

## ОСНОВНЫЕ ПОНЯТИЯ И ОПРЕДЕЛЕНИЯ

### 1.1. Основные понятия реляционных баз данных

Основой современных систем, использующих базы данных, является *реляционная модель данных*. В этой модели данные, представляющие информацию о предметной области, организованы в виде двумерных таблиц, называемых *отношениями*. На рис. 1 приведен пример такой таблицы-отношения и поясняются основные термины реляционной модели.

Код студ	Имя студ	Факультет	Курс
0043	Иванов	Физический	1
2004	Петров	Химический	2
5162	Сидоров	Физический	1
0007	Орлов	Химический	2
0634	Смирнов	Исторический	4
0228	Попов	Физический	3
1735	Кузнецов	Химический	2

Рис. 1

• *Отношение* — это таблица, подобная приведенной на рис. 1, и состоящая из строк и столбцов. Верхняя строка таблицы-отношения называется *заголовком отношения*. Термины *отношение* и *таблица* обычно употребляются как синонимы, однако в языке SQL используется термин *таблица*.

- Строки таблицы – отношения называются кортежами; или - записями. Столбцы называются атрибутами. Термины: атрибут - столбец, колонка, поле - обычно используются как синонимы. Каждый атрибут имеет наименование (имя), которое должно быть уникальным в конкретной таблице-отношении; однако в разных таблицах имена атрибутов могут совпадать.
- Количество кортежей в таблице-отношении называется: кардинальным числом отношения; а количество атрибутов называется степенью отношения.
- Ключ, или первичный ключ отношения - это уникальный идентификатор строк (кортежей), т.е. такой атрибут (набор атрибутов), для которого в любой момент времени в отношении не существует строк с одинаковыми значениями этого атрибута (набора атрибутов). На рис. 1. таблицы ячейка с именем ключевого атрибута имеет нижнюю границу в виде двойной черты.
- Домен отношения – это совокупность значений, из которых могут выбираться значения конкретного атрибута, т.е. конкретный - набор имеющихся в таблице значений атрибута в любой момент времени должен быть подмножеством множества значений домена, на котором определен этот атрибут. В общем случае на одном и том же домене могут быть определены значения разных атрибутов. Важным является то, что домены вводят ограничения на операции сравнения значений различных атрибутов. Эти ограничения состоят в том, что корректным образом можно сравнивать между собой только значения атрибутов определенных на одном — и том же домене. Отношения реляционной базы данных обладают следующими свойствами:

- в отношениях не должно быть кортежей дубликатов;
- кортежи отношений неупорядочены;
- атрибуты отношений также неупорядочены;
- Из этих свойств отношения вытекают следующие важные следствия.
- Из уникальности: кортежем следует, что в отношении всегда имеется атрибут или набор атрибутов, позволяющий идентифицировать кортеж; другими словами, в отношении всегда есть первичный ключ.
- Из неупорядоченности кортежей следует, во-первых, что в отношении не существует другого способа адресации кортежей, кроме адресации по ключу; во-вторых, что в отношении не существует таких понятий как первый кортеж, последний, предыдущий, следующий и т.п.
- Из неупорядоченности атрибутов следует, что единственным способом их адресации в запросах - является использование наименования атрибута.

Относительно свойства реляционного отношения, касающегося отсутствия кортежей - дубликатов, следует сделать важное замечание. В этом пункте SQL не полностью соответствует реляционной модели. А именно, в отношениях, являющихся результатами запросов, SQL допускает наличие одинаковых строк. Для их устранения в запросе используется ключевое слово **DISTINCT** (см. ниже).

Информация в реляционных базах данных, как правило, хранится не в одной таблице-отношении, а в нескольких. При создании нескольких таблиц взаимосвязанной информации появляется возможность выполнения более сложных операций с данными, т.е. более сложной обработки данных. Для работы со связанными данными из нескольких таблиц важным является понятие так называемых внешних ключей.

Внешним ключом таблицы называется атрибут или набор атрибутов этой таблицы, каждое значение которых в текущем состоянии таблицы всегда совпадает со значением атрибутов, являющихся ключом, в другой таблице. Внешние ключи используются для связывания значений атрибутов из разных таблиц. С помощью внешних ключей обеспечивается так называемая ссылочная целостность базы данных, т.е. согласованность данных, описывающих одни и те же объекты, но хранящихся в разных таблицах.

## 1.2. Отличие SQL от процедурных языков программирования

Язык SQL относится к классу непроедурных языков программирования. В отличие от универсальных процедурных языков, которые также могут быть использованы для работы с базами данных, язык SQL ориентирован не на записи, а на множества.

Это означает следующее. В качестве входной информации для формулируемого на языке SQL запроса к базе данных используется множество кортежей-записей одной или нескольких таблиц - отношений. В результате выполнения запроса также образуется множество кортежей результирующей таблицы - отношения. Другими словами, в SQL результатом любой операции над отношениями также является - отношение. Запрос SQL задает не процедуру, т.е. последовательность действий, необходимых для получения результата, а условия, которым должны удовлетворять кортежи результирующего отношения, сформулированные в терминах входного отношения (входных отношений).

## 1.3. Интерактивный и встроенный SQL

Существуют и используются две формы языка SQL: интерактивный SQL и встроенный SQL.

Интерактивный SQL используется для непосредственного ввода SQL-запросов пользователем и получения результата в интерактивном режиме.

Встроенный SQL состоит из команд SQL, встроенных внутрь программ, которые обычно написаны на некотором другом языке (Паскаль, С, С++ и др.). Это делает программы, написанные на таких языках; более мощными, гибкими и эффективными, обеспечивая их применение для работы с данными, хранящимися в реляционных базах. При этом, однако, требуются дополнительные средства обеспечения интерфейса SQL с языком, в который он встраивается.

Данная методичка посвящена интерактивному SQL, поэтому в ней не обсуждаются вопросы построения интерфейсов, позволяющих связать SQL с другими языками программирования.

#### 1.4, Составные части SQL

И интерактивный, и ватроенный SQL подразделяются нв следующие составные части.

Язык Определения Данных - DDL (Data Definition Language):

дает возможность создания, изменения и удаления различных объектов базы данных (таблиц, индексов, пользователей, привилегий и т. п.).

К числу дополнительных функций DDL могут быть отнесены средства определения ограничений целостности данных, определения по- рядка структур хранения данных, описания элемёнтов физического уровня хранения данных.

Язык Обработки Данных — DML (Data Manipulation Language): предоставляет возможность выборки информации из базы данных и ее преобразования.

Тем не менее это не:два различных языка, а:компоненты.единого SQL/

#### 1.5. Типы данных

В языке SQL имеются средства, позволяющие для каждого:атрибута указывать тип данных, которому должны соответствовать все значения этого атрибута.

Следует отметить, что определение типов данных является той частью, в которой коммерческие реализации языка не полностью согласуются с требованиями официального стандарта SQL; Это объясняется, в частности, желанием сделать SQL совместимым с другими языками программирования.

**1.5.1. Тип данных "строка символов".** Тип данных **CHARACTER** или **CHAR** представляет символьные строки фиксированной длины. Его синтаксис имеет вид;

**CHARACTER[(**<длийа>**)]** или

**CHAR[(**<длийа>**)]**.

Текстовые значения:поля таблицы, для которого определен тип **CHAR**, имеют фиксированную длину, которая определяется параметром **<длина>**. Этот параметр может принимать значения от 1 до 255, т.е, строка может содержать до 255 символов. Если во вводимой в поле текстовой константе фактическое число символов меньше числа; определенного параметром **<длина>**, то эта константа автоматически дополняется справа пробелами до заданного числа символов. Квадратные скобки указывают на то, что значение параметра **<длина>** может не указываться явно. В этом случае-длина строки нолагается равной одному символу.

Тип данных для строк переменной длины может обозначаться ключевыми словами **VARCHAR**, **CHARACTER VARYING** или **CHAR VARYING**. Он описывает текстовую строку, которая может иметь произвольную длину до,определенного конкретной реализацией SQL максимума (в Oracle до 2000 символов). В отличие от типа **CHAR**, в этом случае-при вводе текстовой константы, фактическая длина которой меньше заданной, ее дополнение пробелами до заданного максимального значения не производится.

Константы, имеющие тип **CHARACTER** или **VARCHAR**, в выражвнкмх SQL заключаются в одиночные кавычки; например, '**<текст>**'

Следующие предложения. эквивалентны:

**VARCHAR**[(**<длина>**)]; **CHAR VARYING**[(**<длина>**)]; **CHARACTER VARYING**[(**<длина>**)].

Если **длина:строки** не указана явно, она полагается равной одному символу во всех случаях.

По сравнению с типом **CHAR** тип данных **VARCHAR** позволяет более экономно использовать память, выделяемую для хранения текстовых значений, и оказывается более удобным при выполнении операций, связанных со сравнением текстовых констант.

**1.5.2. Числовые типы данных.** Стандартными числовыми типами данных SQL являются:

- **INTEGER** - используется для- представления целых: чисел в диапазоне от -231 до +231,
- **SMALLINT** - используется для представления целых чисел в диапазоне меньшем, чем для **INTEGER**, а именно от -215 до +215;
- **DECIMAL**(**<точность>**[,**<масштаб>**]) - десятичное число с фиксированной точкой; **точность** указывает, сколько значащих цифр имеет число. **Масштаб** указывает максимальное число цифр справа от точки;
- **NUMERIC**(**<точность>**[, **< масштаб >**]) – десятичное число, с фиксированной точкой, такое же, как и **DECIMAL**
- **FLOAT**[(**<точность >**)] - число с плавающей точкой и указанной минимальной точностью;
- **REAL** - такое же число, как и **FLOAT**, за исключением того, что **точность** устанавливается по умолчанию в зависимости от конкретной реализации SQL.
- **DOUBLE PRECISION** - такое же-число, как-и **REAL**, но **точность** в два раза превышает **точность** для **REAL**.

СУБД Oracle использует дополнительно тип данных **NUMBER** для представления всех. числовых данных; целых, с фиксированной или плавающей точкой. Его синтаксис:

- **NUMBER**[(**<точность >**[, **<масштаб>**)])

Если значение параметра - **<точность>** не указано явно, оно: полагается равным 38, Значение параметра **<масштаб>** по умолчанию предполагается равным 0. Значение параметра **<точность>** может изменяться от 1 до 38; значение параметра **<масштаб>** может изменяться от 84 до 128. Использование. отрицательных значений **масштаба** означает сдвиг десятичной точки в сторону старших разрядов. Например, определение **NUMBER**(7, -3) означает округление до тысяч.

Типы **DECIMAL** и **NUMERIC** полностью эквивалентны типу **NUMBER**. Синтаксис:

**DECIMAL**[(**<точность>**[,**<масштаб>**)])

**DEC**[(**<точность>**[,**<масштаб>**)])

**NUMERIC**[(**<точность>**[,**<масштаб>**)])

Напоминаем, что квадратные скобки указывают на. необязательность заключенных в них параметров,

**1.5.3. Дата и время.** Представление дат и времени в SQL. зависит от конкретной СУБД. В Oracle тип данных **DATE** используется для представления даты и времени. Наличие типа данных для хранения даты позволяет поддерживать специальную арифметику дат. Добавление к переменной: типа **DATE** целого числа означает увеличение даты на соответствующее число-дней, а вычитание соответствует определению более ранней даты.

Константы типа **DATE** записываются в зависимости от: формата, принятого в конкретной системе. Например, '03.05.1999' или '12/06/1989', или '03-nov-1999', или '03-apr-99'.

**1.5.4. Неопределенные или отсутствующие данные (NULL).** Для обозначения отсутствующих, пропущенных или неизвестных значений атрибута в SQL используется

ключевое слово **NULL**. Довольно часто можно встретить словосочетание "атрибут имеет значение **NULL**". Строго говоря, **NULL** не является значением в обычном понимании, а используется именно для обозначения. того факта, что действительное значение атрибута на самом деле по каким-либо причинам отсутствует. Это приводит к ряду особенностей, что следует учитывать при использовании значений атрибутов которые могут находиться в состоянии **NULL**.

- В агрегирующих функциях, позволяющих получать сводную- информацию по множеству значений атрибута, например, суммарное или среднее значение, для обеспечения точности и-однозначности толкования результатов отсутствующие или **NULL** -значения атрибутов игнорируются.
- Условные операторы расширяются от булевой двужначной логики истина/ложь до трехзначной логики истина/ложь/неизвестно.
- Все операторы возвращают пустое значение (**NULL**), если значение любого из операндов отсутствует (имеет "значение **NULL**").
- Для проверки на пустое значение следует использовать операторы **IS NULL** и **IS NOT NULL** (использование для этого оператора сравнения "=" является ошибкой).
- Функции преобразования типов, имеющие **NULL** в качестве аргумента, возвращают пустое значение (**NULL**).

### 1.6. Используемые термины и обозначения

Ключевые слова - это используемые в выражениях SQL слова, имеющие специальное назначение (например, они могут обозначать: конкретные команды SQL). Ключевые слова нельзя использовать для других целей, к примеру, в качестве имен объектов базы данных. В книге они выделяются шрифтом: **КЛЮЧЕВОЕСЛОВО**.

Команды, или предложения, являются инструкциями, с помощью которых SQL обращается к базе данных. Команды состоят из нескольких (одной или более) логических частей, называемых предложениями. Предложения начинаются ключевым словом и состоят из ключевых слов и аргументов.

Объекты базы данных, имеющие имена (таблицы, атрибуты и др.): в книге также выделяются особым образом: **ТАБЛИЦА1**, **АТРИБУТ\_2**.

В описании синтаксиса команд SQL оператор определения "::<=" разделяет определяемый элемент (слева от оператора) и собственно его определение (справа от оператора); квадратные скобки "[" "]" указывают необязательный элемент синтаксической конструкции; многоточие "..." указывает, что выражение, предшествующее ему, может повторяться любое число раз; фигурные скобки "{" "}" объединяют последовательность элементов в логическую группу, один из элементов которой должно быть обязательно использован; вертикальная черта "|" указывает, что часть определения, следующая за этим символом, является одним из возможных вариантов; в угловые скобки "< >" заключаются элементы, которые объясняются по мере того, как вводятся.

### 1.7. Учебная база данных

В приводимых примерах построения SQL-запросов и контрольных упражнениях используется база данных, состоящая из следующих таблиц:

**Таблица 1.1 STUDENT(Студент)**

STUDENT_ID	SURNAME	NAME	STIPEND	KURS	CITY	BIRTHDAY	UNIV_ID
1	Иванов	Иван		1	Орёл		10
3	Петров	Петр		3	Курск		10
6	Сидоров	Вадим			Москва		22
10	Кузнецов	Борис					
12	Зайцева	Ольга					
265	Павлов	Андрей					
32	Котов	Павел					
654	Лукин	Артем					
276	Петров	Антон					
55	Белкин	Вадим					

STUDENT\_ID - числовой код, идентифицирующий студента (идентификатор студента)

SURNAME – фамилия студента.

NAME – имя студента.

STIPEND - стипендия которую получает студент.

KURS - курс, на котором учится студент.

CITY - город, в котором живет студент.

BIRTHDAY - дата рождения- студента.

UNIV\_ID - идентификатор университета, в котором учится студент.

**Таблица 1.2. LECTURER (Преподаватель)**

LECTURER_ID	SURNAME	NAME	CITY	UNIV_ID
24	Колесников	Борис	Воронеж	10
46	Никонов	Иван	Воронеж	10
74	Лагутин	Павел	Москва	22
108	Струков	Николай	Москва	22
276	Николаев	Виктор	Воронеж	10
328	Сорокин	Андрей	Орел	10

LECTURER\_ID - идентификатор преподавателя.

SURNAME - фамилия преподавателя.

NAME - имя преподавателя.

CITY - город, в котором живет преподаватель.

UNIV\_ID - идентификатор университета, в котором работает преподаватель.

**Таблица 1.3. SUBJECT (Предмет обучения)**

SUBJ_ID	SUBJ_NAME	HOUR	SEMESTER
10	Информатика	56	1
22	Физика	34	1
43	Математика	56	2
56	История	34	4
94	Английский	56	3
73	Физкультура	34	5

SUBJ\_ID – идентификатор предмета обучения.

SUBJ\_NAME – наименование предмета обучения

HOUR – количество часов, отводимых на изучение предмета.

SEMESTER – семестр в котором изучается данный предмет.

Таблица 1.4. UNIVERSITY (Университет)

UNIV_ID	UNIV_NAME	RATING	CITY
22	МГУ	610	Москва
10	ВГУ	296	Воронеж
11	НГУ	345	Новосибирск
32	РГУ	421	Ростов
14	БГУ	326	Белгород
15	ТГУ	373	Томск
18	ВГМА	327	Воронеж

UNIV\_ID - идентификатор университета.

UNIV\_NAME — название университета.

RATING - рейтинг университета.

CITY - город, в котором расположен университет.

Таблица 1.5. EXAM\_MARKS (Экзаменационные оценки)

EXAM_ID	STUDENT_ID	SUBJ_ID	MARK	EXAM_DATE
145	12	10	5	12/01/2006
34	32	10	4	23/01/2006
75	55	10	5	05/01/2006
238	12	22	3	17/06/2005
639	55	22	NULL	22/06/2005
43	6	22	4	18/01/2006
...	...	...	...	...

EXAM\_ID — идентификатор экзамена.

STUDENT\_ID - идентификатор студента.

SUBJ\_ID - идентификатор предмета обучения.

MARK – экзаменационная оценка.

EXAM\_DATE - дата экзамена.

Таблица.1.6. SUBJ\_LECT (Учебные дисциплины преподавателей)

LECTURER_ID	SUBJ_ID
24	24
46	46
74	74
108	108
276	276
328	328
...	...

LECTURER\_ID - идентификатор преподавателя.

SUBJ\_ID - идентификатор предмета обучения.

## ВОПРОСЫ

1. Какие поля приведенных таблиц являются первичными ключами?
2. Какие данные хранятся в столбце 2 в таблице "Предмет обучения?"
3. Как по-другому называется строка? столбец?
4. Почему мы не можем запросить для просмотра первые Пять строк?.

## ВЫБОРКА ДАННЫХ (ОПЕРАТОР SELECT)

### 2.1. Простейшие SELECT-запросы

Оператор SELECT (ВЫБРАТЬ) языка SQL является самым важным и наиболее часто используемым оператором. Он предназначен для выборки информации из таблиц базы данных. Упрощенный синтаксис оператора SELECT выглядит следующим образом.

**SELECT [DISTINCT]** <список выражений над атрибутами и константами>

**FROM** <список таблиц>

[**WHERE** <условие выборки >]

[**GROUP BY** <список атрибутов>]

[**HAVING** <условие>]

[**UNION** <выражение с оператором **SELECT**>]

[**ORDER BY** <список атрибутов>];

В квадратных скобках указаны элементы, которые могут отсутствовать в запросе.

Ключевое слово **SELECT** сообщает базе данных, что данное предложение является запросом на выборку информации. После слова **SELECT** через запятую перечисляются наименования полей (список атрибутов), содержимое которых запрашивается.

Обязательным ключевым словом в предложении-запросе **SELECT** является слово **FROM** (ИЗ). За ключевым словом **FROM** указывается список разделенных запятыми имен таблиц, из которых извлекается информация.

Например,

**SELECT NAME, SURNAME FROM STUDENT;**

Любой SQL-запрос должен заканчиваться символом ";" (точка с запятой).

Приведенный запрос осуществляет выборку всех значений полей NAME и SURNAME из таблицы STUDENT.

Его результатом является таблица:

Порядок следования столбцов в этой таблице соответствует порядку полей NAME и SURNAME, указанному в запросе; а не их порядку во входной таблице STUDENT.

Если необходимо вывести значения всех столбцов таблицы, то можно вместо перечисления их имен использовать символ "\*" (звездочка).

**SELECT \***

**FROM STUDENT;**

В данном случае в результате выполнения запроса будет получена вся таблица **STUDENT**.

Еще раз обратим внимание на то, что получаемые в результате SQL-запроса таблицы не в полной мере отвечают определению реляционного отношения. В частности, в них могут оказаться кортежи с одинаковыми значениями атрибутов.

Например, запрос "Получить список названий городов, где проживают студенты, сведения о которых находятся в таблице STUDENT", можно записать в следующем виде:

**SELECT CITY FROM STUDENT;**

Его результатом будет таблица

Видно, что в таблице встречаются одинаковые строки (выделены жирным шрифтом).

Для исключения из результата SELECT-запроса повторяющихся записей используется ключевое слово **DISTINCT** (ОТЛИЧНЫЙ). Если запрос **SELECT** извлекает множество



полей, то **DISTINCT** исключает дубликаты строк, в которых значения всех выбранных полей идентичны.

Запрос "Определить список названий различных-горопоа, где проживают студенты, сведения о которых находятся в таблице STUDENT" можно записать в следующем виде.

**SELECT DISTINCT CITY FROM STUDENT;**

В результате получим таблицу, в которой дубликаты строк исключены:

Ключевое слово **ALL** (ВСЕ), в отличие от **DISTINCT**, оказывает противоположное действие, т. е. при его использовании повторяющиеся строки включаются в состав выходных данных. Режим, задаваемый ключевым словом **ALL**, действует по умолчанию, поэтому в реальных запросах для этих целей оно практически не используется.

Использование в операторе **SELECT** предложения, определяемого ключевым словом **WHERE** (ГДЕ), позволяет задавать выражение- условия (предикат), принимающее значение истина или ложь (а также неизвестно при использовании **NULL**) для значений полей строк таблиц, к которым обращается оператор **SELECT**. Предложение **WHERE** определяет, какие строки указанных таблиц должны быть выбраны. В таблицу, являющуюся результатом запроса, включаются только те строки, для которых условие (предикат); указанное в предложении **WHERE**, принимает значение истина.

#### Пример.

Написать запрос, выполняющий выборку имен (NAME) всех студентов с фамилией (SURNAME) Петров, сведения о которых находятся в таблице STUDENT.

**SELECT SURNAME, NAME  
FROM STUDENT  
WHERE SURNAME = 'Петров'**

Результатом этого запроса будет таблица

SURNAME	NAME
Петров	Петр
Петров	Антон

В задаваемых в предложении **WHERE** условиях могут использоваться операции сравнения, определяемые следующими операторами: = (равно), > (больше), < (меньше), >= (больше или равно), <= (меньше или равно), <> (не равно), а также логические операторы **AND**, **OR** и **NOT**.

Например, запрос для получения имен и фамилий студентов, обучающихся на третьем курсе и получающих стипендию (размер стипендии больше нуля) будет выглядеть таким образом:

**SELECT NAME, SURNAME  
FROM STUDENT  
WHERE (KURS = 3 AND STIPEND > 0);**

Результат выполнения этого запроса имеет вид:

SURNAME	NAME
Петров	Петр
Лукин	Артем

## УПРАЖНЕНИЯ

1. Напишите запрос к таблице SUBJECT, выводящий для каждой ее строки идентификатор (номер) предмета обучения, его наименование, семестр, в котором он читается, и количество отводимых на него часов.
2. Напишите запрос, позволяющий вывести -все строки таблицы EXAM\_MARKS, в которых предмет обучения имеет номер (SUBJ\_ID), равный 12.
3. Напишите запрос, выбирающий все данные из таблицы STUDENT, расположив столбцы таблицы в следующем порядке: KURS, SURNAME, MAME, STIPEND.
4. Напишите запрос SELECT, который для каждого предмета обучения (SUBJECT) выполняет вывод его наименования (SUBJ\_NAME) и следом за ним количества часов (HOUR) в 4-м семестре (SEMESTR).
5. Напишите запрос, позволяющий получить из таблицы EXAM\_MARKS значения столбца MARK (экзаменационная оценка) для всех студентов, исключив из списка повторение одинаковых строк.
6. Напишите запрос, который выполняет вывод списка фамилий студентов, обучающихся на третьем и более старших курсах.
7. Напишите запрос, выбирающий данные фамилию, имя и номер курса для студентов, получающих стипендию больше 140.
8. Напишите запрос, выполняющий выборку из таблицы SUBJECT названий всех предметов обучения, на которые отводится более 30 часов.
9. Напишите запрос, который выполняет вывод списка университетов, рейтинг которых превышает 300 баллов.
10. Напишите запрос к таблице STUDENT для вывода списка всех студентов со стипендией не меньше 100, живущих в Воронеже- с указанием фамилии (SURNAME), имени (NAME) и номера курса (KURS)
11. Какие данные будут получены в результате выполнения запроса?  
**SELECT \***  
**FROM STUDENT**  
**WHERE (STIPEND < 100 OR**  
**NOT (BIRTHDAY >= '10/03/1980'**  
**AND STUDENT\_ID > 1003));**
12. Какие данные будут получены в результате. выполнения запроса?  
**SELECT \***  
**FROM STUDENT**  
**WHERE NOT ((BIRTHDAY = '10/03/1980' OR STIPEND > 100)**  
**AND STUDENT\_ID ) >= 1003);**
13. Напишите запрос для получения списка студентов старше 25 лет, обучающихся на 1-м курсе.
14. Напишите запрос для получения списка предметов, для которых в 1-м семестре отведено более 100 часов.
15. Напишите запрос для получения списка преподавателей, живущих в Воронеже.
16. Напишите запрос для получения списка университетов, расположенных в Москве и имеющих рейтинг меньший, чем у ВГУ. Константу в ограничении на рейтинг можно определить по этой же таблице.
17. Напишите запрос для получения списка студентов, проживающих в Воронеже и не получающих стипендию.
18. Напишите запрос для получения списка студентов моложе--20 лет.
19. Напишите запрос для получения списка студентов без определенного места жительства.