Кафедра КИБЭВС

ТУСУР
Томский государственный университет систем управления и радиоэлектроники

Лекция №7

Язык структурированных запросов

Structured Query Language

SQL - Structured Query Language (Язык Структурированных Запросов) – универсальный язык для создания модификации и управления данными в реляционных БД.

Если каждый элемент данных, или значение, определяется пересечением строки. Чтобы найти требуемый элемент данных, необходимо знать:

1.
2.
3.

Если каждый элемент данных, или значение, определяется пересечением строки. Чтобы найти требуемый элемент данных, необходимо знать:

1. имя таблицы;
2.
3.

Если каждый элемент данных, или значение, определяется пересечением строки. Чтобы найти требуемый элемент данных, необходимо знать:

1. имя таблицы;
2. название столбца;
3.

Если каждый элемент данных, или значение, определяется пересечением строки. Чтобы найти требуемый элемент данных, необходимо знать:

1. имя таблицы;
2. название столбца;
3. значение первичного ключа;

В реляционных БД существует 2 типа таблиц:

- пользовательские;
- системные.

Пользовательские таблицы содержат информацию, для которых собственно и создавалась БД.

Системные таблицы (системные каталоги) содержат описание БД.

Независимость

Отражается в двух аспектах:

1. изменение приложения, без изменения структуры БД;
2. изменение структуры БД, без изменения работы приложений.

В БД такое свойство достигается наличием двух структур: *физической и логической морелей БД.*

Физическая независимость данных – представление данных абсолютно не зависит от способа их физического хранения.

Примеры:

1. Недостаток места для хранения информации.
2. Выход из строя устройства.
3. Увеличение производительности системы.

Логическая независимость – изменение взаимосвязей между таблицами, столбцами и строками не влияет на правильное функционирование программных приложений и текущих запросов.

Язык высокого уровня

SQL используется для:

1. манипуляции с данными (data manipulation);
2. определения данных (data definition):
   - выборка (data retrieval);
   - модификации (data modification);
3. администрирование данных (data administration).

Выборка – поиск необходимых данных.

Модификация – добавление, удаление или изменение данных.

Операции выборки:

Select *

From class

Операции выборки:

Select *

From class

Id_Class	Number_Class	Teacher
1	11 А	Иванова В.С.
2	11Б	Петрова Г.В.
3	10 А	Сидорова М.М.
4	10Б	Новикова С.С.

Операции по модификации:

Insert into class

Values (‘5’, ‘9А’, ‘Смирнова А.П.’)

Операции по модификации:

Insert into class

Values (‘5’, ‘9А’, ‘Смирнова А.П.’)

Id_Class	Number_Class	Teacher
1	11 А	Иванова В.С.
2	11Б	Петрова Г.В.
3	10 А	Сидорова М.М.
4	10Б	Новикова С.С.
5	9А	Смирнова А.П.

Создание таблицы:

Create table test

(id int, name char (15))

Операции администрирования или управления данными:

Grant select

On test

To teatcher

Реляционные операции

В управлении реляционными БД упоминается три операции по выборке данных:

- проектирование: *выбирает столбцы*;
- выбор (ограничение): *выбирает строки*;
- объединение: *собирает вместе данные из связанных таблиц*.

Все эти операции записываются с использованием ключевого слова ***Select.***

Общий синтаксис:

SELECT список выбора
FROM список таблиц
WHERE условия поиска

Операция проектирования позволяет указать системе, какие столбцы таблицы вы хотите просмотреть.

Например:

```
Select id_class, number_class
From class
```

Операция проектирования позволяет указать системе, какие столбцы таблицы вы хотите просмотреть.

Например:

Select id_class, number_class

From class

Id_Class	Number_Class
1	11 А
2	11Б
3	10 А
4	10Б
5	9А

Результат: в виде таблицы (результирующая или производная таблица)

Базовая таблица – содержит исходные строки и данные

Операция выбора позволяет получить из таблицы подмножества ее строк.

Например:

```
Select *
From class
Where teacher = “C*”
```

Операция выбора позволяет получить из таблицы подмножества ее строк.

Например:

Select *

From class

Where teacher = “C*”

Id_Class	Number_Class	Teacher
3	10 А	Сидорова М.М.
5	9А	Смирнова А.П.

Операция объединения может работать с одной или несколькими таблицами, соединяя данные таким образом, что можно легко сопоставить или выделить определенную информацию из своей БД.

Пример:

```
Select number_class, name
From class, students
Where class.id_class=titles.id_class
```

Операция объединения может работать с одной или несколькими таблицами, соединяя данные таким образом, что можно легко сопоставить или выделить определенную информацию из своей БД.

Пример:

Select number_class, name

From class, students

Where class.id_class=titles.id_class

Id_class	Name
11А	Ананьев
11А	Анфилофьев
11А	Дель
11Б	Дорожкин
11Б	Ижутов
11Б	Созинова
10А	Сумарокова
И т.д.	И т.д.

Альтернативный способ просмотра данных

Курсор – виртуальная таблица или производная таблица.

Перемещаемая по таблице рамка, через которую можно увидеть только необходимую часть информации.

Виртуальные таблицы не хранятся физически в БД. Это не копия некоторых данных из базовых таблиц, изменяя данные в курсоре – изменяются данные и в базовой таблице.

Нули

При пропуске информации в БД возникает проблема несогласованности, т.е. возможно появление противоречий в логике обработки данных, в следствии чего появляется нарушение целостности БД.

В БД «нуль» не означает пустое поле или обычный математический нуль. Он отображает тот факт, что значение не известно, недоступно или неприменимо.

Безопасность

Понятие **безопасности** связано с необходимостью управления доступом к информации.

Команды SQL **Grant** и **Revoke** позволяют привилегированным пользователям управлять правами других пользователей по просмотру и модификации информации.

Целостность

Целостность очень важный и очень сложный вопрос при управлении реляционными БД.

Несогласованность данных может возникать по ряду причин: сбои системы, ошибки в ПО или логические ошибки в приложениях.
Реляционные системы управления БД защищают данные от такого типа несогласованности, гарантируя, что команда будет выполнена до конца, либо не выполнена полностью. Этот процесс называется **управлением транзакциями.**

Целостность

Объектная целостность связана с корректным проектированием БД. Одно из требований, чтобы ни один первичный ключ не имел нулевого значения.

Ссылочная целостность требует не противоречивости между частями информации повторяющихся в разных таблицах. *Т.е. при изменении информации в одном месте она изменялась и* ***во всех*** *остальных частях.*

- SQL применим как ***к локальным***, так и ***распределенным*** БД.

Важным достоинством SQL при работе с распределенными БД, является не большая загрузка сети, т.к. передаются только запросы и результаты их выполнения.

SQL позволяет:

- формировать состав полей набора данных при выполнении приложения;

- включать в набор данных поля и записи из разных таблиц;

- отбирать записи по сложным условиям и критериям;

- сортировать набор данных по любому полю, в том числе неиндексированному;

- осуществлять поиск данных по частичному совпадению со значениями в поле.

SQL не обладает возможностями полнофункционального языка программирования, а ориентирован на доступ к данным и поэтому включается в средства разработки программного обеспечения.

Выделяют два вида SQL-запросов:

1. статический;
2. динамический.

Статический запрос – запрос включенный в исходный код на этапе разработки программного приложения и в процессе жизненного цикла программы не изменяется.

Динамический запрос – формируется и изменяется в процессе работы программного обеспечения.

Спасибо за внимание!!!

Синтаксис

1. Ключевые слова и операторы SQL всегда записываются прописными буквами (SELECT, FROM, WHERE).
2. Фигурные скобки **{}** вокруг слов или фраз – необходимо выбрать хотя бы одну из заключенных в них опций. Если опции разделены вертикальной чертой **|** - использование только одной из опций. Если опции разделены запятой **(,)** - использование одной или нескольких опций.
3. Квадратные скобки **[]** – заключенные опции не обязательны. Если опции разделены вертикальной чертой **|** - использование только одной из опций или вообще не использовать. Если опции разделены запятой **(,)** - использование одной или нескольких опций, либо вообще не использовать.

Создание БД

Права управления БД:

- Установка прав других пользователей на использование БД.
- Регулярное создание резервных копий и запуск процедур восстановления в случаях сбоев системы.
- Выделение в случае необходимости дополнительного пространства на диске под базу данных.
- Владение большинством производных объектов БД.
- Понимание типов данных БД и умение их

Создание БД

В некоторых стандартах SQL оператор **CREATE DATABASE** не входит его заменяет оператор **CREATE SCHEMA** –определяющий части БД, которыми будут владеть конкретные пользователи.

В зависимости от реализации в запись этого оператора могут входить разные предложения, позволяющие управлять расположением БД, ее размером и др. параметрами

Создание БД

CREATE TABLE table-name

(<column name> <data type>[(size)], <column name> <data type> [(size)] ...)

Команда **CREATE TABLE** определяет *имя таблицы* и *описание набора имён столбцов*, указанных в определенном порядке. Она также определяет *типы данных и размеры столбцов*. Каждая таблица должна иметь по крайней мере один столбец.

Создание БД

Замечание:

- Так как пробелы используются для разделения частей команды SQL, они не могут быть частью имени таблицы (или любого другого объекта, такого как индекс). Знак подчеркивания (_) обычно используется для разделения слов в именах таблиц.

Создание БД

Замечание:

- Значение аргумента размера зависит от типа данных. Если вы его не указываете, ваша система сама будет назначать значение автоматически.

Числовые типы – хороший вариант, т.к. нет необходимости согласовывать размеры для их совместимости, но есть проблемы при использовании больших чисел – вопрос в том достаточно ли велики для того чтобы их вместить.

Создание БД

Таблицы принадлежат пользователю, который их создал, и имена всех таблиц, принадлежащих данному пользователю, должны отличаться друга от друга, как и имена всех столбцов внутри данной таблицы. Отдельные таблицы могут использовать одинаковые имена столбцов, даже если они принадлежат одному и тому же пользователю.

<имя_пользователя>.<имя_таблицы>

Создание БД

Порядок столбцов в таблице определяется порядком, в котором они указаны. Имена столбца не должны разделяться при переносе строки (что делается для удобочитаемости) и отделяются запятыми.

```
CREATE TABLE group
   (id_group integer,
   name group char (5),
    city char (10))
```

Индексы

Таблицы могут иметь большое количество строк, а, так как строки не находятся в каком-нибудь определенном порядке, их поиск по указанному значению может потребовать времени. Индексный адрес это и забота, и в то же время обеспечение способа объединения всех значений в группы из одной или больше строк, которые отличаются одна от другой.

Индекс - это упорядоченный (буквенный или числовой) список столбцов или групп столбцов в таблице.

Индексы

Индекс значительно улучшает эффективность запросов, но использование индекса несколько замедляет операции модификации DML (такие как INSERT и DELETE), а сам индекс занимает память. Следовательно, каждый раз, когда вы создаёте таблицу, вы должны принять решение, индексировать её или нет.

Создание БД

Индексы могут состоять из нескольких полей.

Если больше чем одно поле указывается для одного индекса, второе упорядочивается внутри первого, третье внутри второго, и так далее.

```
CREATE INDEX name_index
    ON name_tables (column_name [,column_name]...)

CREATE UNIQUE INDEX name_index
    ON name_tables (column_name [,column_name]...)
```

Индексы

Замечание:

- создание уникального индекса имеет смысл только тогда когда это диктуется самими данными.
- усилить уникальность поля.
- эта команда будет отклонена, если уже имеются идентичные значения в поле.
- Лучше создавать индексы в процессе создания таблицы и прежде чем введены любые значения.
- для уникального индекса состоящего из более чем одного поля, это - комбинация значений, каждое из которых может и не быть уникальным.

Индексы

Синтаксис для удаления индекса:

DROP INDEX index_name

Удаление индекса не влияет на содержимое полей.

Изменение таблицы после создания

Команда **ALTER TABLE** это широко распространённая и довольно содержательная форма, хотя её возможности несколько ограничены. Она используется для того, чтобы:

- изменить определение существующей таблицы;
- добавить столбцы к таблице;
- удалять столбцы;
- изменять размеры столбца;
- добавлять или удалять ограничения.

Изменение таблицы после создания

Синтаксис добавления столбца к таблице:

ALTER TABLE name_table

ADD <column name> <data type> <size>;

Удаление таблицы

Вы должны быть владельцем (т.е. создателем) таблицы, чтобы иметь возможность удалить её.

При удалении таблицы SQL сначала потребует, чтобы вы очистили таблицу прежде чем удалить её из БД.

Удаление таблицы

Синтаксис для удаления вашей таблицы, если конечно она является пустой, следующий:

DROP TABLE <table_name>

Удаление таблицы

Замечание:

При применении этой команды имя таблицы больше не распознаётся, и нет такой команды, которая могла бы быть дана этому объекту.

Необходимо убедиться, что эта таблица не ссылается внешним ключом к другой таблице, а также нет других команд которые используют в своей работе эту таблицу.

Ограничение значений

Когда вы создаёте таблицу возможно указывать ограничения на значения, которые могут быть введены в поля. Если это сделано, SQL будет отклонять любые значения, нарушающие критерии, которые вы определили.

Есть два основных типа ограничений: ограничение столбца и ограничение таблицы.

Различие между ними в том, что ограничение столбца применяется только к отдельным столбцам, в то время как ограничение таблицы применяется к группам из одного и более столбцов.

Ограничение значений

Ограничение столбца записывается после типа данных и перед запятой. Ограничение таблицы помещается в конец имени таблицы после последнего имени столбца, но перед заключительной круглой скобкой.

CREATE TABLE <table name>

(<column name> <data type> <column constraint>, <column name> <data type> <column constraint> ... <table constraint> (<column name> [, <column name>])...)

Исключение пустых значений

```
CREATE TABLE group
    (id_group integer NOT NULL,
    name group char (5) NOT NULL,
     city char (10))
```

Уникальность столбца

```
CREATE TABLE group
    (id_group integer NOT NULL UNIQUE,
    name group char (5),
     city char (10))
```

Уникальность таблицы

```
CREATE TABLE group
    (id_group integer NOT NULL,
    name group char (5) NOT NULL,
     city char (10),
    UNIQUE (id_group))
```

Определение первичных ключей

```
CREATE TABLE group
    (id_group integer NOT NULL PRIMARY KEY,
    name group char (5) NOT NULL,
     city char (10))
```

Составной первичный ключ

```
CREATE TABLE group
    (id_group integer NOT NULL,
    name group char (5) NOT NULL,
     city char (10),
     PRIMARY KEY (id_group, name))
```

Проверка значений полей

Для установки таких ограничений используется ограничение **CHECK** – устанавливает условие которому должно удовлетворять вводимое значение

```
CREATE TABLE group
   (id_group integer NOT NULL PRIMARY KEY,
   name group char (5) NOT NULL,
   city char (10),
   count integer CHECK (count >=1))
```

Предопределение перечня вводимых значений

```
CREATE TABLE group
    (id_group integer NOT NULL PRIMARY KEY,
    name_group char (5) CHECK,
    (name_group IN (‘529-1’, ‘529-2’, ‘579’ )),
    city char (10),
    count integer CHECK (count >=1))
```

Ограничение базирующиеся на нескольких полях

```
CREATE TABLE group
    (id_group integer NOT NULL PRIMARY KEY,
    name_group char (5),
    city char (10),
    kurs integer,
    CHECK (kurs>1 and kurs<5 and name_group=520 … 526))
```

Установка значений по умолчанию

```
CREATE TABLE group
    (id_group integer NOT NULL PRIMARY KEY,
    name_group char (5) DEFAULT = ‘526-1',
    city char (10),
    count integer CHECK (count >=1))
```

Кафедра КИБЭВС

ТУСУР
Томский государственный университет систем управления и радиоэлектроники

Спасибо за внимание!!!