**ЭКЗАМЕНАЦИОННЫЙ БИЛЕТ № 1**

1. Реляционная модель. Дать определения основных понятий: отношение, кортеж, атрибут, домен, первичный ключ, внешний ключ. Привести примеры. (20 баллов)

Реляционная модель данных - это способ рассмотрения данных, то есть предписание для способа представления данных (посредством таблиц) и для способа работы с таким представлением (посредством операторов). Она связана с тремя аспектами данных: структурой (объекты), целостностью и обработкой данных (операторы).

Цели создания реляционной модели данных:

* обеспечение более высокой степени независимости от данных;
* создание прочного фундамента для решения семантических вопросов и проблем непротиворечивости и избыточности данных;
* засширение языков управления данными за счёт включения операций над множествами.

Реляционная модель данных предусматривает структуру данных, обязательными объектами которой являются:

* отношение;
* атрибут;
* домен;
* кортеж;
* степень;
* кардинальность;
* первичный ключ.

**Отношение** - это плоская (двумерная) таблица, состоящая из столбцов и строк. Отношение является важнейшим понятием и представляет собой двумерную таблицу, содержащую некоторые данные. Также отношениями называют связи между таблицами.

**Кортеж** - это строка отношения.

**Атрибут** - это поименованный столбец отношения. Атрибуты представляют собой свойства, характеризующие сущность.

**Домен** - общая совокупность допустимых значений.

**Первичный ключ** - это поле, которое используется для обеспечения уникальности данных в таблице. Это означает, что значение (информация) в поле первичного ключа в каждой строке (записи) таблицы может быть уникальным.

**Внешний ключ** — это столбец (или группа столбцов), используемый для связи данных между таблицами.

1. Составить схему БД предметной области. Написать SQL-скрипт для создания таблиц. Определить первичные и внешние ключи, тип данных каждого атрибута, а также декларативные ограничения целостности - возможность принимать неопределенное значение, уникальные ключи, проверочные ограничения и т. д. (20 баллов)

Предметная область: Кинотеатр.

* Сущность «Фильм». Атрибуты сущности - наименование, год выпуска, режиссер, страна, жанр, краткое описание.
* Сущность «Зал кинотеатра». Атрибуты сущности - наименование зала, количество рядов, количество мест.
* Сущность «Расписание». Атрибуты сущности - дата и время начала сеанса, продолжительность сеанса, фильм, зал, количество занятых мест.

Create database Bilet1

Go

Use Bilet1

Create table Movies(  
Id int primary key identity(1,1),  
Name nvarchar(max),  
Year datetime2,  
Director nvarchar(max),  
Country nvarchar(max),  
Genre nvarchar(max),  
Description nvarchar(max))  
  
Create table CinemaHalls(  
Id int primary key identity(1,1),  
Name nvarchar(max),  
RowsCount int,  
SeatsCount int)  
  
Create table Seanses(  
Id int primary key identity(1,1),  
StartTime datetime2,  
EndTime datetime2,  
LongTime datetime2,  
MovieId int not null,  
Foreign Key(MovieId) references Movies(Id),  
HallId int not null,  
Foreign Key(HallId) references CinemaHalls(Id),  
DisabledPlacementsCount int)

1. Составить хранимую процедуру для поиска сеансов в выбранном диапазоне дат с определением количества оставшихся свободных мест на каждый сеанс. Используя хранимую процедуру найти все сеансы 2018 года. (20 баллов)

create procedure getSeanses  
@startDate datetime2,  
@endDate datetime2 as  
Select \*, CH.SeatsCount - S.DisabledPlacementsCount as FreeSeats from Seanses S  
Join CinemaHalls CH on S.HallId = [CH.Id](https://vk.com/away.php?utf=1&to=http%3A%2F%2FCH.Id" \t "_blank)  
where StartTime < @startDate and EndTime < @endDate

**ЭКЗАМЕНАЦИОННЫЙ БИЛЕТ № 2**

1. Перечислить теоретико-множественные операции реляционной алгебры. Дать определение операций выборки, проекции и объединения. Привести примеры. (20 баллов)

Реляциционная алгебра состоит из операций над отношениями и их составляющими (атрибутами и кортежами). Результат любой операции реляционной алгебры - новое отношение. Такие системы операций называются замкнутыми. Они разделены на две группы. В первую входят операции, совершаемые над любыми множествами:

* объединение,
* пересечение,
* разность
* декартово произведение.

Во вторую группу входят операции, применимые только к отношениям:

* выборка,
* проекция,
* соединение.

**Выборка** производится над кортежами одного отношения. Результат выборки - новое отношение, состоящее из котежей исходного отношения, удовлетворяющих заданному условию.

*Пример выборки*

Из отношения *Жители* нужно выбрать жителей, младше 30 лет

|  |  |
| --- | --- |
| Отношение *Жители* | |
| **ФИО** | **Возраст** |
| Андреев | 31 |
| Иванов | 21 |
| Перов | 40 |
| Яковлев | 27 |

На языке SQL запрос запрос выглядит так:

SELECT \* FROM Жители

WHERE Возраст > 30;

Результат выборки

|  |  |
| --- | --- |
| **ФИО** | **Возраст** |
| Андреев | 31 |
| Перов | 40 |

**Проекция** также производится над кортежами одного отношения. Результат проекции - новое отношение содержащее только заданные атрибуты исходного отношения.

*Пример проекции*

Из отношения *Жители* нужно выбрать только фамилии жителей

|  |  |  |
| --- | --- | --- |
| Отношение *Жители* | | |
| **Имя** | **ФИО** | **Возраст** |
| Юрий | Иванов | 31 |
| Сергей | Иванов | 21 |
| Владимир | Перов | 40 |
| Игорь | Перов | 27 |

На языке SQL запрос запрос выглядит так:

SELECT DISTINCT ФИО FROM Жители;

Результат выборки

|  |
| --- |
| **ФИО** |
| Иванов |
| Перов |

**Объединение** - операция над двумя отношениями, в результате которой получается новое отношение, состоящее из всех кортежей исходных отношений. Общие для исходных отношений кортежи в новом отношении встречются только по одному разу.

*Пример объединения*

Нужно объединить два отношения *Физ\_лица* и *Юр\_лица*.

|  |  |  |
| --- | --- | --- |
| Отношение *Физ\_лица* | | |
| **ФИО** | **Адр\_регистрации** | **Факт\_адр** |
| Иванов Ю.М. | Москва, Тверская 2 | С.-Петербург,Садовая ул. 12 |
| Сергеев И.А. | С.-Петербург, Седова 23 | С.-Петербург, Гороховая ул. 34 |
| ..... | ..... | ..... |

|  |  |  |
| --- | --- | --- |
| Отношение *Юр\_лица* | | |
| **Наим** | **Адр\_регистрации** | **Адр\_офиса** |
| Альфа | Новгород, Садовая ул. 2 | С.-Петербург,Садовая ул. 42 |
| Бета. | С.-Петербург, Московский пр. 23 | Гатчина, Лесная ул. 34 |
| ..... | ..... | ..... |

Результаты обеих выборок выводятся в общую таблицу. Оба оператора объединяются в один запрос предложением UNION:

SELECT ФИО AS ИМЯ Физ\_лица.Адр\_регистрации AS Адр\_официальный,

Факт\_адр AS Фактический\_адр

FROM Физ\_лица

UNION

SELECT Наим AS ИМЯ Физ\_лица.Адр\_регистрации AS Адр\_официальный,

Адр\_офиса AS Фактический\_адр

FROM Юр\_лица;

Результат запроса:

|  |  |  |
| --- | --- | --- |
| **ИМЯ** | **Адр\_официальный** | **Фактический\_адр** |
| Иванов Ю.М. | Москва, Тверская 2 | С.-Петербург,Садовая ул. 12 |
| Сергеев И.А. | С.-Петербург, Седова 23 | С.-Петербург, Гороховая ул. 34 |
| Альфа | Новгород, Садовая ул. 2 | С.-Петербург,Садовая ул. 42 |
| Бета. | С.-Петербург, Московский пр. 23 | Гатчина, Лесная ул. 34 |
| ..... | ..... | ..... |

1. Составить схему БД предметной области. Написать SQL-скрипт для создания таблиц. Определить первичные и внешние ключи, тип данных каждого атрибута, а также декларативные ограничения целостности - возможность принимать неопределенное значение, уникальные ключи, проверочные ограничения и т. д. (20 баллов)

Предметная область: Библиотека.

* Сущность «Книга». Атрибуты сущности - номер, автор, наименование, издательство, год издания, тираж, количество страниц, аннотация.
* Сущность «Читательский билет». Атрибуты сущности - номер, ФИО, дата рождения, номер телефона, дата выдачи.
* Сущность «Книговыдача». Атрибуты сущности - номер книги, номер читателя, дата выдачи, срок возврата, дата возврата.

Create database Bilet2

go

Use Bilet2

go

Create table Books(

Id int primary key identity(1,1),

Author nvarchar(max),

Name nvarchar(max),

YearOfRelease datetime2,

Publisher nvarchar(max),

PagesCount int,

Annotation nvarchar(max)

)

Create table Ticket(

Id int primary key identity(1,1),

FIO nvarchar(max),

DateOfBirth datetime2,

PhoneNumber int,

PeriodOfExtradition nvarchar,

DateOfExtradition datetime2

)

Create table BookExtradition(

BookId int,

TicketId int,

DateOfExtradition datetime2,

RefundDeadline nvarchar(max),

RefundDate datetime2,

foreign key (TicketId) references Ticket(Id),

foreign key (BookId) references Books(Id)

)

1. Составить хранимую процедуру для поиска должников на заданную дату c определением количества задолженных книг. Используя хранимую процедуру найти должников на 31.12.2018 г. (20 баллов)

create procedure N3  
@date datetime2 as  
Select T.FIO, Count(BE.TicketId) from BookExtradition BE  
Join Books B on BE.BookId = [B.Id](https://vk.com/away.php?utf=1&to=http%3A%2F%2FB.Id" \t "_blank)  
Join Ticket T on BE.TicketId = [T.Id](https://vk.com/away.php?utf=1&to=http%3A%2F%2FT.Id" \t "_blank)  
where BE.RefundDeadline < @date  
group by T.FIO

**ЭКЗАМЕНАЦИОННЫЙ БИЛЕТ № 3**

1. Перечислить теоретико-множественные операции реляционной алгебры. Дать определение операций пересечения, разности, произведения. Привести примеры. (20 баллов)

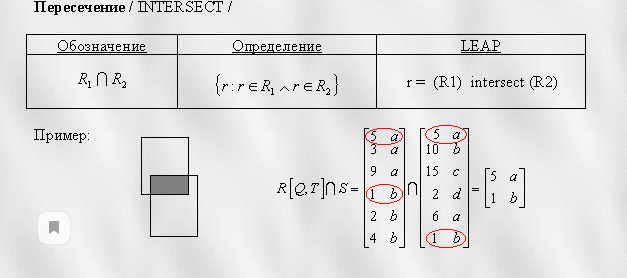
Реляциционная алгебра состоит из операций над отношениями и их составляющими (атрибутами и кортежами). Результат любой операции реляционной алгебры - новое отношение. Такие системы операций называются замкнутыми. Они разделены на две группы. В первую входят операции, совершаемые над любыми множествами:

* объединение,
* пересечение,
* разность
* декартово произведение.

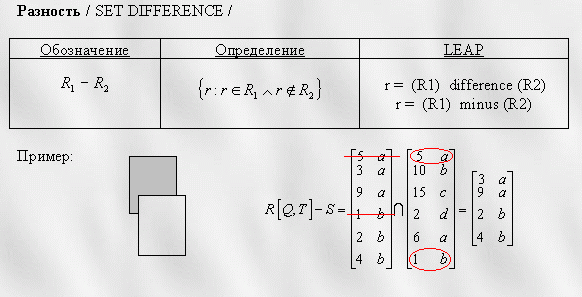
Во вторую группу входят операции, применимые только к отношениям:

* выборка,
* проекция,
* соединение.

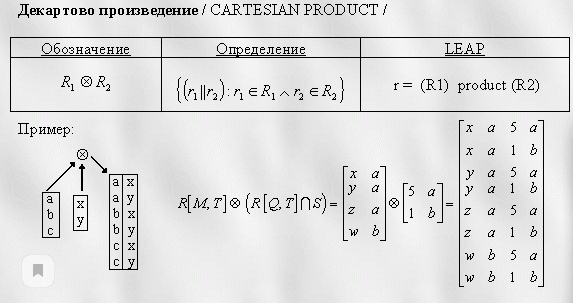
**Пересечение** - операция над двумя отношениями, в результате которой получается новое отношение, состоящее из кортежей, принадлежащих обоим исходным отношениям.



**Разность** - операция над двумя отношениями, в результате которой получается новое отношение, состоящее из кортежей, принадлежащих первому отношению и не принадлежащих второму.



**Декартово произведение** - операция над двумя отношениями, в результате которой получается новое отношение, состоящее из всех возможных кортежей, являющихся попарными сочетаниями кортежей исходных отношений.



1. Составить схему БД предметной области. Написать SQL-скрипт для создания таблиц. Определить первичные и внешние ключи, тип данных каждого атрибута, а также декларативные ограничения целостности - возможность принимать неопределенное значение, уникальные ключи, проверочные ограничения и т. д. (20 баллов)

Предметная область: Расписание экзаменов.

* Сущность «Группа». Атрибуты сущности - номер, наименование, год создания, наименование факультета, ФИО старосты, число студентов.
* Сущность «Дисциплина». Атрибуты сущности - номер, наименование.
* Сущность «Экзамен». Атрибуты сущности - номер дисциплины, ФИО преподавателя, дата экзамена, время экзамена, аудитория для экзамена.

Create database Bilet3

go

Use Bilet3

go

Create table Groop(

Id int Primary key identity(1,1),

Name nvarchar(max),

YearOfCreate datetime2,

FaculityName nvarchar(max),

FIOElder nvarchar(max),

StudentCount int,

)

Create table Subject(

Id int Primary key identity(1,1),

Name nvarchar(max)

)

Create table Exam(

SubjectId int,

Foreign key (SubjectId) references Subject(Id),

GroopId int,

Foreign key (GroopId) references Groop(Id),

FIOTeacher nvarchar(max),

Date datetime2,

PeriodExam nvarchar(max),

ExamCabinet nvarchar(max)

)

1. Составить хранимую процедуру для поиска экзаменов определенной группы с указанием статуса экзамена («не начат» – для еще не наступившей даты экзамена, «в процессе» – для наступившей даты экзамена, «завершен» – если дата прохождения экзамена прошла. Используя хранимую процедуру найти экзамены группы 2 БИ. (20 баллов)

create procedure getSmth  
@GroupId int,  
@status nvarchar  
as  
if @status like '%не начат%'  
Select \* from Exam  
Join Subject S on E.SubjectId = [S.Id](https://vk.com/away.php?utf=1&to=http%3A%2F%2FS.Id" \t "_blank)  
where GroopId = @GroupId and Cast((SELECT DATEDIFF(DAY, Date, GetDate())) as int) < 0  
if @status like '%в процессе%'  
Select \* from Exam  
Join Subject S on E.SubjectId = [S.Id](https://vk.com/away.php?utf=1&to=http%3A%2F%2FS.Id" \t "_blank)  
where GroopId = @GroupId and Cast((SELECT DATEDIFF(DAY, Date, GetDate())) as int) = 0  
if @status like '%завершен%'  
Select [S.Name](https://vk.com/away.php?utf=1&to=http%3A%2F%2FS.Name" \t "_blank) from Exam E  
Join Subject S on E.SubjectId = [S.Id](https://vk.com/away.php?utf=1&to=http%3A%2F%2FS.Id" \t "_blank)  
where GroopId = @GroupId and Cast((SELECT DATEDIFF(DAY, Date, GetDate())) as int) > 0

**ЭКЗАМЕНАЦИОННЫЙ БИЛЕТ № 4**

1. Что такое проектирование реляционной базы данных? Что является результатом проектирования? Какие данные называются избыточными? Привести пример избыточных данных. Каким путем исключается избыточность. (20 баллов)

**Проектирование** – процесс, позволяющий провести некую техническую идею до её инженерной модели.

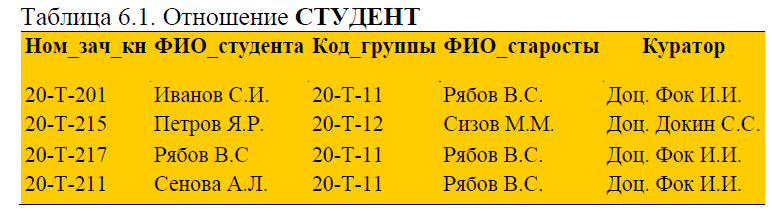
При проектировании схемы реляционной БД можно выделить следующие процедуры:

* определение перечня таблиц и связей между ними;
* определение перечня полей, типов полей, ключевых полей каждой таблицы (схемы таблицы), установление связей между таблицами через внешние ключи;
* установление индексирования для полей в таблицах;
* разработка списков (словарей) для полей с перечислительными данными;
* установление ограничений целостности для таблиц и связей;
* нормализация таблиц, корректировка перечня таблиц и связей.

**Результат проектирования** – проектное решение (совокупность проектных решений), удовлетворяющее заданным требованиям, необходимое для создания объекта проектирования.

**Избыточность данных в БД** относится к нежелательным явлениям, поскольку ведет к увеличению объема памяти, необходимого для физического хранения отношений. Избыточность вызывается, прежде всего, дублированием данных.

Вот характерный пример отношения содержащего нежелательную избыточность:



В данном отношении с первичным ключом Ном\_зач\_кн в каждом кортеже о каждом студенте из одной и той же группы повторяются сведения о коде группы, старосте и кураторе.

**Избыточность данных уменьшают** путём их перекодирования (дополнительного, специального кодирования). Эту операцию называют сжатием или уплотнением данных. Сжатие данных — это операция преобразования исходной выборки данных в результирующую выборку, представляющую адекватную информацию, но имеющую меньший размер.

1. Составить схему БД предметной области. Написать SQL-скрипт для создания таблиц. Определить первичные и внешние ключи, тип данных каждого атрибута, а также декларативные ограничения целостности - возможность принимать неопределенное значение, уникальные ключи, проверочные ограничения и т. д. (20 баллов)

Предметная область: Поликлиника.

* Сущность «Врач». Атрибуты сущности - номер, ФИО, специализация, номер кабинета.
* Сущность «Пациент». Атрибуты сущности - номер, ФИО, дата рождения, адрес, место работы, профессия.
* Сущность «Медицинская карта». Атрибуты сущности - номер пациента, номер врача, дата приема, вид приема - на дому или в больнице, жалобы, диагноз, назначение.

Create database Bilet4

go

Use Bilet4

go

Create table Doctor(

Id int Primary key identity(1,1),

FIO nvarchar(max),

Specialisation nvarchar(max),

CabinetNumber int,

)

Create table Patient(

Id int Primary key identity(1,1),

FIO nvarchar(max),

DateOfBirth datetime2,

Address nvarchar(max),

WorkPlace nvarchar(max),

Profecion nvarchar(max)

)

Create table MedCard(

PatientId int,

Foreign key (PatientId) references Patient(Id),

DoctorId int,

Foreign key (DoctorId) references Doctor(Id),

DateOfReception datetime2,

TypeOfReception text,

Complaints nvarchar(max),

Diagnosis nvarchar(max),

Appointment nvarchar(max),

)

1. Составить хранимую процедуру для вывода медицинской карты определенного пациента (с выводом имени врача, осуществившего прием, и номера кабинета) в хронологическом порядке. Используя хранимую процедуру найти медицинскую карту пациента с фамилией Иванов. (20 баллов)

create procedure getsmth  
@PatientId int  
as  
Select D.FIO, D.CabinetNumber from MedCard M  
Join Doctor D on M.DoctorId = [D.Id](https://vk.com/away.php?utf=1&to=http%3A%2F%2FD.Id" \t "_blank)  
where M.PatientId = @PatientId  
order by M.DateOfReception

**ЭКЗАМЕНАЦИОННЫЙ БИЛЕТ № 5**

1. Виды ключей. Первичный ключ таблицы. Свойства первичного ключа. Привести примеры. (20 баллов)

**Ключ** — это колонка (column) или колонки, не имеющие в строках дублирующих значений.

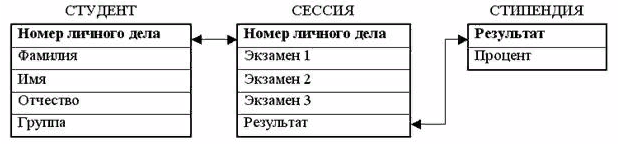
**Внешние ключи.** Внешний ключ — это столбец, значения которого соответствуют значениям первичного ключа другой связанной таблицы.

**Составные ключи.** Если для выполнения условий, накладываемых на значения первичного ключа, заданный ключ включает несколько полей таблицы, то тогда он называется составным.

**Первичный ключ.** Первичный ключ состоит из набора значений, которые однозначно определяют запись базовой таблицы. Любому значению первичного ключа должна соответствовать одна и только одна строка таблицы. Первичный ключ включает одно поле только в том случае, если это поле не содержит повторяющихся значений.

Первичный ключ должен обладать двумя свойствами:

* **однозначная идентификация записи** – запись должна однозначно определяться значением ключа;
* **отсутствие избыточности** – никакое поле нельзя удалить из ключа, не нарушая при этом свойства однозначной идентификации записи.



1. Составить схему БД предметной области. Написать SQL-скрипт для создания таблиц. Определить первичные и внешние ключи, тип данных каждого атрибута, а также декларативные ограничения целостности - возможность принимать неопределенное значение, уникальные ключи, проверочные ограничения и т. д. (20 баллов)

Предметная область: Кулинария.

* Сущность «Блюдо». Атрибуты сущности - номер, наименование, категория, калорийность, жиры, белки, углеводы, рецепт.
* Сущность «Продукт». Атрибуты сущности - номер, наименование, поставщик, цена за единицу, единица измерения.
* Сущность «Продукты блюда». Атрибуты сущности - номер блюда, номер продукта, количество единиц продукта.

Create database Bilet5

go

Use Bilet5

go

Create table Dish(

Id int Primary key identity(1,1),

Name nvarchar(max),

Category nvarchar(max),

Caloric int,

Fats int,

Proteins int,

Carbohydrates int,

Recipe nvarchar(max)

)

Create table Product(

Id int Primary key identity(1,1),

Name nvarchar(max),

Supplier nvarchar(max),

UnitPrice money,

UnitOfMeasurement text

)

Create table ProductOfDish(

DishId int,

foreign key (DishId) references Dish(Id),

ProductId int,

foreign key (ProductId) references Product(Id),

ProductCount int

)

1. Составить хранимую процедуру для вывода стоимости создания заданного блюда (входной параметр) и количества используемых в блюде продуктов. Используя хранимую процедуру найти стоимость создания блюда с наименованием «Борщ» и количество используемых продуктов. (20 баллов)

create procedure  
@DishId int  
as  
Select Count(PD.ProductId) as productNum, PD.ProductCount \* P.UnitPrice as price from ProductOfDish PD  
Join Product P on PD.ProductId = [P.Id](https://vk.com/away.php?utf=1&to=http%3A%2F%2FP.Id" \t "_blank)  
where PD.DishId = @DishId  
group by [P.Id](https://vk.com/away.php?utf=1&to=http%3A%2F%2FP.Id" \t "_blank), Pd.ProductCount, P.UnitPrice

**ЭКЗАМЕНАЦИОННЫЙ БИЛЕТ № 6**

1. Создание связей между таблицами. Виды связей. Главная и подчиненная таблицы. Привести примеры. (20 баллов)

**Виды связей**

* **«один к одному». В этом случае каждой записи одной таблицы соответствует только одна запись другой таблицы;**

Чтобы создать связь "один-ко-многим" в структуре базы данных, добавьте первичный ключ на стороне "один" в таблицу на стороне "многие" в виде дополнительного поля.

* **«один ко многим»**. Это когда одной записи главной таблицы (master) соответствует несколько записей подчиненной таблицы (detail).

При связи "один-к-одному" каждая запись в первой таблице может иметь не более одной связанной записи во второй таблице и наоборот.

* **«многие ко многим»**. Это когда в обоих таблицах существует несколько взаимосвязанных записей

Чтобы представить связь "многие-ко-многим", нужно создать третью (связующую) таблицу, в которой она разбивается на две связи "один-ко-многим". Первичные ключи двух таблиц вставляются в третью таблицу. В результате в третьей таблице сохраняются все экземпляры связи.

Главная таблица {родительская таблица или master) - это таблица, в которой содержатся основные данные.

Подчиненная таблица {дочерняя таблица или detail) - таблица, значения в полях которой зависят от значений главной таблицы.

1. Составить схему БД предметной области. Написать SQL-скрипт для создания таблиц. Определить первичные и внешние ключи, тип данных каждого атрибута, а также декларативные ограничения целостности - возможность принимать неопределенное значение, уникальные ключи, проверочные ограничения и т. д. (20 баллов)

Предметная область: Агентство недвижимости.

* Сущность «Квартира». Атрибуты сущности - номер, описание, адрес, ФИО владельца, площадь, число комнат, этаж, цена.
* Сущность «Агент». Атрибуты сущности – номер, ФИО агента, номер телефона, процент комиссионных.
* Сущность «Сделка». Атрибуты сущности - номер квартиры, номер агента, ФИО покупателя, номер телефона покупателя, сумма для оплаты, вид оплаты (наличными/картой), дата заключения сделки, статус оплаты (оплачено/не оплачено).

Create database Bilet6

go

Use Bilet6

go

Create table Apartment(

Id int Primary key identity(1,1),

Description nvarchar(max),

Address nvarchar(max),

OwnerFIO nvarchar(max),

Area nvarchar(max),

PremisesCount int,

Level int,

Price money

)

Create table Agent(

Id int Primary key identity(1,1),

FIO nvarchar(max),

PhoneNumber int,

PercentOfCommition int,

)

Create table Deal(

ApartmentId int,

Foreign key (ApartmentId) references Apartment(Id),

AgentId int,

Foreign key (AgentId) references Agent(Id),

FIOClient nvarchar(max),

PhoneNumber int,

SumOfPurshare money,

TypeOfPurshare text,

DateOfDeal datetime2,

StatusOfDeal text

)

1. Составить хранимую процедуру для поиска сделок определенного агента (входной параметр) за заданный месяц (входной параметр). Формат вывода - дата заключения сделки, адрес квартиры, цена квартиры, ФИО агента, размер комиссионных агента, размер полученных комиссионных со сделки. Привести пример использования хранимой процедуры. (20 баллов)

**ЭКЗАМЕНАЦИОННЫЙ БИЛЕТ № 7**

1. Что такое тип данных? Перечислить и кратко охарактеризовать основные типы данных MS SQL Server. Привести примеры использования. (20 баллов)

Под **типом** **данных** (ТД) понимается описание **данных** определенного вида, для которых известен их способ представления в памяти (формат), следующие из него размерность и диапазон значений, а также определен набор операций.

**Числовые типы данных**

* **BIT**: хранит значение 0 или 1. Фактически является аналогом булевого типа в языках программирования. Занимает 1 байт.
* **INT**: хранит числа от –2 147 483 648 до 2 147 483 647. Занимает 4 байта. Наиболее используемый тип для хранения чисел.
* **DECIMAL**: хранит числа c фиксированной точностью. Занимает от 5 до 17 байт в зависимости от количества чисел после запятой.
* **NUMERIC**: данный тип аналогичен типу DECIMAL.
* **MONEY**: хранит дробные значения от -922 337 203 685 477.5808 до 922 337 203 685 477.5807. Представляет денежные величины и занимает 8 байт. Эквивалентен типу DECIMAL(19,4).
* **FLOAT**: хранит числа от –1.79E+308 до 1.79E+308. Занимает от 4 до 8 байт в зависимости от дробной части.
* **REAL**: хранит числа от –340E+38 to 3.40E+38. Занимает 4 байта. Эквивалентен типу FLOAT(24).

#### Типы данных, представляющие дату и время

* **DATE**: хранит даты от 0001-01-01 (1 января 0001 года) до 9999-12-31 (31 декабря 9999 года). Занимает 3 байта.
* **TIME**: хранит время в диапазоне от 00:00:00.0000000 до 23:59:59.9999999. Занимает от 3 до 5 байт.
* **DATETIME**: хранит даты и время от 01/01/1753 до 31/12/9999. Занимает 8 байт.
* **DATETIME2**: хранит даты и время в диапазоне от 01/01/0001 00:00:00.0000000 до 31/12/9999 23:59:59.9999999. Занимает от 6 до 8 байт в зависимости от точности времени.
* **SMALLDATETIME**: хранит даты и время в диапазоне от 01/01/1900 до 06/06/2079, то есть ближайшие даты. Занимает от 4 байта.
* **DATETIMEOFFSET**: хранит даты и время в диапазоне от 0001-01-01 до 9999-12-31. Сохраняет детальную информацию о времени с точностью до 100 наносекунд. Занимает 10 байт.

#### Строковые типы данных

* **CHAR**: хранит строку длиной от 1 до 8 000 символов. На каждый символ выделяет по 1 байту. Не подходит для многих языков, так как хранит символы не в кодировке Unicode.
* **VARCHAR**: хранит строку. На каждый символ выделяется 1 байт. Можно указать конкретную длину для столбца - от 1 до 8 000 символов, например, VARCHAR(10). Если строка должна иметь больше 8000 символов, то задается размер MAX, а на хранение строки может выделяться до 2 Гб: VARCHAR(MAX).
* **NCHAR**: хранит строку в кодировке Unicode длиной от 1 до 4 000 символов. На каждый символ выделяется 2 байта. Например, NCHAR(15)
* **NVARCHAR**: хранит строку в кодировке Unicode. На каждый символ выделяется 2 байта.Можно задать конкретный размер от 1 до 4 000 символов: . Если строка должна иметь больше 4000 символов, то задается размер MAX, а на хранение строки может выделяться до 2 Гб.

#### Бинарные типы данных

* **BINARY**: хранит бинарные данные в виде последовательности от 1 до 8 000 байт.
* **VARBINARY**: хранит бинарные данные в виде последовательности от 1 до 8 000 байт, либо до 2^31–1 байт при использовании значения MAX (VARBINARY(MAX)).

#### Остальные типы данных

* **UNIQUEIDENTIFIER**: уникальный идентификатор GUID (по сути строка с уникальным значением), который занимает 16 байт.
* **TIMESTAMP**: некоторое число, которое хранит номер версии строки в таблице. Занимает 8 байт.
* **CURSOR**: представляет набор строк.
* **HIERARCHYID**: представляет позицию в иерархии.
* **SQL\_VARIANT**: может хранить данные любого другого типа данных T-SQL.
* **XML**: хранит документы XML или фрагменты документов XML. Занимает в памяти до 2 Гб.
* **TABLE**: представляет определение таблицы.
* **GEOGRAPHY**: хранит географические данные, такие как широта и долгота.
* **GEOMETRY**: хранит координаты местонахождения на плоскости.

1. Составить схему БД предметной области. Написать SQL-скрипт для создания таблиц. Определить первичные и внешние ключи, тип данных каждого атрибута, а также декларативные ограничения целостности - возможность принимать неопределенное значение, уникальные ключи, проверочные ограничения и т. д. (20 баллов)

Предметная область: Компания.

* Сущность «Поставщик». Атрибуты сущности - номер, Наименование поставщика, город, номер телефона.
* Сущность «Товар». Атрибуты сущности - номер, наименование, тип товара, стоимость единицы товара.
* Сущность «Заказ». Атрибуты сущности - номер заказа, дата заказа, номер товара, номер поставщика, количество единиц товара, стоимость, статус оплаты заказа – оплачен/не оплачен.

Create database Bilet7

go

Use Bilet7

go

Create table Supplier(

Id int Primary key identity(1,1),

Name nvarchar(max),

City nvarchar(max),

PhoneNumber int

)

Create table Product(

Id int Primary key identity(1,1),

Name nvarchar(max),

TypeOfProduct nvarchar(max),

UnitPrice money

)

Create table Orders(

Id int Primary key identity(1,1),

DateOfOrder datetime2,

ProductId int,

Foreign key (ProductId) references Product(Id),

SupplierId int,

Foreign key (SupplierId) references Supplier(Id),

ProductCount int,

Price money,

OrderPurchareStatus text

)

1. Составить представление для получения списка товаров. По каждому товару необходимо выводить общее количество заказов товара, количество оплаченных заказов, суммарную стоимость всех заказов товара. Используя представление вывести все товары. (20 баллов)

**ЭКЗАМЕНАЦИОННЫЙ БИЛЕТ № 8**

1. Перечислить команды языка T-SQL для создания таблиц. Создание полей и управление их свойствами. (20 баллов)

Для **создания таблиц** применяется команда CREATE TABLE. С этой командой можно использовать ряд операторов, которые определяют столбцы таблицы и их атрибуты. И кроме того, можно использовать ряд операторов, которые определяют свойства таблицы в целом.

**Переименование таблицы**

Для переименования таблиц применяется системная хранимая процедура "sp\_rename".

Поля таблицы прописываются согласно следующему примеру синтаксиса:

CREATE TABLE Customers

(

    Id INT PRIMARY KEY,

    Age INT,

    FirstName NVARCHAR(20),

    LastName NVARCHAR(20),

    Email VARCHAR(30),

    Phone VARCHAR(20)

)

Для создания и управления свойствами полей таблицы применяются следующие команды:

* PRIMARY KEY С помощью выражения **PRIMARY KEY** столбец можно сделать первичным ключом.
* IDENTITY Атрибут **IDENTITY** позволяет сделать столбец идентификатором.
* UNIQUE Если мы хотим, чтобы столбец имел только уникальные значения, то для него можно определить атрибут **UNIQUE**.
* NULL и NOT NULL Чтобы указать, может ли столбец принимать значение NULL, при определении столбца ему можно задать атрибут **NULL** или **NOT NULL**. Если этот атрибут явным образом не будет использован, то по умолчанию столбец будет допускать значение NULL.
* DEFAULT Атрибут **DEFAULT** определяет значение по умолчанию для столбца.
* CHECK Ключевое слово **CHECK** задает ограничение для диапазона значений, которые могут храниться в столбце.
* Оператор CONSTRAINT С помощью ключевого слова **CONSTRAINT** можно задать имя для ограничений. В качестве ограничений могут использоваться PRIMARY KEY, UNIQUE, DEFAULT, CHECK.

1. Составить схему БД предметной области. Написать SQL-скрипт для создания таблиц. Определить первичные и внешние ключи, тип данных каждого атрибута, а также декларативные ограничения целостности - возможность принимать неопределенное значение, уникальные ключи, проверочные ограничения и т. д. (20 баллов)

Предметная область: Прокат автомобилей.

* Сущность «Автомобиль». Атрибуты сущности - номер, марка, модель, год выпуска, госномер, цвет, стоимость одного дня проката.
* Сущность «Клиент». Атрибуты сущности - номер, ФИО, серия паспорта, номер паспорта, номер телефона, адрес.
* Сущность «Заказ». Атрибуты сущности - номер, номер автомобиля, номер клиента, дата обращения, дата проката, количество дней проката.

Create database Bilet8

go

Use Bilet8

go

Create table Auto(

Id int Primary key identity(1,1),

Brend nvarchar(max),

Model nvarchar(max),

YearOfRelease datetime2,

StateNumber nvarchar(max),

Colour text,

PriceOfOneDayRental money

)

Create table Client(

Id int Primary key identity(1,1),

FIO nvarchar(max),

PasswordSeries int,

PasswordNumber int,

PhoneNumber int,

Address nvarchar(max)

)

Create table Orders(

Id int Primary key identity(1,1),

AutoId int,

Foreign key (AutoId) references Auto(Id),

ClientId int,

Foreign key (ClientId) references Client(Id),

DateOfPetition datetime2,

DateOfRental datetime2,

DaysOfRentalCount int

)

1. Составить представление для получения списка Заказов автомобилей в следующем виде: номер заказа, дата проката, количество дней проката, ФИО клиента, серия и номер паспорта, марка и модель автомобиля, госномер, стоимость дня проката, стоимость всех дней проката. При помощи представления отобразить список всех заказов за 2018 год. (20 баллов)

**ЭКЗАМЕНАЦИОННЫЙ БИЛЕТ № 9**

1. Перечислить команды языка для создания и изменения структуры таблиц средствами T-SQL. Удаление таблиц. (20 баллов)

Возможно, в какой-то момент мы захотим изменить уже имеющуюся таблицу. Например, добавить или удалить столбцы, изменить тип столбцов, добавить или удалить ограничения. То есть потребуется изменить определение таблицы. Для изменения таблиц используется выражение **ALTER TABLE**.

Для изменения уже существующей структуры таблиц существуют следующие типы команд:

* Добавление нового столбца (alter table (наз. таблицы) add (наз. слобца))
* Удаление столбца (alter table (наз. таблицы) drop column (наз. слобца))
* Изменение типа столбца (alter table (наз. таблицы) alter column (наз. слобца))
* Добавление ограничения CHECK (alter table (наз. таблицы) add check (наз. ограничения))
* Добавление внешнего ключа (alter table (наз. таблицы) add foreign key (наз. поля ключа) references (наз. таблицы(id)))
* Добавление первичного ключа (alter table (наз. таблицы) add primary key (Id))
* Добавление ограничений с именами (alter table (наз. таблицы) add constraint check(наз. ограничения))
* Удаление ограничений (alter table (наз. таблицы) drop (ограничение))

Для удаления таблиц используется команда **DROP TABLE**, которая имеет следующий синтаксис:

|  |  |
| --- | --- |
| 1 | DROP TABLE table1 [, table2, ...] |

1. Составить схему БД предметной области. Написать SQL-скрипт для создания таблиц. Определить первичные и внешние ключи, тип данных каждого атрибута, а также декларативные ограничения целостности - возможность принимать неопределенное значение, уникальные ключи, проверочные ограничения и т. д. (20 баллов)

Предметная область: Учет нарушений правил дорожного движения.

* Сущность «Автомобиль». Атрибуты сущности - номер, марка, модель, год выпуска, госномер, цвет, ФИО владельца.
* Сущность «Вид нарушения». Атрибуты сущности - номер, наименование нарушения, размер штрафа.
* Сущность «Факт нарушения». Атрибуты сущности - номер вида нарушения, номер автомобиля, дата нарушения ПДД, статус – оплачен/не оплачен.

Create database Bilet9

go

Use Bilet9

go

Create table Auto(

Id int Primary key identity(1,1),

Brend nvarchar(max),

Model nvarchar(max),

YearOfRelease datetime2,

StateNumber nvarchar(max),

Colour text,

FIOOwner nvarchar(max)

)

Create table TypeOfViolation(

Id int Primary key identity(1,1),

Name nvarchar(max),

SizeOfPenalty money,

)

Create table FactOfViolation(

TypeOfViolationId int,

foreign key (TypeOfViolationId) references TypeOfViolation(Id),

AutoId int,

foreign key (AutoId) references Auto(Id),

DateOfViolation datetime2,

StatusOfViolation text

)

1. Составить представление для получения списка фактов нарушений. Данные представить в следующем виде: дата нарушения, наименование нарушения, госномер автомобиля, размер штрафа, статус оплаты штрафа. Используя представление, вывести список всех неоплаченных нарушений в порядке убывания размера штрафа. (20 баллов)

**ЭКЗАМЕНАЦИОННЫЙ БИЛЕТ № 10**

1. Общая характеристика, выполняемые функции структурированного языка запросов SQL. Синтаксис основной инструкции SQL для выборки данных. (20 баллов)

**SQL** — **это** специализированный непроцедурный **язык**, позволяющий описывать данные, осуществлять выборку и обработку информации из реляционных СУБД.

COUNT() — функция возвращающая количество записей (строк) таблицы.

AVG() — функция возвращающая среднее значение столбца.

MIN() — функция возвращающая минимальное значение столбца.

MAX() — функция возвращающая максимальное значение столбца таблицы.

SUM() — функция, возвращающая сумму значений столбца таблицы.

ROUND() — функция для округления десятичных чисел.

UCASE() — функция, возвращающая значения столбца или столбцов в верхнем регистре букв.

LCASE() — функция, возвращающая значения столбца или столбцов в нижнем регистре букв.

LEN() — функция, возвращающая длину значения в поле записи.

MID() — функция, выводящая определенное количество символов текстового поля таблицы.

NOW() — функция, возвращающая системное время и дату.

Для получения данных применяется команда **SELECT**. В упрощенном виде она имеет следующий синтаксис:

|  |  |
| --- | --- |
| 1 | SELECT список\_столбцов FROM имя\_таблицы |

1. Составить схему БД предметной области. Написать SQL-скрипт для создания таблиц. Определить первичные и внешние ключи, тип данных каждого атрибута, а также декларативные ограничения целостности - возможность принимать неопределенное значение, уникальные ключи, проверочные ограничения и т. д. (20 баллов)

Предметная область: Туристическое агентство.

* Сущность «Клиент». Атрибуты сущности – номер, ФИО, дата рождения, адрес, серия паспорта, номер паспорта, номер телефона.
* Сущность «Маршрут». Атрибуты сущности - номер, наименование, страна, стоимость 1 дня пребывания, стоимость оформления визы, стоимость транспортных услуг.
* Сущность «Поездка». Атрибуты сущности - номер клиента, номер маршрута, цель поездки, дата начала поездки, количество дней.

Create database Bilet10

go

Use Bilet10

go

Create table Client(

Id int Primary key identity(1,1),

FIO nvarchar(max),

DatOfBirth datetime2,

Address nvarchar(max),

PasswordSeries int,

PasswordNumber int,

PhoneNumber int

)

Create table Route(

Id int Primary key identity(1,1),

Name nvarchar(max),

Country nvarchar(max),

PriceForOneDay money,

PriceForVISA money,

PriceForTransport money

)

Create table Trip(

ClientId int,

Foreign key (ClientId) references Client(Id),

RouteId int,

Foreign key (RouteId) references Route(Id),

TripPorpose nvarchar(max),

DateOfStart datetime2,

DaysCount int

)

1. Составить представление для получения списка поездок. Данные представить в следующем виде: ФИО клиента, наименование маршрута, дата поездки, количество дней, стоимость поездки, рассчитанная по формуле: Стоимость 1 дня пребывания \* Количество дней + Стоимость транспортных услуг + Стоимость оформления визы. Используя представление, вывести список поездок в порядке убывания стоимости поездки. (20 баллов)

**ЭКЗАМЕНАЦИОННЫЙ БИЛЕТ № 11**

1. Выбор данных с сортировкой, удалением дублированных значений, задание условий сравнения, задание диапазонов, принадлежность списку значений, соответствие шаблону. Привести примеры. (20 баллов)

Выбор данных с сортировкой, удалением дублированных значений, задание условий сравнения, задание диапазонов, принадлежность списку значений, соответствие шаблону.

оператор SELECT – один из наиболее важных и самых распространенных операторов *SQL*. Он позволяет производить *выборки* данных из таблиц и преобразовывать к нужному виду полученные результаты. Будучи очень мощным, он способен выполнять действия, эквивалентные операторам реляционной алгебры, причем в пределах единственной выполняемой команды. При его помощи можно реализовать сложные и громоздкие условия отбора данных из различных таблиц.

Оператор SELECT – средство, которое полностью абстрагировано от вопросов представления данных, что помогает сконцентрировать внимание на проблемах доступа к данным. Примеры его использования наглядно демонстрируют один из основополагающих принципов больших (промышленных) *СУБД*: средства хранения данных и доступа к ним отделены от средств представления данных. *Операции* над данными производятся в масштабе наборов данных, а не отдельных записей.

Оператор SELECT имеет следующий формат:

SELECT [ALL | DISTINCT ] {\*|[имя\_столбца

[AS новое\_имя]]} [,...n]

FROM имя\_таблицы [[AS] псевдоним] [,...n]

[WHERE <условие\_поиска>]

[GROUP BY имя\_столбца [,...n]]

[HAVING <критерии выбора групп>]

[ORDER BY имя\_столбца [,...n]]

Оператор SELECT определяет поля (столбцы), которые будут входить в результат выполнения запроса. В списке они разделяются запятыми и приводятся в такой очередности, в какой должны быть представлены в результате запроса. Если используется имя поля, содержащее пробелы или разделители, его следует заключить в квадратные скобки. Символом \* можно выбрать все поля, а вместо имени поля применить выражение из нескольких имен.

Если обрабатывается ряд таблиц, то (при наличии одноименных полей в разных таблицах) в списке полей используется полная спецификация поля, т.е. Имя\_таблицы.Имя\_поля.

Предложение FROM

Предложение FROM задает имена таблиц и представлений, которые содержат поля, перечисленные в операторе SELECT. Необязательный параметр псевдонима – это сокращение, устанавливаемое для имени таблицы.

Обработка элементов оператора SELECT выполняется в следующей последовательности:

* FROM – определяются имена используемых таблиц;
* WHERE – выполняется фильтрация строк объекта в соответствии с заданными условиями;
* GROUP BY – образуются группы строк, имеющих одно и то же значение в указанном столбце;
* HAVING – фильтруются группы строк объекта в соответствии с указанным условием;
* SELECT – устанавливается, какие столбцы должны присутствовать в выходных данных;
* ORDER BY – определяется упорядоченность результатов выполнения операторов.

Порядок предложений и фраз в операторе SELECT не может быть изменен. Только два предложения SELECT и FROM являются обязательными, все остальные могут быть опущены. SELECT – закрытая операция: результат запроса к таблице представляет собой другую таблицу. Существует множество вариантов записи данного оператора, что иллюстрируется приведенными ниже примерами.

Пример 4.1. Составить список сведений о всех клиентах.

SELECT \* FROM Клиент

Пример 4.1. Список сведений о всех клиентах.

Параметр WHERE определяет критерий отбора записей из входного набора. Но в таблице могут присутствовать повторяющиеся записи (дубликаты). Предикат ALL задает включение в выходной набор всех дубликатов, отобранных по критерию WHERE. Нет необходимости указывать ALL явно, поскольку это значение действует по умолчанию.

SELECT ALL Клиент.Фирма FROM Клиент

Или (что эквивалентно)

SELECT Клиент.Фирма FROM Клиент

Результат выполнения запроса может содержать дублирующиеся значения, поскольку в отличие от операций реляционной алгебры оператор SELECT не исключает повторяющихся значений при выполнении выборки данных.

Предикат DISTINCT следует применять в тех случаях, когда требуется отбросить блоки данных, содержащие дублирующие записи в выбранных полях. Значения для каждого из приведенных в инструкции SELECT полей должны быть уникальными, чтобы содержащая их запись смогла войти в выходной набор.

Причиной ограничения в применении DISTINCT является то обстоятельство, что его использование может резко замедлить выполнение запросов.

1. Составить схему БД предметной области. Написать SQL-скрипт для создания таблиц. Определить первичные и внешние ключи, тип данных каждого атрибута, а также декларативные ограничения целостности - возможность принимать неопределенное значение, уникальные ключи, проверочные ограничения и т. д. (20 баллов)

Предметная область: Отдел кадров.

* Сущность «Преподаватель». Атрибуты сущности – номер, ФИО, дата рождения, пол, ученое звание – нет/кандидат наук/доктор наук, кафедра.
* Сущность «Дисциплина». Атрибуты сущности - номер, наименование, наименование группы, количество лекций, количество семинаров.
* Сущность «Нагрузка». Атрибуты сущности - номер преподавателя, номер дисциплины, тип нагрузки – лекция/семинар.

Create database Bilet11

go

Use Bilet11

go

Create table Teacher(

Id int primary key identity(1,1),

FIO nvarchar(max),

DateOfBirth datetime2,

Gender text,

AcademicTitle text

)

Create table Subject(

Id int primary key identity(1,1),

NameOfSubject nvarchar(max),

NameOfGroup nvarchar(max),

LectionsCount int,

SeminarsCount int

)

Create table WorkLoad(

TeacherId int,

Foreign key (TeacherId) references Teacher(Id),

SubjectId int,

Foreign key (SubjectId) references Subject(Id),

TypeOfWorkLoad text

)

1. Составить представление для получения списка кафедр, на которых работает менее 2-х докторов наук. Привести пример использования представления. (20 баллов)

**ЭКЗАМЕНАЦИОННЫЙ БИЛЕТ № 12**

1. Извлечение данных средствами T-SQL. Разделы WHERE, GROUP BY, HAVING, ORDER BY. (20 баллов)

Извлечение данных средствами T-SQL. Разделы WHERE, GROUP BY, HAVING, ORDER BY.

WHERE — оператор в [SQL](https://ru.wikipedia.org/wiki/SQL), указывающий, что оператор языка управления данными ([DML](https://ru.wikipedia.org/wiki/DML)) должен действовать только на записи, удовлетворяющие определенным критериям. Критерии должны быть описаны в форме [предикатов](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B5%D0%B4%D0%B8%D0%BA%D0%B0%D1%82). Раздел WHERE — не обязательный раздел в SQL ([DML](https://ru.wikipedia.org/wiki/DML)) предложениях. Он используется в качестве [условия в SQL-запросе](https://ru.wikipedia.org/wiki/%D0%A3%D1%81%D0%BB%D0%BE%D0%B2%D0%B8%D0%B5_%D0%B2_SQL-%D0%B7%D0%B0%D0%BF%D1%80%D0%BE%D1%81%D0%B5) для ограничения записей обрабатываемых в выражениях SQL ([DML](https://ru.wikipedia.org/wiki/DML)) или возвращаемых запросом.

Команда **GROUP BY** позволяет группировать результаты при выборке из базы данных.

К сгруппированным результатам можно применять любые функции (смотрите примеры).

См. также команду [having](http://old.code.mu/sql/having.html), которая позволяет накладывать условие на группы, созданные с помощью group by.

**Синтаксис**

SELECT \* FROM имя\_таблицы WHERE условие GROUP BY поле\_для\_группировки

Если предложение WHERE определяет предикат для фильтрации строк, то предложение HAVING применяется после группировки для определения аналогичного предиката, фильтрующего группы по значениям агрегатных функций. Это предложение необходимо для проверки значений, которые получены с помощью агрегатной функции не из отдельных строк источника записей, определенного в предложении FROM, а из групп таких строк. Поэтому такая проверка не может содержаться в предложении WHERE.

Сортирует данные, возвращенные запросом в SQL Server. Это предложение используется для следующих целей:

* Упорядочение результирующего набора запроса по заданному списку столбцов и (дополнительно) ограничение числа возвращаемых строк указанным диапазоном. Порядок, в котором строки возвращаются в результирующем наборе, не гарантируется, если не указано предложение ORDER BY.
* Определение порядка, в котором значения [ранжирующей функции](https://docs.microsoft.com/ru-ru/sql/t-sql/functions/ranking-functions-transact-sql?view=sql-server-ver15) применяются к результирующему набору.
* Указывает столбец или выражение, по которому производится сортировка результирующего набора запроса. Столбец сортировки может быть указан с помощью имени или псевдонима столбца или неотрицательного целого числа, представляющего позицию столбца в списке выбора.
* Можно указать несколько столбцов сортировки. Имена столбцов должны быть уникальными. Последовательность столбцов сортировки в предложении ORDER BY определяет организацию упорядоченного результирующего набора. Иными словами, результирующий набор сортируется по первому столбцу, затем упорядоченный список сортируется по второму и т. д.
* Имена столбцов, на которые содержатся ссылки в предложении ORDER BY, должны однозначно соответствовать столбцу или псевдониму столбца в списке выбора либо столбцу, определенному в таблице, указанной в предложении FROM. Если предложение ORDER BY ссылается на псевдоним столбца в списке выбора, псевдоним должен использоваться отдельно, а не как часть выражения в предложении ORDER BY

1. Составить схему БД предметной области. Написать SQL-скрипт для создания таблиц. Определить первичные и внешние ключи, тип данных каждого атрибута, а также декларативные ограничения целостности - возможность принимать неопределенное значение, уникальные ключи, проверочные ограничения и т. д. (20 баллов)

Предметная область: Деканат.

* Сущность «Студент». Атрибуты сущности - номер, ФИО, пол, дата рождения, серия паспорта, номер паспорта, бюджетник – да/нет.
* Сущность «Дисциплина». Атрибуты сущности - номер, наименование.
* Сущность «Успеваемость». Атрибуты сущности - номер студента, номер дисциплины, дата экзамена, оценка.

Create database Bilet12

go

Use Bilet12

go

Create table Student(

Id int primary key identity(1,1),

FIO nvarchar(max),

Gendre text,

DateOfBirth datetime2,

PasswordSeries int,

PasswordNumber int,

Budget text

)

Create table Subject(

Id int primary key identity(1,1),

Name nvarchar(max)

)

Create table AcademicPerfomance(

StudentId int,

Foreign key (StudentId) references Student(Id),

SubjectId int,

Foreign key (SubjectId) references Subject(Id),

DateOfExam datetime2,

Estimation int

)

1. Составить представление для вывода списка студентов. Вывод осуществить в следующем виде: ФИО, бюджетник – да/нет, средний балл студента. Используя представление, найти студентов, средний балл которых превышает 4,5 балла. (20 баллов)

**ЭКЗАМЕНАЦИОННЫЙ БИЛЕТ № 13**

1. Использование предикатов BETWEEN, IN, LIKE в запросах SQL. Привести примеры. (20 баллов)

Использование предикатов BETWEEN, IN, LIKE в запросах SQL. Привести примеры.

### **ОПЕРАТОР IN**

Оператор IN определяет набор значений в которое данное значение может или не может быть включено. В соответствии с нашей учебной базой данных на которой вы обучаетесь по настоящее времен, если вы хотите найти всех продавцов, которые размещены в Barcelona или в London, вы должны использовать следующий запрос ( вывод показывается в Рисунке 5.1 ):

SELECT \*

FROM Salespeople

WHERE city = 'Barcelona'

OR city = 'London';

Имеется и более простой способ получить ту же информацию:

SELECT \*

FROM Salespeople

WHERE city IN ( 'Barcelona', 'London' );

Вывод для этого запроса показывается в Рисунке 5.2.

### **ОПЕРАТОР BETWEEN**

Оператор BETWEEN похож на оператор IN. В отличии от определения по номерам из набора, как это делает IN, BETWEEN определяет диапазон, значения которого должны уменьшаться что делает предикат верным. Вы должны ввести ключевое слово BETWEEN с начальным значением, ключевое AND и конечное значение. В отличие от IN, BETWEEN чувствителен к порядку, и первое значение в предложении должно быть первым по алфавитному или числовому порядку. ( Обратите Внимание что, в отличие от Английского языка, SQL не говорит что "значение находится (между)BETWEEN значением и значением", а просто "значение BETWEEN значение значение". Это применимо и к оператору LIKE). Следующий пример будет извлекать из таблицы Продавцов всех продавцов с комиссионными между .10 и .12 (вывод показывается в Рисунке 5.4):

SELECT \*

FROM Salespeople

WHERE comm BETWEEN .10 AND .12;

Для включенного оператора BETWEEN, значение совпадающее с любым из двух значений границы ( в

### **ОПЕРАТОР LIKE**

LIKE применим только к полям типа CHAR или VARCHAR, с которыми он используется чтобы находить подстроки. Т.е. он ищет поле символа чтобы видеть, совпадает ли с условием часть его строки. В качестве условия он использует групповые символы(wildkards) - специальные символы которые могут соответствовать чему-нибудь. Имеются два типа групповых символов используемых с LIKE:  
LIKE может быть удобен если вы ищете им или другое значение, и если вы не помните как они точно пишутся. Предположим что вы неуверенны как записано по буквам им одного из ваших продавцов Peal или Peel. Вы можете просто использовать ту часть которую вы знаете и групповые символы чтобы находить все возможные

SELECT \*

FROM Salespeople

WHERE sname LIKE 'P \_ \_ l %';

Групповые символы подчеркивания, каждый из которых представляет один символ, добавят только два символа к уже существующим 'P' и 'l' , поэтому им наподобие Prettel не может быть показано. Групповой символ ' % ' - в конце строки необходим в большинстве реализаций если длина пол sname больше чем число символов в имени Peel ( потому что некоторые другие значения sname - длиннее чем четыре символа ). В таком случае, значение пол sname , фактически сохраняемое как им Peel, сопровождается рядом пробелов. Следовательно, символ 'l' не будет рассматриваться концом строки. Групповой символ ' % ' - просто соответствует этим пробелам. Это необязательно, если пол sname имеет тип - VARCHAR.

1. Составить схему БД предметной области. Написать SQL-скрипт для создания таблиц. Определить первичные и внешние ключи, тип данных каждого атрибута, а также декларативные ограничения целостности - возможность принимать неопределенное значение, уникальные ключи, проверочные ограничения и т. д. (20 баллов)

Предметная область: Отдел кадров.

* Сущность «Сотрудник». Атрибуты сущности – номер, ФИО, номер паспорта, серия паспорта, пол, возраст.
* Сущности «Подразделение». Атрибуты сущности - номер, наименование, номер телефона, ФИО начальника, номер телефона начальника.
* Сущность «Сотрудники подразделения». Атрибуты сущности - номер сотрудника, номер подразделения, дата принятия на работу, ставка, оклад.

Create Database Bilet13

go

Use Bilet13

go

create table personal(

id int primary key,

phone int,

name nvarchar (max),

lastname nvarchar(max),

passportID int,

passportSeries int,

age int,

)

create table subdivision(

id int primary key,

number int,

phone int,

nomination nvarchar (max),

FIOchief nvarchar (max),

phonechief nvarchar (max),

)

create table departmentemployees(

id int primary key,

personalId int,

subdivisionID int,

dateofemployment nvarchar (max),

bid nvarchar (max),

salary money,

)

1. Составить представление для отображения среднего возраста сотрудников по подразделениям. Привести пример использования представления. (20 баллов)

**ЭКЗАМЕНАЦИОННЫЙ БИЛЕТ № 14**

1. Использование группировки и итоговых функций в запросах SQL. Привести примеры. (20 баллов)

Использование группировки и итоговых функций в запросах SQL

Агрегирующая функция может применяться ко всем записям БД слоя, к выборке по заданным условиям и, кроме того, возможно группирование записей слоя в несколько групп, и применение агрегирующей функции к каждой группе ([«Группировка записей»](https://www.politerm.com/zuludoc/sql_groupby.html)).

Применяемые агрегирующие функции записываются после ключевого слова SELECT. Также допускается использовать агрегирующие функции в составе выражений, включающих функции, арифметические и побитовые операции. В одном запросе может перечисляться несколько выражений с агрегирующими функциями. Не допускается в запросе одновременно с агрегирующими функциями запрашивать значения полей записей БД, либо использовать в аргументах неагрегирующих функций обращения к полям записей БД . Например, запрос вида SELECT SQRT(Area), SUM(Perimeter) FROM Здания не допускается, поскольку аргументом функции SQRT является название поля данных.

Общая запись агрегирующих функций:

<Функция>([DISTINCT]<выражение>)

В качестве аргумента агрегирующей функции обычно используется название поля, над значениями которого проводятся вычисления. Также допускается в качестве аргумента использовать выражения, включающие в себя произвольную комбинацию названий полей, констант, функций и подзапросов, объединенных арифметическими и побитовыми операциями.

Остальная часть запроса задается стандартным образом.

Перед аргументом функции (кроме функций MAX и MIN)может указываться ключевое слово DISTINCT. В этом случае итоговое значение вычисляется только для различающихся значений аргумента. При использовании ключевого слова DISTINCT в качестве аргумента агрегирующей функции нельзя использовать арифметические выражения, - только названия полей.

В языке SQL используются следующие агрегирующие функции:

SUM([DISTINCT] <выражение>)

Выводит в итоговой таблице сумму значений для выражения по полям выборки. Выражение должно возвращать числовое значение.

AVG([DISTINCT] <выражение>)

Среднее значение для выражения. Выражение должно возвращать числовое значение.

COUNT([DISTINCT] <выражение> |\*)

Подсчитывает число записей, в который выражение не имеет значение Null (поля имеют значение Null, когда никакое значение для них не задано). Выражение может возвращать произвольное значение.

При используемом формате функции COUNT(\*) возвращает общее количество записей в БД слоя.

MAX( <выражение>)

Возвращает максимальное значение выражения для выборки.

MIN(<выражение>)

Возвращает минимальное значение выражения из выборки.

1. Составить схему БД предметной области. Написать SQL-скрипт для создания таблиц. Определить первичные и внешние ключи, тип данных каждого атрибута, а также декларативные ограничения целостности - возможность принимать неопределенное значение, уникальные ключи, проверочные ограничения и т. д. (20 баллов)

Предметная область: Расписание занятий.

* Сущность «Аудитория». Атрибуты сущности – номер, корпус, этаж, номер кабинета, количество мест.
* Сущности «Группа». Атрибуты сущности - номер, наименование.
* Сущность «Расписание». Атрибуты сущности - номер аудитории, номер группы, дисциплина, день недели – пн., вт., ср., чт., пт., номер пары – 1, 2, 3, 4.

create database Bilet14

go

use Bilet14

go

create table audience(

id int primary key,

frame nvarchar (max),

floor int,

cabinetnumber int

)

create table circle(

id int primary key,

audienceID int,

nomination nvarchar(max),

)

create table usual(

id int primary key,

audienceID int,

circleID int,

dayweek nvarchar(max),

pairnumber int,

)

1. Составить хранимую процедуру для отображения расписания заданный группы (входной параметр) на заданный день недели (входной параметр). Вывод осуществить в следующем виде: номер пары, дисциплина, номер кабинета. Привести пример использования хранимой процедуры. (20 баллов)

**ЭКЗАМЕНАЦИОННЫЙ БИЛЕТ № 15**

1. Извлечение данных средствами T-SQL. Подзапросы. Виды подзапросов по типу возвращаемых данных. Соотнесенные подзапросы. Привести примеры. (20 баллов)

Извлечение данных средствами T-SQL. Подзапросы. Виды подзапросов по типу возвращаемых данных. Соотнесенные подзапросы

Подзапросы, внутренние или вложенные запросы – есть не что иное, как запрос внутри запроса. Обычно, подзапрос используется в конструкции WHERE. И, в большинстве случаев, подзапрос используется, когда вы можете получить значение с помощью запроса, но не знаете конкретного результата.

Подзапросы являются альтернативным путем получения данных из множества таблиц.

Наряду с операторами сравнения такими, как =, <, >, >=, <= и др., Вы можете использовать подзапросы с перечисленными ниже конструкциями:

* SELECT
* INSERT
* UPDATE
* DELETE

## **Примеры подзапросов**

1)Обычно, подзапрос возвращает только одну запись, но случается и так, что записей может быть много, тогда в условии WHERE используются такие операторы, как IN, NOT IN. Запрос может выглядеть следующий образом:

SELECT first\_name, last\_name, subject 

FROM student\_details 

WHERE games NOT IN ('Cricket', 'Football');

Тогда результат будет примерно таким:

first\_name last\_name games

------------ ----------- ----------

Shekar Gowda Badminton

Priya Chandra Chess

2)Давайте рассмотрим используемую ранее таблицу student\_details. Если Вы знаете имена студентов, изучающих естествознание (science), то можете получить их id, используя приведенный ниже запрос

SELECT id, first\_name 

FROM student\_details 

WHERE first\_name IN ('Rahul', 'Stephen');

но, если Вы не знаете их имен, то для получения id Вам необходимо написать запрос, описанным ниже способом:

SELECT id, first\_name 

FROM student\_details 

WHERE first\_name IN (SELECT first\_name 

FROM student\_details 

WHERE subject= 'Science');

Результат:

Id first\_name

-------- -------------

100 Rahul

102 Stephen

В описанном выше запросе, вначале выполняется внутренний запрос, затем внешний

3) Вы можете использовать подзапрос с оператором INSERT для добавления данных из одной таблицы в другую. Давайте попробуем сгруппировать всех студентов, которые изучают математику в таблицу math\_group

INSERT INTO maths\_group(id, name) 

SELECT id, first\_name || ' ' || last\_name 

FROM student\_details WHERE subject= 'Maths'

4) Подзапрос может использоваться с оператором SELECT, как описано ниже. Давайте используем таблицы product и order\_items, объединив их между собой

select p.product\_name,

p.supplier\_name,

(select order\_id from order\_items where product\_id = 101) as order\_id

from product p

where p.product\_id = 101

product\_name supplier\_name order\_id

-------------- -------------- ----------

Television Onida 5103

## **Соотнесенный подзапрос**

Запрос называется соотнесенным, когда оба, и внутренний, и внешний, запросы взаимозависимы. Это означает, что для обработки каждой записи внутреннего запроса, должна быть получена также запись внешнего запроса, т.е. внутренний запрос всецело зависит от внешнего.

1. Составить схему БД предметной области. Написать SQL-скрипт для создания таблиц. Определить первичные и внешние ключи, тип данных каждого атрибута, а также декларативные ограничения целостности - возможность принимать неопределенное значение, уникальные ключи, проверочные ограничения и т. д. (20 баллов)

Предметная область: Дневник.

* Сущности «Предмет». Атрибуты сущности - номер, наименование, ФИО учителя.
* Сущность «Ученик». Атрибуты сущности - номер, ФИО, пол, дата рождения, номер свидетельства о рождении.
* Сущность «Дневник». Атрибуты сущности - номер ученика, номер предмета, дата получения оценки, оценка.

create database Bilet15

go

use Bilet15

go

create table subject(

id int primary key,

name nvarchar(max),

FIOteacher nvarchar(max),

)

create table student(

id int primary key,

FIO nvarchar(max),

gender text,

birthday datetime2,

birtcertificatenumber nvarchar(max),

)

create table diary(

id int primary key,

FIOID int,

subjectID int,

dateofreceiptoftheassessment nvarchar (max),

estimates int,

)

1. Составить хранимую процедуру для отображения средних оценок ученика за заданный диапазон времени. Вывести с помощью хранимой процедуры четвертные оценки ученика Иванова по всем предметам за 1-ю четверть (с 1.09.2018 по 04.10.2018. (20 баллов)

**ЭКЗАМЕНАЦИОННЫЙ БИЛЕТ № 16**

1. Извлечение данных средствами T-SQL. Подзапросы. Команды ANY, EXISTS, ALL. Привести примеры. (20 баллов)

Извлечение данных средствами T-SQL. Подзапросы. Команды ANY, EXISTS, ALL.

операторы SOME и ANY - взаимозаменяемы везде и там где мы используем ANY, SOME будет работать точно так же. Различие в терминологии состоит в том чтобы позволить людям использовать тот термин который наиболее однозначен. Это может создать проблему; потому что, как мы это увидим, наша интуиция может иногда вводить в заблуждение. Имеется новый способ нахождения продавца с заказчиками размещенными в их городах ( вывод для этого запроса показывается в Рисунке 13.1 ):

SELECT \*

FROM Salespeople

WHERE city = ANY

(SELECT city

FROM Customers );

Оператор ANY берет все значения выведенные подзапросом, ( для этого случая - это все значения city в таблице Заказчиков ), и оценивает их как верные если любой(ANY) из их равняется значению города текущей строки внешнего запроса.

Это означает, что подзапрос должен выбирать значения такого же типа как и те, которые сравниваются в основном предикате. В этом его отличие от EXISTS, который просто определяет, производит ли под- запрос результаты или нет, и фактически не использует эти результаты.

ИСПОЛЬЗОВАНИЕ ОПЕРАТОРОВ IN ИЛИ EXISTS ВМЕСТО ОПЕРАТОРА ANY

Мы можем также использовать оператор IN чтобы создать запрос аналогичный предыдущему :

SELECT \*

FROM Salespeople

WHERE city IN

( SELECT city

FROM Customers );

Этот запрос будет производить вывод показанный в Рисунке 13.2.  
  
Однако, оператор ANY может использовать другие реляционные опера- торы кроме равняется ( = ), и таким образом делать сравнения которые являются выше возможностей IN. Например, мы могли бы найти всех продавцов с их заказчиками которые следуют им в алфавитном порядке

Все строки были выбраны для Serres и Rifkin, потому что нет других за- казчиков чьи имена следовали бы за ими в алфавитном порядке. Обратите внимание что это является d основным эквивалентом следую- щему запросу с EXISTS.

Любой запрос который может быть сформулирован с ANY ( или, как мы увидим, с ALL ), мог быть также сформулирован с EXISTS, хотя наоборот будет неверно. Строго говор, вариант с EXISTS не абсолютно идентичен вариантам с ANY или с ALL из-за различи в том как обрабатываются пус- тые( NULL ) значения ( что будет обсуждаться позже в этой главе ). Тем ни менее, с технической точки зрения, вы могли бы делать это без ANY и ALL если бы вы стали очень находчивы в использовании EXISTS ( и IS NULL ).  
  
Большинство пользователей, однако, находят ANY и ALL более удобными в использовании чем EXISTS, который требует соотнесенных подзапросов. Кроме того, в зависимости от реализации, ANY и ALL могут, по крайней мере в теории, быть более эффективными чем EXISTS. Подзапросы ANY или ALL могут выполняться один раз и иметь вывод используемый чтобы определять предикат для каждой строки основного зап- роса. EXISTS, с другой стороны, берет соотнесенный подзапрос, который требует чтобы весь подзапрос повторно выполнялся для каждой строки основного запроса. SQL пытается найти наиболее эффективный способ выполнения любой команды, и может попробовать преобразовать менее эффективную формулу запроса в более эффективную (но вы не можете всегда рассчитывать на получение самой эффективной формулировки ).  
  
Основная причина для формулировки EXISTS как альтернативы ANY и ALL в том что ANY и ALL могут быть несколько неоднозначен, из-за способа использования этого термина в Английском языке, как вы это скоро увидите. С приходом понимания различия способов формулирования данного запроса, вы сможете поработать над процедурами которые сейчас кажутся Вам трудными или неудобными.

КАК ANY МОЖЕТ СТАТЬ НЕОДНОЗНАЧНЫМ

Как подразумевалось выше, ANY не полностью однозначен. Если мы создаем запрос чтобы выбрать заказчиков которые имеют больший рейтинг чем любой заказчик в Риме, мы можем получить вывод который несколько отличался бы от того что мы ожидали ( как показано в Рисунке 13.5 ):

SELECT \*

FROM Customers

WHERE rating > ANY

( SELECT rating

FROM Customers

WHERE city = Rome );

В Английском языке, способ которым мы обычно склонны интерпретировать оценку " больше чем любой ( где city = Rome ) " , должен вам сообщить что это значение оценки должно быть выше чем значение оценки в каждом случае где значение city = Rome. Однако это не так, в случае ANY - используемом в SQL . ANY оценивает как верно, если подзапрос находит любое значение которое делает условие верным.

1. Составить схему БД предметной области. Написать SQL-скрипт для создания таблиц. Определить первичные и внешние ключи, тип данных каждого атрибута, а также декларативные ограничения целостности - возможность принимать неопределенное значение, уникальные ключи, проверочные ограничения и т. д. (20 баллов)

Предметная область: Зачетка.

* Сущность «Студент». Атрибуты сущности – номер, ФИО, пол, дата рождения, группа, факультет.
* Сущность «Экзамен». Атрибуты сущности - номер ученика, наименование дисциплины, ФИО преподавателя, дата экзамена, оценка.
* Сущность «Зачет». Атрибуты сущности - номер ученика, наименование дисциплины, ФИО преподавателя, дата зачета, статус - сдан/не сдан.

create database Bilet16

go

use Bilet16

go

create table student(

id int primary key,

FIO nvarchar(max),

gender text,

birthday datetime2,

class nvarchar(max),

faculty nvarchar(max),

)

create table exam(

id int primary key,

studentID int,

nameofthediscipline nvarchar(max),

dateexam datetime2,

grade nvarchar(max),

)

create table offset(

id int primary key,

studentID int,

nameofthediscipline nvarchar(max),

FIOteacher nvarchar(max),

Offsetdate datetime2,

status nvarchar(max),

)

1. Составить представление для отображения в едином списке всех зачетов и экзаменов. Вывод осуществить в следующем виде: ФИО ученика, наименование дисциплины, дата контроля, тип контроля – зачет/экзамен, результат, ФИО преподавателя. Используя представление вывести результаты экзаменов и зачетов студента Иванова за зимнюю сессию 2018-2018 года. (20 баллов)

**ЭКЗАМЕНАЦИОННЫЙ БИЛЕТ № 17**

1. Многотабличные запросы. Назначение и способы объединения таблиц. (20 баллов)

Многотабличные запросы. Назначение и способы объединения таблиц.

JOIN, в переводе на великий и могучий, означает "объединять", то есть собирать из нескольких кусочков единое целое. В базе данных MySQL такими "кусочками" служат столбцы таблиц, которые можно объединять при выборке.

Объединения позволяют извлекать данные из нескольких таблиц без создания временных таблиц и за один запрос.

Таблицы определены и заполнены, настало время делать выборку. Но если приглядеться к данным, которые находятся в таблицах, то можно заметить, что они не в полной мере соответствуют друг другу. Так в таблице nomenclature присутствует товар под номером 2 (Табуретка), для которого нет описания. И в таблице description присутствует описание для товара номер 5 (Зелёная машинка), которого нет в таблице номенклатур .

В зависимости от требований к результату, MySQL позволяет производить три разных типа объединения:

1. INNER JOIN (CROSS JOIN) - внутреннее (перекрёстное) объединение
2. LEFT JOIN - левостороннее внешнее объединение
3. RIGHT JOIN - правостороннее внешнее объединение

## **INNER JOIN (CROSS JOIN) - внутреннее (перекрёстное) объединение**

Этот тип объединения позволяет извлекать строки, которые **обязательно** присутствуют во всех объединяемых таблицах.

LEFT JOIN - Левостороннее внешнее объединение

Левосторонние объединения позволяют извлекать данные из таблицы, дополняя их по возможности данными из другой таблицы.

## **RIGHT JOIN - Правостороннее внешнее объединение**

Этот вид объединений практически ничем не отличается от левостороннего объединения, за тем исключением, что данные берутся из второй таблицы, которая находится **справа** от конструкции JOIN, и сравниваются с данными, которые находятся в таблице, указанной перед конструкцией.

1. Составить схему БД предметной области. Написать SQL-скрипт для создания таблиц. Определить первичные и внешние ключи, тип данных каждого атрибута, а также декларативные ограничения целостности - возможность принимать неопределенное значение, уникальные ключи, проверочные ограничения и т. д. (20 баллов)

Предметная область: Аптека.

* Сущности «Поставщик». Атрибуты сущности - номер, наименование, город, адрес, номер телефона.
* Сущность «Лекарство». Атрибуты сущности – номер, наименование на русском, наименование на латинском, форма выпуска – таблетки/сироп/инъекции, стоимость 1 шт.
* Сущность «Заказ». Атрибуты сущности - номер лекарства, номер поставщика, дата заказа, количество единиц лекарства.

create database Bilet17

go

use Bilet17

go

create table producer(

id int primary key,

nomination nvarchar(max),

city nvarchar(max),

address nvarchar(max),

phone int

)

create table medicine(

id int primary key,

nameinRussian nvarchar(max),

nameinlatin nvarchar(max),

releaseform nvarchar(max),

price money,

)

create table order(

id int primary key,

medicineID int,

producerID int,

orderdate datetime2,

numberoforderunits nvarchar(max),

)

1. Составить хранимую процедуру для поиска заказов лекарств по определенному имени поставщика (входной параметр) и наименованию лекарства (входной параметр). Вывод осуществить в следующем виде: дата заказа, наименование поставщика, наименование лекарства, стоимость 1 шт. лекарства, количество едини лекарства, стоимость всех единиц лекарства (стоимость 1 шт. лекарства \* количество едини лекарства). Привести пример использования хранимой процедуры. (20 баллов)

**ЭКЗАМЕНАЦИОННЫЙ БИЛЕТ № 18**

1. Синтаксис запросов на удаление, вставку и обновление данных в SQL. (20 баллов)

команда**insert**(вставить) позволяет добавлять новые строки в базу данных. Команда **update**(обновить) позволяет изменить существующие в базе данные. Команда **delete**(удалить) удаляет данные из базы. А команда **writetext**(запись текста) позволяет добавлять или изменять тектовые (*text)* и графические (*image*)данные без записи больших массивов в системный журнал транзакций.

Все эти операторы вместе называются **операторами модификации данных**. В этой главе также рассматривается команда **truncate table**(очистить таблицу), удаляющая все строки в таблице. Другим способом добавления данных в таблицу является их непосредственная перепись из файла с помощью программы-утилиты **bcp**. Детальная информация об этих возможностях находится в **Справочном Руководстве SQL сервера** и в **Пособии по утилитам SQL Сервера** для операционной системы пользователя.

Пользователь может модифицировать данные с помощью одного из операторов **insert**, **update** и **delete** только в одной таблице. Однако, исходные для модификации данные могут выбираться из других таблиц, и даже из других баз данных. Эта возможность является расширением языка Transact-SQL по отношению к стандартной версии языка SQL.

1. Составить схему БД предметной области. Написать SQL-скрипт для создания таблиц. Определить первичные и внешние ключи, тип данных каждого атрибута, а также декларативные ограничения целостности - возможность принимать неопределенное значение, уникальные ключи, проверочные ограничения и т. д. (20 баллов)

Предметная область: Спортивная секция.

* Сущности «Учащийся». Атрибуты сущности - номер, ФИО, дата рождения, пол, медицинская группа – основная, подготовительная, специальная.
* Сущность «Норматив». Атрибуты сущности - номер, наименование.
* Сущность «Зачет». Атрибуты сущности - номер учащегося, номер норматива, дата сдачи норматива, результат – сдан/ не сдан.

create database Bilet18

go

use Bilet18

go

create table student(

id int primary key,

FIO nvarchar(max),

Birthday datetime2,

Gender text,

medicalgroup nvarchar(max),

)

create table standard(

id int primary key,

Name nvarchar(max),

)

create table offset(

id int primary key,

studentID int,

standardID int,

dateofpassingthestandard datetime2,

result nvarchar(max),

)

1. Составить представление для отображения списка зачетов. Вывод осуществить в следующем виде: ФИО учащегося, Пол, Возраст, Медицинская группа, Наименование норматива, Результат. Используя вывести количество учащихся, сдавших норматив и количество учащихся, не сдавших норматив (20 баллов)

**ЭКЗАМЕНАЦИОННЫЙ БИЛЕТ № 19**

1. Назначение представлений. Создание, изменение и удаление представлений средствами T-SQL. (20 баллов)

Представления или Views представляют виртуальные таблицы. Но в отличии от обычных стандартных таблиц в базе данных представления содержат запросы, которые динамически извлекают используемые данные.

Представления дают нам ряд преимуществ. Они упрощают комплексные SQL-операции. Они защищают данные, так как представления могут дать доступ к части таблицы, а не ко всей таблице. Представления также позволяют возвращать отформатированные значения из таблиц в нужной и удобной форме.

1. Составить схему БД предметной области. Написать SQL-скрипт для создания таблиц. Определить первичные и внешние ключи, тип данных каждого атрибута, а также декларативные ограничения целостности - возможность принимать неопределенное значение, уникальные ключи, проверочные ограничения и т. д. (20 баллов)

Предметная область: Телефонная книга.

* Сущности «Абонент». Атрибуты сущности - номер, ФИО, дата рождения, адрес, email.
* Сущность «Телефон». Атрибуты сущности – номер - идентификатор, номер абонента, номер телефона абонента, тип номера – домашний/мобильный/факс.
* Сущность «Звонок». Атрибуты сущности - номер абонента, номер телефона - идентификатор, дата звонка, продолжительность, тип звонка – входящий/исходящий/отклонен.

create database Bilet19

go

use Bilet19

go

create table subscriber(

id int primary key,

FIO nvarchar(max),

Birthday datetime2,

address nvarchar(max),

email nvarchar(max),

)

create table phone(

id int primary key,

subscriberID int,

phonesubscriber nvarchar(max),

typeofnumber nvarchar(max),

)

create table call(

id int primary key,

subscriberID int,

phoneID int,

identifier nvarchar(max),

calldate datetime2,

duration nvarchar(max),

calltype nvarchar(max),

)

1. Составить представление для отображения списка звонков. Вывод осуществить в следующем виде: Дата звонка, ФИО абонента, Вид номера, Номер телефона абонента, Тип звонка, Продолжительность. Используя представление найти 3 самых продолжительных звонка. (20 баллов)

**ЭКЗАМЕНАЦИОННЫЙ БИЛЕТ № 20**

1. Назначение представлений. Создание, изменение и удаление представлений средствами T-SQL. (20 баллов)
2. Составить схему БД предметной области. Написать SQL-скрипт для создания таблиц. Определить первичные и внешние ключи, тип данных каждого атрибута, а также декларативные ограничения целостности - возможность принимать неопределенное значение, уникальные ключи, проверочные ограничения и т. д. (20 баллов)

Предметная область: Компьютерный магазин.

* Сущность «Консультант». Атрибуты сущности - номер, ФИО, пол, номер паспорта, серия паспорта, процент комиссионных.
* Сущность «Товар». Атрибуты сущности - номер, категория, марка, модель, цена за единицу, количество на складе.
* Сущность «Продажа». Атрибуты сущности - номер, дата продажи, номер товара, номер консультанта, количество купленных товаров.

create database Bilet20

go

use Bilet20

go

create table consultant(

FIO nvarchar(max),

gender text,

passportID int,

passportseries int,

commissionpercentage nvarchar(max),

)

create table product(

id int primary key,

category nvarchar(max),

brand nvarchar(max),

model nvarchar(max),

price money,

number int

)

create table sale(

id int primary key,

datesale nvarchar(max),

productID int,

consultantID int,

numberofgoodspurchased nvarchar(max),

)

1. Составить хранимую процедуру для поиска продаж товаров заданного консультанта (входной параметр) за заданный месяц (входной параметр). Вывод осуществить в следующем виде: Дата и время продажи, Наименование товара, Цена товара за единицу, Количество купленных товаров, Стоимость купленных товаров, Размер комиссионных консультанта, Размер полученных комиссионных консультанта с продажи товара. Привести пример использования хранимой процедуры. (20 баллов)