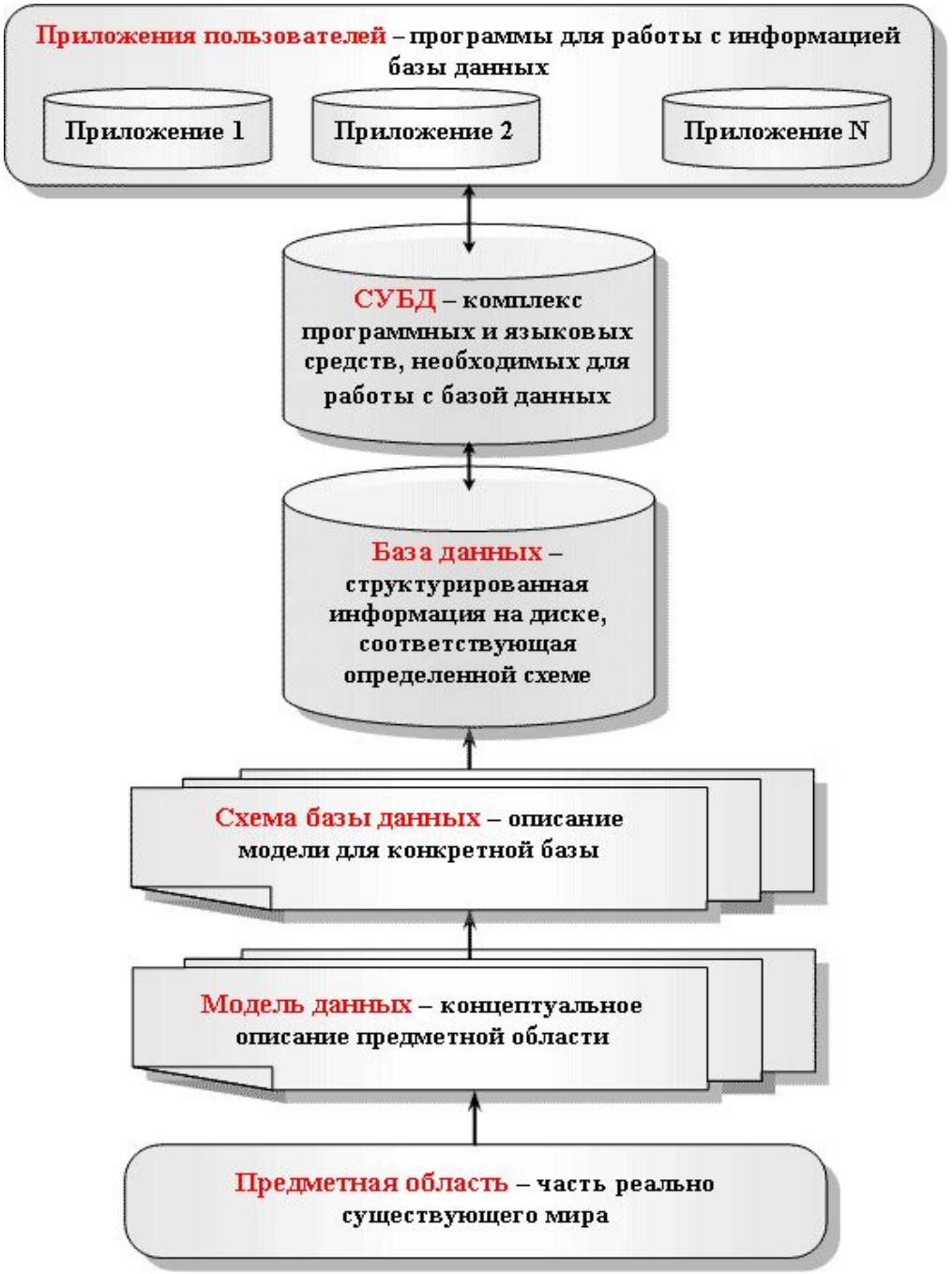
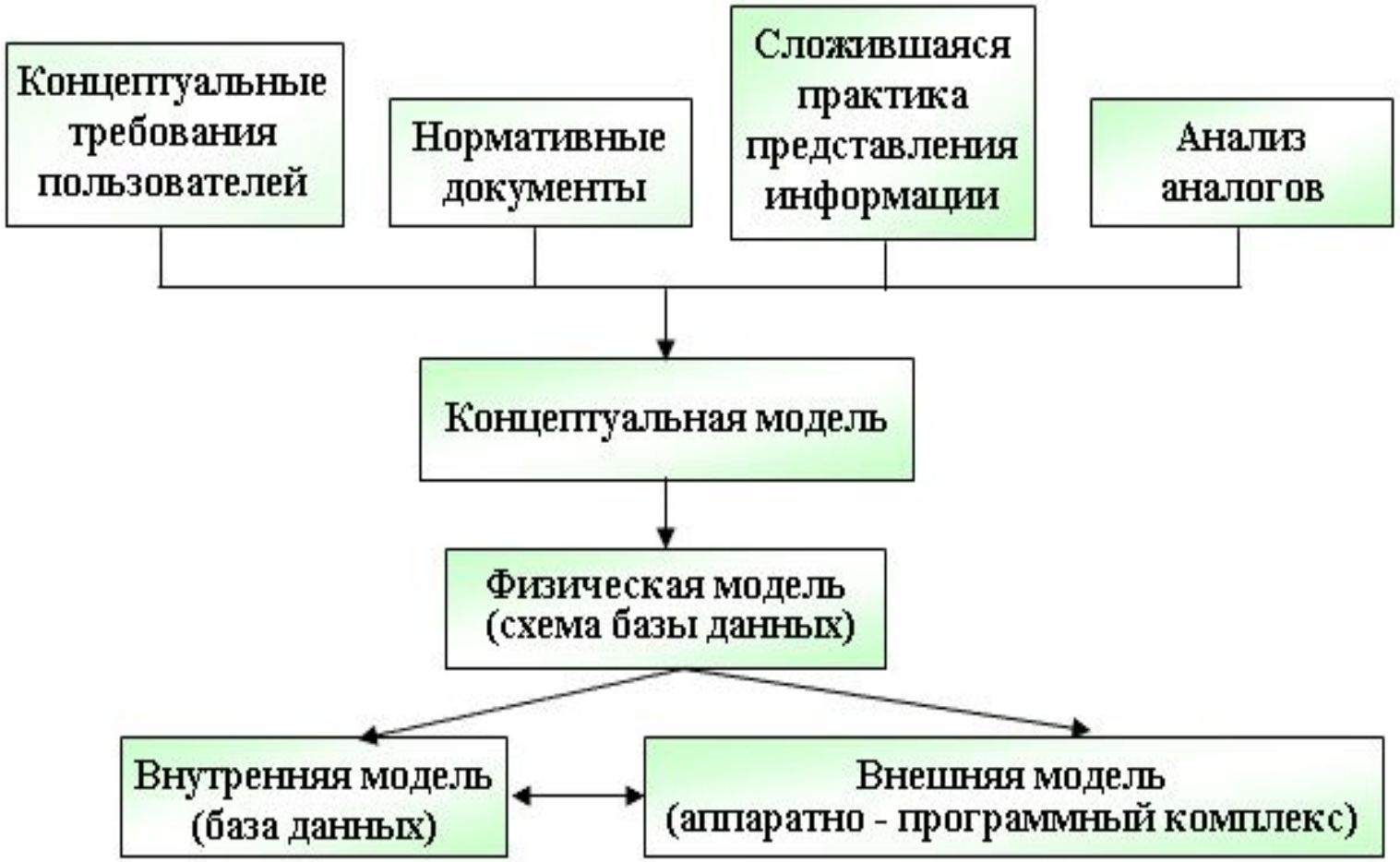
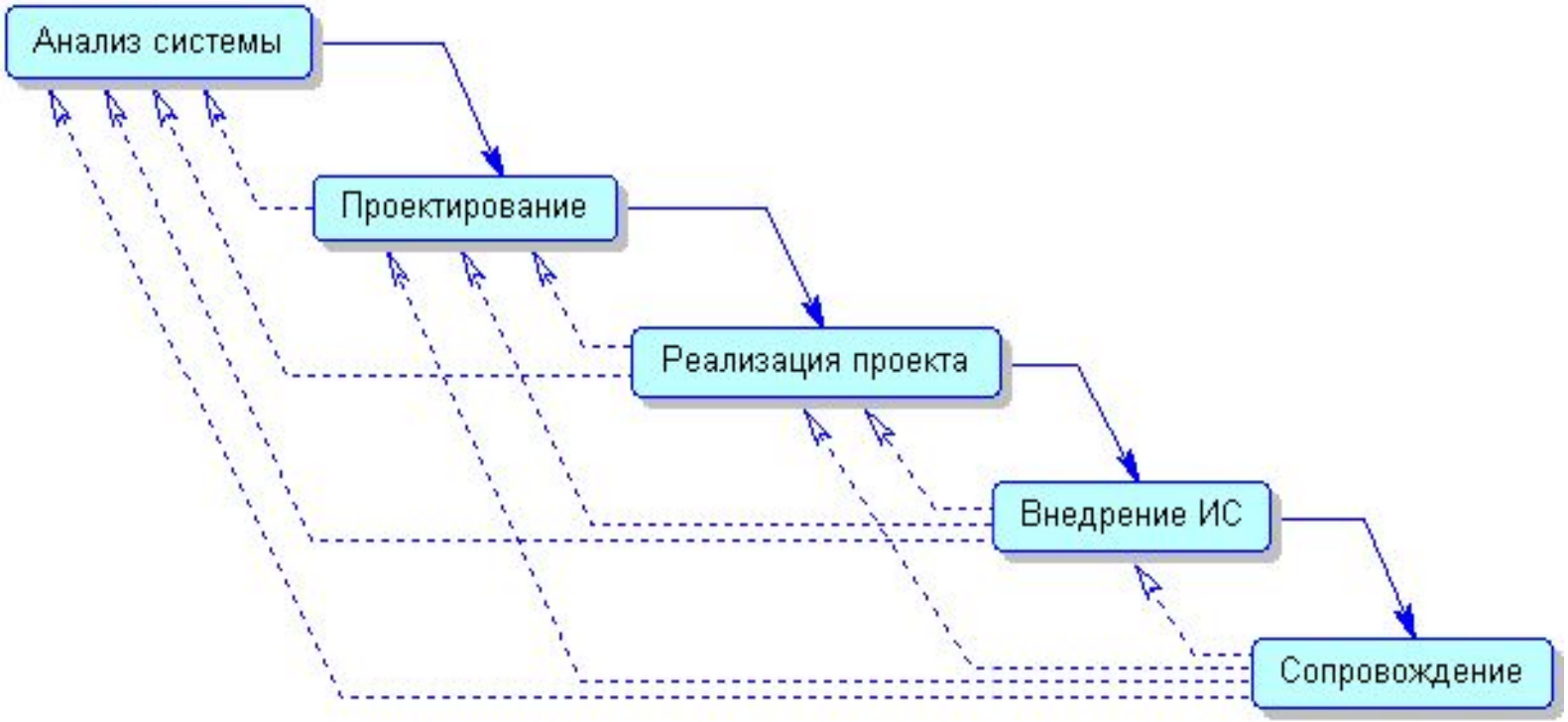

Приложения пользователей – программы для работы с информацией базы данных
Приложение 1
Приложение 2
Приложение N
СУБД – комплекс программных и языковых средств, необходимых для работы с базой данных
База данных – структурированная информация на диске, соответствующая определенной схеме
Схема базы данных – описание модели для конкретной базы
Модель данных – концептуальное описание предметной области
Предметная область – часть реально существующего мира


Концептуальные требования пользователей
Нормативные документы
Сложившаяся практика представления информации
Анализ аналогов
Концептуальная модель
Физическая модель (схема базы данных)
Внутренняя модель (база данных)
Внешняя модель (аппаратно - программный комплекс)

- **Концептуальная модель** - отображает информационные объекты, их свойства и связи между ними без указания способов физического хранения информации (модель предметной области, иногда ее также называют информационно-логической или инфологической моделью).
- Информационными объектами обычно являются **сущности** - обособленные объекты или события, информацию о которых необходимо сохранять, имеющие определенные наборы свойств - **атрибутов**.
- **Физическая модель** - отражает все свойства (атрибуты) информационных объектов базы и связи между ними с учетом способа их хранения - используемой СУБД.

- **Внутренняя модель** - база данных, соответствующая определенной физической модели.
- **Внешняя модель** - комплекс программных и аппаратных средств для работы с базой данных, обеспечивающий процессы создания, хранения, редактирования, удаления и поиска информации, а также решающий задачи выполнения необходимых расчетов и создания выходных печатных форм.



Каскадная схема жизненного цикла ИС

- Жизненный цикл разработки сложной системы в этом случае складывается из этапов анализа, проектирования, программирования и тестирования, внедрения и сопровождения, которые выполняются последовательно.

Лабораторная работа №1

- **1. Описание предметной области**

В вузе имеется несколько факультетов, на которых обучаются студенты. Студенты сдают экзамены по разным предметам и получают соответствующие оценки. Известен перечень экзаменов и возможные значения оценок.

Сущности: СТУДЕНТ, ЭКЗАМЕН, ОЦЕНКА

- СТУДЕНТ (код студента, фамилия)
- ЭКЗАМЕН (код экзамена, предмет, дата экзамена)
- ОЦЕНКА (значение оценки)

Между этими сущностями существуют следующие связи:

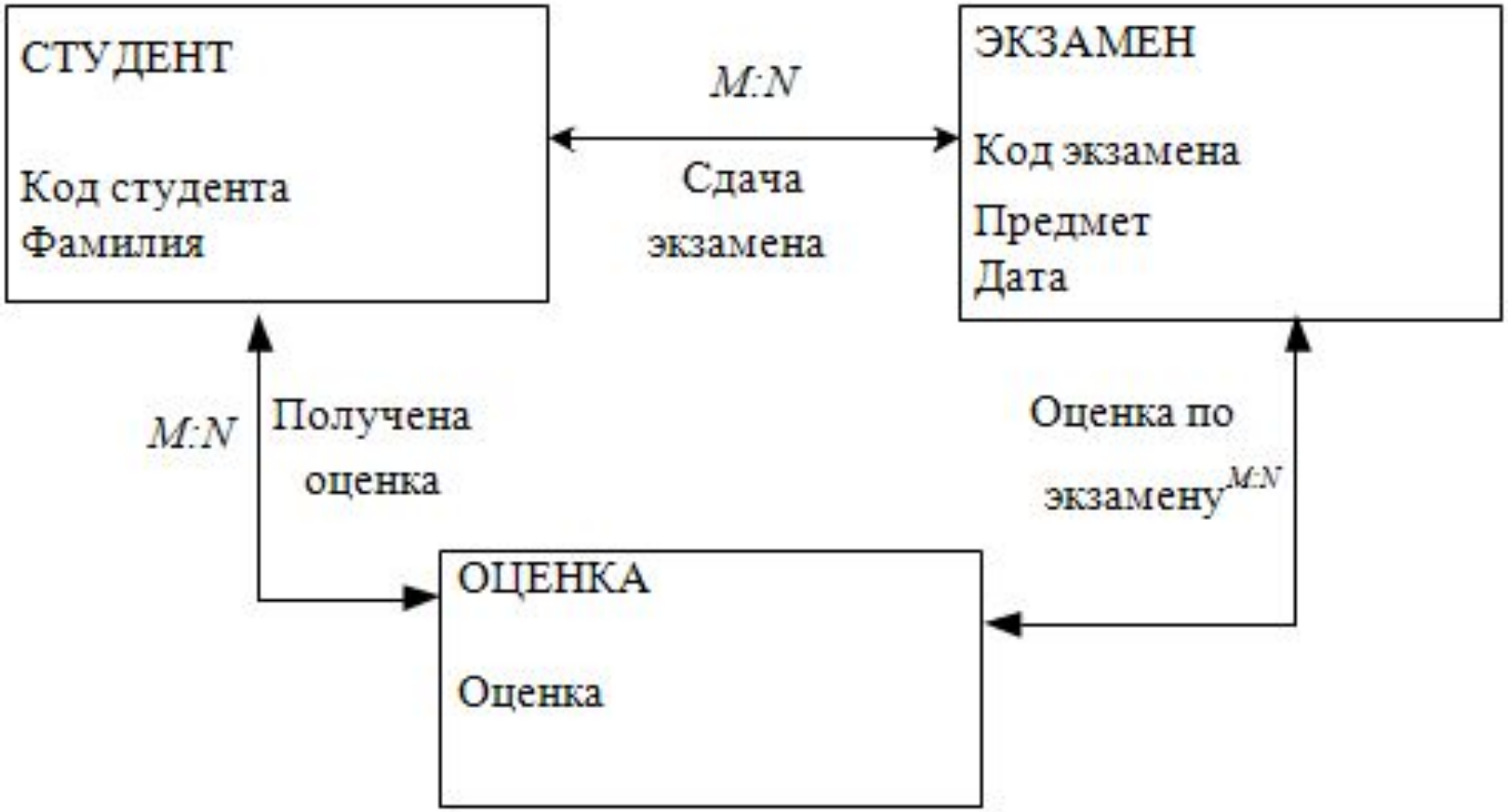
- *студент сдавал экзамен*,
- *студент получил оценку*,
- *по экзамену получены следующие оценки*.

- **2. Сформировать максимально полный перечень возможных запросов к базе данных на основе анализа предметной области.**

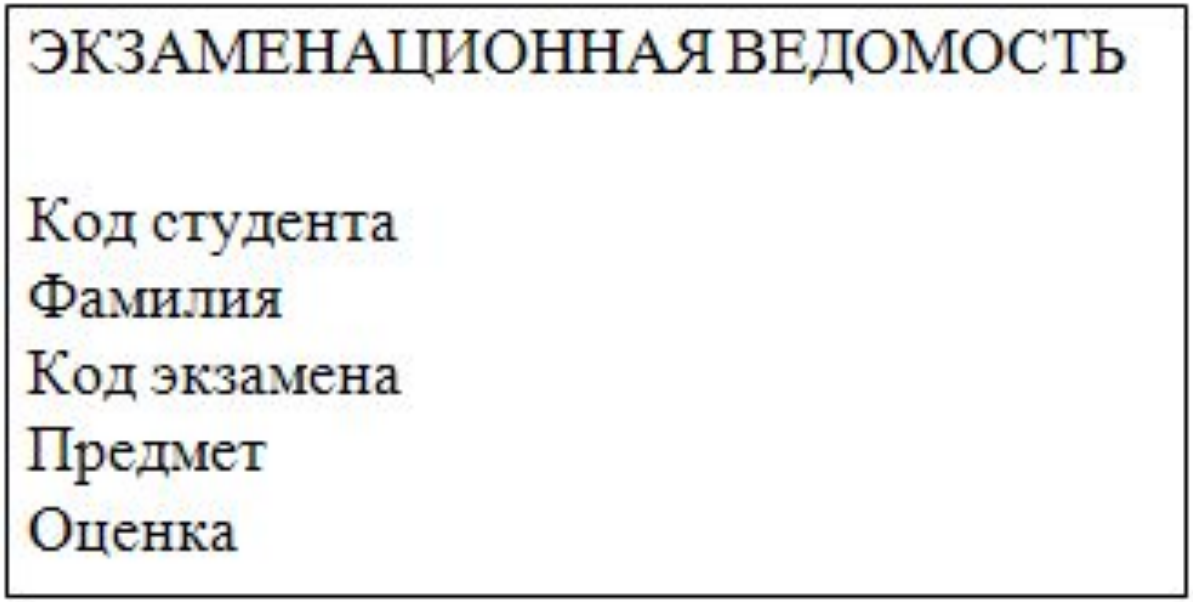
По смыслу задачи к базе данных возможны следующие запросы:

- *Какие оценки получил студент с заданной фамилией (кодом);*
- *Какие студенты получили заданное значение оценки;*
- *Какие экзамены сдал студент с заданной фамилией (кодом);*
- *Какую оценку по конкретному предмету получил студент с заданной фамилией (кодом).*

- **3. Построить концептуальную модель в виде ER-диаграммы.**



- Для реализации последнего запроса предлагается ввести новую агрегированную сущность: ЭКЗАМЕНАЦИОННАЯ ВЕДОМОСТЬ (код студента, фамилия, код экзамена, предмет, дата экзамена, оценка).
- Второй вариант ER-диаграммы:

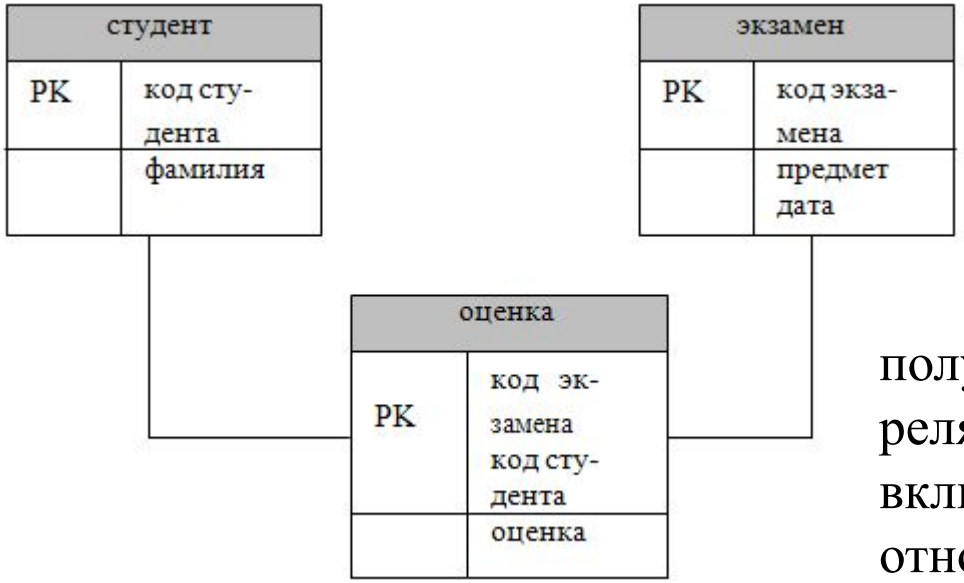


- **4. Представить концептуальную модель в терминах реляционной модели.**
- В терминах концептуальной модели эта модель представляется следующей таблицей.

Код студента	Фамилия	Код экзамена	Предмет	Дата	Оценка

- **5. Описать домены** (допустимые множества значений, которые могут принимать атрибуты), указывая типы соответствующих данных и их характеристики.
- Код студента принимает значения из множества целых чисел, максимальная длина числа 4 знака.
- Код экзамена – из множества целых чисел, максимальная длина числа 4 знака.
- Предмет принимает символьное значение, максимальная длина 20 символов.
- Дата экзамена принимает значение дата в формате 00.00.00.
- Оценка принимает целое значение от 2 до 5.

- **6. Определить ключи и внешние ключи (если они есть).**
- Ключом данного отношения является совокупность атрибутов код студента, код экзамена
- 7. Исходное отношение приведено к трем отношениям: *R*1(**КС**, Ф), *R2*(**КЭ**, П, Д), *R3*(**КС**, **КЭ**, О).



полученная реляционная модель включает три отношения.

SQL

Structured Query Language

- **Репликация базы данных** - создание копий базы данных (реплик), которые могут обмениваться обновляемыми данными или реплицированными формами, отчетами или другими объектами в результате выполнения процесса синхронизации.
- **Транзакция** - изменение информации в базе в результате выполнения одной операции или их последовательности, которое должно быть выполнено полностью или не выполнено вообще. В СУБД существуют специальные механизмы обеспечения транзакций.
- **Язык SQL** (Structured Query Language) - универсальный язык работы с базами данных, включающий возможности ее создания, модификации структуры, отбора данных по запросам, модификации информации в базе и прочие операции манипулирования базой данных.

- Операторы языка SQL можно условно разделить на две категории:
- **язык определения данных** (Data Definition Language — **DDL**)
- **язык манипулирования данными** (Data Manipulation Language — **DML**).

Вид	Название	Назначение
DML	**SELECT**	выборка записей
	UPDATE	изменение записей
	INSERT	вставка новых записей
	DELETE	удаление записей
DDL	**CREATE TABLE**	создание таблицы
	DROP TABLE	удаление таблицы
	ALTER TABLE	изменение структуры таблицы
	CREATE VIEW	создание представления
	DROP VIEW	удаление представления
	GRANT	назначение привилегий
	REVOKE	удаление привилегий

Типы данных SQL

Тип данных	Описание
CHAR(*длина*) CHARACTER(*длина*)	Строки символов постоянной длины
VARCHAR(*длина*) CHAR VARYING(*длина*) CHARACTER VARYING(*длина*)	Строки символов переменной длины
INTEGER INT	Целые числа
SMALLINT	Малые целые числа
NUMERIC(*точность, степень*) DECIMAL(*точность, степень*) DEC(*точность, степень*)	Десятичные числа
FLOAT(*точность*)	Числа с плавающей запятой
REAL	Числа с плавающей запятой низкой точности
DOUBLE PRECISION	Числа с плавающей запятой высокой точности
DATE	Дата
TIME(*точность*)	Время
TIMESTAMP(*точность*)	Дата и время
INTERVAL	Временной интервал

Символ	Обозначение
::=	Равно по определению
|	Необходимость выбора одного из нескольких приведенных значений
<...>	Описанная с помощью *метаязыка* структура языка
{...}	Обязательный выбор некоторой конструкции из списка
[...]	Необязательный выбор некоторой конструкции из списка
[,...n]	Необязательная возможность повторения конструкции от нуля до нескольких раз

Команда SELECT

SELECT
[ALL | DISTINCT | DISTINCTROW | TOP] {* | *table*.* | [*table.*] *field1* [AS *alias1*] [, *table.*] *field2* [AS *alias2*] [, ...]] }
FROM имя_таблицы [псевдоним] [, ...]
[WHERE
условие_по_исходным_данным]
[GROUP BY столбец [, ...]]
[HAVING условие_по_группе]
[ORDER BY столбец [DESC] [, ...]]

Команда	Назначение SQL-выражения
SELECT	Команда/ключевое слово, с которого начинается SQL-выражение; предшествует названию поля (или полей), выбираемого из таблицы
FROM	Указывает имя таблицы или таблиц, содержащих поля, перечисленные после команды SELECT
WHERE	Команда, указывающая условие отбора или ограничение для выводимых записей; используется только в том случае, если необходимо ограничить группу отбираемых записей на основании какого-то условия
ORDER BY	Команда, указывающая порядок вывода данных

- **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;

- Порядок предложений и фраз в операторе SELECT не может быть изменен. Только два предложения SELECT и FROM являются обязательными, все остальные могут быть опущены.

Предикаты SELECT

- ALL
- DISTINCT
- DISTINCTROW
- TOP
- Предикат ALL назначен по умолчанию. Он выбирает все записи, которые в выражении SQL удовлетворяет условию WHERE.
- Предикат DISTINCT необходимо включать, когда из запроса следует исключить одинаковые записи (рассматриваются только поля, включенные в запрос). *Например, при создании запроса, выводящего идентификатор покупателя и день, в который он сделал заказ, нужно использовать следующее*

```
SELECT DISTINCT [CustomerlD], [OrderDate]
```

- DISTINCTROW— это предикат, существующий только в Access. Он работает подобно предикату DISTINCT, но с одним большим отличием: DISTINCTROW проверят совпадение в таблице или таблицах всех полей, а не только выбранных.
- Предикат DISTINCTROW используется для исключения записей, повторяющихся полностью. Он влияет на результат только в том случае, если в запрос включены не все поля из анализируемых таблиц.
- Предикат DISTINCTROW игнорируется, если запрос содержит только одну таблицу.

- Предикат TOP, который также характерен только для Access, ограничивает число выводимых записей, удовлетворяющих условию WHERE. Предикат TOP предназначен для возврата определенного числа записей, находящихся в начале или в конце диапазона, описанного с помощью предложения ORDER BY.

Например, TOP 10 выводит только десять первых записей, удовлетворяющих условию WHERE.

- Предикат TOP имеет один необязательный параметр PERCENT (процент), который указывает не количество первых записей, а их процентное отношение к общему числу отобранных записей.

- Список {* | *table*.* | [*table*.] *field1* [AS *alias1*] [, *table*.] *field2* [AS *alias2*] [, ...]] } (фигурные скобки здесь обозначают список) состоит из имен полей таблиц(ы) запроса.
- Звездочка (*) означает выбор всех полей таблицы. Если в запросе указывается несколько таблиц, то для определения поля используется наименование таблицы, отделяемое от имени поля точкой (.).
- Поле может получать псевдоним при помощи ключевого слова AS.

Выполнение оператора SELECT

1. Выполняется расширенное декартово произведение исходных таблиц;
2. Отбираются те строки получившейся таблицы, которые удовлетворяют (не противоречат) условию по исходным данным;
3. В результирующей таблице добавляются столбцы для всех вычислимых полей:

а) если в выражениях списка выборки использована итоговая функция, то все строки сначала группируются по указанным в секции GROUP BY столбцам (если секция отсутствует, то все строки попадают в одну группу), и для каждой группы выполняется расчет и заполнение вычислимых полей (для расчета используются данные столбца текущей группы).

- После выполнения расчета из множества строк в каждой группе результирующей таблицы оставляется только одна.

- Далее удаляются те оставшиеся строки – представители групп, которые не удовлетворяют условию по группе из раздела HAVING.

б) если в выражениях списка выборки не используется итоговая функция, то для каждой строки выполняется расчет и заполнение вычислимых полей (при расчете используются данные только текущей строки).

4. Если в списке выборки есть подзапросы, то они вычисляются, и результирующая таблица будет представлять собой расширенное декартово произведение таблицы, созданной на предыдущих шагах и таблиц, полученных вычислением подзапросов.

5. Выполняется проекция полученной таблицы на указанные в списке выборки столбцы результата. В случае, если указано ключевое слово DISTINCT, из всех одинаковых строк оставляется один представитель.

6. Выполняется упорядочивание строк результирующей таблицы по значениям указанных в секции ORDER BY столбцов

Пример 1. Составить список сведений о всех клиентах.

```
SELECT * FROM Клиент
```

Пример 2. Составить список всех фирм.

```
SELECT ALL Клиент.Фирма
    FROM Клиент
```

Или (что эквивалентно)

```
SELECT Клиент.Фирма
    FROM Клиент
```

Когда требуется отбросить блоки данных, содержащие дублирующие записи в выбранных полях:

```
SELECT DISTINCT Клиент.Фирма
    FROM Клиент
```

Сравнение

Пример 3. *Показать все операции отпуска товаров объемом больше 20.*

```
SELECT * FROM Сделка
    WHERE Количество>20
```

Пример 4. *Вывести список товаров, цена которых больше или равна 100 и меньше или равна 150.*

```
SELECT Название, Цена
    FROM Товар
    WHERE Цена>=100 And Цена<=150
```

Пример 5. *Вывести список клиентов из Москвы или из Самары.*

```
SELECT Фамилия, ГородКлиента
    FROM Клиент
    WHERE ГородКлиента="Москва" Or
  ГородКлиента="Самара"
```

Диапазон

Пример 6. *Вывести список товаров, цена которых лежит в диапазоне от 100 до 150 (запрос эквивалентен примеру 4).*

```
SELECT Название, Цена
    FROM Товар
    WHERE Цена Between 100 And 150
```

Пример 7. *Вывести список товаров, цена которых не лежит в диапазоне от 100 до 150.*

```
SELECT Товар.Название, Товар.Цена
    FROM Товар
    WHERE Товар.Цена Not Between 100 And 150
```

Или (что эквивалентно)

```
SELECT Товар.Название, Товар.Цена
    FROM Товар
    WHERE (Товар.Цена<100) OR (Товар.Цена>150)
```

Принадлежность множеству

- Оператор IN используется для сравнения некоторого значения со списком заданных значений, при этом проверяется, соответствует ли результат вычисления выражения одному из значений в предоставленном списке. При помощи оператора IN может быть достигнут тот же результат, что и в случае применения оператора OR, однако оператор IN выполняется быстрее.

Пример 8. *Вывести список клиентов из Москвы или из Самары (запрос эквивалентен примеру 5).*

```
SELECT Фамилия, ГородКлиента
    FROM Клиент
    WHERE ГородКлиента in ("Москва", "Самара")
```

- NOT IN используется для отбора любых значений, кроме тех, которые указаны в представленном списке.

Пример 9. *Вывести список клиентов, проживающих не в Москве и не в Самаре.*

```
SELECT Фамилия, ГородКлиента
    FROM Клиент
    WHERE ГородКлиента Not in ("Москва","
Самара")
```

Соответствие шаблону

- С помощью оператора **LIKE** можно выполнять сравнение выражения с заданным шаблоном, в котором допускается использование символов-заменителей:
- Символ % – вместо этого символа может быть подставлено любое количество произвольных символов.
- Символ _ заменяет один символ строки.
- [ ] – вместо символа строки будет подставлен один из возможных символов, указанный в этих ограничителях.
- [^] – вместо соответствующего символа строки будут подставлены все символы, кроме указанных в ограничителях.

- **Пример 10.** *Найти клиентов, у которых в номере телефона вторая цифра – 4.*

```
SELECT Клиент.Фамилия, Клиент.Телефон
    FROM Клиент
    WHERE Клиент.Телефон Like "_4%»
```

- **Пример 11.** *Найти клиентов, у которых в номере телефона вторая цифра – 2 или 4.*

```
SELECT Клиент.Фамилия, Клиент.Телефон
    FROM Клиент
    WHERE Клиент.Телефон Like "_[2,4]%"
```

- **Пример 12.** *Найти клиентов, у которых в номере телефона вторая цифра 2, 3 или 4.*

```
SELECT Клиент.Фамилия, Клиент.Телефон
    FROM Клиент
    WHERE Клиент.Телефон Like "_[2-4]%"
```

- **Пример 13.** *Найти клиентов, у которых в фамилии встречается слог "ро".*

```
SELECT Клиент.Фамилия
    FROM Клиент
    WHERE Клиент.Фамилия Like "%ро%"
```

Значение NULL

- Оператор **IS NULL** используется для сравнения текущего значения со значением **NULL** – специальным значением, указывающим на отсутствие любого значения. **NULL** – это не то же самое, что знак пробела (пробел – допустимый символ) или ноль (0 – допустимое число). **NULL** отличается и от строки нулевой длины (пустой строки).
- **Пример 14.** *Найти сотрудников, у которых нет телефона (поле Телефон не содержит никакого значения).*

```
SELECT Фамилия, Телефон
    FROM Клиент
    WHERE Телефон Is Null
```

- **IS NOT NULL** используется для проверки присутствия значения в поле.

- **Пример 15.** *Выборка клиентов, у которых есть телефон (поле Телефон содержит какое-либо значение).*

```
SELECT Клиент.Фамилия, Клиент.Телефон
    FROM Клиент
    WHERE Клиент.Телефон Is Not Null
```

Предложение ORDER BY

- По умолчанию реализуется сортировка по возрастанию. Явно она задается ключевым словом **ASC**. Для выполнения сортировки в обратной последовательности необходимо после имени поля, по которому она выполняется, указать ключевое слово **DESC**.
- **Пример *16.****Вывести список клиентов в алфавитном порядке*.

```
SELECT Клиент.Фамилия, Клиент.Фирма
    FROM Клиент
    ORDER BY Клиент.Фамилия
```

- **Пример 17.** *Вывести список фирм и клиентов. Названия фирм упорядочить в алфавитном порядке, имена клиентов в каждой фирме отсортировать в обратном порядке.*

```
SELECT Клиент.Фирма, Клиент.Фамилия
  FROM Клиент
  ORDER BY Клиент.Фирма, Клиент.Фамилия
DESC
```

Фраза **ORDER BY** всегда должна быть последним элементом в операторе SELECT.

Инструкция SELECT … INTO

- Данная инструкция позволяет создавать новую таблицу, имя которой задается в качестве параметра имя новой таблицы.

SELECT [предикат] список полей [AS псевдонимы полей]
INTO имя новой таблицы
FROM список таблиц [IN для таблиц внешних баз данных]
[WHERE условие отбора строк]
[GROUP BY группируемые поля]
[HAVING условия отбора групп]
[ORDER BY поля для сортировки и условия их сортировки]
[WITH OWNERACCESS OPTION]

Получить список студентов со средним рейтингом > 90

```
SELECT ZBOOK, AVG (rait) AS r
FROM raiting
GROUP BY ZBOOK
HAVING AVG(rait)>90
```

- Стандарт SQL позволяет использовать **стандартные теоретико-множественные операции над множествами – UNION** (объединение), **INTERSECT** (пересечение), **EXCEPT** (разность).

Пример. Определить список студентов (по номерам зачетных книжек) со средним рейтингом > 90 и не имеющих троек:

```
SELECT ZBOOK FROM raiting
GROUP BY ZBOOK
    HAVING AVG(rait)>90)
EXCEPT(SELECT ZBOOK
        FROM raiting
GROUP BY ZBOOK
    HAVING MIN(rait)>75)
```

Оператор добавления строк в таблицу INSERT

Вставка одной новой строки

INSERT
INTO таблица-адресат [столбец, ...]
VALUES (значение, ...)

- Выполняется вставка новой строки с заданными значениями указанных столбцов: каждому указанному столбцу в списке столбцов должно соответствовать выражение из списка значений (соответствие определяется по порядку следования), а поля оставшихся столбцов заполняются значением NULL или значением по умолчанию.
- Если столбцы не указаны, предполагается, что указаны все столбцы в порядке их определения в схеме таблицы.

Копирование множества строк из одной таблицы в другую

INSERT
INTO таблица-адресат [столбец, ...]
SELECT список полей
FROM выражение

- В указанную таблицу вставляются строки, полученные в результате выполнения предложения **Select**, удовлетворяющего рассмотренным выше требованиям синтаксиса SELECT.
- Перечень столбцов интерпретируется аналогично первому варианту INSERT: **предложения_select** должно соответствовать по структуре и порядку следования списку столбцов, а если список столбцов отсутствует – то структуре таблицы.

```
INSERT
INTO    TEMP   (   P#, WEIGTH )
       SELECT P#, WEIGTH
       FROM   P
       WHERE  COLOR = COLOR('Red') ;
```

- В этом примере подразумевается, что предварительно создана другая таблица **TEMP** с двумя столбцами **P#** и **WEIGTH**. Оператор **INSERT** вставляет в нее номера деталей и соответствующие веса всех деталей с цветом

  ' Red' (красный).

Оператор удаления строк DELETE

DELETE FROM таблица
[WHERE условие]

- В таблице удаляются все строки, удовлетворяющие условию, синтаксис которого соответствует синтаксису раздела **WHERE** оператора **SELECT**. Если **WHERE** отсутствует, будут удалены все строки.

```
DELETE FROM
SP WHERE  P#
=P2
```

- Этот оператор **DELETE** удаляет из таблицы **SP** все строки с информацией о поставках детали с номером **' P2 '**.

Оператор обновления строк UPDATE

UPDATE таблица
SET столбец = значение, ...
[WHERE условие]

- В таблице модифицируются все строки (если указано условие, то только удовлетворяющие условию строки) внесением нового значения поля в столбец.

```
UPDATE S
SET    STATUS = 2 * STATUS ,
       CITY = 'Rome'
WHERE  CITY = 'Paris' ;
```

- Приведенный оператор **UPDATE** увеличивает в два раза код статуса всех поставщиков в Париже и вносит в базу данных информацию о том, что эти поставщики переехали в Рим.

Удаление таблицы DROP TABLE

DROP TABLE имя_таблицы
[RESTRICT | CASCADE]

- С помощью ключевых слов **RESTRICT** и **CASCADE** соответственно указывается, нужно ли удалять каскадно (CASCADE) связанные с таблицей объекты (таблицы, формы, связи и т.д.) или нет (RESTRICT).

CREATE TABLE

CREATE TABLE имя_таблицы
(имя_столбца тип_данных [NULL | NOT NULL] [,...n])

Ключевое слово NULL используется для указания того, что в данном столбце могут содержаться значения NULL.

Значение NULL отличается от пробела или нуля – к нему прибегают, когда необходимо указать, что данные недоступны, опущены или недопустимы.

Если указано ключевое слово NOT NULL, то будут отклонены любые попытки поместить значение NULL в данный столбец.

Если указан параметр NULL, помещение значений NULL в столбец разрешено. По умолчанию стандарт SQL предполагает наличие ключевого слова NULL.

- **Пример 1.** *Создать таблицу для хранения данных о товарах, поступающих в продажу в некоторой торговой фирме. Необходимо учесть такие сведения, как название и тип товара, его цена, сорт и город, где товар производится.*

```
CREATE TABLE Товар
(Название VARCHAR(50) NOT NULL,
Цена MONEY NOT NULL,
 Тип VARCHAR(50) NOT NULL,
 Сорт VARCHAR(50),
 ГородТовара VARCHAR(50))
```

- **Пример 2.***Создать таблицу для сохранения сведений о постоянных клиентах с указанием названий города и фирмы, фамилии, имени и отчества клиента, номера его телефона.*

```
CREATE TABLE Клиент
(Фирма VARCHAR(50) NOT NULL,
 Фамилия VARCHAR(50) NOT NULL,
 Имя VARCHAR(50) NOT NULL,
 Отчество VARCHAR(50),
ГородКлиента VARCHAR(50),
Телефон CHAR(10) NOT NULL)
```