**Лабораторная работа №4**

**Средства языка SQL для выборки данных**

**Цель работы:** изучить основные принципы создания SQL-запросов для выборки данных из таблиц базы данных и представления данных в требуемом виде.

**Основные теоретические сведения**

Одной из основных задач, выполняемых на сервере баз данных, является выборка данных, их первичная обработка и представление в удобном виде. Для получения данных из таблиц базы данных используются SELECT-запросы, которые состоят из нескольких секций, большинство из которых могут отсутствовать.

SELECT-запросы в общем случае имеют следующий вид:

SELECT [TOP top\_N] [DISTINCT] field\_value[, field\_value, ...]

[FROM table\_source]

[WHERE condition]

[GROUP BY field\_name[, field\_name, ...]

[HAVING condition]]

[ORDER BY field\_name [sort\_type][, field\_name [sort\_type], ...]

Фактический порядок выполнения секций SELECT-запроса отличается от порядка их следования в тексте запроса.

**Секция FROM.** В секции FROM указывается таблица данных, из которых выбирается необходимая информация. Это может быть физическая таблица базы данных, соединение нескольких таблиц или табличное выражение (будет рассмотрено в разделе «Вложенные подзапросы»).

В языке SQL существуют операции соединения двух таблиц, которые имеют следующий вид:

table\_1 [[AS] alias\_1] [join\_type] table\_2 [[AS] alias\_2]

[ON condition]

где table\_1 и table\_2 — таблицы или табличные структуры, alias\_1 и alias\_2 — псевдонимы таблиц, которые будут использованы в текущем запросе вместо имени таблицы или в качестве имени табличного выражения, если они указаны, join\_type — тип соединения, condition — условие соединения таблиц. Результатом операции соединения двух таблиц является набор данных — табличная структура, содержащая все столбцы обеих таблиц.

Псевдонимы таблиц обязательны для табличных выражений, а также в случае, если выполняется операция соединения таблицы данных с этой же таблицей — для устранения неоднозначности их использования. В остальных случаях использование коротких имён в качестве псевдонимов позволяет сократить объём текста SQL-запроса.

Существуют различные типы соединения таблиц: CROSS JOIN (перекрёстное соединение), INNER JOIN (внутреннее соединение), LEFT JOIN или LEFT OUTER JOIN («левое» соединение), RIGHT JOIN или RIGHT OUTER JOIN («правое» соединение, FULL JOIN или FULL OUTER JOIN (полное соединение). Подробно о каждом типе соединения ознакомьтесь самостоятельно. Перекрёстное соединение является единственным типом соединения, для которого не задаётся секция ON. Для всех остальных типов условие соединения таблиц является обязательным.

В результате выполнения секции FROM создаётся временная таблица данных, в которой имена и типы столбцов соответствуют столбцам заданных таблиц и табличных структур, а строки представляют результирующий набор данных.

**Секция WHERE.** В секции WHERE указывается условие выборки данных. Каждая строка таблицы, полученной в секции FROM, проверяется на соответствие условию независимо от других строк. В результате выполнения секции WHERE из результирующего набора данных удаляются строки, для которых не выполняется условие выборки данных. Если секция WHERE не задана, все полученные на предыдущем этапе строки сохраняются.

**Секция GROUP BY.** Секция GROUP BY группирует выбранный набор данных для получения набора сводных строк по значениям одного или нескольких столбцов. В одну группу объединяются строки набора данных с одинаковыми значениями полей, столбцы которых перечислены в секции GROUP BY. В результирующий набор данных заносится по одной строке для каждой группы.

**Секция HAVING.** В секции HAVING указывается условие выборки результата группировки, указанной в секции GROUP BY. Каждая группа, проверяется на соответствие условию независимо от других групп. Условие секции HAVING обычно содержит агрегатные функции (будут рассмотрены в разделе «Агрегатные функции»). В результате выполнения секции HAVING из результирующего набора данных удаляются строки для групп, для которых не выполняется условие.

**Секция SELECT.** В секции SELECT содержится описание столбцов результирующей выборки данных в указанном порядке. Синтаксис описания столбца следующий:

column\_value [[AS] column\_name]

где column\_value — выражение, определяющее значение поля, column\_name — имя столбца.

Выражение, в соответствии с которым формируются значения поля столбца, может содержать константные значения, имена столбцов таблицы из секции FROM, вызовы функций, значения которых будут вычисляться для каждой записи, агрегатные функции. Если имя столбца содержится в нескольких таблицах или табличных выражениях в секции FROM, перед именем столбца обязательно указывать имя или псевдоним соответствующей таблицы или табличного выражения.

Если требуется включить все столбцы результирующей выборки, можно использовать символ «\*». Если требуется включить все поля только одной таблицы или табличного выражения из секции FROM, то перед символом «\*» нужно указать имя или псевдоним соответствующей таблицы или табличного выражения.

Если в column\_value указано имя столбца таблицы из секции FROM, это же имя будет назначено данному столбцу в результирующем наборе данных, а column\_name позволяет задать столбцу другое имя. В остальных случаях, если не указано column\_name, столбец не имеет имени.

Имена столбцов результирующей выборки данных могут повторяться. Если требуется обеспечить их уникальность, необходимо задавать для столбцов уникальные значения column\_name.

Если в секции SELECT указано ключевое слово DISTINCT, из результирующей выборки данных удаляются все одинаковые строки. Далее выполняется секция ORDER BY. Если указана секция TOP, из результирующей выборки данных остаётся выбираются не более top\_N строк, а все остальные строки удаляются.

**Секция ORDER BY.** В секции ORDER BY указываются столбцы секции SELECT или FROM, по которым выполняется сортировка. Если указано несколько столбцов, то сначала выполняется сортировка по первому столбцу, затем группы записей с одинаковыми значениями соответствующих полей упорядочиваются по значениям второго поля, затем группы записей с одинаковыми значениями полей первых двух столбцов упорядочиваются по значениям третьего столбца и так далее. Для каждого столбца указывается тип сортировки: ASC — по не-убыванию значений, DESC — по невозрастанию значений. Если тип не указан, используется значение ASC.

**Агрегатная функция** — это статистическая функция, которая принимает в качестве входного параметра имя столбца или выражения, выполняет вычисление на наборе значений и возвращает одиночное значение в качестве результата.

В языке SQL определены различные агрегатные функции. Из основных выделяют, например, COUNT, SUM, AVG, MIN и так далее. Более подробную информацию об агрегатных функциях изучите самостоятельно.

**Вложенные подзапросы.** Один SELECT-запрос может быть помещён в другой SELECT-запрос. В этом случае первый запрос является вложенным, а второй — главным. Вложенные подзапросы заключаются в круглые скобки.

Вложенные подзапросы можно разделить на три типа:

1. скалярные подзапросы — возвращают одно значение, могут использоваться в любом месте главного запроса, где ожидается простое значение соответствующего типа данных;
2. векторные подзапросы — возвращают табличное выражение, которое состоит из одного столбца. Его называют вектором данных. Векторные подзапросы могут использоваться в операции IN, которая проверяет наличие значения в массиве (векторе) данных;
3. табличные подзапросы — возвращают табличное выражение, содержащее произвольное количество строк и столбцов. Могут использоваться в любом месте главного запроса, где ожидается таблица или табличное выражение.

Столбцы табличного подзапроса должны иметь уникальные имена, тип столбцов определяется по значениям полей записей. Вложенные подзапросы могут использовать столбцы таблиц из главного запроса. Доступ к ним осуществляется указанием во вложенном подзапросе имён или псевдонимов таблиц главного запроса.

**Объединение результатов нескольких SELECT-запросов.** В некоторых случаях требуется выбрать данные из разных таблиц или других источников данных, имеющих одинаковую структуру, и объединить все строки полученных результатов в один результирующий набор данных. В этом случае записи можно рассматривать как элементы множеств.

Из двух SELECT-запросов можно получить один результирующий набор данных, указав между ними одну из следующих операций: UNION, UNION ALL, EXCEPT, INTERSECT.

Результатом объединения запросов является табличное выражение, которое также может участвовать в другой операции объединения результатов SELECT-запросов. Поскольку в этом случае не используются вложенные подзапросы, SELECT-запросы не заключаются в круглые скобки.

**Задание к работе**

Составить не менее трех различных SELECT-запросов к своей базе данных, созданной в результате выполнения предыдущих лабораторных работ, которые будут содержать объединение таблиц, агрегатные функции, а также группировку данных.

Выполнить задания по вариантам.

**Вариант 1.** Получить рейтинг (ТОП-5) наиболее популярных команд за сезон. Вывести рейтинг и эффективность игроков за заданный период.

**Вариант 2.** Получить рейтинг популярности товаров на основе предпочтений покупателей, в зависимости от различных групп товаров. Рассчитать сумму среднего чека по магазину в разное время суток.

**Вариант 3.** Получить среднее время простоя для тех водителей компании, которые совершали рейсы в заданную дату. Рассчитать соотношение доходности междугородних и местных доставок.

**Вариант 4.** Вывести наиболее часто заказываемые книги, сгруппированные по жанрам. Получить средний чек по разным группам товаров.

**Вариант 5.** Получить рейтинг продаж для продавцов-консультантов в группировке по различным моделям автомобилей. Получить список рекомендованных к заказу автомобилей на основании имеющегося запаса и популярности модели.

**Вариант 6.** Вывести рейтинг востребованности товаров в разрезе различных групп. Получить рейтинг дефицитности товаров в группировке по различным складам.

**Вариант 7.** Получить список рекомендаций для читателя с учетом его пола, возраста и прочитанных книг на основании общей статистики. Вывести среднее количество книг, прочитанных людьми за заданный период

**Вариант 8.** Составить упорядоченный список неплательщиков по возрастанию суммы долга в разрезе различных энергетических ресурсов. Получить рейтинг исполнителей работ за заданный промежуток времени (упорядочить по количеству отработанных заявок).

**Вариант 9.** Составить рейтинг убыточности товаров (наиболее часто требующих ремонта). Отчет по обеспеченности города мастерскими по разным группам товаров и всему ассортименту в целом.

**Вариант 10.** Подсчитать количество должников по подразделениям. Получить количественную оценку уровня остепененности преподавателей по подразделениям.

**Вариант 11.** Получить количество новых пользователей, зарегистрировавшихся, а также количество пользователей, удаливших аккаунт за указанный период. Подсчитать количество сообщений и комментариев в разрезе тем фоорума.

**Вариант 12.** Получить объем выплат по различным типам страховых случаев за период. Составить рейтинг менеджеров (Топ-5) компании в разрезе заключенных контрактов.

**Вариант 13.** Получить доходность гостиницы в зависимости от сезонности. Оценить актуальность предоставляемых услуг, в разрезе количественных показателей по разным группам услуг.

**Вариант 14.** Получить наиболее популярные направления (по количеству перевезенных пассажиров) в различные времена года. Посчитать рейтинг поездов (по количеству проданных билетов) в группировке по типам вагонов.

**Вариант 15.** Отчет по результатам предоставляемых услуг. Получить статистику пролеченных случаев заболеваний за заданный период.

**Пример выполнения задания**

**Вывести топ три компании с максимальным бюджетом**

SELECT executor.id, contact.fcs, budget

FROM contract

JOIN executor on executor.id = contract.executor\_id

JOIN contact on contact.id = executor.contact\_id

GROUP BY executor.id

ORDER BY budget DESC

LIMIT 3;

**Вывести ФИО исполнителей в обратном алфавитном порядке, чьи работы завершены к текущей дате**

SELECT contact.fcs

FROM contract

JOIN executor on executor.id = contract.customer\_id

JOIN contact on contact.id = executor.contact\_id

WHERE end\_date <= CURRENT\_DATE

GROUP BY contact.id

ORDER BY contact.fcs DESC;

**Вывести ФИО заказчиков в обратном алфавитном порядке, чьи договора находятся на стадии исполнения**

SELECT contact.fcs

FROM contract

JOIN customer on contract.customer\_id = customer.id

JOIN contact on contact.id = customer.contact\_id

WHERE end\_date > CURRENT\_DATE

GROUP BY contact.id

ORDER BY contact.fcs DESC;

В процессе выполнения лабораторной работы были изучены основные принципы создания SQL-запросов для выборки данных из таблиц базы данных и представления данных в требуемом виде.