Оператор выбора SELECT

Один и тот же *запрос* может быть реализован несколькими способами, и, будучи все правильными, они, тем не менее, могут существенно отличаться *по* времени исполнения, и это особенно важно для больших баз данных.

Синтаксис оператора SELECT имеет следующий вид:

SELECT[ALL|DISTINCT](<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 упорядочить вывод>]

Здесь *ключевое слово* ALL означает, что в результирующий набор строк включаются все строки, удовлетворяющие условиям запроса. Значит, в результирующий набор могут попасть одинаковые строки. И это нарушение принципов теории отношений (в отличие от *реляционной алгебры*, где *по* умолчанию предполагается отсутствие дубликатов в каждом результирующем отношении). *Ключевое слово* DISTINCTозначает, что в результирующий набор включаются только различные строки, то есть дубликаты строк результата не включаются в набор.

Символ *. (звездочка) означает, что в результирующий набор включаются все столбцы из исходных таблиц запроса.

В разделе FROM задается перечень исходных отношений (таблиц) запроса.

В разделе WHERE задаются условия отбора строк результата или условия соединения кортежей исходных таблиц, подобно *операции* условного соединения в *реляционной алгебре*.

В разделе GROUP BY задается *список* полей группировки.

В разделе HAVING задаются предикаты-условия, накладываемые на каждую группу.

В части ORDER BY задается *список* полей упорядочения результата, то есть *список* полей, который определяет порядок сортировки в результирующем отношении. Например, если первым полем списка будет указана Фамилия, а вторым Номер группы, то в результирующем отношении сначала будут собраны в алфавитном порядке студенты, и если найдутся

однофамильцы, то они будут расположены в порядке возрастания номеров групп.

В выражении условий раздела WHERE могут быть использованы следующие предикаты:

- *Предикаты сравнения* { =, <>, >,<, >=,<= },которые имеют традиционный смысл.
- Предикат Between A and B —принимает значения между A и B. Предикат истинен, когда сравниваемое значение попадает в заданный диапазон, включая границы диапазона. Одновременно в стандарте задан и противоположный предикат Not Between A and B,который истинен тогда, когда сравниваемое значение не попадает в заданный интервал, включая его границы.
- Предикат вхождения в множество IN (множество) истинен тогда, когда сравниваемое значение входит в множество заданных значений. При этом множество значений может быть задано простым перечислением или встроенным подзапросом. Одновременно существует противоположный предикат NOT IN (множество), который истинен тогда, когда сравниваемое значение не входит в заданное множество.
- *Предикаты сравнения* с образцом LIKE и NOT LIKE. Предикат LIKE требует задания шаблона, с которым сравнивается заданное значение, предикат истинен, если сравниваемое значение соответствует шаблону, и ложен в противном случае. Предикат NOT LIKEимеет противоположный смысл.

  По стандарту в шаблон могут быть включены специальные символы:

- символ подчеркивания (_) — для обозначения любого одиночного символа;
- символ процента (%) — для обозначения любой произвольной последовательности символов;
- остальные символы, заданные в шаблоне, обозначают самих себя.
- *Предикат сравнения* с неопределенным значением IS NULL. Понятие неопределенного значения было внесено в концепции баз данных позднее. Неопределенное значение интерпретируется в реляционной модели как значение, неизвестное на данный момент времени. Это значение при появлении дополнительной информации в любой момент времени может быть заменено на некоторое конкретное значение. При сравнении неопределенных значений не действуют стандартные правила сравнения: одно неопределенное значение никогда не считается равным другому неопределенному значению. Для выявления равенства значения

некоторого атрибута неопределенному применяют специальные стандартные предикаты:

<имя атрибута>IS NULL и <имя атрибута> IS NOT NULL.

Если в данном кортеже (в данной строке) указанный атрибут имеет неопределенное значение, то предикат IS NULL принимает значение "Истина" (TRUE), а предикат IS NOT NULL — "Ложь" (FALSE), в противном случае предикат IS NULL принимает значение "Ложь", а предикат IS NOT NULL принимает значение "Истина".

Введение Null-значений вызвало необходимость модификации классической двузначной логики и превращения ее в трехзначную. Все логические операции, производимые с неопределенными значениями, подчиняются этой логике в соответствии с заданной таблицей истинности:

A	**B**	**Not A**	$A \wedge B$	$A \vee B$
TRUE	TRUE	FALSE	TRUE	TRUE
TRUE	FALSE	FALSE	FALSE	TRUE
TRUE	Null	FALSE	Null	TRUE
FALSE	TRUE	TRUE	FALSE	TRUE
FALSE	FALSE	TRUE	FALSE	FALSE
FALSE	Null	TRUE	FALSE	Null
Null	TRUE	Null	Null	TRUE
Null	FALSE	Null	FALSE	Null
Null	Null	Null	Null	Null

- Предикаты существования EXISTS и несуществования NOT EXISTS. Эти предикаты относятся к встроенным подзапросам, и подробнее мы рассмотрим их, когда коснемся вложенных подзапросов.

В условиях поиска могут быть использованы все рассмотренные ранее предикаты.

Отложив на время знакомство с группировкой, рассмотрим детально первые три строки оператора SELECT:

- SELECT — ключевое слово, которое сообщает СУБД, что эта команда — запрос. Все запросы начинаются этим словом с последующим пробелом. За ним может следовать способ выборки — с удалением дубликатов (DISTINCT) или без удаления (ALL, подразумевается по умолчанию). Затем следует список перечисленных через запятую столбцов, которые выбираются запросом из таблиц, или символ '*'(звездочка) для выбора всей строки. Любые столбцы, не перечисленные здесь, не будут включены в

результирующее отношение, соответствующее выполнению команды. Это, конечно, не значит, что они будут удалены или их информация будет стерта из таблиц, потому что запрос не воздействует на информацию в таблицах — он только показывает данные.

- FROM — ключевое слово, подобно SELECT, которое должно быть представлено в каждом запросе. Оно сопровождается пробелом и затем именами таблиц, используемых в качестве источника информации. В случае если указано более одного имени таблицы, неявно подразумевается, что над перечисленными таблицами осуществляется операция декартова произведения. Таблицам можно присвоить имена-псевдонимы, что бывает полезно для осуществления операции соединения таблицы с самой собою или для доступа из вложенного подзапроса к текущей записи внешнего запроса (вложенные подзапросы здесь не рассматриваются).

  Все последующие разделы оператора SELECT являются необязательными.

  Самый простой запрос SELECT без необязательных частей соответствует просто декартову произведению. Например, выражение

  SELECT * FROM R1, R2

  соответствует декартову произведению таблиц R1 и R2. Выражение

  SELECT R1.A, R2.B FROM R1, R2

  соответствует проекции декартова произведения двух таблиц на два столбца A из таблицы R1 и B из таблицы R2, при этом дубликаты всех строк сохранены, в отличие от операции проектирования в *реляционной алгебре*, где при проектировании по умолчанию все дубликаты кортежей уничтожаются.

- WHERE — ключевое слово, за которым следует предикат — условие, налагаемое -на запись в таблице, которому она должна удовлетворять, чтобы попасть в выборку, аналогично операции селекции в *реляционной алгебре*.

Рассмотрим базу данных, которая моделирует сдачу сессии в некотором учебном заведении. Пусть она состоит из трех отношений R1, R2, R3. Будем считать, что они представлены таблицами R1, R2 и R3 соответственно.

R1 = (ФИО, Дисциплина, Оценка) ; R2 = (ФИО, Группа) ; R_3 = (Группы, Дисциплина)

R1

ФИО	Дисциплина	Оценка
Петров Ф. И.	Базы данных	5
Сидоров К. А.	Базы данных	4
Миронов А. В.	Базы данных	2
Степанова К. Е.	Базы данных	2
Крылова Т. С.	Базы данных	5
Сидоров К. А.	Теория информации	4
Степанова К. Е.	Теория информации	2
Крылова Т. С.	Теория информации	5
Миронов А. В.	Теория информации	Null
Владимиров В. А.	Базы данных	5
Трофимов П. А.	Сети и телекоммуникации	4
Иванова Е. А.	Сети и телекоммуникации	5
Уткина Н. В.	Сети и телекоммуникации	5
Владимиров В. А.	Английский язык	4
Трофимов П. А.	Английский язык	5
Иванова Е. А.	Английский язык	3
Петров Ф. И.	Английский язык	5

R2

ФИО	Группа
Петров Ф. И.	4906
Сидоров К. А.	4906
Миронов А. В.	4906
Крылова Т. С.	4906
Владимиров В. А.	4906
Степанова К. Е.	4906
Трофимов П. А.	4807
Иванова Е. А.	4807
Уткина Н. В.	4807

R3

Группа	Дисциплина
4906	Базы данных
4906	Теория информации
4906	Английский язык
4807	Английский язык
4807	Сети и телекоммуникации

Приведем несколько примеров использования оператора SELECT.

- Вывести список всех групп (без повторений), где должны пройти экзамены.

SELECT DISTINCT Группы FROM R3

Результат:

Группа
4906
4807

- Вывести список студентов, которые сдали экзамен по дисциплине "Базы данных" на "отлично".

```
SELECT ФИО
FROM R1
WHERE Дисциплина = "Базы данных" AND Оценка = 5
```

Результат:

ФИО
Петров Ф. И.
Крылова Т. С.
Владимирова В. А.

- Вывести список всех студентов, которым надо сдавать экзамены с указанием названий дисциплин, по которым должны проводиться эти экзамены.

```
SELECT ФИО,Дисциплина
FROM R2,R3
WHERE R2.Группа = R3.Группа;
```

Здесь часть WHERE задает условия *соединения отношений* R2 и R3, при отсутствии условий соединения в части WHERE результат будет эквивалентен расширенному декартову произведению, и в этом случае каждому студенту были бы приписаны все дисциплины из отношения R3, а не те, которые должна сдавать его группа.

Результат:

ФИО	Дисциплина
Петров Ф. И.	Базы данных
Сидоров К. А.	Базы данных
Миронов А. В.	Базы данных
Степанова К. Е.	Базы данных
Крылова Т. С.	Базы данных
Владимиров В. А.	Базы данных
Петров Ф. И.	Теория информации

Сидоров К. А.	Теория информации
Миронов А. В.	Теория информации
Степанова К. Е.	Теория информации
Крылова Т. С.	Теория информации
Владимиров В. А.	Теория информации
Петров Ф. И.	Английский язык
Сидоров К. А.	Английский язык
Миронов А. В.	Английский язык
Степанова К. Е.	Английский язык
Крылова Т. С.	Английский язык
Владимиров В. А.	Английский язык
Трофимов П. А.	Сети и телекоммуникации
Иванова Е. А.	Сети и телекоммуникации
Уткина Н. В.	Сети и телекоммуникации
Трофимов П. А.	Английский язык
Иванова Е. А.	Английский язык
Уткина Н. В.	Английский язык

- Вывести список лентяев, имеющих несколько двоек.
  - SELECT DISTINCT R1.ФИО
  - FROM R1 a, R1 b
  - WHERE a.ФИО = b.ФИО AND
  - a.Дисциплина <> b.Дисциплина AND
    a.Оценка <= 2 AND b.Оценка <= 2;

  Здесь мы использовали псевдонимы для именования отношения R1 a и b, так как для записи условий поиска нам необходимо работать сразу с двумя экземплярами данного отношения.

  Результат:

ФИО
Степанова К. Е.

Из этих примеров хорошо видно, что логика работы оператора выбора (*декартово произведение*—селекция—*проекция*) не совпадает с порядком описания в нем данных (сначала *список* полей для проекции, потом *список* таблиц для декартова произведения, потом условие соединения). Дело в том, что *SQL* изначально разрабатывался для применения конечными пользователями, и его стремились сделать возможно ближе к языку естественному, а не к языку алгоритмическому. *По* этой причине *SQL* на первых порах вызывает путаницу и раздражение у начинающих его изучать профессиональных

программистов, которые привыкли разговаривать с машиной именно на алгоритмических языках.

Наличие неопределенных (Null) значений повышает гибкость обработки информации, хранящейся в *БД*. В наших примерах мы можем предположить ситуацию, когда студент пришел на экзамен, но не сдавал его *по* некоторой причине, в этом случае оценка *по* некоторой дисциплине для данного студента имеет неопределенное *значение*. В данной ситуации можно поставить вопрос: "Найти студентов, пришедших на экзамен, но не сдававших его с указанием названия дисциплины". Оператор SELECT будет выглядеть следующим образом:

```
SELECT ФИО, Дисциплина
FROM R1
WHERE Оценка IS NULL
```

Результат:

ФИО	Дисциплина
Миронов А. В.	Теория информации