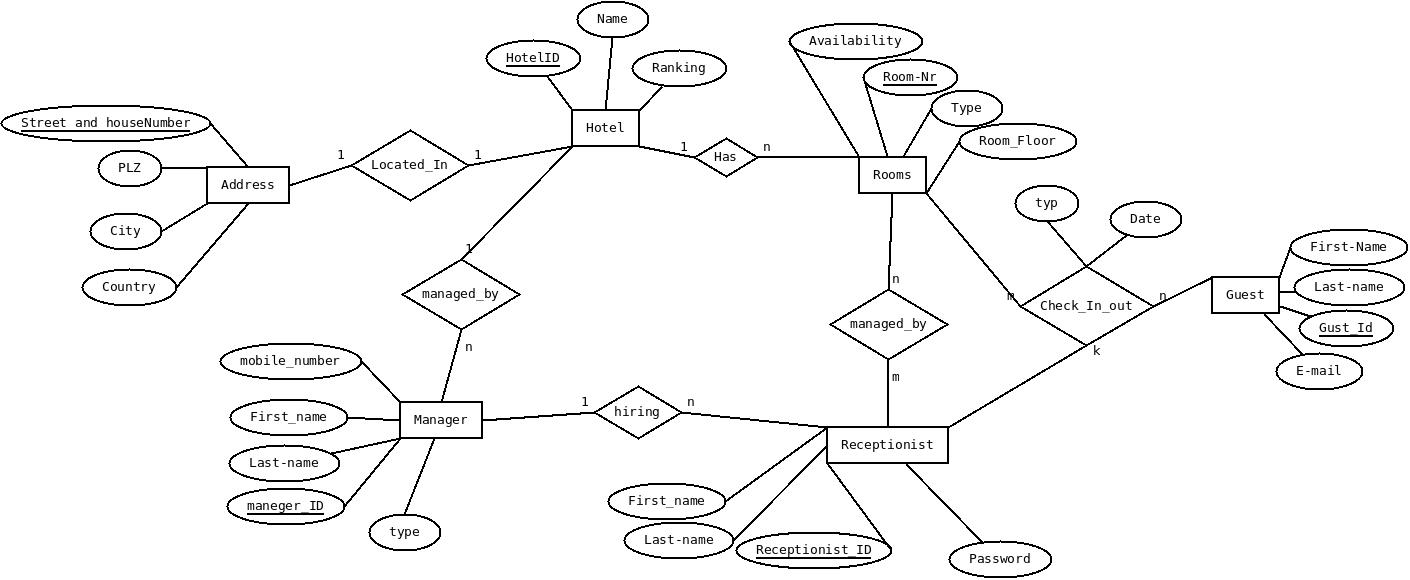
**Datenbanken Belegarbeit**

Khaled Halabieh (556290)

Georg Westerholt (556752)

­

Im folgenden Entity-Relationship-Modell wird das skalierbare Datenbankensystem eines Hotels dargestellt. Das Hotel bekommt Buchungsanfragen und teilt den Gesten die passenden Zimmer zu.   
Die Datenbank erfasst das Hotel, dessen Adresse und Räume, Manager, Rezeptionisten und Gäste.  
Rezeptionisten managen die Räume und kümmern sich um den Check- In/Out der Gäste. Die Rezeptionisten werden von einem Hotelmanager eingestellt, welcher zusätzlich das Hotel managet. Das Hotel hat mehrere Zimmer für die Gäste sowie eine Adresse.



Der Implementationsentwurf des DBS ergibt sich nun aus dem ER-Modell:

Hotel:

|  |  |  |
| --- | --- | --- |
| Name | Rank | HotelID |
| Sql Hotel | 4 | 1 |
| Kalix Hotel | 1 | 2 |

Address:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Street\_and\_hauseNumber | PLZ | City | Country | HotelId |
| SqlStrasse 22 | 10212 | Berlin | Germany | 1 |
| Nollendorfstr.21A | 10777 | Berlin | Germany | 2 |

Rooms:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Room\_nr | Availability | Type | Room\_floor | HotelId |
| 1 | false | King | 1 | 1 |
| 2 | false | Double | 1 | 1 |
| 3 | false | Single | 1 | 1 |
| 4 | false | Double | 2 | 1 |
| 5 | true | Double | 2 | 1 |
| 6 | true | Single | 2 | 1 |
| 7 | true | Single | 3 | 1 |
| 8 | true | Single | 3 | 1 |
| 10 | true | King | 1 | 2 |
| 11 | true | Double | 1 | 2 |
| 12 | true | Single | 2 | 2 |

Manager:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Manager\_ID | First\_name | Last\_name | mobile\_number | Type | HotelId |
| 1 | Khaled | Halabieh | 556290 | Operations Management | 1 |
| 2 | Georg | Westerholt | 556752 | Financial Management | 1 |
| 3 | Kal | Hal | 1234567 | human resources | 2 |

Receptionist:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Receptionist\_ID | First\_name | Last\_name | Manager\_ID | Password |
| 1 | Donald | Duck | 1 | 123 |
| 2 | Jackie | chan | 2 | 123 |
| 3 | anna | laza | 3 | 123 |
| 4 | dani | mark | 3 | 123 |

managed\_by:

|  |  |
| --- | --- |
| Receptionist\_ID | Room\_nr |
| 1 | 1 |
| 1 | 2 |
| 1 | 3 |
| 1 | 4 |
| 1 | 5 |
| 2 | 6 |
| 2 | 7 |
| 2 | 8 |
| 3 | 10 |
| 3 | 11 |
| 4 | 12 |

Guest:

|  |  |  |  |
| --- | --- | --- | --- |
| Guest\_ID | First\_name | Last\_name | E\_mail |
| 1 | Salim | Make | salem@make.com |
| 2 | Lena | mert | Lena@mert.com |
| 3 | Karla | zeit | Karla@zeit.com |
| 4 | Modi | Schneider | Modi@Schneider.com |

Check\_in\_out:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Date | Type | Guest\_ID | Room\_nr | Receptionist\_ID |
| 2017-06-17 | Check\_in | 1 | 1 | 1 |
| 2017-04-10 | Check\_in | 2 | 2 | 1 |
| 2017-05-1 | Check\_in | 3 | 3 | 1 |
| 2017-06-1 | Check\_in | 4 | 4 | 2 |
| 2017-05-12 | Check\_out | 3 | 3 | 1 |
| 2017-06-16 | Check\_out | 4 | 4 | 2 |

Mit folgendem SQL-Skript wird das Datenbankensystem erstellt:

CREATE TABLE IF NOT EXISTS Hotel (

name varchar(100) NOT NULL,

Rank Integer NOT NULL ,

HotelId Integer NOT NULL,

PRIMARY KEY (HotelId)

) ;

CREATE TABLE IF NOT EXISTS Address (

Street\_and\_hauseNumber varchar(100) NOT NULL,

PLZ Integer NOT NULL ,

City varchar(20) NOT NULL,

Country varchar(20) NOT NULL,

HotelId Integer NOT NULL,

PRIMARY KEY (Street\_and\_hauseNumber),

FOREIGN KEY (HotelId) REFERENCES Hotel(HotelId)

) ;

CREATE TABLE IF NOT EXISTS Rooms(

Room\_nr Integer NOT NULL,

Availability boolean NOT NULL,

Type varchar(30) NOT NULL,

Room\_floor Integer NOT NULL,

HotelId Integer NOT NULL,

PRIMARY KEY(Room\_nr),

FOREIGN KEY (HotelId) REFERENCES Hotel(HotelId)

);

CREATE TABLE IF NOT EXISTS Manager(

Manager\_ID Integer NOT NULL,

First\_name varchar(20) NOT NULL,

Last\_name varchar(20) NOT NULL,

mobile\_number Bigint NOT NULL,

Type varchar(50) default 'Manager',

HotelId Integer NOT NULL,

PRIMARY KEY(Manager\_ID),

FOREIGN KEY (HotelId) REFERENCES Hotel(HotelId)

);

CREATE TABLE IF NOT EXISTS Receptionist (

Receptionist\_ID Integer NOT NULL,

First\_name varchar(20) NOT NULL,

Last\_name varchar(20) NOT NULL,

Password varchar(20) NOT NULL,

Manager\_ID Integer NOT NULL,

PRIMARY KEY(Receptionist\_ID),

FOREIGN KEY (Manager\_ID) REFERENCES Manager(Manager\_ID)

);

CREATE TABLE IF NOT EXISTS managed\_by(

Receptionist\_ID Integer NOT NULL,

Room\_nr Integer NOT NULL,

FOREIGN KEY (Receptionist\_ID) REFERENCES Receptionist(Receptionist\_ID),

FOREIGN KEY (Room\_nr) REFERENCES Rooms(Room\_nr)

);

CREATE TABLE IF NOT EXISTS Guest(

Guest\_ID Integer NOT NULL,

First\_name varchar(20) NOT NULL,

Last\_name varchar(20) NOT NULL,

E\_mail varchar(30) NOT NULL,

PRIMARY KEY(Guest\_ID)

);

CREATE TABLE IF NOT EXISTS Check\_in\_out(

Date date NOT NULL DEFAULT CURRENT\_TIMESTAMP,

Type varchar (10) NOT NULL,

Guest\_ID Integer NOT NULL REFERENCES Guest(Guest\_ID),

Room\_nr Integer NOT NULL REFERENCES Rooms(Room\_nr),

Receptionist\_ID Integer NOT NULL REFERENCES Receptionist(Receptionist\_ID),

CONSTRAINT Check\_in\_out\_pkey PRIMARY KEY(Guest\_ID, Room\_nr, Receptionist\_ID,Type)

);

Folgendes Skript fügt nun Datensätze in die Datenbank ein:

INSERT INTO Hotel(name , Rank , HotelId) VALUES

('Sql Hotel' , 4 , 1),

('kalix Hotel' , 1 , 2);

INSERT INTO Address(Street\_and\_hauseNumber , PLZ, City , Country , HotelId) VALUES

('SqlStrasse 22' , 10212 , 'Berlin', 'Germany' , 1),

('Nollendorfstr.21A' , 10777 , 'Berlin' , 'Germany' , 2);

INSERT INTO Rooms(Room\_nr , Availability ,Type ,Room\_floor ,HotelId) VALUES

(1, false ,'King' , 1 , 1),

(2, false ,'Double' , 1 , 1),

(3, false ,'Single' , 1 , 1),

(4, false ,'Double' , 2 , 1),

(5, true ,'Double' , 2 , 1),

(6, true ,'Single' , 2 , 1),

(7, true ,'Single' , 3 , 1),

(8, true ,'Single' , 3 , 1),

(10, true ,'King' , 1 , 2),

(11, true ,'Double' , 1 , 2),

(12, true ,'Single' , 2 , 2);

INSERT INTO Manager(Manager\_ID ,First\_name , Last\_name ,mobile\_number , Type ,HotelId ) VALUES

(1 ,'Khaled' , 'Halabieh' , 556290 ,'Operations Management' , 1 ),

(2 ,'Georg' , 'Westerholt' , 556752 ,'Financial Management' , 1 ),

(3 ,'Kal' , 'Hal' , 1234567 ,'human resources', 2);

INSERT INTO Receptionist (Receptionist\_ID, First\_name, Last\_name , Manager\_ID , Password) VALUES

(1 , 'Donald' , 'Duck' , 1,123),

(2 , 'Jackie' , 'chan' , 2,123),

(3 , 'anna' , 'laza' , 3,123),

(4 , 'dani' , 'mark' , 3,123);

INSERT INTO managed\_by (Receptionist\_ID , Room\_nr) VALUES

(1 , 1),

(1 , 2),

(1 , 3),

(1 , 4),

(1 , 5),

(2 , 6),

(2 , 7),

(2 , 8),

(3 , 10),

(3 , 11),

(4 , 12);

INSERT INTO Guest(Guest\_ID ,First\_name , Last\_name ,E\_mail ) VALUES

(1 , 'Salim' , 'Make' , 'salem@make.com'),

(2 , 'Lena' , 'mert' , 'Lena@mert.com'),

(3 , 'Karla' , 'zeit' , 'Karla@zeit.com'),

(4 , 'Modi' , 'Schneider' , 'Modi@Schneider.com');

INSERT INTO Check\_in\_out(Date , Type ,Guest\_ID , Room\_nr , Receptionist\_ID ) VALUES

(DATE '2017-06-17' , 'Check\_in' , 1 , 1 , 1),

(DATE '2017-04-10' , 'Check\_in' , 2 , 2 , 1),

(DATE '2017-05-1' , 'Check\_in' , 3 , 3 , 1),

(DATE '2017-06-1' , 'Check\_in' , 4 , 4 , 2),

(DATE '2017-05-12' , 'Check\_out' , 3 , 3 , 1),

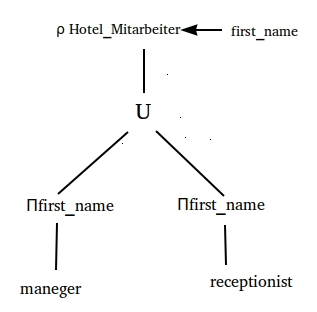
(DATE '2017-06-16' , 'Check\_out' , 4 , 4 , 2);

1- UNION:

- Alle Mitarbeiter namen ( manager und receptionist) die im Hotel arbeiten .

select first\_name AS "Hotel\_Mitarbeiter" from manager UNION (select first\_name from receptionist) ;

Π*ρ Hotel\_Mitarbeiter* (first\_name) (maneger) ∪ (Πfirst\_name (receptionist))



2- JOIN

a-

Listet für jeden Receptionist die jeweiligen Räume auf die er managet.

select re.first\_name, ro.room\_nr , ro.type from rooms as ro natural join receptionist as re natural join managed\_by

b-

Listet Check-In und Check-out von allen Gästen mit Datum

select g.First\_name , c.Type , c.date from Guest as g LEFT JOIN Check\_in\_out as c on g.Guest\_ID = c.Guest\_ID

3.exists

Listet die Gäse auf die bereits ausgecheckt sind.

select g.First\_name ,g.last\_name from Guest as g where EXISTS ( select \* from Check\_in\_out as c where c.guest\_id=g.guest\_id and c.type='Check\_out' )

b-NOT EXISTS

Listet die noch unbelegten Räume auf.

select r.Room\_nr , r.type from rooms as r where NOT EXISTS (select \* from check\_in\_out as c where c.Room\_nr = r.Room\_nr)

4. COUNT

Listet die Anzahl an Manager pro Hotel auf

select h.name , COUNT(m.HotelId) AS "Nr.Managers" from Manager as m natural join Hotel as h group by h.HotelId

b

Listet die Anzahl der Räume pro Hotel auf.

select h.name , COUNT(r.Room\_nr) AS "Nr.Rooms" from Rooms as r natural join Hotel as h group by h.HotelId ;

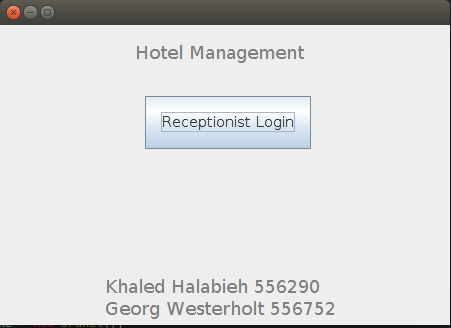
5. HAVING:

Listet alle Hotels auf die mehr als 5 Räume haben

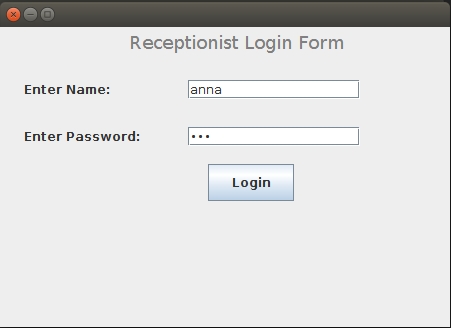
select h.name from Rooms as r natural join Address as a natural join Hotel as h group by h.HotelId HAVING COUNT(r.Room\_nr)>5

**Beschreibung des Programms zur bearbeitungen der Tabelle Guest**

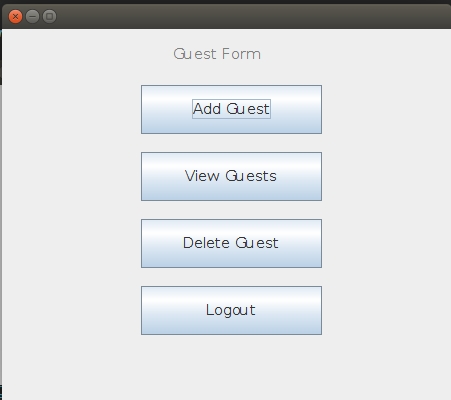
1. Fenster mit Button zu den Login Fenster des Rezeptionisten



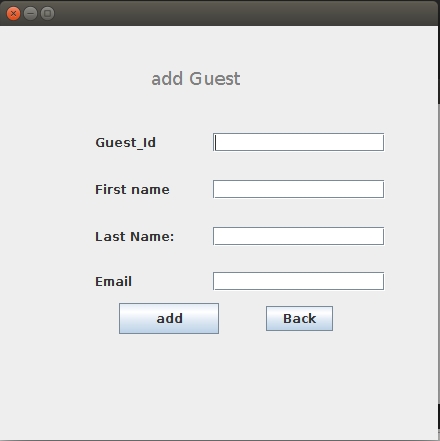
1. Login Fenster  
   ‚



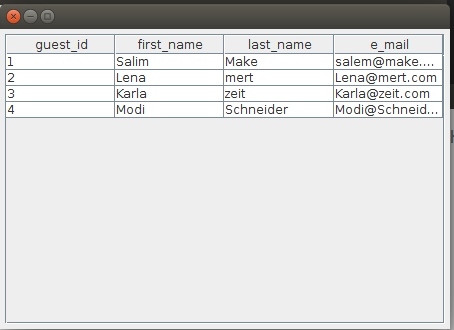
1. Menüfenster mit den Buttons „Add Guest“, „View Guests“, „Delete Guest“ und „Logout“



1. Fenster „Add Guest“ zum hinzufügem eines neuen Gastes mit den gewünschten Daten



1. Fenster „View Guest“ mit der Geste Tabelle



1. Fenster „Delete Guest“

