**Софийски университет ,,Св. Климент Охридски“**

Факултет по математика и информатика

Специалност: Информационни системи,

Курс 2, Група 1

**Курсов**

**проект**

по Бази данни

Изготвен от:

**Екип 3 – Хотел**

Веселина Чотрева

Соня Стоянова

Кристина Цекова

* **Дефиниране на задачата.**

Базата от данни за хотел съхранява информация за данните на служителите, гостите и направените регистрации в хотела –„ ХилТест“. Хотелът разполага с леглова база от 48 легла, разпределени в 24 стаи на 4 етажа. В базата от данни за хотел са включени седем множества същности.

**1. Описание на множествата същности.**

* **Rooms /Стаи/**

Множеството същности Rooms описва стая в хотел, като негови атрибути са roomNumber /номер на стая/, roomType /тип/ и pricePerNight /цена/. Ключовият атрибут е roomNumber. Стаите могат да са единични, двойни и апартаменти (тройни). В базата от данни те са означени с цифрите 1 – единична, 2 – двойна, 3 – апартамент. И трите атрибута са цели положителни числа.

* **Staff /Служители/**

Множеството от същности Staff характеризира служителите в хотела, като всички те имат следните атрибути: код на служител /ID/, име /name/, заплата /salary/, бонуси /bonuses/, дата на постъпване на работа /hireDate/, телефонен номер /phoneNumber/, имейл /email/. Заплатата и бонусите са положителни числа, датата на постъпване е от тип дата. ID е ключът и е представлява символен низ с точно 6 символа. Името е символен низ с дължина 50 символа. Телефонният номер и имейлът също са символни низове с дължини съответно 10 и 30 символа. Служителите могат да бъдат: Управители /Managers/, Рецепционисти/Receptionists/ и Камериери /Attendants/. Всеки от тях наследява множеството Staff, като добавя и свои собствени атрибути, които го отличават от останалите служители.

* **Managers /Управители/**

Множеството същности Managers описва харакетистиките на управители в хотел. Един управител отговаря за определени служители в хотела – както за камериерите, така и за рецепционистите. Мениджърът се отличава от останалите служители с фиксирана заплата, като също може да получава бонуси.

* **Receptionists /Рецепционисти/**

Множеството същности Receptionists описва рецепционист в хотел. То наследява множеството същности Staff, като добавя и собствен атрибут – образование /education/, което е от тип символен низ. Означенията му в базата от данни са следните: А – средно, В – средно специално, С – висше.

Рецепционистите отговарят за регистрирането на гостите и тяхното заплащане зависи от степента на образование, което притежават.

* **Attendants /Камериери/**

Множеството същности Attendants описва служителите в хотела, които отговарят за чистотата и състоянието на стаите. Освен наследените атрибути са добавени и: изработени часове /worked hours/ и ставка /moneyPerHour/. И двата трябва да положителни числа, но ставката не е задължително да е цяло число. Всеки камериер отговаря за определени стаи. Заплащането му се формира като броят на изработените часове се умножава по определен коефициент (ставка).

* **Guests /Гости/**

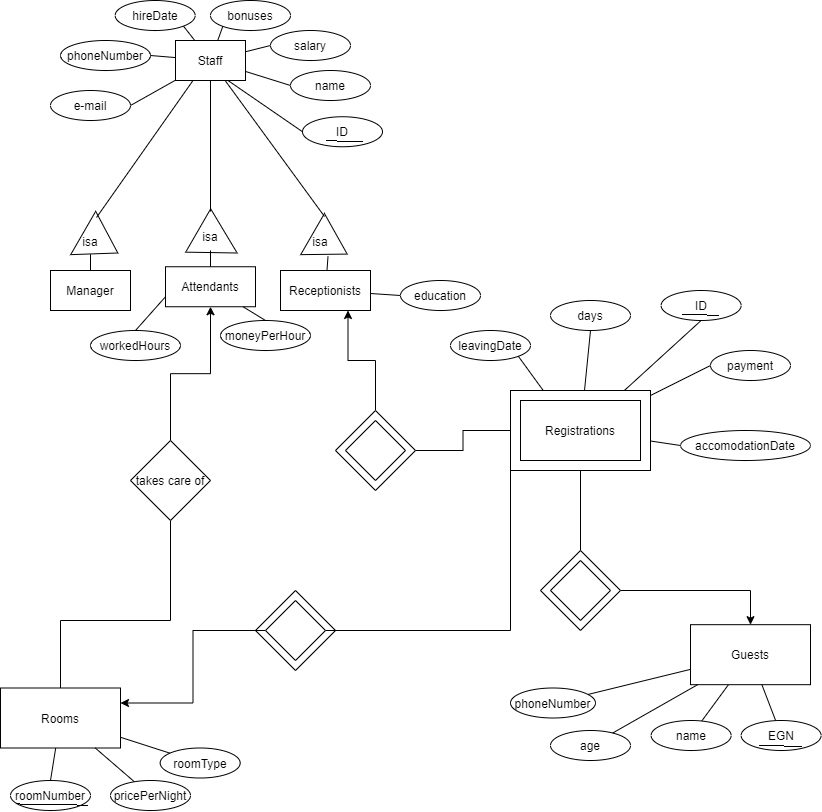
Това множество същности описва гостите в хотела. Те имат ЕГН /EGN/, име /name/, възраст /age/ и телефонен номер /phoneNumber/. ЕГН-то е символен низ с точно 10 символа. Това е и ключът. Името и телефонният номер също са символни низове.

* **Registrations /Регистрации/**

Това множество описва регистрациите в хотела. Негови атрибути са: номер на регистрация /ID/, дата на настаняване /accomodationDate/, дата на напускане /leavingDate/, начин на плащане /payment/ и престой в дни /days/.

Ключът тук е номерът на регистрацията, който е цяло положително число. Начинът на плащане в базата от данни се бележи с също с цяло положително число в интервала [0, 2], като 0 означава плащане в брой, 1 – плащане с карта и 2 – по банков път.

**2. E/R диаграма на модела на БД.**

****

**3. Преобразуване от E/R модел към релационен модел.**

В модела имаме една isa-йерархия. Нея преобразуваме с помощта на ООП-подхода, като накрая получаваме 8 релации:

***Staff*** *(ID, name, salary, bonus, emplDate, phoneNumber, e-mail)*

***Staff\_Manager*** *(ID, name, salary, bonuses, emplDate, phoneNumber, e-mail)*

***Staff\_Attendants*** *(ID, name, salary, bonuses, emplDate, phoneNumber, e-mail, workedHours, moneyPerHour)*

***Staff\_Receptionists*** *(ID, name, salary, bonuses, emplDate, phoneNumber, e-mail, education)*

***Staff\_Receptionists\_Managers****(ID,name, salary, bonuses, emplDate, phoneNumber, email, education)*

***Staff\_Receptionists\_Attendants*** *(ID, name, salary, bonuses, emplDate, phoneNumber, е-mail, education, workedHours, moneyPerHour)*

***Staff\_Managers\_Attendants*** *(ID, name, salary, bonuses, emplDate, phoneNumber, e-mail, workedHours, moneyPerHour)*

***Staff\_Manager\_Attendants\_Receptionists*** *(ID, name, salary, bonuses, emplDate, phoneNumber, e-mail, workedHours, moneyPerHour, education)*

Тъй като нямаме служител, който е едновременно управител и камериер, или управител и рецепционист, или камериер и рецепционист, както и управител, камериер и рецепционист, изключваме тези релации, защото са невалидни. Остават само три:

***Staff\_Manager*** *(ID, name, salary, bonuses, emplDate, phoneNumber, e-mail)*

***Staff\_Attendants*** *(ID, name, salary, bonuses, emplDate, phoneNumber, e-mail, workedHours, moneyPerHour)*

***Staff\_Receptionists*** *(ID, name, salary, bonuses, emplDate, phoneNumber, e-mail, education)*

След оптимизация на отношението takesCareOf цялостната релационна схема изглежда по следния начин:

***Rooms*** *(roomNumber, pricePerNight, roomType, attendantID)*

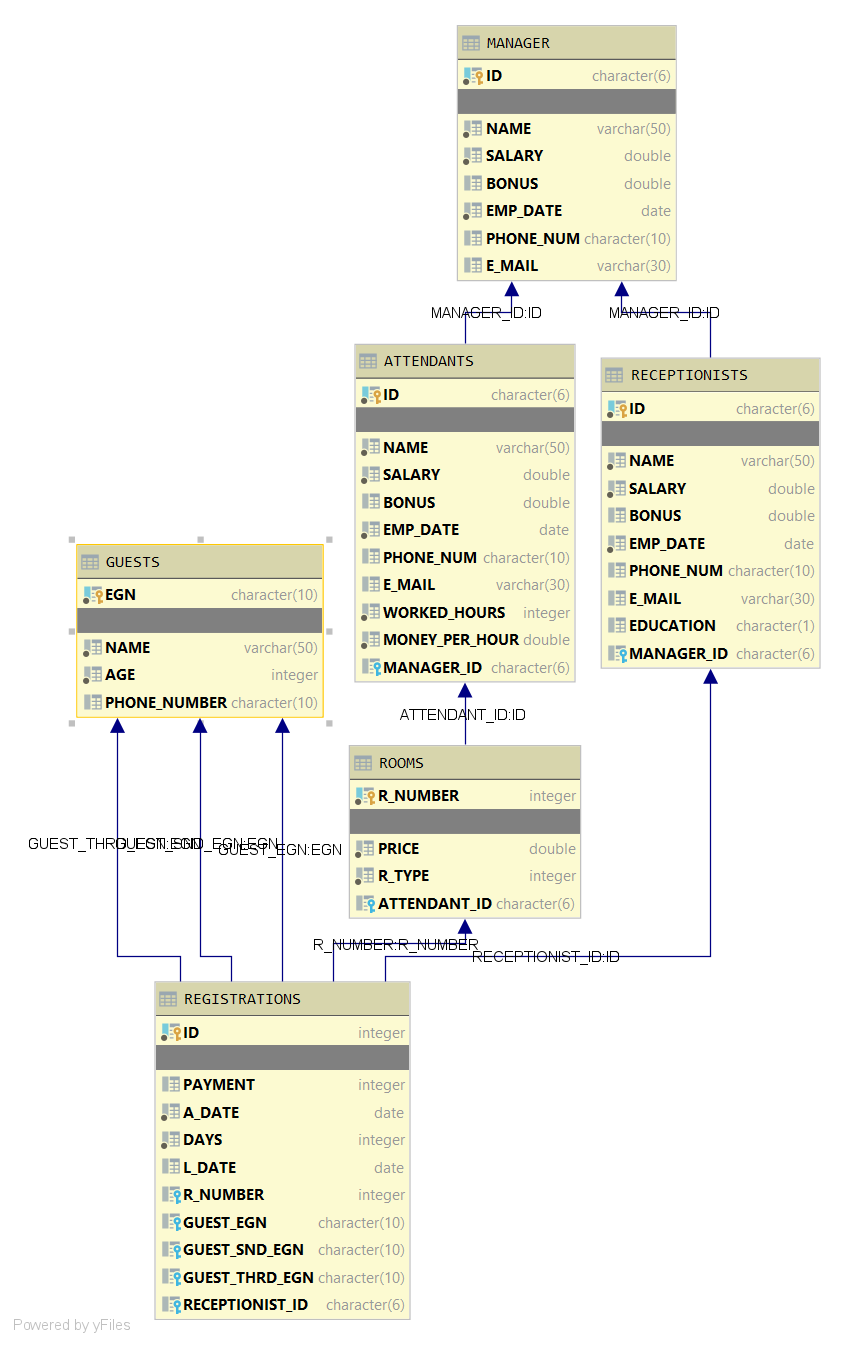
***Guests (****EGN, name, age, phoneNumber)*

***Registrations*** *(ID, payment, accomodationDate, days, leavingDate, roomNumber, guestEGN, guest2EGN , guest3EGN, receptionistID)*

***Manager*** *(ID, name, salary, bonus, emplDate, phoneNumber, e-mail)*

***Receptionists*** *(ID, name, salary, bonus, emplDate, phoneNumber, e-mail, education, managerID)*

***Attendants*** *(ID, name, salary, bonuses, emplDate, phoneNumber, e-mail, workedHours, moneyPerHour, managerID)*

**4. Картинка на релационния модел.**

**5. Описание на изгледите**.

Преди да създадем изгледите трябва да добавим стойности в таблиците от схемата ни. Това става с помощта на INSERT INTO <name-of-table> VALUES (<values-for-attributes>). Добавяме стойности за всички атрибути от таблиците, но може и да има такива с <null> стойности:

* Попълване на таблицата MANAGER

**INSERT INTO** MANAGER(ID, **NAME**, SALARY, BONUS, EMP\_DATE, PHONE\_NUM, E\_MAIL)  
**VALUES** (**'MNG092'**, **'Ivan Ivanov'**, 2500, 300, **'2015-05-01'**, **'0987645383'**, **'i.ivanov@gmail.com'**),  
 (**'MNG221'**, **'Petur Dimitrov'**, 2200, 400, **'2016-01-15'**, **'0987883283'**, **'p\_dimm@gmail.com'**),  
 (**'MNG733'**, **'Ivaila Mineva'**, 2900, 450, **'2015-05-01'**, **'0888821573'**, **'ivailla@gmail.com'**),  
 (**'MNG205'**, **'Maria Todorova'**, 2100, 200, **'2018-10-08'**, **'0887435383'**, **'mm\_ttva@gmail.com'**);

* Попълване на таблицата RECEPTIONISTS

**INSERT INTO** RECEPTIONISTS(ID, **NAME**, SALARY, BONUS, EMP\_DATE, PHONE\_NUM, E\_MAIL, EDUCATION, MANAGER\_ID)  
**VALUES** (**'REC924'**, **'Sonya Stoyanova'**, 800, 150, **'2019-03-02'**, **'0833145383'**, **'sonya.st@gmail.com'**,**'A'**, **'MNG092'**),  
 (**'REC014'**, **'Veselina Chotreva'**, 1000, 400, **'2012-01-15'**, **'0835883283'**, **'vess.chotreva@gmail.com'**, **'B'**, **'MNG221'**),  
 (**'REC621'**, **'Mirela Peneva'**, 1200, 450, **'2010-11-14'**, **'0943882157'**, **'mirela.geor@gmail.com'**, **'C'**, **'MNG733'**),  
 (**'REC872'**, **'Denitsa Petrova'**, 600, 100, **'2020-01-18'**, **'0817435383'**, **'d.petrova@gmail.com'**, **'B'**, **'MNG205'**);

* Попълване на таблицата ATTENDANTS

**INSERT INTO** ATTENDANTS(ID, **NAME**, SALARY,BONUS, EMP\_DATE, PHONE\_NUM,E\_MAIL,WORKED\_HOURS,MONEY\_PER\_HOUR,MANAGER\_ID)  
**VALUES** (**'ATT333'**,**'Iva Petrova'**, 800, 300, **'2015-04-04'**, **'0889675645'**, **'i.ivanova@gmail.com'**,**'8'**,700,**'MNG733'**),  
 (**'ATT133'**,**'Ivana Georgieva'**, 800, 300, **'2015-04-04'**, **'0834675645'**, **'ivana,ib@gmail.com'**,**'8'**,700,**'MNG733'**),  
 (**'ATT233'**,**'Stefan Stefanov'**, 800, 100, **'2015-03-08'**, **'085675645'**, **'st.stefanov@gmail.com'**,**'8'**,700,**'MNG205'**),  
 (**'ATT433'**,**'Kiril Metodiev'**, 800, 200, **'2015-04-07'**, **'0889675305'**, **'k.metodiev@gmail.com'**,**'8'**,700,**'MNG205'**);

* Попълване на таблицата ROOMS:

**INSERT INTO** ROOMS(R\_NUMBER, PRICE, R\_TYPE, ATTENDANT\_ID)  
**VALUES** (101, 150, 3, **'ATT333'**),  
 (102, 150, 3, **'ATT333'**),  
 (103, 70, 2, **'ATT333'**),  
 (104, 70, 2, **'ATT333'**),  
 (105, 30, 1, **'ATT333'**),  
 (106, 30, 1, **'ATT333'**),  
 (201, 150, 3, **'ATT133'**),  
 (202, 150, 3, **'ATT133'**),  
 (203, 70, 2, **'ATT133'**),  
 (204, 70, 2, **'ATT133'**),  
 (205, 30, 1, **'ATT133'**),  
 (206, 30, 1, **'ATT133'**),  
 (301, 150, 3, **'ATT233'**),  
 (302, 150, 3, **'ATT233'**),  
 (303, 70, 2, **'ATT233'**),  
 (304, 70, 2, **'ATT233'**),  
 (305, 30, 1, **'ATT233'**),  
 (306, 30, 1, **'ATT233'**),  
 (401, 150, 3, **'ATT433'**),  
 (402, 150, 3, **'ATT433'**),  
 (403, 70, 2, **'ATT433'**),  
 (404, 70, 2, **'ATT433'**),  
 (405, 30, 1, **'ATT433'**),  
 (406, 30, 1, **'ATT433'**);

* Попълване на таблицата GUESTS

**INSERT INTO** GUESTS(EGN, **NAME**, AGE, PHONE\_NUMBER)  
**VALUES** (**'7008104334'**, **'Krasimira Atanasova'**, 50, **'0877342536'**),  
 (**'7802026745'**, **'Lidiya Mihaleva'**, 42, **'0886124225'**),  
 (**'6807246341'**, **'Teodor Todorov'**, 52, **'0865344256'**),  
 (**'8912236473'**, **'Asen velikov'**, 31, **'0823656564'**),  
 (**'8803078443'**, **'Silviya Velikova'**, 32, **'0897856447'**),  
 (**'8201074532'**, **'Stoyan Penev'**, 38, **'0895243009'**),  
 (**'7905185667'**, **'Bojidara Ilieva'**, 41, **'0867566320'**),  
 (**'8304265448'**, **'Kristian Stefanov'**, 37, **'0893487616'**),  
 (**'9007024343'**, **'Stefani Vasileva'**, 30, **'0887338526'**),  
 (**'7801235465'**, **'Ivanka Georgieva'**, 42, **'0887926388'**),  
 (**'8010125463'**, **'Valq Cekova'**, 40, **'0897452458'**),  
 (**'9304156224'**, **'Kostadin Ivanov'**, 27, **'0882456312'**),  
 (**'9902184678'**, **'Kristina Cekova'**, 21, **'0897456345'**),  
 (**'9907265463'**, **'Aleksandra Petrova'**, 21, **'0878923456'**),  
 (**'9705163427'**, **'Ilian Dimitrov'**, 23, **'0882367190'**),  
 (**'9801244635'**, **'Nikoleta Stefanova'**, 22, **'0897625341'**),  
 (**'9009284562'**, **'Martina Rizova'**, 29, **'0897456666'**),  
 (**'9006175463'**, **'Petq Drobreva'**, 29, **'0897456558'**),  
 (**'9803134573'**, **'Atanas Stoqnov'**, 21, **'0897456398'**),  
 (**'9906175463'**, **'Georgi Petrov'**, 22, **'0897491243'**),  
 (**'9603134573'**, **'Petur Georgiev'**, 24, **'0897456388'**);

* Попълване на таблицата REGISTRATIONS

**INSERT INTO** REGISTRATIONS( PAYMENT, A\_DATE, DAYS, L\_DATE, R\_NUMBER, GUEST\_EGN, RECEPTIONIST\_ID, GUEST2\_EGN, GUEST3\_EGN)  
**VALUES**

(0, **'2018-12-10'**, 5, **'2019-12-15'**, 201, **'9902184678'**, **'9907265463'**, **'9705163427'**, **'REC621'**),  
(1, **'2019-01-01'**, 2, **'2019-01-03'**, 205, **'9906175463'**, **'REC014'**, **NULL**, **NULL**),  
(1, **'2019-01-01'**, 2, **'2019-01-03'**, 105, **'9603134573'**, **'REC621'**, **NULL**, **NULL**),

(1, **'2019-01-01'**, 2, **'2019-01-03'**, 306, **'7008104334'**, **'REC621'**, **NULL**, **NULL**),

(0, **'2019-02-18'**, 5, **'2019-02-23'**, 303, **'6807246341'**, **'REC014'**, **'7802026745'**, **NULL**)

(1, **'2019-03-12'**, 4, **'2019-03-16'**, 101, **'7801235465'**, **'8010125463'**, **'9304156224'**, **'REC014'**),  
(1, **'2019-05-05'**, 3, **'2019-05-08'**, 404, **'8912236473'**, **'REC872'**, **'8803078443'**, **NULL**),  
(0, **'2019-08-15'**, 10, **'2019-08-25'**, 203, **'8201074532'**, **'REC924'**, **'7905185667'**, **NULL**)

(2, **'2019-09-01'**, 6, **'2019-09-07'**, 104, **'9007024343'**, **'REC621'**, **'8304265448'**, **NULL**),  
(1, **'2019-09-09'**, 3, **'2019-09-12'**, 305, **'9803134573'**, **'REC014'**, **NULL**, **NULL**),

(2, **'2020-01-23'**, 6, **'2020-01-29'**, 302, **'9801244635'**, **'9009284562'**, **'9006175463'**, **'REC872'**);

* Изглед ***MANAGERS\_BEFORE\_2017***

**CREATE VIEW** MANAGERS\_BEFORE\_2017  
**AS  
 SELECT** ID, **NAME**, EMP\_DATE **FROM** MANAGER  
 **WHERE** EMP\_DATE < **'2017-01-01'**;

Този изглед извежда идентификационния номер, името и датата на назначаване на тези мениджъри, които са постъпили на работа преди 2017 г.

* Изглед ***RECEPTIONISTS\_REGISTERED\_ON\_3\_FLOOR***

**CREATE VIEW** RECEPTIONISTS\_REGISTERED\_ON\_3\_FLOOR  
**AS  
 SELECT** ID, **NAME FROM** RECEPTIONISTS  
 **WHERE** ID **IN** (  
 **SELECT DISTINCT** RECEPTIONIST\_ID **FROM** REGISTRATIONS  
 **WHERE** R\_NUMBER > 300 **AND** R\_NUMBER < 400);

Чрез този изглед можем да изведем идентификационния номер и името на тези рецепционисти, които са извършили регистрации на третия етаж. За целта взимаме съответната информация от таблицата REGISTRATIONS

**DISTINCT** RECEPTIONIST\_ID **FROM** REGISTRATIONS  
 **WHERE** R\_NUMBER > 300 **AND** R\_NUMBER < 400);

и от таблицата RECEPTIONISTS взимаме само тези рецепционисти, които удовлетворяват условието.

* Изглед ***REGISTRATION\_CURR\_YEAR***

**CREATE VIEW** REGISTRATION\_CURR\_YEAR  
**AS  
 SELECT** R\_NUMBER, A\_DATE, DAYS  
 **FROM** REGISTRATIONS  
 **WHERE YEAR**(A\_DATE) = **'2019'  
 AND MONTH**(L\_DATE) = **'01'**;

Целта на този изглед е да се изведе информация за номера на стаята, датата на настаняване и броя на дните, за които е направена регистрацията:

**SELECT** R\_NUMBER, A\_DATE, DAYS

**FROM** REGISTRATIONS

Като са дадени годината от датата на настаняване и месецът на датата на напускане:

**YEAR**(A\_DATE) = **'2019'  
 MONTH**(L\_DATE) = **'01'**;

* Изглед ***TWO\_GUESTS\_ROOMS***

**CREATE VIEW** TWO\_GUESTS\_ROOMS  
**AS  
 SELECT** *\** **FROM** ROOMS  
 **WHERE R\_TYPE** = 2  
 **AND ATTENDANT\_ID** = **'ATT333'**;

Този изглед извежда всички колони от таблицата ROOMS, за които е изпълнено, че типът на стаята (R\_TYPE) е 2, което означава, че тя е двойна, както и че номерът на камериера (ATTENDANT\_ID), който обслужава тази стая, е 'ATT333'.

* Изглед ***ATTENT\_PHONE\_BONUS***

**CREATE VIEW** ATTENT\_PHONE\_BONUS  
**AS  
 SELECT NAME**, PHONE\_NUM, BONUS  
 **FROM** ATTENDANTS  
 **WHERE** PHONE\_NUM **LIKE '088%' AND** BONUS > 200;

Целта на този изглед е да изведе името, телефонния номер и заплатата на тези служители, чийто телефонен номер започва с **'088%'**, абонусът е по – голям от **200.**

* Изглед ***GUESTS\_AGE\_NAME***

**CREATE VIEW** GUESTS\_AGE\_NAME  
**AS  
 SELECT** *\** **FROM** GUESTS

**WHERE** AGE **BETWEEN** 20 **AND** 40  
 **AND NAME LIKE '%ov'**;

Изгледът GUESTS\_AGE\_NAME ни извежда данните на всички гости на хотела, които отговарят на условието (**'%ov'**) и да са на възраст между 20 и 40 години(AGE **BETWEEN** 20 **AND** 40).

* Изтриването на изгледите става по следния начин:

**DROP VIEW** RECEPTIONISTS\_REGISTERED\_ON\_3\_FLOOR;

**DROP VIEW** MANAGERS\_BEFORE\_2017;

**DROP VIEW** REGISTRATION\_CURR\_YEAR;

**DROP VIEW** TWO\_GUESTS\_ROOMS;

**DROP VIEW** ATTENT\_PHONE\_BONUS;

**DROP VIEW** GUEST\_AGE\_NAME;

**6. Описание на функциите.**

В базата от данни за Хотел имаме общо 9 функции – 5 от тях са скаларни, а 4 са таблични. Първо, ще опишем скаларните функции и след това табличните.

**6.1. Скаларни функции**

* Функция ***GET\_ROOM\_TYPE***

**CREATE** FUNCTION GET\_ROOM\_TYPE(R\_NUM **INTEGER**)  
RETURNS **INTEGER**RETURN  
 **SELECT** R\_TYPE  
 **FROM** ROOMS  
 **WHERE** R\_NUMBER = R\_NUM;

Функцията извежда типа на стаята от таблицата ROOMS според нейния номер. Типът на връщане е INTEGER.

Има два начина за извикване. При първия начин задавайки стойност за атрибута R\_NUM = 201, в резултат се връща само типът на тази стая, отговаряща на този номер. В случая резултатът е 3, тоест стаята е апартамент. При втория начин извеждаме номер на стая и цена от таблицата ROOMS за всеки номер, тоест информация за всички стаи.

**VALUES** FN71840.GET\_ROOM\_TYPE(201);**SELECT** FN71840.GET\_ROOM\_TYPE(R\_NUMBER) **AS** R\_NUM, PRICE **AS** R\_PRICE  
**FROM** ROOMS;

* Функция ***GET\_REG\_DATE***

**CREATE** FUNCTION GET\_REG\_DATE(REG\_ID **INTEGER**)  
RETURNS **DATE**RETURN  
 **SELECT** A\_DATE  
 **FROM** REGISTRATIONS  
 **WHERE** ID = REG\_ID;

Тази фуннкция извежда датата на регистрация по зададена дата на настаняване. Типът на връщаната стойност е DATE.

При първия начин извикваме функцията по зададен номер на регистрация – в случая той е 3. Връща ни съответно датата на настаняване: 2020-01-23. Вторият начин извежда датата на регистрация и датата на напускане за всички регистрации от таблицата REGISTRATIONS.

**VALUES** FN71840.GET\_REG\_DATE(3);

**SELECT** FN71840.GET\_REG\_DATE(ID) **AS** REG\_A\_DATE, L\_DATE **AS** REG\_L\_DATE  
 **FROM** REGISTRATIONS;

* Функция ***GET\_MANAGER\_EMAIL***

**CREATE** FUNCTION GET\_MANAGER\_EMAIL(MNG\_ID **CHARACTER**(6))  
RETURNS **VARCHAR**(30)  
RETURN  
 **SELECT** E\_MAIL  
 **FROM** MANAGER  
 **WHERE** ID = MNG\_ID;

Тази функция връща имейл адреса на мениджър по зададен идентификационен номер. Стойността, която връща функцията е от тип VARCHAR(30).

Можем да извикаме функцията само с даден идентификационен номер, в случая - 'MNG205'. Връща ни съответния имейл - mm\_ttva@gmail.com

**VALUES** FN71840.GET\_MANAGER\_EMAIL(**'MNG205'**);

Можем да използваме функцията и в заявка. Като по този начин тя ще изведе електронните пощи на всички мениджъри в таблицата.

**SELECT NAME AS** MNG\_NAME, FN71840.GET\_MANAGER\_EMAIL(ID)

**AS** MNG\_EMAIL

**FROM** MANAGER;

* Функция ***GET\_ATT\_NAME***

**CREATE** FUNCTION GET\_ATT\_NAME(ATT\_ID **CHARACTER**(6))  
RETURNS **VARCHAR**(50)  
RETURN  
 **SELECT NAME  
 FROM** ATTENDANTS  
 **WHERE** ID = ATT\_ID;

Тази функция извежда името на служителя по зададен идентификационен номер от таблицата ATTENDANTS, като ни връща VARCHAR(50). Извикваме я с даден идентификационен номер - 'ATT133'.

**VALUES** FN71840.GET\_ATT\_NAME(**'ATT133'**);

След това функцията ни връща името, съответстващо на този номер, а именно Ivana Georgieva.

Второто извикване на функцията извежда имената на всички служители от таблицата ATTENDANTS.

**SELECT** FN71840.GET\_ATT\_NAME(ID) **AS** ATT\_NAME  
**FROM** ATTENDANTS;

* Функция ***GET\_GUEST\_PHONE***

**CREATE** FUNCTION GET\_GUEST\_PHONE(G\_EGN **CHARACTER**(10))  
RETURNS **CHARACTER**(10)  
RETURN  
 **SELECT** PHONE\_NUMBER  
 **FROM** GUESTS  
 **WHERE** EGN = G\_EGN ;

Функцията ни извежда телефонния номер на госта по неговото ЕГН и ни връща CHARACTER(10).

Съответно извикваме функцията с подаденото ЕГН, като в нашия случай това е '7801235465'. По този начин получаваме телефонния номер на нашия гост - Ivanka Georgieva.

**VALUES** FN71840.GET\_GUEST\_PHONE(**'7801235465'**);

По втория начин, извиквайки функцията, се извеждат телефонните номера на всички гости на хотела, взети от таблицата GUESTS.

**SELECT** FN71840.GET\_GUEST\_PHONE(EGN) **AS** G\_EGN  
**FROM** GUESTS;

**6.2. Таблични функции**

* Функция ***REGISTRATION\_DETAILS***

**CREATE FUNCTION** *REGISTRATION\_DETAILS*(R\_ID **INTEGER**)

**RETURNS TABLE**(**ADATE DATE**, **LDATE DATE**, **RNUM INTEGER**)  
**RETURN  
 SELECT A\_DATE**, **L\_DATE**, **R\_NUMBER  
 FROM** REGISTRATIONS  
 **WHERE ID** = R\_ID;

Функцията извежда таблица с информация за датата на настаняване, датата на напускане и номера на стаята за съответната регистрация по зададен номер на регистрация R\_ID. Извикването на функцията става по следния начин:

**SELECT** *\****FROM TABLE** (FN71840.*REGISTRATION\_DETAILS*(1)) R;

* Функция ***MANAGER\_INFO***

**CREATE** FUNCTION MANAGER\_INFO(MNG\_ID **CHARACTER**(6))  
RETURNS **TABLE**(M\_NAME **VARCHAR**(50), M\_SALARY **DOUBLE**, M\_BONUS **DOUBLE**, M\_EMAIL **VARCHAR**(30))  
RETURN  
 **SELECT NAME**, SALARY, BONUS, E\_MAIL  
 **FROM** MANAGER  
 **WHERE** ID = MNG\_ID;

По зададен идентификационен номер на мениджър ('MNG205'), функцията връща таблица със следната информация – име, заплата, бонус и имейл адрес на съответния служител.

**SELECT** *\****FROM TABLE**(FN71840.MANAGER\_INFO(**'MNG205'**)) T;

* Функция ***RECEPTIONIST\_INFO***

**CREATE** FUNCTION RECEPTIONIST\_INFO(REC\_ID **CHARACTER**(6))  
RETURNS **TABLE**(R\_NAME **VARCHAR**(50), R\_SALARY **DOUBLE**, R\_BONUS **DOUBLE**)  
RETURN  
 **SELECT NAME**, SALARY, BONUS  
 **FROM** RECEPTIONISTS  
 **WHERE** ID = REC\_ID;

Чрез тази функция по аналогичен начин извеждаме таблица с името, заплатата и бонуса на рецепционист, по зададен идентификационен номер - 'REC621'.

**SELECT** *\** **FROM TABLE**(FN71840.RECEPTIONIST\_INFO(**'REC621'**)) T;

* Функция ***FUNCTION GUEST\_DETAILS***

**CREATE** FUNCTION GUEST\_DETAILS(G\_EGN **CHARACTER**(10))  
RETURNS **TABLE**(G\_NAME **VARCHAR**(50), G\_AGE **INT**, G\_MOBILE **CHAR**(10))  
RETURN  
 **SELECT** G.NAME, G.AGE, G.PHONE\_NUMBER  
 **FROM** GUESTS G  
 **WHERE** G.EGN = G\_EGN;

С помощта на тази функция се извежда името, възрастта и телефонният номер на гостите на хотела по зададеното ни ЕГН - '7801235465'.

**SELECT** *\****FROM TABLE**(FN71840.GUEST\_DETAILS(**'7801235465'**)) G;

**7. Описание на тригерите.**

Тригерите са обекти от базата от данни, дефинирани върху таблица и изпълнявани при командите UPDATE, INSERT и DELETE. Нашите тригери са общо 9 – 5 са after, останалите 4 са before.

**7.1. After triggers**

* Тригер ***TRIG\_NEW\_REG***

Започваме със създаването на копие на таблицата, с която ще работим, в случай че не искаме да променяме оригиналните данни. След това копираме и стойностите от оригиналната таблица. Създаваме таблица AUDIT\_REG, която има две колони:

**CTIME\_REG TIMESTAMP** – показва времето, в което сме променили данните от таблицатаREGISRATIONS\_NEW;

**TEXT VARCHAR**(200) – в тази колона ще се извежда информация за номера на регистрацията, броят на дните преди промяната и броят на дните след промяната. Тази таблица извежда данните след като сме ги обновили.

В случая искаме към дните на регистрация с номер 10 да добавим още 5 дни. След като това условие се изпълни успешно се активира и самият тригер.

*-- Copy table***CREATE TABLE** REGISRATIONS\_NEW **LIKE** REGISTRATIONS;  
*-- Insert into copied table***INSERT INTO** REGISRATIONS\_NEW  
**SELECT** *\** **FROM** REGISTRATIONS;  
*-- Create table***CREATE TABLE** AUDIT\_REG(**CTIME\_REG TIMESTAMP**, **TEXT VARCHAR**(200));  
*-- Create the trigger***CREATE TRIGGER** TRIG\_NEW\_REG  
 **AFTER UPDATE OF DAYS ON** REGISRATIONS\_NEW  
 **REFERENCING OLD AS** OLD\_REG **NEW AS** NEW\_REG  
 **FOR EACH ROW  
 WHEN** (OLD\_REG.**DAYS** != NEW\_REG.**DAYS**)  
 **BEGIN  
 DECLARE** V\_TEXT **VARCHAR**(200);  
 **SET** V\_TEXT = **' REG\_NO = '** || OLD\_REG.**ID** || **' OLD DAYS = '** || *INT*(OLD\_REG.**DAYS**)  
 || **' NEW DAYS = '** || *INT*(NEW\_REG.**DAYS**);  
 **INSERT INTO** AUDIT\_REG **VALUES**(**CURRENT\_TIMESTAMP**, V\_TEXT);  
 **END**;  
  
**UPDATE** REGISRATIONS\_NEW  
 **SET DAYS** = **DAYS** + 5  
 **WHERE ID** = 10;

* Тригер ***TRIG\_NEW\_ROOM***

Създаваме копие на таблицата ROOMS и вмъкваме стойностите от старата таблица. Създаваме си и таблица AUDIT\_ROOM, в която ще се показва резултатът след изпълнението на тригера.

Целта на тригера TRIG\_NEW\_ROOM е да се обнови цената на стая от тип 3, като към старата цена се добави 20.

*-- Copy table***CREATE TABLE** MY\_ROOMS\_NEW **LIKE** ROOMS;  
*-- Insert into copied table***INSERT INTO** MY\_ROOMS\_NEW  
**SELECT** *\** **FROM** ROOMS;  
*-- Create table***CREATE TABLE** AUDIT\_ROOM(**CTIME\_REG TIMESTAMP**, **TEXT VARCHAR**(200));  
*-- Create the trigger***CREATE TRIGGER** TRIG\_NEW\_ROOM  
 **AFTER UPDATE OF PRICE ON** MY\_ROOMS\_NEW  
 **REFERENCING OLD AS** OLD\_ROOM **NEW AS** NEW\_ROOM  
 **FOR EACH ROW  
 WHEN** (OLD\_ROOM.**PRICE** != NEW\_ROOM.**PRICE**)  
 **BEGIN  
 DECLARE** V\_TEXT **VARCHAR**(200);  
 **SET** V\_TEXT = **' ROOM\_NO = '** || OLD\_ROOM.**R\_NUMBER** || **' OLD ROOM PRICE = '** || *INT*(OLD\_ROOM.**PRICE**)  
 || **' NEW ROOM PRICE = '** || *INT*(NEW\_ROOM.**PRICE**);  
 **INSERT INTO** AUDIT\_ROOM **VALUES**(**CURRENT\_TIMESTAMP**, V\_TEXT);  
 **END**;

**UPDATE** MY\_ROOMS\_NEW  
 **SET PRICE** = **PRICE** + 20  
 **WHERE R\_TYPE** = 3;

* Тригер ***TRIG\_UPD\_MNG***

Отново създаваме нова таблица MANAGER\_COPY с копие на информацията от MANAGERS, както и таблица AUDIT\_MNG, в която ще записваме променените записи. В този случай увеличаваме заплатата на даден мениджър с 20%, след което новата информация се записва в създадената таблица чрез тригера.

*-- Copy table***CREATE TABLE** MANAGER\_COPY **LIKE** FN71840.MANAGER;  
*-- Insert into copied table***INSERT INTO** MANAGER\_COPY  
**SELECT** *\** **FROM** FN71840.MANAGER;  
*-- Create table***CREATE TABLE** AUDIT\_MNG(**C\_TIME TIMESTAMP**, **TEXT VARCHAR**(200));  
*--Create trigger***CREATE** TRIGGER TRIG\_UPD\_MNG  
 AFTER **UPDATE OF** SALARY **ON** MANAGER\_COPY  
 REFERENCING OLD **AS** O NEW **AS** N  
 **FOR** EACH ROW  
 **WHEN** (O.SALARY != N.SALARY)  
 **BEGIN  
 DECLARE** V\_TEXT **VARCHAR**(200);  
 **SET** V\_TEXT = **' MANAGER ID = '** || O.ID  
 || **' OLD SALARY = '** || **CHAR**(O.SALARY)  
 || **' NEW SALARY = '** || **CHAR**(N.SALARY);  
 **INSERT INTO** AUDIT\_MNG **VALUES**(*CURRENT\_TIMESTAMP*, V\_TEXT);  
 **END**;  
  
**SELECT NAME**, SALARY, BONUS  
**FROM** MANAGER\_COPY  
**WHERE** ID = **'MNG221'**;  
  
**UPDATE** MANAGER\_COPY  
 **SET** SALARY = SALARY + 0.2 \* SALARY  
 **WHERE** ID = **'MNG221'**;

* Тригер ***TRIG\_UPD\_REC***

По аналогичен начин първо създаваме копие на таблицата RECEPTIONISTS, за да не загубим първоначалната информация, след това и таблица за новите данни - AUDIT\_REC.

След като обновим данните, например при намаляване наполовина бонуса на даден рецепционист, информацията се записва в AUDIT\_REC.

*-- Copy table***CREATE TABLE** RECEPTIONISTS\_COPY **LIKE** FN71840.RECEPTIONISTS;  
*-- Insert into copied table***INSERT INTO** RECEPTIONISTS\_COPY  
**SELECT** *\** **FROM** FN71840.RECEPTIONISTS;  
*-- Create table***CREATE TABLE** AUDIT\_REC(**C\_TIME TIMESTAMP**, **TEXT VARCHAR**(200));  
*--Create trigger***CREATE** TRIGGER TRIG\_UPD\_REC  
 AFTER **UPDATE OF** BONUS **ON** RECEPTIONISTS\_COPY  
 REFERENCING OLD **AS** O NEW **AS** N  
 **FOR** EACH ROW  
 **WHEN** (O.BONUS != N.BONUS)  
 **BEGIN  
 DECLARE** V\_TEXT **VARCHAR**(200);  
 **SET** V\_TEXT = **' RECEPTIONIST ID = '** || O.ID  
 || **' OLD BONUS = '** || **INT**(O.BONUS)  
 || **' NEW BONUS = '** || **INT**(N.BONUS);  
 **INSERT INTO** AUDIT\_REC **VALUES**(*CURRENT\_TIMESTAMP*, V\_TEXT);  
 **END**;  
  
**SELECT NAME**, SALARY, BONUS, EMP\_DATE  
**FROM** RECEPTIONISTS\_COPY  
**WHERE** ID = **'REC621'**;  
  
**UPDATE** RECEPTIONISTS\_COPY  
 **SET** BONUS = BONUS / 2  
 **WHERE** ID = **'REC621'**;

* Тригер ***TRIG\_NEW\_ATT***

За да не загубим първоначалната информация си създаваме пак нова таблица, копие на таблицата ATTENDANTS, по която ще работим, а именно - ATTENDANTS\_NEW.

В този пример искаме да увеличим бонуса на служител по подаден идентицикационен номер (при нас е 'ATT233'). На служителя с този номер (Stefan Stefanov) , който отговаря на това уловие, се увелича бонуса с 300 лева. Информацията се записва в ATTENDANTS\_NEW.

*-- Copy table***CREATE TABLE** ATTENDANTS\_NEW **LIKE** ATTENDANTS;  
*-- Insert into copied table***INSERT INTO** ATTENDANTS\_NEW  
**SELECT** *\** **FROM** ATTENDANTS;  
*-- Create table***CREATE TABLE** AUDIT\_ATT(**CTIME TIMESTAMP**, **TEXT VARCHAR**(200));  
**SELECT** *\** **FROM** AUDIT\_ATT;  
  
**CREATE** TRIGGER TRIG\_NEW\_ATT  
 AFTER **UPDATE OF** BONUS **ON** ATTENDANTS\_NEW  
 REFERENCING OLD **AS** O NEW **AS** N  
 **FOR** EACH ROW  
 **WHEN** (O.BONUS != N.BONUS)  
 **BEGIN  
 DECLARE** V\_TEXT **VARCHAR**(200);  
 **SET** V\_TEXT = **' ATT\_NO = '** || O.ID  
 || **' OLD BONUS = '** || **INT**(O.BONUS)  
 || **' NEW BONUS = '** || **INT**(N.BONUS);  
 **INSERT INTO** AUDIT\_ATT **VALUES**(*CURRENT\_TIMESTAMP*, V\_TEXT);  
 **END**;  
  
**SELECT NAME**, SALARY, BONUS **FROM** ATTENDANTS\_NEW  
**WHERE** ID = **'ATT233'**;  
  
**UPDATE** ATTENDANTS\_NEW  
 **SET** BONUS = BONUS + 300  
 **WHERE** ID = **'ATT233'**;

**7.2. Before triggers**

* Тригер ***TRIG\_ROOM\_NEW***

*-- Copy table***CREATE TABLE** ROOMS\_NEW **LIKE** ROOMS;  
*-- before trigger***CREATE TRIGGER** TRIG\_ROOM\_NEW  
 **BEFORE INSERT ON** ROOMS\_NEW  
 **REFERENCING NEW AS** N  
 **FOR EACH ROW  
 WHEN** (N.**PRICE IS NOT NULL**)  
 **SET** N.**PRICE** = N.**PRICE** + 50;  
*-- Insert into copied table***INSERT INTO** ROOMS\_NEW  
**SELECT** *\** **FROM** ROOMS;

Отново създаваме копие на таблицата ROOMS, с което ще работим.

Целта на този тригер е да се промени цената на стая, като се увеличи с 50. Това е валидно само за цени, които са различни от NULL.

Тригерът се активира преди вмъкването на информацията от старата таблица в новата, тоест в новата таблица имаме нови стойности за цените на стаите.

* Тригер ***TRIG\_REC\_NEW***

*-- Copy table***CREATE TABLE** RECEPTIONISTS\_NEW **LIKE** FN71840.RECEPTIONISTS;  
*--Create trigger***CREATE** TRIGGER TRIG\_REC\_NEW  
 BEFORE **INSERT ON** RECEPTIONISTS\_NEW  
 REFERENCING NEW **AS** N  
 **FOR** EACH ROW  
 **WHEN** (N.BONUS **IS NOT NULL**)  
 **SET** N.BONUS = N.BONUS + 100;  
*-- Insert into copied table***INSERT INTO** RECEPTIONISTS\_NEW  
**SELECT** *\** **FROM** RECEPTIONISTS;

Работим с копие на таблицата RECEPTIONISTS. В нея ще записваме информация след като сме изпълнили тригера – първо увеличаваме бонуса на съответния служител и тогава вмъкваме в таблицата.

* Тригер ***TRIG\_GUESTS\_NEW***

*-- Copy table***CREATE TABLE** GUESTS\_NEW **LIKE** GUESTS;  
  
**CREATE** TRIGGER TRIG\_GUESTS\_NEW  
 BEFORE **INSERT ON** GUESTS\_NEW  
 REFERENCING NEW **AS** N  
 **FOR** EACH ROW  
 **WHEN** (N.AGE **IS NOT NULL**)  
 **SET** N.AGE = N.AGE + 1;  
*-- Insert into copied table***INSERT INTO** GUESTS\_NEW  
**SELECT** *\** **FROM** GUESTS;

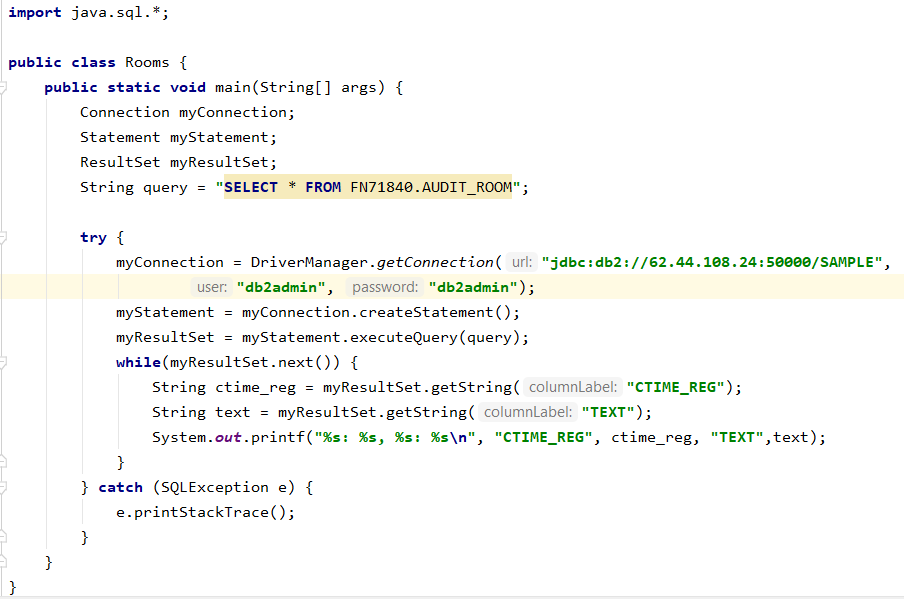
Отново създаваме копие на таблицата GUESTS, наречена GUESTS\_NEW. В нея ще записваме информация след като сме изпълнили тригера – първо ще се увеличи на възрастта на госта и след това ще бъде вмъкнато и в GUESTS\_NEW.

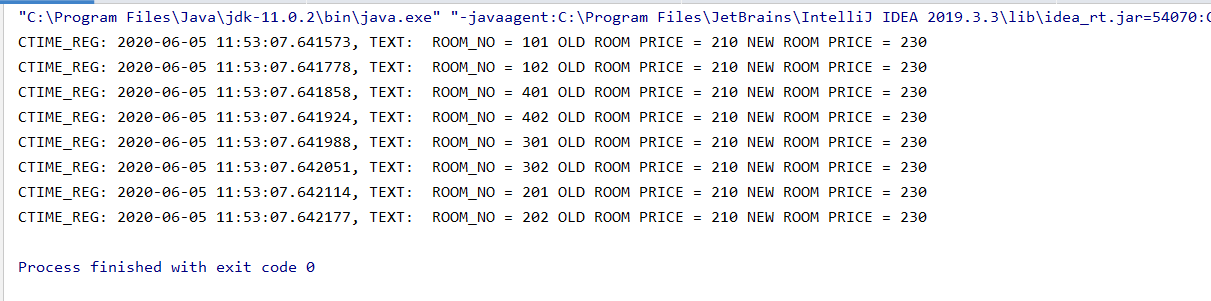
* Тригер ***TRIG\_ATT\_OTHER***

**CREATE TABLE** ATT\_OTHER **LIKE** ATTENDANTS;  
  
**CREATE** TRIGGER TRIG\_ATT\_OTHER  
 BEFORE **INSERT ON** ATT\_OTHER  
 REFERENCING NEW **AS** N  
 **FOR** EACH ROW  
 **WHEN** (N.SALARY **IS NOT NULL**)  
 **SET** N.SALARY = N.SALARY + 500;  
  
**INSERT INTO** ATT\_OTHER  
**SELECT** *\** **FROM** ATTENDANTS;

Пак първата ни стъпка тук е да създадем копие на таблицата, с която искаме да работим – ATT\_OTHER. В нея след като първо се увеличи заплатата на даден служител, ще последва тази нова информация да бъде вмъкната в таблицата ATT\_OTHER.

**8. Описание на приложението за достъп с резултатите от заявката.**

**8.1. Достъп до таблицата GUESTS.**

Statement позволява да се изпълняват заявки върху базата и да вземем резултата, съхранявайки го в ResultSet. Connection – връзка с базата.

**8.2. Достъп до таблицата AUDIT\_ROOM**



