalten aus der inserted-Tabelle in einem Cursor gespeichert, der in seinen Variablen die Elemente der Spalten speichert. Daraufhin wird jede einzelne Zeile des Cursors ausgelesen und überprüft ob für die eingefügte Zeile bereits eine zugehörige Buchung in der Einnahmentabelle existiert. Der Trigger reagiert jedoch nicht nur bei Insert-Befehlen, er steuert auch die Delete und Update Befehle.

```
CREATE TRIGGER Veranstaltungen_Sponsoren_insert on dbo.Veranstaltungen_Sponsoren
INSTEAD OF INSERT,UPDATE, DELETE
AS

SET NOCOUNT ON
	Declare @vorgang char(1)

	If (Select Count(*) from deleted) = 0
		Set @vorgang = 'I'
	Else
		If (Select COUNT(*) from inserted) = 0
		Set @vorgang = 'D'
		Else
		Set @vorgang = 'U'
If @vorgang = 'I'

Begin
	If EXISTS (Select count(ve_veranst_ID) from Veranstaltungen_Einnahmen having
	count(ve_veranst_ID)= null)
	Rollback transaction
	Set NOCOUNT on;

	If Not Exists (Select * from Veranstaltungen_Einnahmen)

	Begin
	RAISERROR('Befüllen Sie erst die Tabelle Veranstaltungen_Einnahmen mit
	Werten',11,1)
	Rollback transaction
END

ELSE
	Declare @vsp_sponsoren_betrag money, @einnahmenkategorie int,@vsp_veranst_ID
	int,@vsp_sponsoren_ID int


	Declare Import_Cursor Cursor For
	Select i.vsp_sponsoren_ID,i.vsp_veranst_ID, i.vsp_sponsoren_betrag from inserted i

	Open Import_Cursor

	Fetch next from Import_Cursor into @vsp_sponsoren_ID,@vsp_veranst_ID,
	@vsp_sponsoren_betrag

	While @@FETCH_STATUS = 0

	BEGIN
		INSERT INTO Veranstaltungen_Sponsoren (vsp_sponsoren_ID, vsp_veranst_ID,
		vsp_sponsoren_betrag)
		VALUES (@vsp_sponsoren_ID,@vsp_veranst_ID,@vsp_sponsoren_betrag)
```

```
                IF (@vsp_sponsoren_betrag > 0)

                        Begin

                        Begin
                                SET @einnahmenkategorie = 400003
                                If (Exists(Select ve_einnahmen_hoehe from
                                Veranstaltungen_Einnahmen where ve_einnahmen_ID =
                                @einnahmenkategorie and ve_veranst_ID =
                                @vsp_veranst_ID))

                                Begin
                                Update Veranstaltungen_Einnahmen
                                Set ve_einnahmen_hoehe = ve_einnahmen_hoehe +
                                @vsp_sponsoren_betrag
                                where ve_einnahmen_ID = @einnahmenkategorie and
                                ve_veranst_ID = @vsp_veranst_ID
                                End

                Else


                        Begin
                                INSERT INTO Veranstaltungen_Einnahmen(ve_einnahmen_ID,
                                ve_veranst_ID, ve_einnahmen_hoehe)
                                VALUES (@einnahmenkategorie, @vsp_veranst_ID,
                                @vsp_sponsoren_betrag)
                        End

                        END
                        End


        Fetch next from Import_Cursor into @vsp_sponsoren_ID, @vsp_veranst_ID,
        @vsp_sponsoren_betrag
        End
                Close Import_Cursor
                Deallocate Import_Cursor

        EXECUTE sp_refreshview N'Sponsoringvolumentabelle'

End
If @vorgang = 'D'

Begin
        Declare Delete_Cursor Cursor For
        Select d.vsp_sponsoren_ID, d.vsp_veranst_ID, d.vsp_sponsoren_betrag from deleted
        d

        Open Delete_Cursor

        Fetch next from Delete_Cursor into @vsp_sponsoren_ID,@vsp_veranst_ID,
        @vsp_sponsoren_betrag

        While @@FETCH_STATUS = 0
```

```
                    BEGIN
                        Delete from Veranstaltungen_Sponsoren
                        where vsp_sponsoren_ID = @vsp_sponsoren_ID

                        SET @einnahmenkategorie = 400003

                        IF ( Exists(Select @vsp_veranst_ID from deleted where
                        @vsp_veranst_ID in (Select ve_veranst_ID from
                        Veranstaltungen_Einnahmen )))

                            Begin
                                Declare @volumen int

                                Set @volumen = (Select ve_einnahmen_hoehe from
                                Veranstaltungen_Einnahmen where  ve_einnahmen_ID
                                = @einnahmenkategorie and ve_veranst_ID =
                                @vsp_veranst_ID)

                                If @volumen = @vsp_sponsoren_betrag




                                Begin
                                Delete from Veranstaltungen_Einnahmen
                                where ve_veranst_ID = @vsp_veranst_ID and
                                ve_einnahmen_ID = @einnahmenkategorie and
                                ve_einnahmen_hoehe = @vsp_sponsoren_betrag
                                End
                            End

                        Else

                            Begin
                                Update Veranstaltungen_Einnahmen
                                Set ve_einnahmen_hoehe = @volumen -
                                @vsp_sponsoren_betrag
                                where ve_veranst_ID = @vsp_veranst_ID and
                                ve_einnahmen_ID = @einnahmenkategorie
                            END

                Fetch next from Delete_Cursor into @vsp_sponsoren_ID,@vsp_veranst_ID,
                @vsp_sponsoren_betrag
                End

                Close Delete_Cursor
                Deallocate Delete_Cursor

                EXECUTE sp_refreshview N'Sponsoringvolumentabelle'
        End


        If @vorgang = 'U'
```

```
Begin

        Declare Update_Cursor Cursor For
        Select i.vsp_sponsoren_ID, i.vsp_veranst_ID, i.vsp_sponsoren_betrag from inserted i

        Open Update_Cursor
        Fetch next from Update_Cursor into @vsp_sponsoren_ID,@vsp_veranst_ID,
        @vsp_sponsoren_betrag
        While @@FETCH_STATUS = 0

                BEGIN
                        Declare @summe int
                        Set @summe = (Select Sum(vsp_sponsoren_betrag) from
                        Veranstaltungen_Sponsoren where vsp_sponsoren_ID <>
                        @vsp_sponsoren_ID and vsp_veranst_ID = @vsp_veranst_ID)

                        Update Veranstaltungen_Sponsoren
                        Set vsp_sponsoren_ID =  @vsp_sponsoren_ID , vsp_veranst_ID =
                        @vsp_veranst_ID , vsp_sponsoren_betrag = @vsp_sponsoren_betrag

                        where vsp_sponsoren_ID = @vsp_sponsoren_ID and vsp_veranst_ID =
                        vsp_veranst_ID

                        IF (@vsp_sponsoren_betrag >= 0)

                                Begin
                                Begin

                                        SET @einnahmenkategorie = 400003
                                        If (Exists(Select ve_einnahmen_hoehe from
                                        Veranstaltungen_Einnahmen where ve_einnahmen_ID
                                        = @einnahmenkategorie and ve_veranst_ID =
                                        @vsp_veranst_ID))


                                        Begin
                                                If (@summe >0)
                                                        Begin
                                                                Update
                                                                Veranstaltungen_Einnahmen
                                                                Set ve_einnahmen_hoehe =
                                                                @summe +
                                                                @vsp_sponsoren_betrag
                                                                where ve_einnahmen_ID =
                                                                @einnahmenkategorie and
                                                                ve_veranst_ID =
                                                                @vsp_veranst_ID
                                                        End
                                                Else
```

```
                                        Begin
                                            Update
                                            Veranstaltungen_Einnahmen
                                            Set ve_einnahmen_hoehe =
                                            @vsp_sponsoren_betrag
                                            where ve_einnahmen_ID =
                                            @einnahmenkategorie and
                                            ve_veranst_ID =
                                            @vsp_veranst_ID

                                        End
                                End

                                Else
                                        Begin
                                            INSERT INTO
                                            Veranstaltungen_Einnahmen(ve_einnah
                                            men_ID,ve_veranst_ID,
                                            ve_einnahmen_hoehe)
                                            VALUES (@einnahmenkategorie,
                                            @vsp_veranst_ID,
                                            @vsp_sponsoren_betrag)
                                        End

                            END
                            END
        Fetch next from Update_Cursor into @vsp_sponsoren_ID, @vsp_veranst_ID,
        @vsp_sponsoren_betrag
                End

        Close Update_Cursor
        Deallocate Update_Cursor

        EXECUTE sp_refreshview N'Sponsoringvolumentabelle'

End
```

Nachdem die Trigger implementiert sind und einen reibungslosen Datenimport ermöglichen, ist es nun nötig die Views zu definieren, die spätere Abfragen erleichtern sollen. Dabei sollen vor allem komplexere Abfragen verschlankt werden, indem Teile aus materialized-Views genommen werden. Dies soll zu einer verbesserten Abfragegeschwindigkeit und einer einfacheren Wartung, im Falle von Fehlern oder nötigen Anpassungen, führen.

3.3 View-Definition

Die Views werden dann bei späteren Datenimporten sukzessive mit Daten gefüllt und es wird über die durch Trigger gesteuerte Aktualisierung auch erreicht, dass die Views immer aktuell sind.

```
use EVENTUS
CREATE VIEW Sponsoringvolumentabelle

AS
Select vsn_sponsoren_name AS Sponsor, SUM(vsp_sponsoren_betrag) AS
Sponsoringvolumen
FROM Veranstaltungen_Sponsoren INNER JOIN Veranstaltungen_Sponsoren_namen ON
vsp_sponsoren_ID = vsn_sponsoren_ID
GROUP BY vsn_sponsoren_name


CREATE VIEW nächste_Veranstaltung
AS
SELECT vd_veranst_ID as Veranstaltungsnummer, vd_veranst_von as Veranstaltungsbeginn
FROM Veranstaltungen_Daten
Where vd_veranst_von > GETDATE()

CREATE VIEW Besucheranzahl
AS
SELECT COUNT(gav_gaeste_ID) AS Besucheranzahl,gav_veranst_ID
FROM Gaeste_Aussteller_Veranstaltungen
GROUP BY gav_veranst_ID

CREATE VIEW Einnahmen
AS
SELECT vd_veranst_auftraggeber AS Auftraggeber_E, SUM(ve_einnahmen_hoehe)AS
Summe_E
FROM Veranstaltungen_Daten,Veranstaltungen_Einnahmen
WHERE ve_veranst_ID = vd_veranst_ID
GROUP BY vd_veranst_auftraggeber


CREATE VIEW Ausgaben
As
SELECT SUM(va_ausgaben_betrag) AS Summe,vd_veranst_auftraggeber AS Auftraggeber
FROM Veranstaltungen_Ausgaben,Veranstaltungen_Daten
WHERE va_veranst_ID = vd_veranst_ID
GROUP BY vd_veranst_auftraggeber
```

Mithilfe der Views ist es nun möglich, die Anforderungen bezüglich der Abfragen zu effizienter zu erfüllen. Die expliziten SQL-Abfragen folgen dann in **Kapitel 4.**

An nächster Stelle folgt der Datenimport, auf dessen Darstellung hier verzichtet wird. Für den detaillierten Datenimport, benutzen Sie die bereitgestellten SQL-Dateien.

4 SQL– Queries

Eine weitere Anforderung ist es SQL-Abfragen für die erstellte und mit Daten befüllte DB zu entwickeln. Die folgenden 12 Abfragen beinhalten die anforderungsgemäße Beantwortung der Abfragewünsche.

```
use EVENTUS
--Query1: Alle vergangenen Events
SELECT vd_veranst_name AS Veranstaltung, vd_veranst_von AS Anfangstermin,
vd_veranst_bis AS Endtermin
FROM Veranstaltungen_Daten
WHERE GETDATE() > Veranstaltungen_Daten.vd_veranst_von And GETDATE() >
Veranstaltungen_Daten.vd_veranst_bis

--Query2: Alle Sponsoren die mehr als 2* gespendet haben
SELECT vsn_sponsoren_name as Sponsor, Count(vsp_sponsoren_ID) as #Spenden
FROM Veranstaltungen_Sponsoren_namen INNER JOIN Veranstaltungen_Sponsoren on
vsn_sponsoren_ID = vsp_sponsoren_ID
GROUP BY vsn_sponsoren_name,vsp_sponsoren_ID
HAVING Count(vsp_sponsoren_ID) >2

--Query3: Alle MA die Vollzeit sind
SELECT mad_mitarbeiter_ID AS Personalnummer, mad_mitarbeiter_vname AS Vorname,
mad_mitarbeiter_nname AS Nachname
FROM Mitarbeiter_Daten
WHERE mad_mitarbeiter_stunden LIKE '[0-9][0-9]h' AND mad_mitarbeiter_stunden > '39h'

--Query4: Sponsor mit höchstem Sponsoringvolumen
SELECT Sponsor, MAX(Sponsoringvolumen) AS Sponsoringvolumen
FROM Sponsoringvolumentabelle
WHERE Sponsoringvolumen = (SELECT MAX(Sponsoringvolumen) FROM
Sponsoringvolumentabelle)
GROUP BY Sponsor

--Query5: # Showacts pro Veranstaltung
SELECT vd_veranst_ID AS Veranstaltungsnummer,vd_veranst_name AS Veranstaltung,
COUNT(vds_showact_ID) AS #Showacts
FROM Veranstaltungen_Daten_Showacts INNER JOIN Veranstaltungen_Daten  ON
vds_veranst_ID = vd_veranst_ID
GROUP BY vd_veranst_name, vd_veranst_ID
ORDER BY COUNT(vds_showact_ID) ASC

---Query6: Liste Reservierungen für nächstes Event
SELECT gav_gaeste_ID AS Gästenummer , gad_gaeste_vname AS
Vorname,gad_gaeste_nname AS Name,vd_veranst_von AS Veranstaltungsbeginn,
vd_veranst_bis AS Veranstaltungsende,vd_veranst_name AS Veranstaltungsname
FROM (Gaeste_Aussteller_Veranstaltungen gav
inner join Gaeste_Aussteller_Daten gad ON gav.gav_gaeste_ID = gad.gad_gaeste_ID)
inner join Veranstaltungen_Daten vd ON gav.gav_veranst_ID = vd.vd_veranst_ID
WHERE gav_veranst_ID =      (          SELECT Veranstaltungsnummer
                                       FROM nächste_Veranstaltung
                            WHERE  Veranstaltungsbeginn = (SELECT
                            MIN(Veranstaltungsbeginn) FROM nächste_Veranstaltung))

--Query7: bester Kunde (auf Basis #Umsatzvolumen)
```

```
SELECT vaa_auftraggeber_name AS Kunde
FROM Veranstaltungen_Auftraggeber INNER JOIN Einnahmen ON vaa_auftraggeber_ID =
Auftraggeber_E
WHERE Summe_E = (SELECT MAX(Summe_E) FROM Einnahmen)

--Query8: durchschnittl. Besucherzahl Veranstaltungen
SELECT AVG(Besucheranzahl) AS durchschnittliche_Besucheranzahl
FROM Besucheranzahl

--Query9: VIP- Zuständiger MA
SELECT DISTINCT  mad_mitarbeiter_ID AS Mitarbeiternummmer,mad_mitarbeiter_vname AS
Vorname, mad_mitarbeiter_nname AS Nachname
FROM (Mitarbeiter_Daten md
        INNER JOIN Mitarbeiter_Veranstaltungen mv
        ON md.mad_mitarbeiter_ID = mv.mav_mitarbeiter_ID)
        INNER JOIN Gaeste_Aussteller_Veranstaltungen gav
        ON mv.mav_veranst_ID = gav.gav_veranst_ID
        INNER JOIN Gaeste_Aussteller_Daten gad
        ON gav.gav_gaeste_ID = gad.gad_gaeste_ID
 WHERE gad.gad_gaeste_vip = 'Ja'

--Query10:Veranstaltung mit dem meisten Gewinn
SELECT vag.vaa_auftraggeber_name AS Kunde
FROM (Einnahmen e
        inner join Ausgaben a
        ON e.Auftraggeber_E = a.Auftraggeber)
        inner join Veranstaltungen_Auftraggeber vag
        ON e.Auftraggeber_E= vag.vaa_auftraggeber_ID
WHERE Summe_E - Summe = (SELECT MAX(Summe_E - Summe) FROM Einnahmen e

                                inner join Ausgaben a ON e.Auftraggeber_E = a.Auftraggeber)

--Query11: Konzerte die noch keine Reservierungen haben
SELECT vd_veranst_ID AS VeranstaltungsID, vd_veranst_name AS Veranstaltung
FROM Veranstaltungen_Daten
WHERE vd_veranst_name like '%konzert%' Or vd_veranst_info like '%konzert%' And
vd_veranst_ID not in (SELECT vds_veranst_ID FROM Veranstaltungen_Daten_Showacts )

--Query12: Liste aller DL inkl. Veranstaltungen
SELECT lfdd_lieferanten_name AS Lieferant_Dienstleister, vd_veranst_name AS
Veranstaltung
FROM (Lieferanten_Dienstleister_Daten
                FULL JOIN Lieferanten_Veranstaltungen
                ON lfdd_lieferanten_ID = lv_lieferanten_ID)
                LEFT OUTER JOIN Veranstaltungen_Daten
                ON lv_veranst_ID = vd_veranst_ID
```

Hierbei ist anzumerken, dass Query4, Query6, Query7,Query8 und Query10 zum Teil auf Abfragen aus gespeicherten Views basieren.

Damit das RDBMS vor unautorisierten Zugriffen von Benutzern abgesichert ist, benötigt es ein Rechte- und Rollenkonzept, welches im folgenden Kapitel implementiert wird.

5 Rechte- und Rollenverwaltung

Die Datenbankrollen der EVENTUS GmbH lassen sich grob in die Benutzerrollen Projektleitung, Assistenz, Geschäftsführung, Werkstudenten und Administrator einteilen. Dabei wird dem freien Mitarbeiter Volker Reinhardt die Administratorrolle zugeordnet. Es wird angenommen, dass er die EVENTUS GmbH bei Wartungsarbeiten der DB unterstützt. Alle anderen Benutzerrollen entsprechen den jeweiligen Positionen der Mitarbeiter im Unternehmen. Darüber hinaus wird von dem mittelständischen Unternehmen angenommen, dass auch die Geschäftsführung am operativen Tagesgeschäft teilnimmt, wodurch auch dieser Rolle teilweise Update, Insert und Delete Rechte eingeräumt werden.

```
use EVENTUS
--user add

CREATE LOGIN EVDH01 WITH PASSWORD = 'EVDH01';
CREATE LOGIN EVDH02 WITH PASSWORD = 'EVDH02';
CREATE LOGIN EVGR03 WITH PASSWORD = 'EVGR03';
CREATE LOGIN EVJH04 WITH PASSWORD = 'EVJH04';
CREATE LOGIN EVMA05 WITH PASSWORD = 'EVMA05';
CREATE LOGIN EVNR06 WITH PASSWORD = 'EVNR06';
CREATE LOGIN EVPS07 WITH PASSWORD = 'EVPS07';
CREATE LOGIN EVPR08 WITH PASSWORD = 'EVPR08';
CREATE LOGIN EVVR09 WITH PASSWORD = 'EVVR09';
USE EVENTUS
GO
CREATE USER Daniela_Hoffmann FOR LOGIN EVDH01;
CREATE USER Danilo_Hoffmann FOR LOGIN EVDH02;
CREATE USER Gustav_Richter FOR LOGIN EVGR03;
CREATE USER Janina_Hagenreiter FOR LOGIN EVJH04;
CREATE USER Mareike_Adam FOR LOGIN EVMA05;
CREATE USER Nadine_Reimer FOR LOGIN EVNR06;
CREATE USER Paula_Schmidt FOR LOGIN EVPS07;
CREATE USER Peggy_Richter FOR LOGIN EVPR08;
CREATE USER Volker_Reinhart FOR LOGIN EVVR09;
GO
EXEC sp_addsrvrolemember @loginame = N'EVVR09', @rolename = N'sysadmin'
GO


--Rechte für einzelne Tabellen vergeben
use EVENTUS
CREATE ROLE Projektleitung
Go
ALTER ROLE Projektleitung ADD MEMBER Daniela_Hoffmann
ALTER ROLE Projektleitung ADD MEMBER Mareike_Adam
GO
CREATE ROLE Assistenz
ALTER ROLE Assistenz ADD MEMBER Janina_Hagenreiter
ALTER ROLE Assistenz ADD MEMBER Peggy_Richter
GO
```

```
CREATE ROLE Geschäftsführung
ALTER ROLE Geschäftsführung ADD MEMBER Paula_Schmidt
GO
CREATE ROLE Werkstudenten
ALTER ROLE Werkstudenten ADD MEMBER Danilo_Hoffmann
ALTER ROLE Werkstudenten ADD MEMBER Gustav_Richter
ALTER ROLE Werkstudenten ADD MEMBER Nadine_Reimer
--Veranstaltungen Auftraggeber
GRANT UPDATE ON [dbo].[Veranstaltungen_Auftraggeber] TO
Projektleitung,Assistenz,Geschäftsführung
GRANT SELECT ON dbo.Veranstaltungen_Auftraggeber TO
Projektleitung,Assistenz,Geschäftsführung,Werkstudenten
GRANT INSERT ON dbo.Veranstaltungen_Auftraggeber TO
Projektleitung,Assistenz,Geschäftsführung
GRANT DELETE ON dbo.Veranstaltungen_Auftraggeber TO
Projektleitung,Assistenz,Geschäftsführung

--Veranstaltungen Daten
GRANT UPDATE ON [dbo].[Veranstaltungen_Daten] TO
Projektleitung,Assistenz,Geschäftsführung
GRANT SELECT ON dbo.Veranstaltungen_Daten TO
Projektleitung,Assistenz,Geschäftsführung,Werkstudenten
GRANT INSERT ON dbo.Veranstaltungen_Daten TO
Projektleitung,Assistenz,Geschäftsführung
GRANT DELETE ON dbo.Veranstaltungen_Daten TO
Projektleitung,Assistenz,Geschäftsführung

--Veranstaltungen_Showact_Kategorie
GRANT UPDATE ON [dbo].[Veranstaltungen_Showact_Kategorie] TO
Projektleitung,Assistenz,Geschäftsführung
GRANT SELECT ON dbo.Veranstaltungen_Showact_Kategorie TO
Projektleitung,Assistenz,Geschäftsführung,Werkstudenten
GRANT INSERT ON dbo.Veranstaltungen_Showact_Kategorie TO
Projektleitung,Assistenz,Geschäftsführung
GRANT DELETE ON dbo.Veranstaltungen_Showact_Kategorie TO
Projektleitung,Assistenz,Geschäftsführung
--Veranstaltungen_Showacts
GRANT UPDATE ON [dbo].[Veranstaltungen_Showacts] TO
Projektleitung,Assistenz,Geschäftsführung
GRANT SELECT ON dbo.Veranstaltungen_Showacts TO
Projektleitung,Assistenz,Geschäftsführung,Werkstudenten
GRANT INSERT ON dbo.Veranstaltungen_Showacts TO
Projektleitung,Assistenz,Geschäftsführung
GRANT DELETE ON dbo.Veranstaltungen_Showacts TO
Projektleitung,Assistenz,Geschäftsführung
--Veranstaltungen_Daten_Showacts
GRANT UPDATE ON [dbo].[Veranstaltungen_Daten_Showacts] TO
Projektleitung,Assistenz,Geschäftsführung
GRANT SELECT ON dbo.Veranstaltungen_Daten_Showacts TO
Projektleitung,Assistenz,Geschäftsführung,Werkstudenten
GRANT INSERT ON dbo.Veranstaltungen_Daten_Showacts TO
Projektleitung,Assistenz,Geschäftsführung
GRANT DELETE ON dbo.Veranstaltungen_Daten_Showacts TO
Projektleitung,Assistenz,Geschäftsführung
```

```
--Veranstaltungen_Ort_Standort
GRANT UPDATE ON [dbo].[Veranstaltungen_Ort_Standort] TO
Projektleitung,Assistenz,Geschäftsführung
GRANT SELECT ON dbo.Veranstaltungen_Ort_Standort TO
Projektleitung,Assistenz,Geschäftsführung,Werkstudenten
GRANT INSERT ON dbo.Veranstaltungen_Ort_Standort TO
Projektleitung,Assistenz,Geschäftsführung
GRANT DELETE ON dbo.Veranstaltungen_Ort_Standort TO
Projektleitung,Assistenz,Geschäftsführung
--Veranstaltungen_Ort
GRANT UPDATE ON [dbo].[Veranstaltungen_Ort] TO
Projektleitung,Assistenz,Geschäftsführung
GRANT SELECT ON dbo.Veranstaltungen_Ort TO
Projektleitung,Assistenz,Geschäftsführung,Werkstudenten
GRANT INSERT ON dbo.Veranstaltungen_Ort TO Projektleitung,Assistenz,Geschäftsführung
GRANT DELETE ON dbo.Veranstaltungen_Ort TO Projektleitung,Assistenz,Geschäftsführung
--Veranstaltungen_Einnahmen_Kategorie
GRANT UPDATE ON [dbo].[Veranstaltungen_Einnahmen_Kategorie] TO Assistenz
GRANT SELECT ON dbo.Veranstaltungen_Einnahmen_Kategorie TO
Assistenz,Geschäftsführung,Projektleitung
GRANT INSERT ON dbo.Veranstaltungen_Einnahmen_Kategorie TO Assistenz
GRANT DELETE ON dbo.Veranstaltungen_Einnahmen_Kategorie TO Assistenz
--Veranstaltungen_Einnahmen
GRANT UPDATE ON [dbo].[Veranstaltungen_Einnahmen] TO Assistenz
GRANT SELECT ON dbo.Veranstaltungen_Einnahmen TO
Assistenz,Geschäftsführung,Projektleitung
GRANT INSERT ON dbo.Veranstaltungen_Einnahmen TO Assistenz
GRANT DELETE ON dbo.Veranstaltungen_Einnahmen TO Assistenz
--Veranstaltungen_Ausgaben_Kategorien
GRANT UPDATE ON [dbo].[Veranstaltungen_Ausgaben_Kategorien] TO Assistenz
GRANT SELECT ON dbo.Veranstaltungen_Ausgaben_Kategorien TO
Assistenz,Geschäftsführung,Projektleitung
GRANT INSERT ON dbo.Veranstaltungen_Ausgaben_Kategorien TO Assistenz
GRANT DELETE ON dbo.Veranstaltungen_Ausgaben_Kategorien TO Assistenz
--Veranstaltungen_Ausgaben
GRANT UPDATE ON [dbo].[Veranstaltungen_Ausgaben] TO Assistenz
GRANT SELECT ON dbo.Veranstaltungen_Ausgaben TO
Assistenz,Geschäftsführung,Projektleitung
GRANT INSERT ON dbo.Veranstaltungen_Ausgaben TO Assistenz
GRANT DELETE ON dbo.Veranstaltungen_Ausgaben TO Assistenz
--Veranstaltungen_Sponsoren_namen
GRANT UPDATE ON [dbo].[Veranstaltungen_Sponsoren_namen] TO Assistenz
GRANT SELECT ON dbo.Veranstaltungen_Sponsoren_namen TO
Assistenz,Geschäftsführung,Projektleitung,Werkstudenten
GRANT INSERT ON dbo.Veranstaltungen_Sponsoren_namen TO Assistenz
GRANT DELETE ON dbo.Veranstaltungen_Sponsoren_namen TO Assistenz
--Veranstaltungen_Sponsoren
GRANT UPDATE ON [dbo].[Veranstaltungen_Sponsoren] TO Assistenz
GRANT SELECT ON dbo.Veranstaltungen_Sponsoren TO
Assistenz,Geschäftsführung,Projektleitung
GRANT INSERT ON dbo.Veranstaltungen_Sponsoren TO Assistenz
GRANT DELETE ON dbo.Veranstaltungen_Sponsoren TO Assistenz
--Lieferanten_Dienstleister_Daten
GRANT UPDATE ON [dbo].[Lieferanten_Dienstleister_Daten] TO Assistenz
GRANT SELECT ON dbo.Lieferanten_Dienstleister_Daten TO
Projektleitung,Assistenz,Geschäftsführung,Werkstudenten
GRANT INSERT ON dbo.Lieferanten_Dienstleister_Daten TO Assistenz
GRANT DELETE ON dbo.Lieferanten_Dienstleister_Daten TO Assistenz
```

```
--Lieferanten_Veranstaltungen
GRANT UPDATE ON [dbo].[Lieferanten_Veranstaltungen] TO
Projektleitung,Assistenz,Geschäftsführung
GRANT SELECT ON dbo.Lieferanten_Veranstaltungen TO
Projektleitung,Assistenz,Geschäftsführung,Werkstudenten
GRANT INSERT ON dbo.Lieferanten_Veranstaltungen TO
Projektleitung,Assistenz,Geschäftsführung
GRANT DELETE ON dbo.Lieferanten_Veranstaltungen TO
Projektleitung,Assistenz,Geschäftsführung
--Gaeste_Aussteller_Daten
GRANT UPDATE ON [dbo].[Gaeste_Aussteller_Daten] TO Assistenz
GRANT SELECT ON dbo.Gaeste_Aussteller_Daten TO
Projektleitung,Assistenz,Geschäftsführung,Werkstudenten
GRANT INSERT ON dbo.Gaeste_Aussteller_Daten TO Assistenz
GRANT DELETE ON dbo.Gaeste_Aussteller_Daten TO Assistenz
--Gaeste_Aussteller_Veranstaltungen
GRANT UPDATE ON [dbo].[Gaeste_Aussteller_Veranstaltungen] TO
Projektleitung,Assistenz,Geschäftsführung
GRANT SELECT ON dbo.Gaeste_Aussteller_Veranstaltungen TO
Projektleitung,Assistenz,Geschäftsführung,Werkstudenten
GRANT INSERT ON dbo.Gaeste_Aussteller_Veranstaltungen TO
Projektleitung,Assistenz,Geschäftsführung
GRANT DELETE ON dbo.Gaeste_Aussteller_Veranstaltungen TO
Projektleitung,Assistenz,Geschäftsführung
--Mitarbeiter_Positionen
GRANT UPDATE ON [dbo].[Mitarbeiter_Positionen] TO Assistenz
GRANT SELECT ON dbo.Mitarbeiter_Positionen TO Assistenz,Geschäftsführung
GRANT INSERT ON dbo.Mitarbeiter_Positionen TO Assistenz
GRANT DELETE ON dbo.Mitarbeiter_Positionen TO Assistenz
--Mitarbeiter_Daten
GRANT UPDATE ON [dbo].[Mitarbeiter_Daten] TO Assistenz
GRANT SELECT ON dbo.Mitarbeiter_Daten TO Assistenz,Geschäftsführung
GRANT INSERT ON dbo.Mitarbeiter_Daten TO Assistenz
GRANT DELETE ON dbo.Mitarbeiter_Daten TO Assistenz
--Mitarbeiter_Veranstaltungen
GRANT UPDATE ON [dbo].[Mitarbeiter_Veranstaltungen] TO
Projektleitung,Assistenz,Geschäftsführung
GRANT SELECT ON dbo.Mitarbeiter_Veranstaltungen TO
Projektleitung,Assistenz,Geschäftsführung
GRANT INSERT ON dbo.Mitarbeiter_Veranstaltungen TO
Projektleitung,Assistenz,Geschäftsführung
GRANT DELETE ON dbo.Mitarbeiter_Veranstaltungen TO
Projektleitung,Assistenz,Geschäftsführung
```

6 XML- Integration

Das folgende Codebeispiel zeigt eine XML-Instanz, welche in einer SQL-Datenbank in einer Zelle hinterlegt werden kann und später mittels SQL/Xquery abgefragt werden kann.

```
use EVENTUS
create table xml_Veranstaltungen(
        xml_ID varchar(50) Primary Key,
        veranstaltung xml)


Insert into xml_Veranstaltungen (xml_ID, veranstaltung)
Values ('xml_1','
                <Veranstaltungen_Daten Veranstaltung="7. AutomobilzuliefererTage"
Startdatum="2014-01-23" Enddatum="2014-01-24">
  <Veranstaltungen_Ort Veranstaltungsort="Messe">
   <Veranstaltungen_Ort_Standort Standort="Erfurt">
    <Veranstaltungen_Showacts Titel="" Vorname="Ralf" Name="Ruprecht"
Firma="Volkswagen AG">
     <Veranstaltungen_Showact_Kategorie Kategorie="Auftaktredner" />
    </Veranstaltungen_Showacts>
    <Veranstaltungen_Showacts Titel="" Vorname="" Name="Pianissimo" Firma="">
     <Veranstaltungen_Showact_Kategorie Kategorie="Streichquartett" />
    </Veranstaltungen_Showacts>
   </Veranstaltungen_Ort_Standort>
  </Veranstaltungen_Ort>
</Veranstaltungen_Daten>
<Veranstaltungen_Daten Veranstaltung="8. AutomobilzuliefererTage" Startdatum="2015-
01-09" Enddatum="2015-01-10">
  <Veranstaltungen_Ort Veranstaltungsort="Messe">
   <Veranstaltungen_Ort_Standort Standort="Erfurt">
    <Veranstaltungen_Showacts Titel="" Vorname="Matthias" Name="Jürgens"
Firma="BMW Group München">
     <Veranstaltungen_Showact_Kategorie Kategorie="Auftaktredner" />
    </Veranstaltungen_Showacts>
    <Veranstaltungen_Showacts Titel="" Vorname="" Name="Jazzorchester Mannheim"
Firma="">
     <Veranstaltungen_Showact_Kategorie Kategorie="Jazzorchester" />
    </Veranstaltungen_Showacts>
   </Veranstaltungen_Ort_Standort>
  </Veranstaltungen_Ort>
</Veranstaltungen_Daten>
<Veranstaltungen_Daten Veranstaltung="Dental 2014" Startdatum="2014-09-27"
Enddatum="2014-09-27">
  <Veranstaltungen_Ort Veranstaltungsort="Friedrich-Schiller-Universität">
   <Veranstaltungen_Ort_Standort Standort="Jena">
    <Veranstaltungen_Showacts Titel="Prof." Vorname="Georg" Name="Rimmel"
Firma="">
     <Veranstaltungen_Showact_Kategorie Kategorie="Auftaktredner" />
    </Veranstaltungen_Showacts>
    <Veranstaltungen_Showacts Titel="" Vorname="Ronald" Name="Treller" Firma="">
     <Veranstaltungen_Showact_Kategorie Kategorie="Moderation" />
    </Veranstaltungen_Showacts>
   </Veranstaltungen_Ort_Standort>
  </Veranstaltungen_Ort>
</Veranstaltungen_Daten>
```

```
<Veranstaltungen_Daten Veranstaltung="Firmenlauf 2015" Startdatum="2015-09-25"
Enddatum="2015-09-25">
  <Veranstaltungen_Ort Veranstaltungsort="Innenstadt">
    <Veranstaltungen_Ort_Standort Standort="Naumburg">
      <Veranstaltungen_Showacts Titel="" Vorname="Rene" Name="Lindner" Firma="">
        <Veranstaltungen_Showact_Kategorie Kategorie="Moderation" />
      </Veranstaltungen_Showacts>
      <Veranstaltungen_Showacts Titel="" Vorname="" Name="Akuwaba" Firma="">
        <Veranstaltungen_Showact_Kategorie Kategorie="Trommlergruppe" />
      </Veranstaltungen_Showacts>
    </Veranstaltungen_Ort_Standort>
  </Veranstaltungen_Ort>
</Veranstaltungen_Daten>
<Veranstaltungen_Daten Veranstaltung="Firmenlauf 2016" Startdatum="2016-09-30"
Enddatum="2016-09-30">
  <Veranstaltungen_Ort Veranstaltungsort="Innenstadt">
    <Veranstaltungen_Ort_Standort Standort="Naumburg">
      <Veranstaltungen_Showacts Titel="" Vorname="Rene" Name="Lindner" Firma="">
        <Veranstaltungen_Showact_Kategorie Kategorie="Moderation" />
      </Veranstaltungen_Showacts>
      <Veranstaltungen_Showacts Titel="" Vorname="" Name="Voice &amp;quot;n&amp;quot; Fun"
Firma="">
        <Veranstaltungen_Showact_Kategorie Kategorie="Band" />
      </Veranstaltungen_Showacts>
      <Veranstaltungen_Showacts Titel="" Vorname="" Name="Augustiner Vicalkreis"
Firma="">
        <Veranstaltungen_Showact_Kategorie Kategorie="Chor" />
      </Veranstaltungen_Showacts>
    </Veranstaltungen_Ort_Standort>
  </Veranstaltungen_Ort>
</Veranstaltungen_Daten>
<Veranstaltungen_Daten Veranstaltung="JeNAH Jubiläum" Startdatum="2016-04-01"
Enddatum="2016-04-01">
  <Veranstaltungen_Ort Veranstaltungsort="Volkshaus">
    <Veranstaltungen_Ort_Standort Standort="Arnstadt-Alkersleben">
      <Veranstaltungen_Showacts Titel="" Vorname="Tim" Name="Tölke" Firma="">
        <Veranstaltungen_Showact_Kategorie Kategorie="Moderation" />
      </Veranstaltungen_Showacts>
      <Veranstaltungen_Showacts Titel="" Vorname="Albrecht" Name="Schröter" Firma="">
        <Veranstaltungen_Showact_Kategorie Kategorie="Oberbürgermeister Jena" />
      </Veranstaltungen_Showacts>
      <Veranstaltungen_Showacts Titel="" Vorname="" Name="Pianissimo" Firma="">
        <Veranstaltungen_Showact_Kategorie Kategorie="Streichquartett" />
      </Veranstaltungen_Showacts>
      <Veranstaltungen_Showacts Titel="" Vorname="Tim" Name="Tölke" Firma="">
        <Veranstaltungen_Showact_Kategorie Kategorie="Moderation" />
      </Veranstaltungen_Showacts>
      <Veranstaltungen_Showacts Titel="" Vorname="Albrecht" Name="Schröter" Firma="">
        <Veranstaltungen_Showact_Kategorie Kategorie="Oberbürgermeister Jena" />
      </Veranstaltungen_Showacts>
      <Veranstaltungen_Showacts Titel="" Vorname="" Name="Abba99" Firma="">
        <Veranstaltungen_Showact_Kategorie Kategorie="Abba-Coverband" />
      </Veranstaltungen_Showacts>
      <Veranstaltungen_Showacts Titel="" Vorname="" Name="Diamonds" Firma="">
        <Veranstaltungen_Showact_Kategorie Kategorie="Showtanzgruppe" />
      </Veranstaltungen_Showacts>
    </Veranstaltungen_Ort_Standort>
  </Veranstaltungen_Ort>
```

```
</Veranstaltungen_Daten>
<Veranstaltungen_Daten Veranstaltung="Kinderzirkus Weimar" Startdatum="2015-06-01"
Enddatum="2015-06-01">
  <Veranstaltungen_Ort Veranstaltungsort="Kinderland Bummi e.V.">
    <Veranstaltungen_Ort_Standort Standort="Dresden">
      <Veranstaltungen_Showacts Titel="" Vorname="Nathalie" Name="Reimann" Firma="">
        <Veranstaltungen_Showact_Kategorie Kategorie="Feuershow" />
      </Veranstaltungen_Showacts>
      <Veranstaltungen_Showacts Titel="" Vorname="Peter" Name="Jacobi" Firma="">
        <Veranstaltungen_Showact_Kategorie Kategorie="Kaninchendompteur" />
      </Veranstaltungen_Showacts>
      <Veranstaltungen_Showacts Titel="" Vorname="Familie" Name="Altmann" Firma="">
        <Veranstaltungen_Showact_Kategorie Kategorie="Trapez" />
      </Veranstaltungen_Showacts>
    </Veranstaltungen_Ort_Standort>
  </Veranstaltungen_Ort>
</Veranstaltungen_Daten>
<Veranstaltungen_Daten Veranstaltung="Klassik für jedermann " Startdatum="2015-05-08"
Enddatum="2015-05-08">
  <Veranstaltungen_Ort Veranstaltungsort="Mammuthalle">
    <Veranstaltungen_Ort_Standort Standort="Dornburg">
      <Veranstaltungen_Showacts Titel="" Vorname="" Name="MDR Orchester" Firma="">
        <Veranstaltungen_Showact_Kategorie Kategorie="Orchester" />
      </Veranstaltungen_Showacts>
    </Veranstaltungen_Ort_Standort>
  </Veranstaltungen_Ort>
</Veranstaltungen_Daten>
<Veranstaltungen_Daten Veranstaltung="Krimiabend " Startdatum="2016-03-20"
Enddatum="2016-03-20">
  <Veranstaltungen_Ort Veranstaltungsort="Marien-Magdalen Kirche">
    <Veranstaltungen_Ort_Standort Standort="Gebesee">
      <Veranstaltungen_Showacts Titel="" Vorname="Anna-Maria" Name="Schenkel"
Firma="">
        <Veranstaltungen_Showact_Kategorie Kategorie="Krimiautorin" />
      </Veranstaltungen_Showacts>
      <Veranstaltungen_Showacts Titel="" Vorname="Martin" Name="Meisner" Firma="">
        <Veranstaltungen_Showact_Kategorie Kategorie="Krimiautor" />
      </Veranstaltungen_Showacts>
      <Veranstaltungen_Showacts Titel="" Vorname="Peter" Name="Decker" Firma="">
        <Veranstaltungen_Showact_Kategorie Kategorie="Kriminalhauptkomissar" />
      </Veranstaltungen_Showacts>
    </Veranstaltungen_Ort_Standort>
  </Veranstaltungen_Ort>
</Veranstaltungen_Daten>
<Veranstaltungen_Daten Veranstaltung="Museumsnacht Weimar " Startdatum="2015-06-
10" Enddatum="2015-06-10">
  <Veranstaltungen_Ort Veranstaltungsort="">
    <Veranstaltungen_Ort_Standort Standort="Kranichfeld">
      <Veranstaltungen_Showacts Titel="" Vorname="Erwin" Name="Roth" Firma="">
        <Veranstaltungen_Showact_Kategorie Kategorie="Klangkunst" />
      </Veranstaltungen_Showacts>
      <Veranstaltungen_Showacts Titel="" Vorname="" Name="Blue Moon" Firma="">
        <Veranstaltungen_Showact_Kategorie Kategorie="Jazzkombo" />
      </Veranstaltungen_Showacts>
      <Veranstaltungen_Showacts Titel="" Vorname="Bas" Name="Böttcher" Firma="">
        <Veranstaltungen_Showact_Kategorie Kategorie="Poetry Slam" />
      </Veranstaltungen_Showacts>
      <Veranstaltungen_Showacts Titel="" Vorname="" Name="FolklOHR" Firma="">
```

```
      <Veranstaltungen_Showact_Kategorie Kategorie="Band" />
     </Veranstaltungen_Showacts>
    </Veranstaltungen_Ort_Standort>
  </Veranstaltungen_Ort>
</Veranstaltungen_Daten>
<Veranstaltungen_Daten Veranstaltung="Museumsnacht Weimar " Startdatum="2016-06-
05" Enddatum="2016-06-05">
  <Veranstaltungen_Ort Veranstaltungsort="">
    <Veranstaltungen_Ort_Standort Standort="Kranichfeld">
     <Veranstaltungen_Showacts Titel="" Vorname="Erwin" Name="Roth" Firma="">
      <Veranstaltungen_Showact_Kategorie Kategorie="Klangkunst" />
     </Veranstaltungen_Showacts>
     <Veranstaltungen_Showacts Titel="" Vorname="Ben" Name="Becker" Firma="">
      <Veranstaltungen_Showact_Kategorie Kategorie="Sprecher" />
     </Veranstaltungen_Showacts>
     <Veranstaltungen_Showacts Titel="" Vorname="" Name="Amsterdam Klezmer"
Firma="">
      <Veranstaltungen_Showact_Kategorie Kategorie="Band" />
     </Veranstaltungen_Showacts>
     <Veranstaltungen_Showacts Titel="" Vorname="" Name="StreichZart" Firma="">
      <Veranstaltungen_Showact_Kategorie Kategorie="Klassik" />
     </Veranstaltungen_Showacts>
    </Veranstaltungen_Ort_Standort>
  </Veranstaltungen_Ort>
</Veranstaltungen_Daten>
<Veranstaltungen_Daten Veranstaltung="Speeddays" Startdatum="2016-10-01"
Enddatum="2016-10-01">
  <Veranstaltungen_Ort Veranstaltungsort="Flugplatz">
    <Veranstaltungen_Ort_Standort Standort="Arnstadt-Alkersleben">
     <Veranstaltungen_Showacts Titel="" Vorname="Dieter" Name="Zucker" Firma="">
      <Veranstaltungen_Showact_Kategorie Kategorie="Moderation" />
     </Veranstaltungen_Showacts>
     <Veranstaltungen_Showacts Titel="" Vorname="Voker" Name="Ziemann" Firma="">
      <Veranstaltungen_Showact_Kategorie Kategorie="DJ" />
     </Veranstaltungen_Showacts>
    </Veranstaltungen_Ort_Standort>
  </Veranstaltungen_Ort>
</Veranstaltungen_Daten>')
```

Für diese XML- Instanz zeigt Abbildung 6 ein allgemeines XML-Schema.

XML-Instanz1.xsd | XML-Instanz1.xml | SQL_XML.sql - PAU...PAULSPC\Paul (56)) | SQL_Trigger_View.s...PAULSPC\Paul (55))*

```xml
<?xml version="1.0" encoding="utf-8"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="Veranstaltungen_Daten">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="Veranstaltungen_Ort">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="Veranstaltungen_Ort_Standort">
                                <xs:complexType>
                                    <xs:sequence>
                                        <xs:element maxOccurs="unbounded" name="Veranstaltungen_Showacts">
                                            <xs:complexType>
                                                <xs:sequence>
                                                    <xs:element name="Veranstaltungen_Showact_Kategorie">
                                                        <xs:complexType>
                                                            <xs:attribute name="Kategorie" type="xs:string" use="required" />
                                                        </xs:complexType>
                                                    </xs:element>
                                                </xs:sequence>
                                                <xs:attribute name="Titel" type="xs:string" use="required" />
                                                <xs:attribute name="Vorname" type="xs:string" use="required" />
                                                <xs:attribute name="Name" type="xs:string" use="required" />
                                                <xs:attribute name="Firma" type="xs:string" use="required" />
                                            </xs:complexType>
                                        </xs:element>
                                    </xs:sequence>
                                    <xs:attribute name="Standort" type="xs:string" use="required" />
                                </xs:complexType>
                            </xs:element>
                        </xs:sequence>
                        <xs:attribute name="Veranstaltungsort" type="xs:string" use="required" />
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
            <xs:attribute name="Veranstaltung" type="xs:string" use="required" />
            <xs:attribute name="Startdatum" type="xs:date" use="required" />
            <xs:attribute name="Enddatum" type="xs:date" use="required" />
        </xs:complexType>
    </xs:element>
</xs:schema>
```

Abbildung 6: XML-Schema

7 Fazit

Die Erstellung einer relationalen Datenbank erfordert es einen genauen Projekt- und Zeitplan zur umfassenden Erfüllung aller gestellten Anforderungen.
Der Projektstart war im November 2016, bei dem zunächst die Anforderungsprofile auf Seiten des Datenbankentwurfs diskutiert wurden. Dabei stand immer im Fokus eine leistungsfähige und gut strukturierte Datenbank zu entwerfen. Die Implementierung in Microsoft SQL-Server 2016 startete Ende November und konnte bereits Mitte Dezember abgeschlossen werden. Daraufhin lag der Fokus auf den SQL-Abfragen und der Rechte- und Rollenverwaltung. Auch dieser Projektschritt konnte im Dezember 2016 noch abgeschlossen werden. Die Erstellung von XML-Daten aus bereits vorrätigen Daten in Tabellen der RDBMS EVENTUS erfolgte Anfang Januar. Als Resultat dieses Projektabschnittes, ergab sich ein XML-Schema, welches als Validierungsschema für zukünftig erstellbare Flyer denkbar wäre.

Die fertiggestellte Datenbank mit ihren zusätzlichen Features kann nun in andere Software eingebettet werden. Es ist beispielsweise denkbar, die Datenbank im Hintergrund an ein Programm zum Veranstaltungsmanagement zu koppeln, wodurch die Bearbeitung von veranstaltungsrelevanten Daten schneller, sicherer und weniger fehleranfällig ist. Ein Beispielprogramm, welches ein Grundmodul zur Veranstaltungsdatenpflege bereitstellt, wurde im Rahmen des Projekts entwickelt und soll als Ausblick dienen. Die Anwendung enthält zusätzlich noch ein Modul zur Mitarbeiterverwaltung, indem sämtliche relevanten mitarbeiterbezogenen Daten hinterlegt werden können. Ein drittes Modul umfasst den Aspekt der Standard- und Benutzerabfragen, indem eine Auswahl an Standardabfragen zur Verfügung stehen aber auch über ein Eingabefeld individuelle Abfragen gestellt werden können.

Mithilfe des Anwendungsprogramms wird die Schnittstelle zwischen Datenbankentwicklung und Anwendbarkeit durch Dritte (z.B. Unternehmen) aufgezeigt.