МГТУ ИМ. Н.Э. БАУМАНА

Базы Данных

Семинар 2

Москва, 2018

DDL (Продолжение)

Мы рассмотрели создание объектов (create). Следующий блок интсрукций — alter. Изменение уже существующих объектов в системе.

Что можо сделать с помощью интсрукции ALTER. Обратить внимание, на примере изменения типа атрибута, что синтаксис для разных БД отличается:

1. Добавить атрибут (столбец)

```
ALTER TABLE table_name ADD column_name datatype;
ALTER TABLE table_name ADD (column_name_1 datatype,
                            column_name_2 datatype,
                            ...
                            column_name_n datatype);
```

2. Создание ограничений

```
ALTER TABLE table_name_1 ADD CONSTRAINT constant_name
        FOREIGN KEY (column_name_1, ..., column_name_n)
                REFERENCES table_name_1 (column_name_1, ...,
column_name_n)
        ON Update Cascade
        ON Delete Cascade;
ALTER TABLE table_name ADD CONSTRAINT primary_key_name
        PRIMARY KEY (column_name_1, ..., column_name_n);
```

3. Удалить атрибут/ограничения

```
ALTER TABLE table_name DROP COLUMN column_name;
ALTER TABLE table_name DROP CONSTRAINT constraint_name;
```

3. Изменить тип атрибута

```
ALTER TABLE table_name ALTER COLUMN column_name datatype; (SQL Server)
ALTER TABLE table_name MODIFY COLUMN column_name datatype; (MySQL,
Oracle)
ALTER TABLE table_name ALTER COLUMN column_name TYPE datatype;
```

(PostgreSQL)

Работает не для всех типов, например, пока в талица пустая можно изменять типы атрибутов как угодно. Если в таблице уже есть данные, например, при изменении varchar на number,будет ошибка, ddl не обновится. Orale(11) также не допускает и обратного пробразования из number в varchar.

4. Переименование атрибута

ALTER TABLE *table_name* CHANGE *column_old column_new;* (MySql)
ALTER TABLE *table_name* RENAME COLUMN *column_old* TO *column_new;* (Oracle)

5. Преименование объекта

ALTER TABLE *table_name* RENAME TO *new_table_name;*

Для удаления объектов, таки как таблицы, представления, индексы, церы БД используется инструкция DROP.

DROP TABLE *table_name;*

DML

Язык дня проведения манипуляций с данными. Используется для вставки/обновления/удаления данных. При этом сами объекты не изменяются. Языки DML могут существенно различаться у различных производителей СУБД. Существует стандарт SQL, установленный ANSI, но производители СУБД часто предлагают свои собственные «расширения» языка. Функции языков DML определяются первым словом в предложении (часто называемом **запросом**), которое почти всегда является глаголом. В случае с SQL эти глаголы — «select» («выбрать»), «insert» («вставить»), «update» («обновить»), и «delete» («удалить»). Это превращает природу языка в ряд обязательных утверждений (команд) к базе данных.

Название SQL (Structured Query Language – язык структурированных запросов) отражает тот факт, что запросы являются наиболее часто используемым элементом SQL. Запрос – это оператор, который посылает

команду Системе Управления Базой Данных (СУБД) произвести манипуляцию или отобразить определенную информацию. Все запросы по выборке данных в SQL конструируются с помощью оператора **SELECT**. Он позволяет выполнять довольно сложные проверки и обработку данных.

Запрос может выводить данные из определенного столбца или изо всех столбцов таблицы. Чтобы создать простейших SELECT запрос, необходимо указать имя столбца и название таблицы. Также необходимо сказать, что SQL код является регистронезависимым. Пример запроса:

```
SELECT column_list
FROM table_name
[WHERE условие]
[GROUP BY условие]
[HAVING условие]
[ORDER BY условие]
```

Разберем каждую из инструкций:

SELECT - Ключевое слово, которое сообщает базе данных о том, что оператор является запросом. Все запросы начинаются с этого слова, за ним следует пробел. Column_list - Список столбцов таблицы, указанный через запятую, которые выбираются запросом. Столбцы, не указанные в операторе, не будут включены в результат. Если необходимо вывести данные всех столбцов, можно использовать сокращенную запись. Звездочка (*) означает полный список столбцов.

```
SELECT *
FROM my_example_table;

SELECT column_1, column_2, column_3 AS myALIAS
FROM my_example_table AS myTable;
```

MyALIAS — Псевдоним, который используется как название столбца при отображении результата запроса на экран. Псевдоним может быть как у столбца, так и у всей таблицы. Псевдонимы для таблиц используются, в основном, при соединении таблиц.

FROM - Ключевое слово, которое должно присутствовать в каждом запросе. После него через пробел указывается имя таблицы, являющейся источником данных.

Код в скобках является не обязательным в операторе SELECT. Он необходим для более точного определения запроса.

WHERE - Оператор SQL WHERE служит для задания дополнительного условия выборки, операций вставки, редактирования и удаления записей. После ключевого слова следует логическое выражение - *condition* (Принимает значение true/false). Условие может включать в себя предикаты AND, OR, NOT, LIKE, BETWEEN, IS, IN, ключевое слово NULL, операторы сравнения и равенства (<, >, =).

Для примера необходимо написать несложную таблицу и заполнить парой записей — на усмотрение семинариста. Обязательно указывать результат до и после запроса. По настроению можно рассказать про некоторые функции

```
-- простой селект 1
SELECT * FROM Planets
WHERE Radius BETWEEN 3000 AND 9000;

-- простой селект 2
SELECT * FROM Planets
WHERE Radius = 5000;

-- Два равноценных запроса
SELECT PlanetName, OpeningYear, Opener
FROM Planets
WHERE PlanetName NOT LIKE '%s' AND PlanetName NOT LIKE 'S
%';
-- равноценно
SELECT PlanetName, OpeningYear, Opener
FROM Planets
WHERE UPPER(PlanetName) NOT LIKE 'S%';
```

GROUP BY + HAVING - Оператор SQL GROUP BY используется для объединения результатов выборки по одному или нескольким столбцам. Оператор SQL HAVING является указателем на результат выполнения агрегатных функций.

Агрегатной функцией в языке SQL называется функция, возвращающая какое-либо одно значение по набору значений столбца. Такими функциями являются: COUNT(), MIN(), MAX(), AVG(), SUM(). Оператор SQL HAVING аналогичен оператору WHERE за тем исключением, что применяется не для всего набора столбцов таблицы, а для набора созданного оператором GROUP BY и применяется всегда строго после него.

```
SELECT Singer, SUM(Sale)
FROM Artists
GROUP BY Singer
HAVING SUM(Sale) > 2000000;


SELECT Singer, MIN(Year)
FROM Artists
GROUP BY Singer
HAVING MIN(Year) < 1995;
```

ORDER BY - Выполняет сортировку выходных значений. Оператор SQL ORDER BY можно применять как к числовым столбцам, так и к строковым. В последнем случае, сортировка будет происходить по алфавиту.

Оператор SQL ORDER BY имеет следующий синтаксис:

```
ORDER BY column_name [ASC | DESC]
```

Сортировка может производиться как по возрастанию, так и по убыванию значений.

- Параметр ASC (по умолчанию) устанавливает порядок сортирования во возрастанию, от меньших значений к большим.
- Параметр DECS устанавливает порядок сортирования по убыванию, от больших значений к меньшим.

```
SELECT *
FROM Artists
ORDER BY Singer;


SELECT Singer, Album, Year
```

```
FROM Artists
WHERE Year > 2005
ORDER BY Year DESC;
```

Порядок написания инструкций: select -> from -> where -> group by -> having -> orber by

Порядок, в котором обрабатывает инструкции СУБД: from -> where -> group by -> having -> select -> orber by

* Именно поэтому если переименовывать переменные в select, новые имена не видны в остальных нструкция (кроме order by)