

Технически университет-София



Дисциплина: „Бази данни - проект”

Изработил: Христо Ненков

Проверил:

Фак.№: 121223001

Факултет: ФКСТ

Група: 45

Дата: 13.02.2024

/...../

I. ВЪВЕДЕНИЕ

Съответната база данни е имплементирана на езика MySQL, сървърът MySQL Server и средата (Management Studio) ORACLE MySQL Workbench 8.0.

Архитектурата на базата е значително по – усложнена от зададената в критериите, като съм използвал за помощ при архитектурата й това видео:

<https://www.youtube.com/watch?v=lrjO1bHK-w>

Базата данни е съставена от таблици съответстващи на:

- Централен Офис
- Клон Офиси
- Служители – Оценителен експерт, Застрахователен агент
- Клиенти – Клиент (Физическо лице)
- 2 типа застрахователни полици – на автомобилно застраховане и на здравно застраховане, като всеки тип застраховане си има своите разновидности зададени с **ENUM**
- Разширена информация на застрахован автомобил и на шофьора, който ще кара автомобила (свързани с автомобилна полица)
- Разширена информация на Личен лекар (свързана с здравна полица)
- **МНОГО КЪМ МНОГО** връзки между агент, клиент, съотвена полица и съответна допълнителна информация
- Система за заплащане на дадена застраховка
- Система за издаване на фактури и извлечения, съответстващи на дадена платена полица

Базата данни използва връзки от типа **ЕДНО КЪМ МНОГО** и **МНОГО КЪМ МНОГО**, имплементирани чрез допълнителни таблици.

При разработката съм дефинирал свои собствени **CONSTRAINT** – и, а **INDEX** – ите се генерират сами от средата за работа.

Базата данни е конструирана и имплементирана на български език, а след това преведна и на английски. Причината е доста по – голямата разбираемост при представянето на данни на български език и по – лесното им манипулиране на английски.

Примерните данни са генерирани за всяка от 18 - те таблици, по 25 реда, с помощта на Изкуствен интелект.

Всички файлове от курсовият проект са прикачени в имейла, но за по – лесно разглеждане, можете да го видите и в моя **GitHub**: <https://github.com/hristo-nenkov-6>

II. Имплементиране на базата и генериране на информация

- 1. Да се проектира база от данни и да се представи ER диаграма със съответни CREATE TABLE заявки за средата MySQL.**

Тук не съм представил кода от имплементацията на английски, нито пък генерирането на примерната информация. Тези файлове могат да бъдат намерени, прикачени към имейла.

Реализация на MySQL (Български език):

```
CREATE SCHEMA IF NOT EXISTS `ЗастахователнаКомпания` ;
USE `ЗастахователнаКомпания` ;

CREATE TABLE IF NOT EXISTS `Автомобил` (
  `AutomobileId` BIGINT UNSIGNED NOT NULL AUTO_INCREMENT,
  `CarRegistrationNumber` VARCHAR(255) NOT NULL,
  `Марка` VARCHAR(255) NOT NULL,
  `Модел` VARCHAR(255) NOT NULL,
  `Гориво` VARCHAR(255) NOT NULL,
  `Година на производство` YEAR NOT NULL,
  `Кубатура на двигателя` INT NOT NULL,
  PRIMARY KEY (`AutomobileId`)
);

CREATE TABLE IF NOT EXISTS `Застахователна Компания` (
  `InsuranceCompanyId` BIGINT UNSIGNED NOT NULL AUTO_INCREMENT,
  `Адрес на централа` VARCHAR(255) NOT NULL,
  `Град` BIGINT NOT NULL,
  `Телефонен номер на централа` VARCHAR(255) NOT NULL,
  PRIMARY KEY (`InsuranceCompanyId`)
);

CREATE TABLE IF NOT EXISTS `Клон на застрахователна компания` (
  `OfficeId` BIGINT UNSIGNED NOT NULL AUTO_INCREMENT,
  `InsuranceCompanyId` BIGINT UNSIGNED NOT NULL,
  `Адрес` BIGINT NOT NULL,
  `Град` BIGINT NOT NULL,
  `Телефонен номер на офис` BIGINT NOT NULL,
  PRIMARY KEY (`OfficeId`),
  CONSTRAINT `Клон на застрахователна компания_insurancecompanyId_foreign`
    FOREIGN KEY (`InsuranceCompanyId`) REFERENCES `Застрахователна
Компания` (`InsuranceCompanyId`)
);

CREATE TABLE IF NOT EXISTS `Застрахователен Агент` (
  `AgentId` BIGINT UNSIGNED NOT NULL AUTO_INCREMENT,
  `OfficeId` BIGINT UNSIGNED NOT NULL,
  `Име на агент` VARCHAR(255) NOT NULL,
  PRIMARY KEY (`AgentId`),
  CONSTRAINT `Застрахователен агент_officeid_foreign`
    FOREIGN KEY (`OfficeId`) REFERENCES `Клон на застрахователна компания` (`OfficeId`)
);

CREATE TABLE IF NOT EXISTS `Полица (застраховка автомобил)` (
  `PolicyId` BIGINT UNSIGNED NOT NULL AUTO_INCREMENT,
  `Тип на автозастраховката` ENUM('Катастрофа', 'Кражба', 'Пожар') NOT NULL,
  `Максимална покриваща сума при вина на сключилия` DOUBLE NOT NULL,
  `Максимална покриваща сума при потърпевшо лице` DECIMAL(8,2) NOT NULL,
  PRIMARY KEY (`PolicyId`)
);
```

```

CREATE TABLE IF NOT EXISTS `Застрахователен Агент` (
  `AgentId` BIGINT UNSIGNED NOT NULL AUTO_INCREMENT,
  `OfficeId` BIGINT UNSIGNED NOT NULL,
  `Име на агент` VARCHAR(255) NOT NULL,
  PRIMARY KEY (`AgentId`),
  CONSTRAINT `Застрахователен агент_officeid_foreign`
    FOREIGN KEY (`OfficeId`) REFERENCES `Клон на застрахователна компания` (`OfficeId`)
);

CREATE TABLE IF NOT EXISTS `Полица (застраховка автомобил)` (
  `PolicyId` BIGINT UNSIGNED NOT NULL AUTO_INCREMENT,
  `Тип на автозастраховката` ENUM('Катастрофа', 'Кражба', 'Пожар') NOT NULL,
  `Максимална покриваща сума при вина на сключилия` DOUBLE NOT NULL,
  `Максимална покриваща сума при потърпевшо лице` DECIMAL(8,2) NOT NULL,
  PRIMARY KEY (`PolicyId`)
);

CREATE TABLE IF NOT EXISTS `Автомобилна полица - Агент` (
  `PolicyId` BIGINT UNSIGNED NOT NULL,
  `AgentId` BIGINT UNSIGNED NOT NULL,
  CONSTRAINT `Автомобилна полица _ Агент_agentid_foreign`
    FOREIGN KEY (`AgentId`) REFERENCES `Застрахователен Агент` (`AgentId`),
  CONSTRAINT `Автомобилна полица _ Агент_policyid_foreign`
    FOREIGN KEY (`PolicyId`) REFERENCES `Полица (застраховка автомобил)` (`PolicyId`)
);

CREATE TABLE IF NOT EXISTS `Клиент (физическо лице)` (
  `ClientId` BIGINT UNSIGNED NOT NULL AUTO_INCREMENT,
  `Пол` ENUM('Мъж', 'Жена', 'Друго') NOT NULL,
  `Име` VARCHAR(255) NOT NULL,
  `Презиме` VARCHAR(255) NULL DEFAULT NULL,
  `Фамилия` VARCHAR(255) NOT NULL,
  `ЕГН/ЛНЧ` VARCHAR(255) NOT NULL,
  `Телефонен номер` BIGINT NOT NULL,
  `Адрес по лична карта` VARCHAR(255) NOT NULL,
  `Настоящ адрес` VARCHAR(255) NOT NULL,
  `Образование` VARCHAR(255) NULL DEFAULT NULL,
  `Месторабота` VARCHAR(255) NULL DEFAULT NULL,
  `Осигурителен доход` VARCHAR(255) NOT NULL,
  PRIMARY KEY (`ClientId`)
);

CREATE TABLE IF NOT EXISTS `Шофьор` (
  `DriverId` BIGINT UNSIGNED NOT NULL AUTO_INCREMENT,
  `DriversLicenseNumber` VARCHAR(255) NOT NULL,
  `Пол` ENUM('Мъж', 'Жена', 'Друго') NOT NULL,
  `Име` VARCHAR(255) NOT NULL,
  `Презиме` VARCHAR(255) NULL DEFAULT NULL,
  `Фамилия` VARCHAR(255) NOT NULL,
  `Място на издаване на свидетелство за управление` BIGINT NOT NULL,
  `Дата на издаване на свидетелство за управление` BIGINT NOT NULL,
  PRIMARY KEY (`DriverId`)
);

```

```

CREATE TABLE IF NOT EXISTS `Агент - Клиент (физическо лице)` (
  `AgentId` BIGINT UNSIGNED NOT NULL,
  `ClientId` BIGINT UNSIGNED NOT NULL,
  CONSTRAINT `Агент _ Клиент (Физическо лице)_agentid_foreign`
    FOREIGN KEY (`AgentId`) REFERENCES `Застрахователен Агент` (`AgentId`),
  CONSTRAINT `Агент _ Клиент (Физическо лице)_clientid_foreign`
    FOREIGN KEY (`ClientId`) REFERENCES `Клиент (физическо лице)` (`ClientId`)
);

CREATE TABLE IF NOT EXISTS `Личен лекар` (
  `GPId` BIGINT UNSIGNED NOT NULL AUTO_INCREMENT,
  `Име` VARCHAR(255) NOT NULL,
  `Адрес` VARCHAR(255) NOT NULL,
  `Телефонен номер` VARCHAR(255) NOT NULL,
  PRIMARY KEY (`GPId`)
);

CREATE TABLE IF NOT EXISTS `Полица (Застраховка живот и здраве)` (
  `PolicyId` BIGINT UNSIGNED NOT NULL AUTO_INCREMENT,
  `Тип на застраховката` ENUM('Живот', 'Здравна') NOT NULL,
  `Максимална застраховка при инцидент` DOUBLE NOT NULL,
  `Максимална застраховка при смърт` DOUBLE NOT NULL,
  `Номер на медицински картон в НЗОК` VARCHAR(255) NOT NULL,
  `Пушач` BOOLEAN NOT NULL,
  PRIMARY KEY (`PolicyId`)
);

CREATE TABLE IF NOT EXISTS `Полица живот и здраве - клиент` (
  `PolicyId` BIGINT UNSIGNED NOT NULL,
  `ClientId` BIGINT UNSIGNED NOT NULL,
  `GPId` BIGINT UNSIGNED NOT NULL,
  `Отговорник за болния` BIGINT NULL,
  CONSTRAINT `Полица живот и здраве _ Клиент_GPId_Foreign`
    FOREIGN KEY (`GPId`) REFERENCES `Личен лекар` (`GPId`),
  CONSTRAINT `Полица живот и здраве _ Клиент_clientid_foreign`
    FOREIGN KEY (`ClientId`) REFERENCES `клиент (физическо лице)` (`ClientId`),
  CONSTRAINT `Полица живот и здраве _ Клиент_policyid_foreign`
    FOREIGN KEY (`PolicyId`) REFERENCES `Полица (Застраховка живот и здраве)`
      (`PolicyId`)
);

CREATE TABLE IF NOT EXISTS `Полица живот и здраве - Агент` (
  `PolicyId` BIGINT UNSIGNED NOT NULL,
  `AgentId` BIGINT UNSIGNED NOT NULL,
  CONSTRAINT `Полица живот и здраве _ Агент_agentid_foreign`
    FOREIGN KEY (`AgentId`) REFERENCES `застрахователен агент` (`AgentId`),
  CONSTRAINT `Полица живот и здраве _ Агент_policyid_foreign`
    FOREIGN KEY (`PolicyId`) REFERENCES `Полица (Застраховка живот и здраве)`
      (`PolicyId`)
);

```

```

CREATE TABLE IF NOT EXISTS `Извлечение` (
  `InvoiceId` BIGINT UNSIGNED NOT NULL AUTO_INCREMENT,
  `PolicyId` BIGINT UNSIGNED NOT NULL,
  `Име на платилия` BIGINT NOT NULL,
  `Платена сума` BIGINT NOT NULL,
  `Дата на плащане` BIGINT NOT NULL,
  PRIMARY KEY (`InvoiceId`),
  CONSTRAINT `Извлечение_policyid_foreign_Life`
    FOREIGN KEY (`PolicyId`) REFERENCES `Полица (Застраховка живот и здраве)`
    (`PolicyId`),
  CONSTRAINT `Извлечение_policyid_foreign_Auto`
    FOREIGN KEY (`PolicyId`) REFERENCES `полица (застраховка автомобил)` (`PolicyId`));

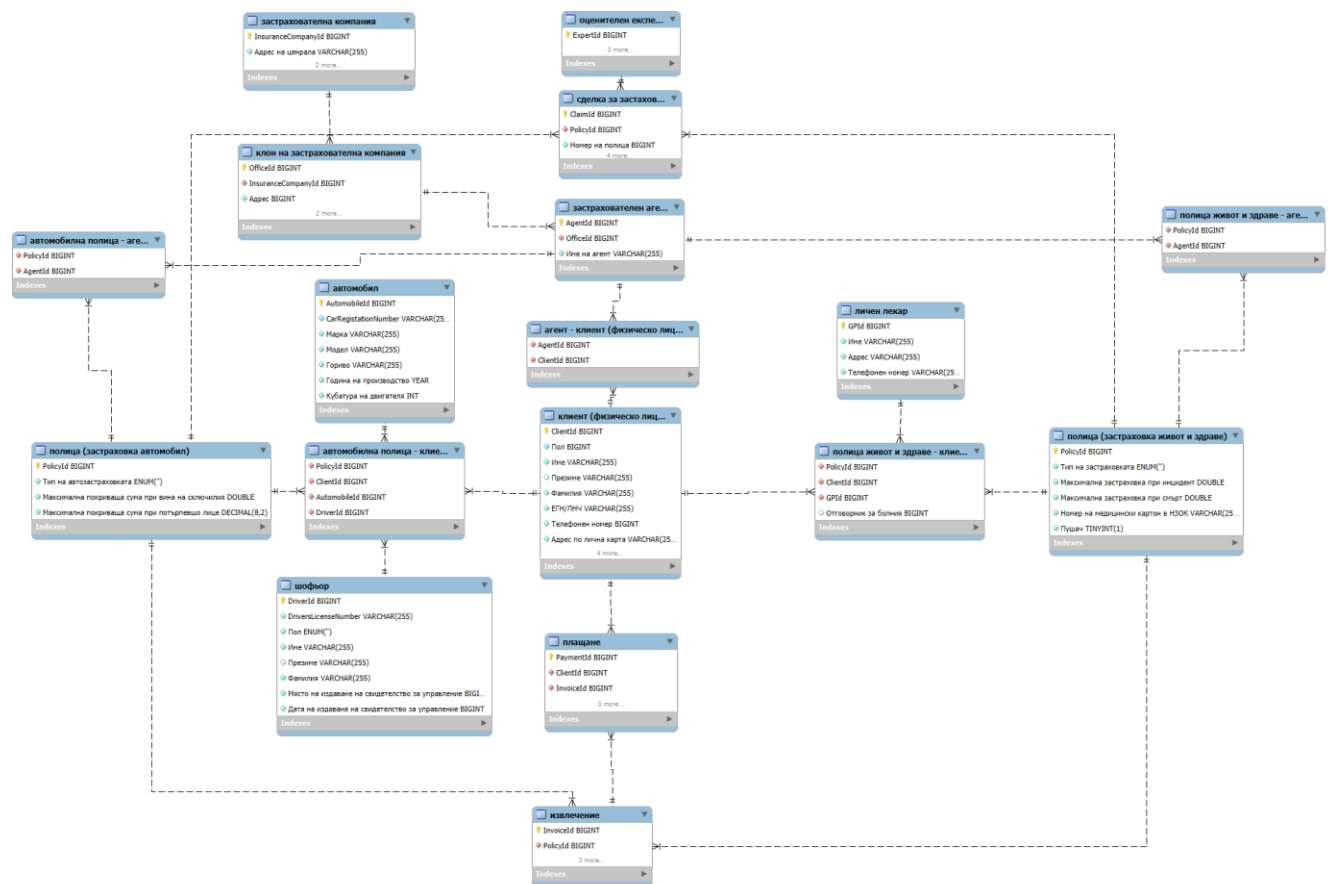
CREATE TABLE IF NOT EXISTS `Оценителен експерт` (
  `ExpertId` BIGINT UNSIGNED NOT NULL AUTO_INCREMENT,
  `Телефонен номер` BIGINT NOT NULL,
  `Адрес` BIGINT NOT NULL,
  `Град` BIGINT NOT NULL,
  PRIMARY KEY (`ExpertId`));

CREATE TABLE IF NOT EXISTS `Плащане` (
  `PaymentId` BIGINT UNSIGNED NOT NULL AUTO_INCREMENT,
  `ClientId` BIGINT UNSIGNED NOT NULL,
  `InvoiceId` BIGINT UNSIGNED NOT NULL,
  `Платена сума` DOUBLE NOT NULL,
  `Дата на платената сума` DATETIME NOT NULL,
  `Начин на плащане` ENUM('В брой', 'С карта', 'Банков трансфер') NOT NULL,
  PRIMARY KEY (`PaymentId`),
  CONSTRAINT `Плащане_clientid_foreign`
    FOREIGN KEY (`ClientId`) REFERENCES `Клиент (физическо лице)` (`ClientId`),
  CONSTRAINT `Плащане_invoiceid_foreign`
    FOREIGN KEY (`InvoiceId`) REFERENCES `Извлечение` (`InvoiceId`));

CREATE TABLE IF NOT EXISTS `Сделка за застраховка` (
  `ClaimId` BIGINT UNSIGNED NOT NULL AUTO_INCREMENT,
  `PolicyId` BIGINT UNSIGNED NOT NULL,
  `Номер на полица` BIGINT NOT NULL,
  `Внесена сума` DOUBLE NOT NULL,
  `Дата на сключване` DATETIME NOT NULL,
  `Валидност (месеци)` BIGINT NOT NULL,
  `ExpertId` BIGINT UNSIGNED NOT NULL,
  PRIMARY KEY (`ClaimId`),
  CONSTRAINT `Сделка за застраховка_expertid_foreign`
    FOREIGN KEY (`ExpertId`) REFERENCES `Оценителен експерт` (`ExpertId`),
  CONSTRAINT `Сделка за застраховка_policyid_foreignAuto`
    FOREIGN KEY (`PolicyId`) REFERENCES `Полица (застраховка автомобил)` (`PolicyId`),
  CONSTRAINT `Сделка за застраховка_policyid_foreignLife`
    FOREIGN KEY (`PolicyId`) REFERENCES `Полица (Застраховка живот и здраве)`
    (`PolicyId`));

```


Архитектура на базата данни:



III. Манипулация на таблици от базата данни

2. Напишете заявка, в която демонстрирате SELECT с ограничаващо условие по избор.

Извличане на имена, започващи с **a** от таблица.

```
SELECT FirstName, LastName, PhoneNumber, Income  
FROM `Client`  
WHERE FirstName LIKE '%a';
```

	FirstName	LastName	PhoneNumber	Income
►	Mia	Wright	444-555-6666	180000
	Olivia	Green	666-777-8888	200000
	Pamela	Baker	777-888-9999	210000
	Sara	Carter	000-111-2222	240000
	Uma	Roberts	222-333-4444	260000

3. Напишете заявка, в която използвате агрегатна функция и GROUP BY по ваш избор.

Намиране на броя на клиентите със съответен тип образование.

```
SELECT COUNT(ClientId) AS `Count of Clients`, Education
FROM `Client`
GROUP BY Education
ORDER BY COUNT(ClientId);
```

	Count of Clients	Education
▶	8	Master
	8	PhD
	9	Bachelor

4. Напишете заявка, в която демонстрирате INNER JOIN по ваш избор.

Показване на регистрационните номера, марката и модела на определени застраховани коли

```
SELECT CarRegistrationNumber, Brand, Model
FROM car
INNER JOIN carpolicyclient
ON car.AutomobileId = carpolicyclient.AutomobileId
WHERE car.AutomobileId BETWEEN 10 AND 18;
```

	CarRegistrationNumber	Brand	Model
▶	BCD890	Nissan	Qashqai
	EFG123	Toyota	RAV4
	HIJ456	Honda	CR-V
	KLM789	Ford	Mustang
	NOP012	BMW	3 Series
	QRS345	Audi	Q7
	TUV678	Mercedes	E-Class
	VWX901	Volkswagen	Passat
	YZA234	Hyundai	i30

5. Напишете заявка, в която демонстрирате OUTER JOIN по ваш избор.

Показване на тип застраховка за коли застраховани над определена сума:

```
SELECT PayerName, AmountPaid, cp.InsuranceType
FROM invoice as i
LEFT JOIN carpolicy as cp
ON i.PolicyId = cp.PolicyId
WHERE i.AmountPaid >= 580;
```

	PayerName	AmountPaid	InsuranceType
▶	Bob Brown	890	Crash
	Charlie White	1020.25	Theft
	David Black	1100	Fire
	Eve Green	975.5	Crash
	Henry Lewis	910.5	Crash
	Ivy Clark	1050.75	Theft
	Kathy Young	999.99	Crash
	Leo Walker	1120.3	Theft
	Paul Baker	890.6	Fire
	Quinn Turner	1025.9	Crash
	Rachel Evans	1105.5	Theft
	Sam Carter	970.3	Fire
	Victor Allen	920.75	Fire
	Wendy Hill	1080.9	Crash

6. Напишете заявка, в която демонстрирате вложен SELECT по ваш избор

Показване информация за всички клиенти мъже, с доход над средния

```
SELECT FirstName, LastName, Income, Gender
FROM `Client` as c
LEFT JOIN Payment as p
ON p.ClientId = c.ClientId
WHERE c.income > (SELECT AVG(AmountPaid)
                  FROM Invoice
                  WHERE c.Gender = 'Male');
```

	FirstName	LastName	Income	Gender
►	John	Doe	50000	Male
	Bob	Brown	80000	Male
	Frank	Wilson	110000	Male
	Ivy	Walker	140000	Male
	Liam	King	170000	Male
	Olivia	Green	200000	Male
	Ryan	Nelson	230000	Male
	Uma	Roberts	260000	Male
	Xander	Evans	290000	Male

7. Напишете заявка, в която демонстрирате едновременно JOIN и агрегатна функция.

Показване на собственика, марката и кубатурата на колите с най-големи двигатели от всяка марка.

```

SELECT d.FirstName, d.LastName,
       Partial.Brand, Partial.MaxEngineSize
FROM Driver AS d
LEFT JOIN carpolicyclient AS cpc ON d.DriverId =
cpc.DriverId
LEFT JOIN
(
    SELECT c.Brand, MAX(c.EngineSize) AS MaxEngineSize
    FROM car AS c
    INNER JOIN carpolicyclient AS cpc ON c.AutomobileId =
cpc.AutomobileId
    GROUP BY c.Brand
) AS Partial
ON cpc.AutomobileId = (
    SELECT c2.AutomobileId
    FROM car AS c2
    WHERE c2.Brand = Partial.Brand
    LIMIT 1
)
WHERE
    Partial.Brand IS NOT NULL;

```

	FirstName	LastName	Brand	MaxEngineSize
▶	John	Doe	Toyota	2000
	Jane	Smith	Honda	2200
	Alice	Johnson	Ford	5000
	Bob	Brown	BMW	3000
	Charlie	Davis	Audi	3000
	Eve	White	Mercedes	2500
	Frank	Wilson	Volkswagen	1800
	Grace	Lee	Hyundai	1800
	Henry	Clark	Kia	2000
	Ivy	Walker	Nissan	1600

8. Създайте тригер по ваш избор.

Тригер, запълващ автоматично информация в отделна таблица при добавяне или изтриване на информация от таблица Car.

```

CREATE TABLE car_log(
    Id INT AUTO_INCREMENT PRIMARY KEY,
    CarId INT,
    ActionMade VARCHAR(50)
);

DELIMITER //
CREATE TRIGGER after_car_insert
AFTER INSERT ON car
FOR EACH ROW
BEGIN
    INSERT INTO car_log(CarId, ActionMade)
    VALUES (NEW.AutomobileId, 'Inserted new car');
END //
DELIMITER ;
DELIMITER //
CREATE TRIGGER after_car_delete
AFTER DELETE ON car
FOR EACH ROW
BEGIN
    INSERT INTO car_log(CarId, ActionMade)
    VALUES (OLD.AutomobileId, 'Deleted car');
END //

DELIMITER ;

DROP TRIGGER after_car_delete;

INSERT INTO `Car` (`CarRegistrationNumber`, `Brand`,
`Model`, `Fuel`, `Year`, `EngineSize`) VALUES
('ABC123', 'Toyota', 'Corolla', 'Petrol', 2018, 1600),
('DEF456', 'Honda', 'Civic', 'Petrol', 2019, 1800),
('GHI789', 'Ford', 'Focus', 'Diesel', 2020, 2000),
('JKL012', 'BMW', 'X5', 'Petrol', 2021, 3000);

select * FROM car_log;

DELETE FROM car WHERE AutomobileId = 40;

SELECT * FROM car_log;

```


	Id	CarId	ActionMade
►	1	38	Inserted new car
	2	39	Inserted new car
	3	40	Inserted new car
	4	41	Inserted new car
	5	40	Deleted car

9. Създайте процедура, в която демонстрирате използване на курсор

Тук използвам друга таблица, тъй като таблицата от базата данни е дефинирана по друг начин.

Процедура, извличаща името на града от адресна колона и добавяйки го към колона за градове.

```
CREATE TABLE IF NOT EXISTS Expert (  
    ExpertId BIGINT UNSIGNED NOT NULL AUTO_INCREMENT,  
    PhoneNumber VARCHAR(255) NOT NULL,  
    Address VARCHAR(255) NOT NULL,  
    City VARCHAR(255) NULL,  
    PRIMARY KEY (ExpertId)  
);  
  
INSERT INTO Expert (PhoneNumber, Address, City) VALUES  
( '123-456-7890', '123 Main St, New York', NULL),  
( '987-654-3210', '456 Elm St, Los Angeles', NULL),  
( '555-123-4567', '789 Oak St, Chicago', NULL),  
( '444-987-6543', '321 Pine St, Houston', NULL),  
( '333-222-1111', '654 Maple St, Phoenix', NULL),  
( '777-888-9999', '987 Cedar St, San Francisco', NULL),  
( '666-555-4444', '741 Birch St, Miami', NULL),  
( '111-222-3333', '852 Walnut St, Seattle', NULL),  
( '999-000-1111', '369 Ash St, Boston', NULL),  
( '222-333-4444', '147 Spruce St, Denver', NULL);  
  
SELECT * FROM Expert;
```

	ExpertId	PhoneNumber	Address	City
►	1	123-456-7890	123 Main St, New York	NULL
	2	987-654-3210	456 Elm St, Los Angeles	NULL
	3	555-123-4567	789 Oak St, Chicago	NULL
	4	444-987-6543	321 Pine St, Houston	NULL
	5	333-222-1111	654 Maple St, Phoenix	NULL
	6	777-888-9999	987 Cedar St, San Francisco	NULL
	7	666-555-4444	741 Birch St, Miami	NULL
	8	111-222-3333	852 Walnut St, Seattle	NULL
	9	999-000-1111	369 Ash St, Boston	NULL
	10	222-333-4444	147 Spruce St, Denver	NULL

```

DELIMITER //
CREATE PROCEDURE UpdateExpertCity()
BEGIN
    DECLARE Expert_Id BIGINT;
    DECLARE Phone_Number VARCHAR(255);
    DECLARE _Address VARCHAR(255);
    DECLARE _City VARCHAR(255);
    DECLARE _Done BOOLEAN DEFAULT FALSE;

    DECLARE Expert_Cursor CURSOR FOR
        SELECT ExpertId, PhoneNumber, Address,
SUBSTRING_INDEX(TRIM(Address), ',', -1)
        FROM expert;

    DECLARE CONTINUE HANDLER FOR NOT FOUND
    SET _Done = TRUE;

    OPEN Expert_Cursor;

    UPDATE_LOOP: LOOP
        FETCH Expert_Cursor
        INTO Expert_Id, Phone_Number, _Address, _City;

        IF _Done THEN
            LEAVE UPDATE_LOOP;
        END IF;

        UPDATE Expert
        SET City = _City
        WHERE ExpertId = Expert_Id AND
        City IS NULL;

    END LOOP;

    CLOSE Expert_Cursor;

END //
DELIMITER ;
CALL UpdateExpertCity();
SELECT * FROM Expert;

```

	ExpertId	PhoneNumber	Address	City
►	1	123-456-7890	123 Main St, New York	New York
	2	987-654-3210	456 Elm St, Los Angeles	Los Angeles
	3	555-123-4567	789 Oak St, Chicago	Chicago
	4	444-987-6543	321 Pine St, Houston	Houston
	5	333-222-1111	654 Maple St, Phoenix	Phoenix
	6	777-888-9999	987 Cedar St, San Francisco	San Francisco
	7	666-555-4444	741 Birch St, Miami	Miami
	8	111-222-3333	852 Walnut St, Seattle	Seattle
	9	999-000-1111	369 Ash St, Boston	Boston
	10	222-333-4444	147 Spruce St, Denver	Denver

IV. Заключение и бъдеще на проекта

Тази примерна база данни на застрахователна компания би могла да бъде разширена към още типове застраховки (Застраховка на имущество, Застраховка на заем и т.н.), както и да бъде добавен вариант за застраховане на Бизнес клиенти. Това разбира се е малка част от възможните разширения на базата.

За да бъде използвана максимално много, можем да разглеждаме тази база от данни като сървър (каквата самата тя представлява).

Framework – овете - Spring и Spring – Boot, могат да бъдат използване съответно като **ORM** и абстракция на базата, за да бъде създаден истински работещ уебсайт, с Back - End – нашият сървър.