# SQL Scripts

2.1 SQL script to create the database structure in SQL server

CREATE TABLE Booking (

booking\_number INTEGER NOT NULL IDENTITY,

event\_id INTEGER NOT NULL,

status VARCHAR(20) NOT NULL,

datetime DATETIME NOT NULL,

booking\_memo VARCHAR(50),

quantity INTEGER NOT NULL,

CONSTRAINT pk\_Booking PRIMARY KEY(booking\_number),

CONSTRAINT fk\_Booking FOREIGN KEY(event\_id) REFERENCES Event(event\_id),

CONSTRAINT ck\_Booking CHECK(status IN ('reserved', 'refunded', 'sold'))

);

CREATE TABLE Event (

event\_id INTEGER NOT NULL IDENTITY,

ticket\_price FLOAT NOT NULL,

status VARCHAR(20) NOT NULL,

start\_date DATE NOT NULL,

end\_date DATE NOT NULL,

type VARCHAR(50) NOT NULL,

name VARCHAR(50) NOT NULL,

venue\_id INTEGER NOT NULL,

CONSTRAINT pk\_Event PRIMARY KEY(event\_id),

CONSTRAINT fk\_Event FOREIGN KEY(venue\_id) REFERENCES Venue(venue\_id),

CONSTRAINT ck\_Event CHECK(status IN ('cancelled', 'running'))

) ;

CREATE TABLE Artist (

artist\_id INTEGER NOT NULL IDENTITY,

firstname VARCHAR(50) NOT NULL,

lastname VARCHAR(50),

country VARCHAR(50) NOT NULL,

phone\_number VARCHAR(20) NOT NULL,

rider VARCHAR(200),

CONSTRAINT pk\_Artist PRIMARY KEY(artist\_id)

) ;

CREATE TABLE Venue (

venue\_id INTEGER NOT NULL IDENTITY,

venue\_name VARCHAR(50) NOT NULL,

seat\_capacity INTEGER NOT NULL,

CONSTRAINT PK\_Venue PRIMARY KEY(venue\_id),

);

CREATE TABLE Event\_Artist (

event\_id INTEGER NOT NULL,

artist\_id INTEGER NOT NULL,

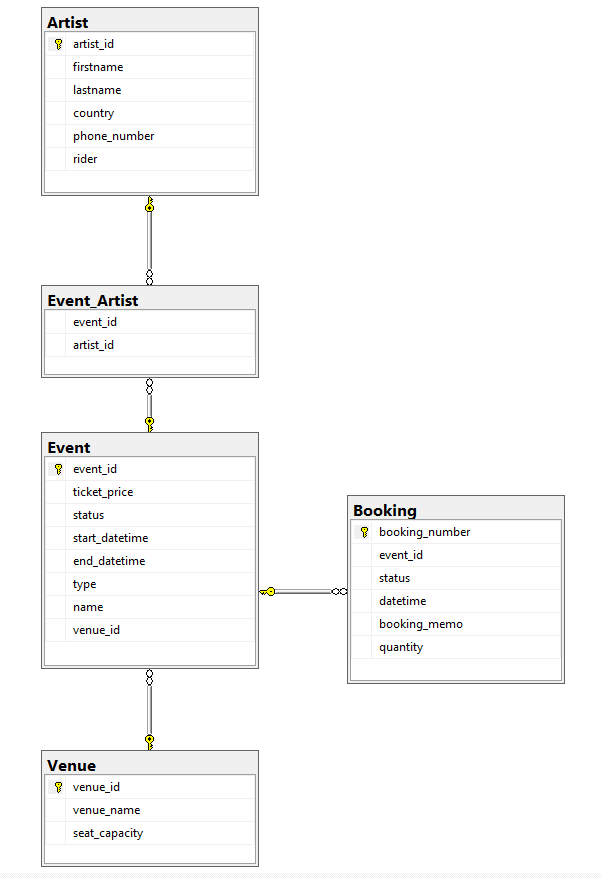
CONSTRAINT pk\_Event\_Artist PRIMARY KEY(event\_id, artist\_id),

CONSTRAINT fk\_Event\_Artist\_Event FOREIGN KEY(event\_id) REFERENCES Event(event\_id),

CONSTRAINT fk\_Event\_Artist\_Artist FOREIGN KEY(artist\_id) REFERENCES Artist(artist\_id)

);

2.2 Database Diagram



*Pic. 2. Database Diagram*

2.3 SQL script to create the indexes

The following indexes are automatically created by SQL Server: pk\_Artist, pk\_Booking, pk\_Event, pk\_Event\_Artist, pk\_Venue. Indexes to create are:

CREATE INDEX ix\_event\_artist\_event ON Event\_Artist(event\_id);

CREATE INDEX ix\_event\_artist\_artist ON Event\_Artist(artist\_id);   
CREATE INDEX ix\_event\_booking ON Booking(event\_id);   
CREATE INDEX ix\_event\_venue ON Event(venue\_id);

2.4 SQL script to populate the database with data for testing

Booking:

insert into Booking (event\_id, status, datetime, booking\_memo, quantity)

values ('1', 'reserved', '2018-12-11 15:20:00 ', '123456789', '2');

insert into Booking (event\_id, status, datetime, booking\_memo, quantity)

values ('2', 'refunded', '2018-12-12 14:10:12', NULL, '4');

insert into Booking (event\_id, status, datetime, booking\_memo, quantity)

values ('3', 'sold', '2018-12-10 12:15:00', '123456788', '5');

Event:

insert into Event (ticket\_price, status, start\_datetime, end\_datetime, type, name, venue\_id)

values ('120.50', 'running', '2019-01-11 12:00:00', '2019-01-12 17:00:00', 'ice-show', 'Plyuschenko and co', '1');

insert into Event (ticket\_price, status, start\_datetime, end\_datetime, type, name, venue\_id)

values ('78.00', 'running', '2019-01-02 15:00:00', '2019-01-02 18:00:00', 'concert', 'Britney Spears Show', '1');

insert into Event (ticket\_price, status, start\_datetime, end\_datetime, type, name, venue\_id)

values ('12.50', 'cancelled', '2019-02-11 12:00:00', '2019-02-11 15:00:00', 'movie', 'Smth about love', '2');

insert into Event (ticket\_price, status, start\_datetime, end\_datetime, type, name, venue\_id)

values ('60.50', 'cancelled', '2019-02-11 12:00:00', '2019-02-11 15:00:00', 'con-cert', 'Suomi maa', '2');

insert into Event (ticket\_price, status, start\_datetime, end\_datetime, type, name, venue\_id)

values ('45.50', 'running', '2018-06-30 13:00:00', '2018-07-11 16:30:00', 'dance', 'Finnish summer', '1');

insert into Event (ticket\_price, status, start\_datetime, end\_datetime, type, name, venue\_id)

values ('15.50', 'running', '2018-12-10 13:00:00', '2018-12-20 16:30:00', 'con-cert', 'Rock today', '2');

Venue:

insert into Venue (venue\_name, seat\_capacity)

values ('Grand Arena', '3500');

insert into Venue (venue\_name, seat\_capacity)

values ('Small Arena', '70');

Artist:

insert into Artist (firstname, lastname, country, phone\_number, rider)

values ('Britney', 'Spears', 'USA', '1234566', 'Valio milk, Paulig coffee, Fazer chocolate, white roses');

insert into Artist (firstname, lastname, country, phone\_number, rider)

values ('John', 'Travolta', 'USA', '1234568', NULL);

insert into Artist (firstname, lastname, country, phone\_number, rider)

values ('Evgeny', 'Plyuschenko', 'Russia', '1237568', 'Valio milk, Paulig coffee, Fazer chocolate, karelian pies');

insert into Artist (firstname, lastname, country, phone\_number, rider)

values ('Adele', 'Winston', 'UK', '74574743', 'Indian tea, Karellian pirakka, allergy to any flowers');

insert into Artist (firstname, lastname, country, phone\_number, rider)

values ('Saara', 'Aalto', 'Finland', '4543536', 'Irish coffee, fruits, roses');

insert into Artist (firstname, lastname, country, phone\_number, rider)

values ('Jorma', ' Uotinen', 'Finland', '9768567', 'coffee, meet balls, any flowers');

insert into Artist (firstname, lastname, country, phone\_number, rider)

values ('ZZ Top', NULL, 'USA', '546547', NULL);

Event\_Artist:

insert into Event\_Artist (event\_id, artist\_id)

values (2, 1);

insert into Event\_Artist (event\_id, artist\_id)

values (1, 3);

insert into Event\_Artist (event\_id, artist\_id)

values (4, 5);

insert into Event\_Artist (event\_id, artist\_id)

values (5, 6);

insert into Event\_Artist (event\_id, artist\_id)

values (6, 7);

--checking constraints out--

insert into Booking (event\_id, status, datetime, booking\_memo, ticket\_number)

values ('1', 'hello', '2018-12-11 13:00:00', '123456789', '123');

insert into Event (ticket\_price, status, start\_datetime, end\_datetime, type, name, venue\_id)

values ('120.50', 'hello', '2019-01-11', '2019-01-11', 'ice-show', 'Plyuschenko and co', '1');

2.5 SQL script to test the database

--1.

UPDATE Booking

SET status = 'reserved'

WHERE booking\_number = 7

--2.

UPDATE Booking

SET status = 'refunded'

WHERE booking\_number = 7

--3.

UPDATE Booking

SET quantity = 6

WHERE booking\_number = 8

--4.

UPDATE Booking

SET status = 'sold'

WHERE booking\_number = 7

--5.

DELETE FROM Booking

WHERE status = 'reserved' AND DATEDIFF(DAY, datetime, GETDATE()) >= 3

--6.

DELETE Booking FROM Booking

JOIN Event ON (Event.event\_id = Booking.event\_id)

WHERE Booking.status NOT LIKE 'sold' AND Event.end\_datetime < (SELECT GETDATE())

--7.

UPDATE Event

SET status = 'cancelled'

WHERE event\_id = 1

--8.

UPDATE Booking

SET status = 'refunded'

WHERE event\_id IN (SELECT event\_id FROM Event WHERE status = 'cancelled')

--9.

SELECT name FROM Event

WHERE DATENAME(MONTH, start\_datetime) = 'January'

--10.

SELECT name FROM Event

WHERE DATENAME(MONTH, start\_datetime) = 'January' AND type = 'theater'

--11.

SELECT phone\_number FROM Artist

WHERE firstname = 'Adele'

--12.

SELECT firstname, lastname, rider, Event.start\_datetime FROM Artist

JOIN Event\_Artist ON (Artist.artist\_id = Event\_Artist.artist\_id)

JOIN Event ON (Event.event\_id = Event\_Artist.event\_id)

WHERE lastname = 'Aalto' AND firstname = 'Saara'

--13.

SELECT SUM(quantity) FROM Booking

JOIN Event ON (Event.event\_id = Booking.event\_id)

JOIN Event\_Artist ON (Booking.event\_id = Event\_Artist.event\_id)

JOIN Artist ON (Artist.artist\_id = Event\_Artist.artist\_id)

WHERE Artist.lastname = 'Uotinen' AND Artist.firstname = 'Jorma' AND Event.name = 'Finnish summer' AND Event.start\_datetime = '2018-06-30' AND type = 'dance'

--14.

SELECT (Venue.seat\_capacity - Booking.quantity) FROM Booking

JOIN Event ON (Event.event\_id = Booking.event\_id)

JOIN Event\_Artist ON (Booking.event\_id = Event\_Artist.event\_id)

JOIN Artist ON (Artist.artist\_id = Event\_Artist.artist\_id)

JOIN Venue ON (Event.venue\_id = Venue.venue\_id)

WHERE firstname = 'ZZ Top' AND Event.type = 'con-cert' AND Event.start\_datetime = '2018-12-10'

--15.

SELECT SUM(ticket\_price\*quantity) FROM Event

JOIN Booking ON (Booking.event\_id = Event.event\_id)

WHERE YEAR(datetime) = YEAR(GETDATE())