

**МИНИСТЕРСТВО ЦИФРОВЫХ ТЕХНОЛОГИЙ  
РЕСПУБЛИКИ УЗБЕКИСТАН**

**ТАШКЕНТСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ  
ТЕХНОЛОГИЙ ИМЕНИ МУХАММАДА АЛХОРАЗМИЙ**

**ПРЕДМЕТ “База Данных” ПРАКТИЧЕСКАЯ РАБОТА № 1, 2,  
3**

**Выполнил: Курбаналиев Саят**

**Группа: DBM202-2**

**Ташкент 2024**

**Практическая работа № 2.1.**

**Тема:** Проектирование баз данных и создание модели «сущность-связь»

**Цель лабораторной работы:** научиться проводить исследование заданной предметной области, спроектировать базу данных, разрабатывать логическую модель «СУЩНОСТЬ-СВЯЗЬ».

**Краткие теоретические сведения:** Проектирование базы данных начинается с определения сущностей, то есть объектов или понятий, которые важны для системы. Каждая сущность будет представлена таблицей в базе данных. Например, в каршеринговой системе можно выделить такие сущности, как Владелец, Клиент, Автомобиль, Аренда и Счета. Каждая из этих сущностей обладает своими характеристиками, которые называются атрибутами. Например, у сущности "Автомобиль" могут быть атрибуты марка, модель, год выпуска и регистрационный номер.

После определения сущностей и атрибутов, необходимо установить связи между сущностями. Связь определяет, как одна сущность связана с другой. Например, Владелец может владеть несколькими автомобилями, а один Клиент может арендовать несколько автомобилей. Для таких связей часто используется понятие кардинальности, которое указывает, сколько экземпляров одной сущности может быть связано с экземплярами другой.

Связи бывают разные. Например, один Владелец может быть связан с несколькими Автомобилями (связь "один ко многим"). В свою очередь, один Автомобиль может быть связан с несколькими Арендками (связь "один ко многим"), а один Клиент может иметь несколько Аренд (связь "многие ко многим"). Такие связи важно учитывать при проектировании, чтобы база данных была логичной и могла эффективно хранить данные.

Процесс проектирования базы данных с использованием модели «сущность-связь» позволяет структурировать информацию о системе, определить связи между данными и подготовить её для дальнейшей реализации.

### **Предметная область каршеринга:**

Включает в себя управление автопарком, клиентами, тарифами, процессами аренды и возврата, техническим обслуживанием, финансовыми операциями и отчетностью. Это охватывает учет автомобилей, регистрацию клиентов, определение тарифов, управление персоналом, процессы аренды, обслуживание автомобилей, финансовые операции и аналитическую отчетность для управленческих решений.

## Сущности и их атрибуты в моей работе:

- Аренда:
  1. Цена
  2. Время резерва 3. Продолжительность
- Клиент:
  1. Имя
  2. Номер 3. Местоположение
- Машина:
  1. Марка машины
  2. Модель машины
  3. Регистрационный номер
  4. Класс машины 5. ID Страхования
- Владелец:
  1. Имя
  2. Техпаспорт
  3. Номер • Чек:
    1. Общая стоимость
    2. Время начало аренды
    3. Время конца аренды

## ER-Диаграмма:

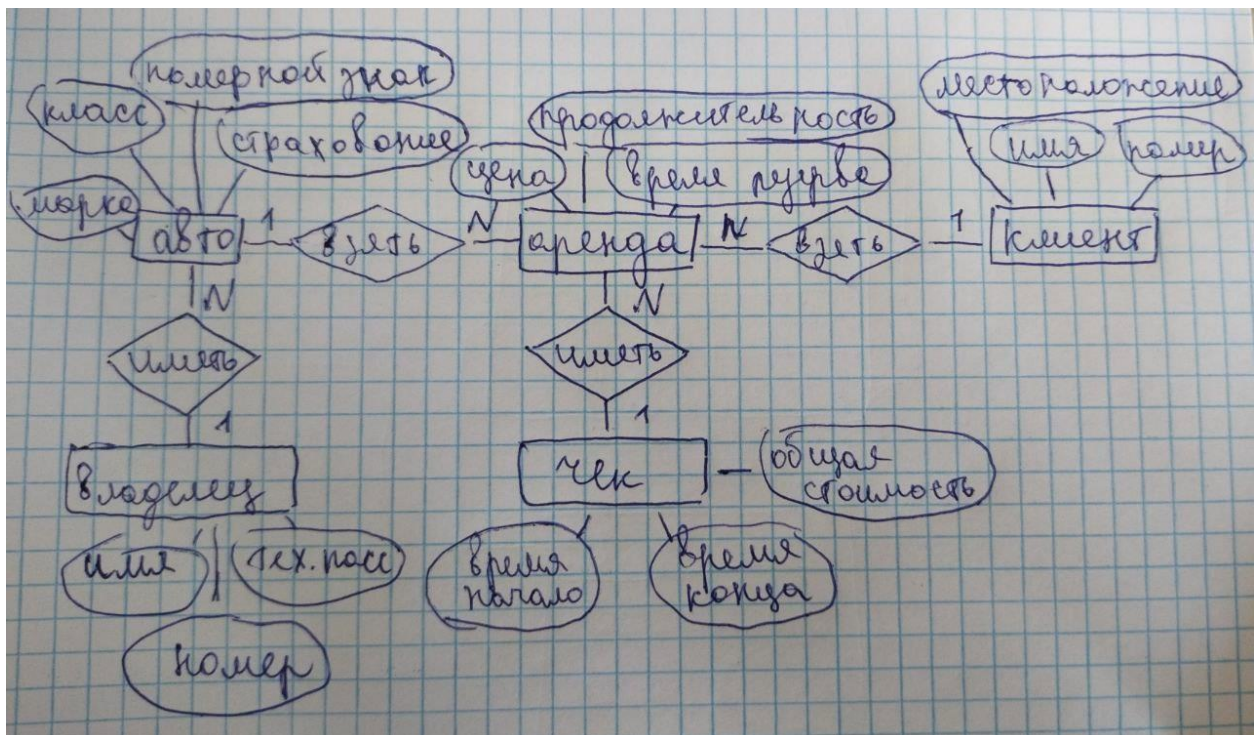


Таблица 1:

Сущность	Атрибуты	Сущность	Атрибуты
<b>Владелец</b>	Имя Техпаспорт Номер	<b>Чек</b>	Общая стоимость Время начало аренды Время конца аренды
<b>Аренда</b>	Цена Время резерва Продолжительность	<b>Клиент</b>	Имя Номер Местоположение
<b>Авто</b>	Марка машины Модель машины Регистрационный номер Класс машины ID Страхования		

Сущность	Связь	Сущность	Тип	Модальность
Машина	Взять	Аренда	Один- многим	Может- Должен
Клиент	Взять	Аренда	Один- многим	Может- Должен
Машина	Иметь	Владелец	Многие- одному	ДолженМожет
Аренда	Иметь	Чек	Многие- одному	Должен- Должен

## **Практическая работа № 2.2.**

**Тема:** Проектирование реляционной базы данных

**Цель лабораторной работы:** получить практические навыки структурирования предметной области и преобразования ER-диаграммы в реляционную модель данных.

### **Краткие теоретические сведения:**

После того, как построена модель сущность-связь (ER-модель) на следующем этапе проектирования необходимо преобразовать ее в реляционную. Основной структурой реляционной модели является отношение (relation), графической интерпретацией которого служит таблица. Каждое отношение состоит из некоторого ограниченного числа кортежей, а содержание каждого кортежа определяется набором атрибутов отношения. Каждый атрибут имеет определенный тип, значение которого берется из определенного домена. Кортежам отношения соответствуют строки таблицы, количество столбцов таблицы равно количеству атрибутов отношения, а тип величины, находящейся в соответствующем столбце определяется типом соответствующего атрибута. Отношения могут быть связаны между собой посредством набора атрибутов, одинаково содержащихся в обоих отношениях. Связи между отношениями в реляционной модели, в отличие от ER-модели, могут иметь только тип “один ко многим”, то есть одно отношение всегда будет основным, а второе – подчиненным и одному кортежу основного отношения могут соответствовать несколько кортежей подчиненного отношения. Данное соответствие означает, что у обоих кортежей значения набора атрибутов, по которому связаны отношения, совпадают. Обычно у основного отношения данный набор атрибутов является первичным ключом, и, следовательно, уникальным для каждого кортежа. У второго отношения данный набор атрибутов называется внешним ключом для данной связи.

Кортежей с одинаковым значением внешнего ключа может быть сколько угодно, но не может существовать кортежей со значением внешнего ключа,

которому не соответствовал какой-либо первичный ключ основного отношения. Связь “один к одному” рассматривается, как частный случай связи “один ко многим”.

Связи “многие ко многим” в реляционной модели быть не может. Чтобы преобразовать ER-модель в реляционную модель необходимо выполнить следующие действия.

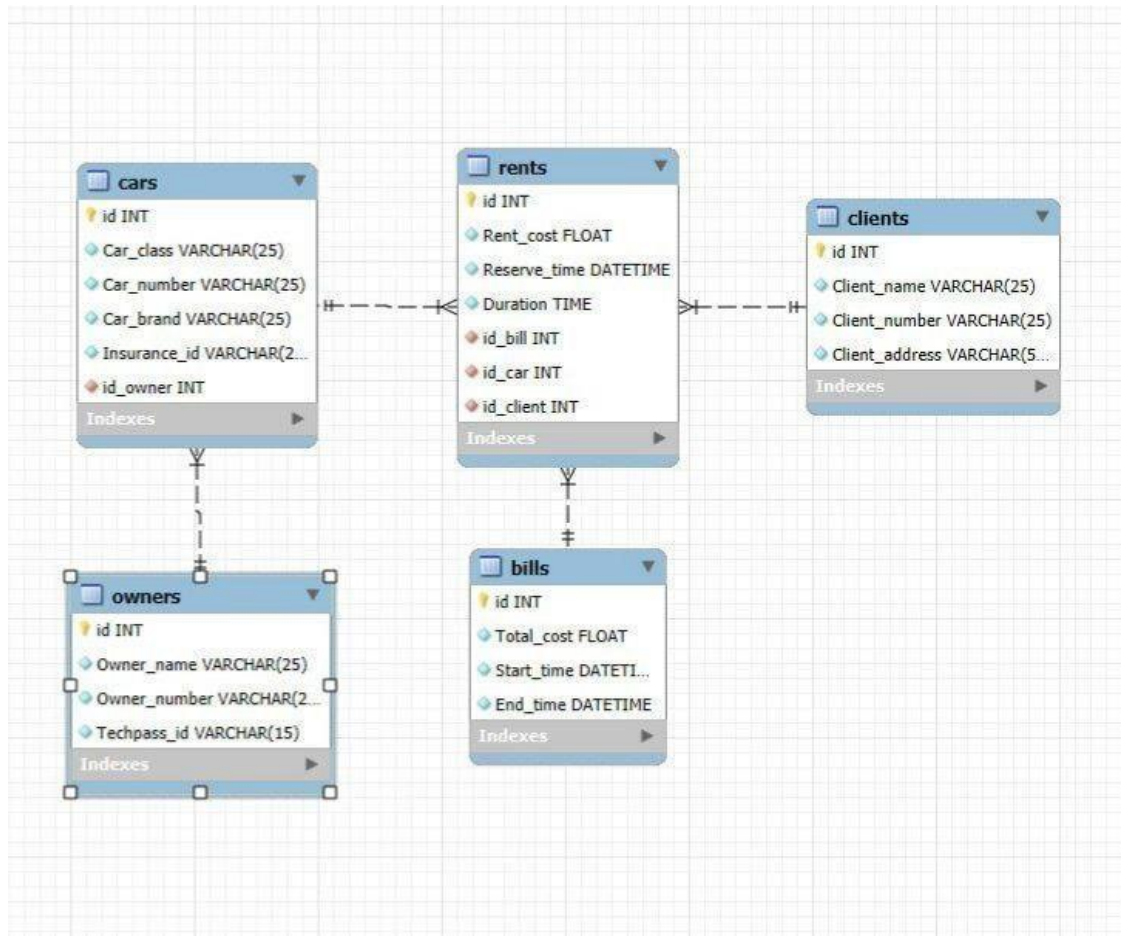
1. Поставить каждой сущности ER-модели в соответствие отношение реляционной модели, при этом каждому атрибуту сущности ставится в соответствие атрибут отношения реляционной модели. Ключ сущности становится первичным ключом соответствующего отношения (PRIMARY KEY). При этом имена сущностей и отношений, равно, как и атрибутов, могут не совпадать. Желательно при указании имен отношений и атрибутов реляционной модели использовать латиницу, поскольку эти имена чаще всего являются идентификаторами в некотором языке программирования.

2. В каждое отношение, соответствующее подчиненной сущности, добавляется набор атрибутов, соответствующий ключу основной сущности, если, конечно, он там уже не присутствовал. В любом случае этот набор атрибутов становится внешним ключом в подчиненном отношении (FOREIGN KEY).

3. При обязательном характере связи у атрибутов, соответствующих внешнему ключу, устанавливается свойство отсутствия неопределенных значений (NOT NULL)

4. Если в ER-модели имеются связи “многие ко многим”, то их надо преобразовать в связи “один ко многим”, поскольку связи “многие ко многим” в реляционной модели не допускаются. Для этого в реляционную модель добавляется связующее отношение, атрибуты которого соответствуют атрибутам первичных ключей обоих отношений, участвующих в связи “многие ко многим”. Связующее отношение будет находиться в связи “один ко многим” с каждым из этих отношений.

## Реляционная модель:



## Практическая работа № 3

**Тема:** Создание, редактирование, удаление таблиц в SQL

**Цель лабораторной работы:** научиться составлять и выполнять SQLзапросы для работы с данными БД.

### Краткие теоретические сведения:

SQL (Structured Query Language) представляет собой непроцедурный язык, используемый для управления данными реляционных СУБД. Термин «непроцедурный» означает, что на данном языке можно сформулировать, что нужно сделать с данными, но нельзя проинструктировать, как именно это следует сделать. Иными словами, в этом языке отсутствуют алгоритмические конструкции, такие как метки, операторы цикла, условные переходы и др.

Язык SQL был создан в начале 70-х годов в результате исследовательского проекта IBM, целью которого было создание языка манипуляции реляционными данными. Первоначально он назывался SEQUEL (Structured English Query Language), затем — SEQUEL/2, а затем — просто SQL. Официальный стандарт SQL был опубликован ANSI (American National Standards Institute — Национальный институт стандартизации, США) в 1986 году (это наиболее часто используемая ныне реализация SQL). Данный стандарт был расширен в 1989 и 1992 годах, поэтому последний стандарт SQL носит название SQL92. В настоящее время ведется работа над стандартом SQL3, содержащим некоторые объектно-ориентированные расширения.

Существует три уровня соответствия стандарту ANSI — начальный, промежуточный и полный. Многие производители серверных СУБД, такие как IBM, Informix, Microsoft, Oracle и Sybase, применяют собственные реализации SQL, основанные на стандарте ANSI (отвечающие как минимум начальному уровню соответствия стандарту) и содержащие некоторые расширения, специфические для данной СУБД.

Язык SQL предназначен для выполнения операций над таблицами (создание, удаление, изменение структуры) и над данными таблиц (выборка, изменение, добавление и удаление), а также некоторых сопутствующих операций. SQL является непроцедурным языком и не содержит операторов управления, организации подпрограмм, ввода-вывода и т.п. В связи с этим SQL автономно не используется, обычно он погружен в среду встроенного языка программирования СУБД.

**Data Definition Language (DDL)** содержит операторы, позволяющие создавать, изменять и уничтожать базы данных и объекты внутри них (таблицы, представления и др.). Эти операторы перечислены в табл.3.1.

**Таблица 3.2. Операторы DDL**

Оператор	Описание
----------	----------



CREATE TABLE	Применяется для добавления новой таблицы к базе данных
DROP TABLE	Применяется для удаления таблицы из базы данных
ALTER TABLE	Применяется для изменения структуры имеющейся таблицы
CREATE VIEW	Применяется для добавления нового представления к базе данных
DROP VIEW	Применяется для удаления представления из базы данных
CREATE INDEX	Применяется для создания индекса для данного поля
DROP INDEX	Применяется для удаления существующего индекса
CREATE SCHEMA	Применяется для создания новой схемы в базе данных
DROP SCHEMA	Применяется для удаления схемы из базы данных
CREATE DOMAIN	Применяется для создания нового домена
ALTER DOMAIN	Применяется для переопределения домена
DROP DOMAIN	Применяется для удаления домена из базы данных

SQL поддерживает следующие типов данных, которые можно разделить на три категории: числовые данные, дата и время, и данные типа строка.

**Таблица 3.2.** Наиболее популярные типы данных в SQL

Строки	
Тип данных	Описание
CHAR(size)	Строки фиксированной длиной (могут содержать буквы, цифры и специальные символы). Фиксированный размер указан в скобках. Можно записать до 255 символов
VARCHAR(size)	Может хранить не более 255 символов.
TINYTEXT	Может хранить не более 255 символов.

TEXT	Может хранить не более 65 535 символов.
BLOB	Может хранить не более 65 535 символов.
MEDIUMTEXT	Может хранить не более 16 777 215 символов.
MEDIUMBLOB	Может хранить не более 16 777 215 символов.
LONGTEXT	Может хранить не более 4 294 967 295 символов.
LOBLOB	Может хранить не более 4 294 967 295 символов.
ENUM(x,y,z,etc.)	<p>Позволяет вводить список допустимых значений. Можно ввести до 65535 значений в ENUM список.Если при вставке значения не будет присутствовать в списке ENUM, то мы получим пустое значение.</p> <p>Ввести возможные значения можно в таком формате: ENUM ( 'X', 'Y', 'Z')</p>
SET	Напоминает ENUM за исключением того, что SET может содержать до 64 значений.
<b>Дробные числа</b>	
Тип данных	Описание
TINYINT(size)	Может хранить числа от -128 до 127
SMALLINT(size)	Диапазон от -32 768 до 32 767
MEDIUMINT(size)	Диапазон от -8 388 608 до 8 388 607
INT(size)	Диапазон от -2 147 483 648 до 2 147 483 647
BIGINT(size)	Диапазон от -9 223 372 036 854 775 808 до 9 223 372 036 854 775 807
FLOAT(size,d)	Число с плавающей точкой небольшой точности.
DOUBLE(size,d)	Число с плавающей точкой двойной точности.
DECIMAL(size,d)	Дробное число, хранящееся в виде строки.
<b>Дата и время</b>	
Тип данных	Описание

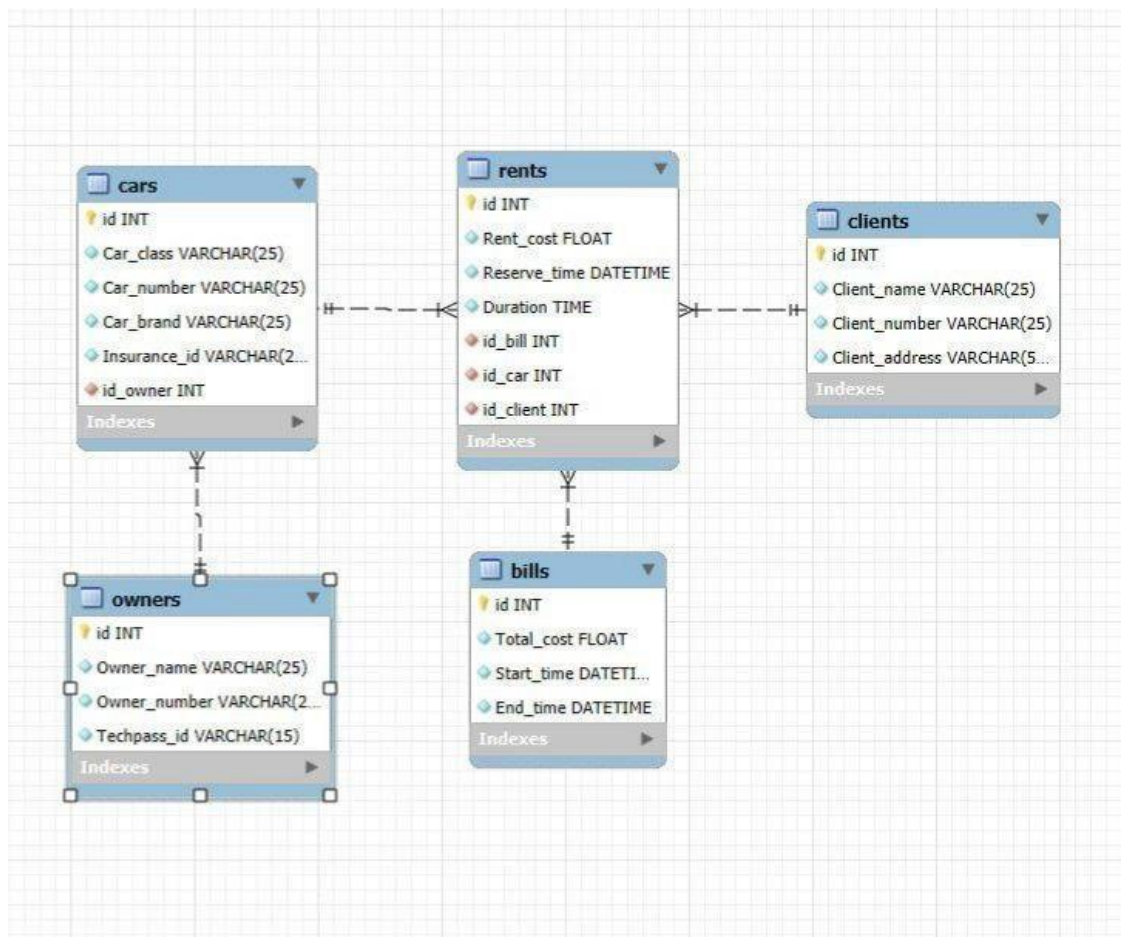
DATE()	Дата в формате ГГГГ-ММ-ДД
DATETIME()	Дата и время в формате ГГГГ-ММ-ДД ЧЧ:ММ:СС
TIMESTAMP()	Дата и время в формате timestamp. Однако при получении значения поля оно отображается не в формате timestamp, а в виде ГГГГ-ММ-ДД ЧЧ:ММ:СС
TIME()	Время в формате ЧЧ:ММ:СС
YEAR()	Год в двух значной или в четырехзначном формате.

Основным назначением языка SQL (как и других языков для работы с базами данных) является подготовка и выполнение запросов. В результате выборки данных из одной или нескольких таблиц может быть получено множество записей, называемое представлением.

Представление, по существу является таблицей, формируемой в результате выполнения запроса. Оно является разновидностью хранимого запроса. По одним и тем же таблицам можно построить несколько представлений.

**Задания:**

**Реляционный модель:**



-- Создаем базу данных

**CREATE DATABASE Carsharing;**

**USE Carsharing;**

-- Создаем таблицу Owners

**CREATE TABLE Owners (**

**id INT NOT NULL,**

**Owner\_name VARCHAR(25) NOT NULL,**

**Owner\_number VARCHAR(25) NOT NULL,**

**Techpass\_id VARCHAR(15) NOT NULL,**

**CONSTRAINT pk\_owner PRIMARY KEY (id)**

**);**

**-- Создаем таблицу Bills**

```
CREATE TABLE Bills ( id INT  
NOT NULL, Total_cost FLOAT  
NOT NULL,  
Start_time DATETIME NOT NULL,  
End_time DATETIME NOT NULL,  
CONSTRAINT pk_bill PRIMARY KEY (id)  
);
```

**-- Создаем таблицу Clients**

```
CREATE TABLE Clients (  
id INT NOT NULL,  
Client_name VARCHAR(25) NOT NULL,  
Client_number VARCHAR(25) NOT NULL,  
Client_address VARCHAR(50) NOT NULL,  
CONSTRAINT pk_client PRIMARY KEY (id)  
);
```

**-- Создаем таблицу Cars**

```
CREATE TABLE Cars (  
id INT NOT NULL,  
Car_class VARCHAR(25) NOT NULL,  
Car_number VARCHAR(25) NOT NULL,  
Car_brand VARCHAR(25) NOT NULL,  
Insurance_id VARCHAR(25) NOT NULL,  
id_owner INT NOT NULL, CONSTRAINT  
pk_car PRIMARY KEY (id), CONSTRAINT  
vk_owner FOREIGN KEY (id_owner)  
REFERENCES Owners(id)
```



);

-- Создаем таблицу Rents

```
CREATE TABLE Rents (  
  id INT NOT NULL,  
  Rent_cost FLOAT NOT NULL,  
  Reserve_time DATETIME NOT NULL,  
  Duration TIME NOT NULL,    id_bill  
  INT NOT NULL,    id_car INT NOT  
  NULL,    id_client INT NOT NULL,  
  CONSTRAINT pk_rent PRIMARY KEY (id),  
  CONSTRAINT vk_bill FOREIGN KEY (id_bill) REFERENCES  
Bills(id),  
  CONSTRAINT vk_car FOREIGN KEY (id_car) REFERENCES  
Cars(id),  
  CONSTRAINT vk_client FOREIGN KEY (id_client) REFERENCES  
Clients(id)  
);
```

✓	9	11:03:36	CREATE DATABASE Carsharing	1 row(s) affected
✓	10	11:03:36	USE Carsharing	0 row(s) affected
✓	11	11:03:36	CREATE TABLE Owners ( id INT NOT NULL, Owner_name VARCHAR(25) NOT NULL, Owner_numb...	0 row(s) affected
✓	12	11:03:36	CREATE TABLE Bills ( id INT NOT NULL, Total_cost FLOAT NOT NULL, Start_time DATETIME NO...	0 row(s) affected
✓	13	11:03:36	CREATE TABLE Clients ( id INT NOT NULL, Client_name VARCHAR(25) NOT NULL, Client_number ...	0 row(s) affected
✓	14	11:03:36	CREATE TABLE Cars ( id INT NOT NULL, Car_class VARCHAR(25) NOT NULL, Car_number VARC...	0 row(s) affected
✓	15	11:03:36	CREATE TABLE Rents ( id INT NOT NULL, Rent_cost FLOAT NOT NULL, Reserve_time DATETIME...	0 row(s) affected

- 1 • `use carsharing;`
- 2 • `show tables;`

Result Grid		Filter Rows:		Export:		Wrap Cell Content:	
	Tables_in_carsharing						
▶	bills						
	cars						
	clients						
	owners						
	rents						