Оператор выборки данных SELECT SQL

Операторы языка SQL

Язык SQL включает в себя операторы разных категорий. Любой SQL-оператор состоит из зарезервированных слов и слов, определяемых пользователем в соответствии с установленными синтаксическими правилами. Как и во многих языках программирования, большинство компонентов операторов языка не чувствительны к регистру. Исключение из этого правила как обычно составляют символьные данные, при задании которых необходимо помнить о регистре и использовать тот, который необходим для представления данных.

Для записи операторов в языке принят свободный формат, позволяющий посредством отступов и выравниваний придать SQL-программе более читабельный вид.

При записи операторов рекомендуется придерживаться следующих правил:

- каждая фраза в операторе должна начинаться с новой строки;
- начало каждой фразы должно быть выровнено с началом остальных фраз оператора;
- каждая часть фразы должна начинаться с новой строки с некоторым отступом относительно начала всей фразы, что позволит выделить подчиненные части;
- для записи операторов применяются некоторые соглашения:
- для записи зарезервированных слов используются прописные буквы;
- для записи определяемых пользователем слов используются строчные буквы;
- вертикальная черта "|" указывает на необходимость выбора одного из нескольких значений;
- фигурные скобки определяют обязательный элемент;
- квадратные скобки определяют необязательный элемент;
- многоточие "..." используется для указания необязательной возможности повторения конструкции, от нуля до нескольких раз.

Операторы каждой категории приведены в отдельной таблице.

Операторы определения данных (табл.1) применяются для описания структур используемых данных. В состав этой категории входят следующие операторы: CREATE TABLE, DROP TABLE, ALTER TABLE, CREATE VIEW, ALTER VIEW, DROP VIEW.

***Таблица 1**. Операторы определения данных*

Оператор	**Пояснение**
CREATE TABLE	Создать таблицу
DROP TABLE	Удалить таблицу
ALTER TABLE	Изменить таблицу
CREATE VIEW	Создать представление
ALTER VIEW	Изменить представление
DROP VIEW	Удалить представление

Операторы манипулирования данными, образующие следующую категорию операторов, предназначены для заполнения таблиц данными и для обновления загруженной в них информации. К данной категории относятся следующие операторы: DELETE, INSERT, UPDATE (табл.2).

***Таблица 2**. Операторы манипулирования данными*

Оператор	**Пояснение**
DELETE	Удаляет одну или несколько строк, соответствующих условиям фильтрации, из базовой таблицы
INSERT	Вставляет одну строку в базовую таблицу
UPDATE	Обновляет значения одного или нескольких столбцов в одной или нескольких строках, соответствующих условиям фильтрации

Для выборки информации из базы данных предназначен язык запросов, который в языке SQL представлен одним оператором SELECT (табл.3).

***Таблица 3.** Язык запросов*

Оператор	**Пояснение**
SELECT	Выбирает строки; оператор, позволяющий сформировать результирующую таблицу, соответствующую запросу

Кроме указанных категорий операторов, назначение которых не сложно представить, прочитав пояснения в таблицах, необходимо выделить еще две: операторы управления транзакциями (табл. 4) и средства администрирования данных (табл. 5).

Таблиц 4. *Управление транзакциями*

Оператор	**Пояснение**
COMMIT	Завершить транзакцию— завершить обработку информации, объединенную в транзакцию
ROLLBACK	Откатить транзакцию— отменить изменения, проведенные в ходе выполнения транзакции
SAVEPOINT	Сохранить промежуточную точку выполнения транзакции - сохранить промежуточное состояние БД, пометить его для того, чтобы можно было в дальнейшем к нему вернуться

Таблица 5*. Администрирование данных*

Оператор	**Пояснение**
ALTER DATABASE	Изменить набор основных объектов в базе данных, ограничений, касающихся всей базы данных
ALTER DBAREA	Изменить ранее созданную область хранения
ALTER PASSWORD	Изменить пароль для всей базы данных
CREATE DATABASE	Создать новую базу данных
CREATE DBAREA	Создать новую область хранения и сделать ее доступной для размещения данных
DROP DATABASE	Удалить существующую базу данных
DROP DBAREA	Удалить существующую область хранения (если в ней на настоящий момент не располагаются активные данные)
GRANT	Предоставить права доступа на ряд действий над некоторым объектом БД
REVOKE	Лишить прав доступа к некоторому объекту или некоторым действиям над объектом

В коммерческих СУБД набор основных операторов расширен. В большинство СУБД включены операторы определения и удаления индекса запуска хранимых процедур и операторы определения триггеров.

Начинать знакомство с данным языком принято с рассмотрения возможностей языка запросов, который в языке SQL представлен одним оператором SELECT, потому что этот мощный оператор, естественно, является и самым сложным. К тому же в дальнейшем интересно посмотреть, как его можно использовать совместно с операторами манипулирования данными.

1. Оператор выбора *SELECT.* Формирование запросов к базе данных

Назначение оператора SELECT состоит в выборке и отображении данных одной или нескольких таблиц БД. Этот исключительно мощный, наиболее часто используемый оператор реализует все операции реляционной алгебры. Один и тот же запрос может быть реализован несколькими способами, которые могут существенно отличаться по времени исполнения.

Формат оператора SELECT:

```
SELECT [DISTINCT | ALL]   * | <список полей>  FROM <список  таблиц>
[WHERE <предикат-условие выборки или соединения>]
[GROUP BY <список полей результата>]
[HAVING <предикат-условие для группы>]
[ORDER BY <список полей, по которым требуется упорядочить вывод>]
```

Указанный порядок следования фраз в операторе SELECT не может быть изменен, но не все его части являются обязательными. К обязательным предложениям относятся только фразы SELECT и FROM. Все остальные части оператора могут быть использованы по усмотрению программиста. Пояснение:

□ Фраза **SELECT**:

• наличие ключевого слова **ALL** (по умолчанию) означает, что в результирующую таблицу включаются все строки, удовлетворяющие условиям запроса, что может привести к появлению в результирующей таблице одинаковых строк;

• ключевое слово **DISTINCT** предназначено для приведения таблицы в соответствие с принципами теории отношений, где предполагается отсутствие дубликатов строк;

• символ "*" определяет очень часто встречаемую ситуацию, когда в результирующий набор включаются все столбцы из исходной таблицы запроса.

□ Во фразе **FROM** задается перечень исходных таблиц запроса.

□ Во фразе **WHERE** определяются условия отбора строк результата или условия соединения строк исходных таблиц, подобно операции условного соединения в реляционной алгебре. В качестве условий отбора могут быть использованы следующие предикаты:

• сравнения "= , <>, >, <, >=, <=" — для сравнения результатов вычисления двух выражений; более сложные выражения строятся с помощью логических операторов AND, OR, NOT; значения выражений вычисляются в порядке, который определяется приоритетом используемых операторов и наличием скобок в выражении;

• **BETWEEN A AND B** — предикат истинен, когда вычисленное значение выражения попадает в заданный диапазон (предикат **NOT BETWEEN A AND B** истинен тогда, когда сравниваемое значение не попадает в заданный интервал);

• **IN** — предикат истинен тогда, когда сравниваемое значение входит в множество заданных значений; при этом множество значений может быть задано простым перечислением или встроенным подзапросом (предикат NOT IN истинен тогда, когда сравниваемое значение не входит в заданное множество);

• **LIKE** и **NOT LIKE**— предикаты, смысл которых противоположен, требуют задания шаблона, с которым сравнивается заданное значение; предикат LIKE истинен тогда, когда сравниваемое значение соответствует шаблону, и ложен в противном случае;

• **IS NULL** — предикат, применяющийся для выявления равенства значения некоторого атрибута неопределенному значению:

  - **<имя атрибута> IS NULL** — принимает значение TRUE, если в данной строке указанный атрибут имеет неопределенное значение и значение FALSE, в противном случае;
  - **<имя атрибута> IS NOT NULL** — все происходит наоборот.

• **EXIST** и **NOT EXIST**, используемые во встроенных подзапросах.

□ Во фразе **GROUP BY** задается список полей группировки.

□ Во фразе **HAVING** задаются предикаты-условия, накладываемые на каждую группу.

□ Во фразе **ORDER BY** задается список полей упорядочения результата, то есть список полей, который определяет порядок сортировки в результирующей таблице.

В стандарте SQL определено понятие NULL-значения, которое вызвало необходимость применения трехзначной логики, где все логические операции выполняются в соответствии с приведенной ниже таблицей истинности (табл.6).

Таблица 6. *Таблица истинности*

A	B	Not A	A AND B	A OR B
TRUE	TRUE	FALSE	TRUE	TRUE
TRUE	FALSE	FALSE	FALSE	TRUE
TRUE	NOLL	FALSE	NULL	TRUE
FALSE	TRUE	TRUE	FALSE	TRUE
FALSE	FALSE	TRUE	FALSE	FALSE
FALSE	NULL	TRUE	FALSE	NULL
MULL	TRUE	NULL	NULL	TRUE
NULL	FALSE	NULL	FALSE	NULL
NULL	NULL	NULL	NULL	NULL

1.1. Простые запросы

Запрос 1

Вывести сведения о кафедрах университета.

Данная задача сводится к выборке и выводу информации из одной таблицы, причем выводу подлежат все ее строки и все ее столбцы:

SELECT * FROM kafedra

Результатом выполнения такого запроса будет являться таблица, содержащая сведения обо всех кафедрах университета:

Kod kaf	Name kaf	Nom_telef	Nom_Auditoria	Col_sotr	Zav_kaf
001	Физики	23-34-24	132	25	Иванов Т.М.
002	Общей математики	23-65-43	003	22	Махов К Л.
003	Истории	23-78-72	465	16	Росс Л.Т.
004	Графики	23-99-77	385	18	Фирсов С.С.
005	Прикладной математики	23-66-62	028	24	Ляхова И.Т.

Запрос 2

Вывести номера телефонов кафедр университета.

Результат такого запроса должен содержать только два столбца: **Name_kaf** и **Nom_telef,** поэтому сам запрос должен выглядеть следующим образом:

SELECT Name_kaf, Nom_telef FROM kafedra

Результирующая таблица:

Name kaf	Nomjelef
Физики	23-34-24
Общей математики	23-65-43
Истории	23-78-72
Графики	23-99-77
Прикладной математики	23-66-62

В сформированных выше запросах требовалось вывести все строки таблицы, указанной в предложении FROM. Если при выборке требуется ограничить количество выводимых строк в соответствии с каким-то условием, то этого можно достичь, используя в запросе предложение WHERE. В предложение WHERE можно включить одно или несколько условий отбора строк.

Запрос 3

Вывести сведения о кафедре графики.

Запрос будет выглядеть следующим образом:

SELECT * FROM kafedra WHERE Name_kaf = 'Графики'

Ответ на такой запрос будет содержать только одну строку:

Kod.kaf	Name_kaf	Nomjelef	Nom_Auditoria	Col_sotr	Zav_kaf
004	Графики	23-99-77	385	18	Фирсов С.С.

Запрос 4

Вывести сведения о кафедрах университета, находящихся на первом этаже, учитывая тот факт, что номера аудиторий первого этажа лежат в диапазоне от 1 до 99.

Запрос будет выглядеть следующим образом:

SELECT * FROM kafedra WHERE Nom_Auditoria BETWEEN 1 AND 99

Результат запроса:

Kodjcaf	Name_kaf	Norn lelef	Norn Audit oria	Coi_sotr	Zavkaf
002	Общей математики	23-65-43	003	22	Махов К.Л.
005	Прикладной математики	23-66-62	028	24	Ляхова И.Т.

В общем случае строки в результирующей таблице выводятся в неупорядоченном каким-либо образом состоянии. Просматривать и анализировать такой материал не всегда удобно. Для сортировки строк по какому-либо столбцу применяется фраза ORDER BY. Она включает список разделенных запятыми наименований столбцов, по которым требуется упорядочить выводимую информацию. Данная фраза должна всегда располагаться последней в операторе SELECT и при ее наличии появляется возможность отсортировать строки по возрастанию (ASC) или убыванию (DESC) значений указанного столбца или комбинации указанных столбцов, независимо о" того, присутствуют эти столбцы в результирующей таблице или нет.

Запрос 5

Вывести сведения о кафедрах университета в виде, отсортированном т столбцу Name_kaf в порядке возрастания.

Запрос будет выглядеть следующим образом:

```
SELECT   * FROM kafedra ORDER BY Name_caf ASC
```

Результат данного запроса:

Kod_kaf	Name_kaf	Nomjelef	Nom_Auditoria	Col_sotr	Zav kaf
004	Графики	23-Э9-77	385	18	ФирсовС.С.
003	Истории	23-78-72	465	16	Росс Л.Т.
002	Общей математики	23-65-43	003	22	Махов К.Л.
005	Прикладной математики	23-66-62	028	24	Ляхова И.Т.
001	Физики	23-34-24	132	25	Иванов Т.М.

Часто для улучшения наглядности выводимую информацию полезно отсортировать по нескольким столбцам. Для этого имена столбцов сортировки необходимо перечислить через запятую во фразе ORDER BY. При этом выводимая таблица будет содержать строки, упорядоченные по первому указанному во фразе ORDER BY столбцу, а строки, имеющие равные значения в этом столбце, будут упорядочены по значениям второго столбца и т. д. слева направо.

1.2. Агрегатные функции языка

В стандарте языка SQL определено несколько агрегатных функций:

- **COUNT** —- возвращает количество значений в указанном столбце;
- **SUM** — возвращает сумму значений в указанном столбце;
- **AVG** — возвращает усредненное значение в указанном столбце;
- **MIN** — возвращает минимальное значение в указанном столбце;
- **MAX** — возвращает максимальное значение в указанном столбце.

В качестве операнда данных функций может использоваться наименование только одного столбца, и все они возвращают единственное значение, С функциями SUM и AVG могут использоваться только числовые поля. С функциями COUNT, MAX и MIN могут использоваться как числовые, так и символьные поля. При вызове всех перечисленных выше функций, кроме функции COUNT(*)., осуществляется исключение всех пустых значений и только после этого операция применяется к оставшимся значениям столбца, Функция COUNT(*) призвана осуществлять подсчет всех строк таблицы независимо от того, какие значения в них находятся.

Агрегатные функции нельзя использовать в предложении WHERE, потому что предикаты оцениваются в терминах одиночной строки, а агрегатные функции — в терминах групп строк.

Запрос 6

Подсчитать и вывести общее число кафедр университета:

```
SELECT  COUNT (*) AS count FROM kafedra
```

Ответ на данный запрос:

```
Count
   5
```

Запрос 7

Определить среднее число сотрудников, работающих на кафедрах университет:

```
SELECT AVG(Col_sotr) AS avg  FROM  kafedra
```

Ответ на запрос:

```
avg
21
```

1.3. Группирование результатов

Выбираемые пользователем данные могут быть подвергнуты различного рода анализу и обобщению, и язык SQL имеет для этого свои средства. Только что рассмотренные агрегатные функции являются примером таких средств. В приведенных выше запросах агрегатные функции применялись ко всей таблице и выдавали сводные данные на основании обработки значений всего выделенного столбца. Такие результаты обычно размещаются в конце отчета и сжимаются в итоговую строку.

Однако часто встречаются ситуации, когда в отчет необходимо поместить и промежуточные результаты, опирающиеся на вычисления обобщенных групповых значений. Для применения агрегатных функций в подобных случаях предполагается предварительная операция группировки. Суть операции группировки состоит в том, что все множество строк таблицы разбивается на группы, в каждой из которых собираются строки, имеющие одинаковые

значении атрибутов, которые заданы в списке группировки. Обработка такой информации реализуется путем применения агрегатных функций уже к каждой отдельной группе и выдаче полученных итогов.

В языке SQL для осуществления операции группировки в оператор SELECT включается фраза GROUP BY. Запрос, в котором присутствует фраза GROUP BY, называется *группирующим запросом,* а столбцы, перечисленные в этой фразе, называются *группирующими столбцами.*

Стандарт языка требует, чтобы предложение SELECT и фраза GROUP BY были тесно связаны между собой. Если в запросе должна использоваться группировка, то каждый элемент списка в предложении SELECT должен иметь единственное значение для всей группы. К тому же в предложении SELECT могут включаться в этом случае только следующие типы данных:

- имена столбцов;
- агрегатные функции;
- константы;
- выражения, состоящие из перечисленных выше элементов.

Таким образом, предложение GROUP BY позволяет определять подмножество значений в особом поле в терминах другого поля и применять функцию агрегата к подмножеству. Это дает возможность объединять поля и агрегатные функции в едином предложении SELECT. В дальнейшем в качестве примера будем работать с двумя БД: НИР и Сессия.

БД НИР состоит из одной таблицы *R,* в которой хранится информация о производимых выплатах специалистам за проделанную работу по определенным этапам НИР: R = (ФИО, Этап, Начисления).

Пусть таблица содержит следующие данные.

R

ФИО	Этап	Начисления (руб)
Семенов Т.Т.	Этап_1	1000
Просов СМ.	Этап_1	2000
Мехова И.И.	Этап_1	500
Семенов Т.Т.	Этап_2	500
Просов СМ.	Этап_2	500
Мехова И.И.	Этап_2	1000
Просов СМ.	Этап_3	1000
Мехова И.И.	Этап_3	1000
Немцов Я.Ю.	Этап_3	2000
Немцов Я.Ю.	Этап_4	2000
Яров И.М.	Этап_4	3000

БД Сессия включает в себя сводную таблицу, где представлены экзаменационные оценки студентов, полученные ими в сессию по определенным дисциплинам:

***S* = (ФИО, Дисциплина, Оценка);**

ФИО	Дисциплина	Оценка
Мур С.М.	Физика	4
Цуканов Т.Т.	Физика	5
Думская М.Т.	Физика	3
Дрозд Г. Р.	Физика	4
МурС.М.	История	4
Цуканов Т.Т.	История	5
Думская М.Т.	История	3
Цуканов Т.Т.	Математика	5
Думская М.Т.	Математика	4
Дрозд Г. Р.	Математика	5
Петрова СО.	Электротехника	5
Часов И.И.	Электротехника	4
Иванова Я.С.	Электротехника	5
КрисссР.О.	Электротехника	3
Часов И.И.	Иностр_яэык	5
Иванова Я.С.	Иностр_яэык	4
Часов И.И.	Экономика	4
Иванова Я.С.	Экономика	4
Крисе Р.О.	Экономика	5
Фирсова Л.Р.	Экономика	3

Запрос 8

БД НИР. Для каждого специалиста определить сумму, выплаченную за работу по данной теме, и количество сделанных ему выплат.

Для формирования запроса включим в предложение SELECT следующую информацию: **ФИО, COUNT (Начисления) AS count, SUM (Начисления)** AS **sum,** где в качестве имен для двух вычисляемых столбцов используются

псевдонимы. Группировку будем производить по столбцу ФИО. Для того чтобы проще было просматривать результаты, выводимые данные представим в отсортированном по столбцу ФИО виде:

SELECT ФИО AS ФИО_сотрудника, COUNT (*) AS Кол_вып_этапов, SUM (Начисления) AS Начисления FROM R GROUP BY ФИО ORDER BY ФИО

Результат запроса:

ФИО_сотрудника	Кол_вып_этапов	Начисления
Мехова И.И.	3	2500
Просов СМ.	3	3500
Семенов Т.Т.	2	1500
Немцов Я.Ю.	2	4000
Яров И.М.	1	3000

Для того чтобы сформировать результаты в таком виде, все строки были сгруппированы по атрибуту ФИО, в каждой группе было подсчитано количество строк и сумма начислений.

Запрос 9

БД Сессия. Для каждой дисциплины определить количество студентов, сдавших экзамен.

SELECT Дисциплина, COUNT (*) AS Количество_студентов FROM S GROUP BY Дисциплина ORDER BY Дисциплина

Результат запроса:

Дисциплина	Количество_студентов
Иностр_язык	2
История	3
Математика	3
Физика	4
Экономика	4
Электротехника	4

Поскольку в таблице отсутствуют неопределенные значения, т. е. в ней находятся сведения только о студентах, успешно сдавших экзамен, то в запрос была включена функция COUNT (*), у которой аргумент представлен символом "*", определяющем в данном случае подсчет всех строк в группе.

Совместно с фразой GROUP BY может быть использована фраза HAVING, предназначенная для задания ограничений отбора групп, которые будут помешены в результирующую таблицу запроса. Стандарт языка требует, чтобы имена столбцов во фразе HAVING обязательно присутствовали в списке фразы GROUP BY или применялись в агрегатных функциях.

В частности, если раздел HAVING присутствует в табличном выражении, не содержащем GROUP BY, ТО результатом его выполнения будет либо пустая таблица, либо результат выполнения предыдущих разделов табличного выражения, рассматриваемый как одна группа без столбцов группирования.

Агрегатные функции могут применяться не только в выражении вывода результатов строки SELECT, но и в выражении условия обработки сформированных групп HAVING. В этом случае каждая агрегатная функция вычисляется для каждой выделенной группы. Значения, полученные при вычислении агрегатных функций, могут быть использованы для вывода соответствующих результатов или для условия отбора групп.

В результат можно включить значение поля группировки и несколько агрегатных функций, а в условиях группировки можно использовать несколько полей. При этом группы образуются по набору заданных полей группировки. Операции с агрегатными функциями могут быть применены к объединению множества исходных таблиц.

Запрос 10

БД НИР. В условиях предыдущего запроса вывести информацию, касающуюся только тех специалистов, которым производились начисления более одного раза.

Для вывода такой информации в текст предыдущего запроса необходимо добавить фразу HAVING COUNT (Начисления) > 1:

SELECT ФИО AS ФИО_сотрудника, COUNT (*) AS Кол_вып_этапов, SUM (Начисления) AS Начисления FROM R GROUP BY ФИО HAVING COUNT(Начисления) > 1 ORDER BY ФИО

Результаты запроса:

ФИО_сотрудника	Кол_вып_этапов	Начисления
Мехова И.И.	3	2500
Просов СМ.	3	3500
Семенов Т.Т.	2	1500

Немцов Я.Ю.	2	4000

Запрос11

БД Сессия. В условиях предыдущего запроса к данной базе вывести информацию, касающуюся только тех дисциплин, количество сдавших экзамен по которым превосходит три:

SELECT Дисциплина, COUNT (*) AS Количество_студентов FROM S GROUP BY Дисциплина HAVING COUNT(Дисциплина)> 3 ORDER BY Дисциплина

Результат запроса:

Дисциплина	Количество_студентов
Физика	4
Электротехника	4
Экономика	4

1.4. Вложенные запросы

Стандарт языка позволяет в тело одного оператора SELECT внедрять другой оператор SELECT. В такой ситуации можно рассматривать внешний и внутренний (внедряемый) операторы запроса. Обычно внутренний запрос генерирует значение, которое проверяется в предикате внешнего запроса (в предложении WHERE ИЛИ HAVING), определяющего, верно оно или нет. Если внутренний оператор запроса помещен в предложения WHERE И HAVING внешнего оператора SELECT, ТО создается ситуация вложенных запросов (подзапросов).

В сочетании с другими возможностями оператора выбора, такими как группировка, возможность вкладывать запросы внутрь друг друга представляет собой мощное средство для достижения нужного результата. Во фразе FROM оператора SELECT, если при формировании запроса требуется более чем один экземпляр некоторой таблицы, допустимо применять синонимы к именам таблицы. СИНОНИМЫ (АЛИАСЫ) задаются с использованием ключевого слова AS:

FROM Rl AS A, R2 AS B.

Подзапросы могут быть нескольких видов:

- скалярный подзапрос, возвращающий единственное значение;
- , возвращающий несколько значений в виде одной строки;
- табличный подзапрос, возвращающий данные в виде таблицы.

Подзапрос может указываться непосредственно после операторов сравнения (!=, <, >, <=, >=) и его текст должен быть заключен в скобки.

Запрос 12

БД НИР. Вывести список платежей, где величина единовременных начислений превысила среднее значение.

Стандарт языка не позволяет применять агрегатные функции в предложении WHERE, поэтому единовременные начисления необходимо сравнивать с результатом подзапроса, вычисляющего среднее значение начислений всех строк таблицы. Данный скалярный подзапрос вернет значение, равное 1208.33, и уже с этим значением будут сравниваться все начисления и отбираться для помещения в таблицу вывода.

Запрос будет выглядеть следующим образом:

SELECT ФИО, Этап, Начисления FROM R WHERE Начисления > (SELECT avg(Начисления) FROM R)

Результат запроса:

ФИО	Этап	Начисления(руб)
Просов См.	Этап_1	2000
Немцов Я.Ю.	Этап_3	2000
Немцов Я.Ю.	Этап_4	2000
Яров И.М.	Этап_4	2000

Реализация вложенных запросов требует соблюдения определенных правил и ограничений:

- в подзапросах не должна использоваться фраза ORDER BY.
- список в предложении SELECT может включать только имена столбцов и составленные из них выражения.
- по умолчанию имена столбцов в подзапросе относятся к таблице, указанной во фразе FROM.
- подзапрос, участвующий в операции, может быть только правым операндом.

В стандарте SQL2 операторы сравнения расширены до многократных сравнений с использованием ключевых слов **ANY** и **ALL**. ЭТО расширение используется при сравнении значения определенного столбца со столбцом данных, возвращаемым вложенным запросом.

Ключевое слово ANY, поставленное в любом предикате сравнения, означает, что предикат будет истинен, если хотя бы для одного значения из подзапроса предикат сравнения истинен. Ключевое слово ALL требует, чтобы предикат сравнения был бы истинен при сравнении со всеми строками подзапроса.

Запрос 13

БД НИР. Определить начисления, которые превышают начисления хотя бы одного специалиста, выполнявшего Этап_1.

Подобный запрос может быть реализован разными путями. Например, можно организовать подзапрос для нахождения минимальных начислений по Этапу_1, а затем во внешнем запросе путем сравнения с полученным значением вывести те начисления, которые превосходят это значение.

Как будет выглядеть запрос с использованием ключевого слова ANY: в этом случае внутренний запрос должен сформировать набор числовых значений начислений для ситуации, где Этап='Этап_1'. Внешний же запрос выберет сведения о начислениях, больших любого из значений в этом наборе:

```
SELECT ФИО, Этап, Начисления FROM R WHERE Начисления > ANY (SELECT Начисления
FROM R
WHERE Этап = 'Этап_1')
```

Результат запроса:

ФИО	Этап	Начисления (руб)
Семенов Т.Т.	Этап_1	1000
Просов СМ.	Этап_1	2000
МеховаИ.И.	Этап_2	1000
Просов СМ.	Этап_3	1000
МеховаИ.И.	Этап_3	1000
Немцов Я.Ю.	Этап_3	2000
Немцов Я.Ю.	Этап_4	2000
Яров И.М.	Этап_4	3000

Запрос 14

БД НИР. Определить начисления, которые превышают начисления любого специалиста, выполнявшего Этап_:.

```
SELECT ФИО,  Этап,  Начисления FROM R WHERE Начисления  > ALL (SELECT  Начисления FROM R WHERE  Этап =
'Этап_1');
```

В данном случае результат запроса содержит только одну строку, поскольку только эта выплата больше 2 000. Именно такое значение имеют наибольшие начисления по первому этапу.

ФИО	Этап	Начисления(руб)
Яров И.М.	Этап_4	3000

Запрос 15

БД Сессия. Найти студентов, которые сдали все экзамены на оценку не ниже чем «хорошо»:

```
SELECT  ФИО FROM S  WHERE  4  >=  ALL  (SELECT Оценка FROM S AS SI WHERE S.ФИО =S1.ФИО)
```

Результат запроса:

ФИО
Мур СМ.
Цуканов Т.Т.
Дрозд Г.Р.
Петрова СО.
Часов И.И.
Иванова Я.С.

Здесь также внутренний запрос формирует набор оценок определенного студента, которые сравниваются со значением 4. В том случае, если все оценки равны или превосходят данное значение, ФИО студента помещается в таблицу вывода.

Совместно с подзапросами могут быть использованы специально предназначенные для этого предикаты EXISTS И NOT EXISTS. Результат их обработки имеет значение TRUE ИЛИ FALSE.

Для ключевого слова EXISTS результат равен TRUE В ТОМ И ТОЛЬКО В ТОМ случае, если в возвращаемой подзапросом таблице присутствует хотя бы одна строка. Если результирующая таблица подзапроса пуста, результат обработки этого ключевого слова будет иметь значение FALSE.

Для ключевого слова NOT EXISTS наблюдается полностью противоположная ситуация. Он возвращает истину, если вывод подзапроса пуст, и ложь, если вывод подзапроса не пуст.

Возможности использования данных предикатов будут проиллюстрированы ниже при обсуждении многотабличных запросов.

1.5. Многотабличные запросы

При работе с базами данных потребности пользователей не ограничиваются только реализацией простых запросов данных из одной таблицы. Во многих случаях для получения ответа на запрос необходимо объединить информацию из нескольких исходных таблиц. Для того чтобы осуществить такое объединение в результирующей таблице, необходимо выполнить операцию соединения, при которой объединение информации из двух таблиц происходит посредством образования пар связанных строк, выбранных из каждой таблицы. Таблицам можно присвоить имена-псевдонимы, что бывает полезно для осуществления операции соединения таблицы с самой собою и в ряде других ситуаций.

Если в операторе SELECT указано более одного имени таблицы, неявно подразумевается, что над перечисленными таблицами осуществляется операция декартова произведения. Самый простой запрос SELECT такого рода без необязательных частей выглядит следующим образом:

SELECT * FROM Rl, R2;

и соответствует декартову произведению таблиц *R1* и *R2*. Выражение

SELECT Rl.A, R2.B FROM El, R2;

соответствует проекции декартова произведения двух таблиц на два столбца *A* из таблицы *RI* и *B* из таблицы *R2*.

Рассмотрим базу данных, в которой хранится информация о производимых выплатах специалистам за проделанную работу по определенным этапам НИР. Пусть она состоит из трех отношений *Rl, R2* и *R3*. Будем считать, что они представлены таблицами *RJ, R2* и *R3* соответственно.

***R1*= (ФИО, Отдел);**
***R2*= (Отдел, Этап);**
***R3* = (ФИО, Этап, Начисления).**

R1

ФИО	Отдел
Семенов Т.Т.	03
Просов СМ.	03
Мехова И.И.	03
Немцов Я.Ю.	04
Яров И.М.	04

R2

Отдел	Этап
03	Этап_1
03	Этап_2
03	Этап_3
04	Этап_3
04	Этап_4

R3

ФИО	Этап	Начисления(руб)
Семенов ТТ.	Этап_1	1000
Просов СМ.	Этап_1	2000
Мехова И.И.	Этал_1	500
Семенов Т.Т.	Этап_2	500
Просов СМ.	Этап_2	500
Мехова И.И.	Этап_2	1000
Просов СМ.	Этап_3	1000
Мехова И.И.	Этап_3	1000
Немцов Я.Ю.	Этап_3	2000
Чемцов Я.Ю.	Этап_4	2000
Яров И.М.	Этап_4	3000

Для запросов будет использоваться и расширенная база данных Сессия, представленная таблицами:

***S1* = (ФИО, Дисциплина, Оценка)** — содержащей сведения о результатах сессии;
***S2*= (ФИО, Группа)** — содержащей сведения о составе групп;
***S3* = (Группа, Дисциплина)** — содержащей перечень экзаменов, подлежащих сдаче.

S1

ФИО	Дисциплина	Оценка
Мур СМ.	Физика	**4**
Цуканов Т.Т.	Физика	**5**
Думская М.Т.	Физика	**3**
Дрозд Г. Р.	Физика	**4**
Мур СМ.	История	**4**
Цуканов ТТ.	История	**5**
Думская М.Т.	История	**3**
Цуканов Т.Т.	Математика	**5**
Думская М.Т.	Математика	**4**

Дрозд Г. Р.	Математика	**5**
Петрова СО.	Экономика	**5**
Часов И.И.	Электротехника	**4**
Иванова Я.С	Электротехника	**5**
Крисе Р.О.	Электротехника	**3**
Часов И.И.	Иностр_яэык	**5**
Иванова Я.С.	Иностр_язык	**4**
Часов И. И.	Экономика	**4**
Иванова Я.С.	Экономика	**4**
Крисе Р.О.	Экономика	**5**
Фирсова Л. Р.	Экономика	**3**

S2

ФИО	**Группа**
Мур СМ.	02-КТ-21
Цуканов Т.Т.	02-КТ-21
Думская М.Т.	02-КТ-21
Дрозд Г. Р.	02-КТ-21
Петрова С.О.	02-КТ-12
Часов И.И.	02-КТ-12
Иванова Я.С.	02-КТ-12
Крисс Р.О.	02-КТ-12
Фирсова Л.Р.	02-КТ-12

S3

Группа	**Дисциплина**
02-КТ-21	Физика
02-КТ-21	История
02-КТ-21	Математика
02-КТ-12	Экономика
02-КТ-12	Электротехника
02-КТ-12	Инострязык

Запрос 16

БД НИР. Вывести список сотрудников отдела 03, которые участвовали в выполнении Этапа_3:

```
SELECT R3 .ФИО,   R3.Этап FROM RI,   R3
WHERE  RI.Отдел =  '03'   AND
R1.ФИО = R3.ФИО AND
R3.Этап =  'Этап_3'
```

Результат запроса:

ФИО	**Этап**
Просов С.М. .	Этап_3
Мехова И.И.	Этап_3

Запрос 17

Вывести группы, в которых по одной дисциплине на экзаменах получено больше одной пятерки:

```
SELECT  S2.Группа FROM SI,   S2
WHERE   SI.ФИО =  S2.ФИО AND
S1.Оценка = 5
GROUP BY 52.Группа,   SI.Дисциплина HAVING   count (*) > 1
```

Результатом выполнения раздела HAVING является сгруппированная таблица, содержащая только те группы строк, для которых результат вычисления условия поиска есть TRUE.

Результат запроса:

Группа
02-КТ-21
02-КТ-12

Пример запроса, где будет использован предикат NOT EXISTS. Его возможности удобно проиллюстрировать в тексте многотабличного запроса.

Запрос 18

Вывести список тех студентов, кто должен был сдавать экзамен по истории, но пока еще не сдавал:

```
SELECT ФИО  FROM S2,   S3
WHERE S2.ґруппа=S3.Группа AND
Дисциплина =   'История'   AND NOT EXISTS   (SELECT  ФИО FROM SI
WHERE ФИО = S2.ФИО AND Дисциплина =  'История')
```

Результат запроса:

ФИО
Дрозд Г. Р.

Предикат EXISTS истинен, когда подзапрос не пуст, то есть содержит хотя бы один кортеж, в противном случае предикат EXISTS ложен. Предикат NOT EXISTS — истинен только тогда, когда подзапрос пуст.

Обработка такого запроса состоит в том, что для каждого студента, обучающегося в группе, студентам которой необходимо сдавать экзамен по истории, проверяется истинность предиката NOT EXISTS. Истинность его устанавливается по факту присутствия во внутреннем запросе значений. Если подзапрос пуст, то данный студент еще не сдавал экзамен по истории, предикат NOT EXISTS имеет значение TRUE, и ФИО студента помещается в результирующую таблицу вывода.