Практическое задание 4

Язык SQL

Создание таблицы

Базовый синтаксис оператора *создания таблицы* имеет следующий вид:

```
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`.

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.

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

Изменение таблицы

Структура существующей *таблицы* может быть модифицирована с помощью команды `ALTER TABLE`, упрощенный синтаксис которой представлен ниже:

```
ALTER TABLE имя_таблицы
  ADD [COLUMN] имя_столбца тип_данных
 [ NULL | NOT NULL ]
  | DROP [COLUMN] имя_столбца
```

Эта команда удалит не только указанную *таблицу*, но и все входящие в нее *строки* данных. Если требуется удалить из *таблицы* лишь данные, сохранив структуру *таблицы*, следует воспользоваться командой `DELETE`.

Оператор `DROP TABLE` дополнительно позволяет указывать, следует ли операцию *удаления* выполнять каскадно. Если в операторе указано ключевое слово `RESTRICT`, то при наличии в *базе данных* хотя бы одного объекта, существование которого зависит от удаляемой *таблицы*, выполнение оператора `DROP TABLE` будет отменено. Если указано ключевое слово `CASCADE`, автоматически удаляются и все прочие объекты *базы данных*, чье существование зависит от удаляемой *таблицы*, а также другие объекты, зависящие от удаляемых объектов. Общий эффект от выполнения оператора `DROP TABLE` с ключевым словом `CASCADE` может оказаться весьма ощутимым, поэтому подобные операторы следует использовать с максимальной осторожностью.

Пример 3. Добавить в *таблицу* `Клиент` поле для номера расчетного счета.

Удаление таблицы

С течением времени структура *базы данных* меняется: создаются новые *таблицы*, а прежние становятся ненужными и удаляются из *базы данных* с помощью оператора:

```
DROP TABLE имя_таблицы [RESTRICT | CASCADE]
```

Индексы

Индексы в стандарте языка

Индексы представляют собой структуру, позволяющую выполнять ускоренный доступ к *строкам таблицы* на основе значений одного или более ее *столбцов*
Индекс – это дополнительная таблица, содержащая собственный первичный ключ, внешний ключ и индексируемые поля.
Поскольку *индексы* должны обновляться системой при каждом внесении *изменений* в их базовую *таблицу*, они создают дополнительную нагрузку на систему.
Индексы обычно создаются с целью удовлетворения определенных критериев поиска после того, как *таблица* уже находилась некоторое время в работе и увеличилась в размерах. *Создание индексов* не предусмотрено стандартом SQL, однако большинство диалектов поддерживают как минимум следующий оператор:

```
CREATE [ UNIQUE ] INDEX имя_индекса
  ON имя_таблицы(имя_столбца[ASC|DESC][,...n])
```

Указанные в операторе *столбцы* составляют *ключ индекса*. *Индексы* могут создаваться только для базовых *таблиц*, но не для представлений. Если в операторе указано ключевое слово `UNIQUE`, уникальность значений *ключа индекса* будет автоматически поддерживаться системой.

Удаление индекса

Если созданный *индекс* впоследствии окажется ненужным, его можно удалить с помощью оператора:

```
DROP INDEX 'имя_индекса'[,...n]
```

Пример 4. Создать уникальный *кластерный индекс* для *таблицы* `Клиент` по *столбцу* `Фамилия`.

Оператор SELECT

Оператор `SELECT` – один из наиболее важных и самых распространенных операторов SQL. Он позволяет производить *выборки* данных из таблиц и преобразовывать к нужному виду полученные результаты. Будучи очень мощным, он способен выполнять действия, эквивалентные операторам реляционной алгебры, причем в пределах единственной выполняемой команды. При его помощи можно реализовать сложные и громоздкие условия отбора данных из различных таблиц.

Оператор `SELECT` имеет следующий формат:

```
SELECT [ALL | DISTINCT] *|имя_столбца
  [AS   новое_имя] [,...n]
  FROM имя_таблицы [[AS] псевдоним] [,...n]
  [WHERE  <условие_поиска>]
  [GROUP BY  имя_столбца [,...n]]
  [HAVING <критерии выбора групп>]
  [ORDER BY имя_столбца [,...n]]
```

Оператор `SELECT` определяет поля (столбцы), которые будут входить в *результат выполнения запроса*. В списке они разделяются запятыми и приводятся в такой очередности, в какой должны быть представлены в *результате запроса*. Если используется имя поля, содержащее пробелы или разделители, его следует заключить

в квадратные скобки. Символом * можно выбрать все поля, а вместо имени поля применить выражение из нескольких имен.

Если обрабатывается ряд таблиц, то (при наличии одноименных полей в разных таблицах) в списке полей используется полная спецификация поля, т.е. Имя_таблицы.Имя_поля.

Предложение FROM

Предложение FROM задает имена таблиц и просмотров, которые содержат поля, перечисленные в операторе SELECT. Необязательный параметр псевдонима – это сокращение, устанавливаемое для имени таблицы.

Обработка элементов оператора SELECT выполняется в следующей последовательности:

- FROM – определяются имена используемых таблиц;
- WHERE – выполняется *фильтрация строк* объекта в соответствии с заданными условиями;
- 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;
- ORDER BY – определяется упорядоченность результатов выполнения операторов.

Порядок предложений и фраз в операторе SELECT не может быть изменен. Только два предложения SELECT и FROM являются обязательными, все остальные могут быть опущены. SELECT – закрытая операция: *результат запроса* к таблице представляет собой другую таблицу.

Пример 5. Составить список сведений о всех клиентах.

Параметр WHERE определяет критерий отбора записей из входного набора. Но в таблице могут присутствовать повторяющиеся записи (дубликаты). Предикат ALL задает включение в выходной набор всех дубликатов, отобранных по критерию WHERE. Нет необходимости указывать ALL явно, поскольку это значение действует по умолчанию.

Пример 6. Составить список всех фирм.

Результат выполнения запроса может содержать *дублирующиеся значения*, поскольку в отличие от операций реляционной алгебры оператор SELECT не исключает повторяющихся значений при выполнении *выборки данных*.

Предикат DISTINCT следует применять в тех случаях, когда требуется отбросить блоки данных, содержащие *дублирующие записи*. Учтите, что DISTINCT резко замедлить выполнение запросов.

Пример 7. Сделать то же самое, но без повторений

Предложение WHERE

параметр WHERE определяет, какие данные из приведенных в списке FROM таблиц появятся в *результате запроса*. За ключевым словом WHERE следует перечень *условий поиска (*пять основных типов *условий поиска* (или предикатов):

- *Сравнение*: сравниваются результаты вычисления одного выражения с результатами вычисления другого. Используем операторы сравнения {=, >,<, >=,<=, <>} и/или логические операторы{ AND, OR, NOT}.
- *Диапазон*: проверяется, попадает ли результат вычисления выражения в заданный *диапазон* значений. { `BETWEEN, NOT BETWEEN` }
- *Принадлежность множеству*: проверяется, принадлежит ли результат вычислений выражения заданному множеству значений{ `IN`,`NOT IN` }.
- *Соответствие шаблону*: проверяется, отвечает ли некоторое строковое значение заданному шаблону{ `LIKE` }.
- *Значение* `NULL`: проверяется, содержит ли данный столбец определитель `NULL` (неопределенное значение) { Is Null, Is Not Null }.

Пример 8. Показать все операции отпуска товаров объемом больше 20.

Пример 9. Вывести список товаров, цена которых больше или равна 100 и меньше или равна 150.

Пример 10. Вывести список клиентов из Москвы или из Самары.

Диапазон

Пример 11. Вывести список товаров, цена которых лежит в диапазоне от 100 до 150

Пример 12. Вывести список товаров, цена которых не лежит в диапазоне от 100 до 150."

Принадлежность множеству

Пример 13. Вывести список клиентов из Москвы или из Самары

Пример 14. Вывести список клиентов, проживающих не в Москве и не в Самаре.

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 15. Найти клиентов, у которых в номере телефона вторая цифра – 4.

Пример 16. Найти клиентов, у которых в номере телефона вторая цифра – 2 или 4.

Пример 17. Найти клиентов, у которых в номере телефона вторая цифра 2, 3 или 4.

Пример 18. Найти клиентов, у которых в фамилии встречается слог «ро».

Значение NULL

Пример 19. Найти сотрудников, у которых нет телефона (поле Телефон не содержит никакого значения).

Пример 20. Выборка сотрудников, у которых есть телефон (поле Телефон содержит какое-либо значение).

Предложение ORDER BY

Сортировка может выполняться по нескольким полям, в этом случае они перечисляются за ключевым словом ORDER BY через запятую. Способ сортировки{ **ASC, DESC** }. Фраза ORDER BY всегда должна быть последним элементом в операторе SELECT.

Пример 21.Вывести список клиентов в алфавитном порядке.

Пример 22. Вывести список фирм и клиентов. Названия фирм упорядочить в алфавитном порядке, имена клиентов в каждой фирме отсортировать в обратном порядке.

Домашнее задание

1. **Придумать *4 «сложных» запроса, т.е. содержащих как min два условия в WHERE***