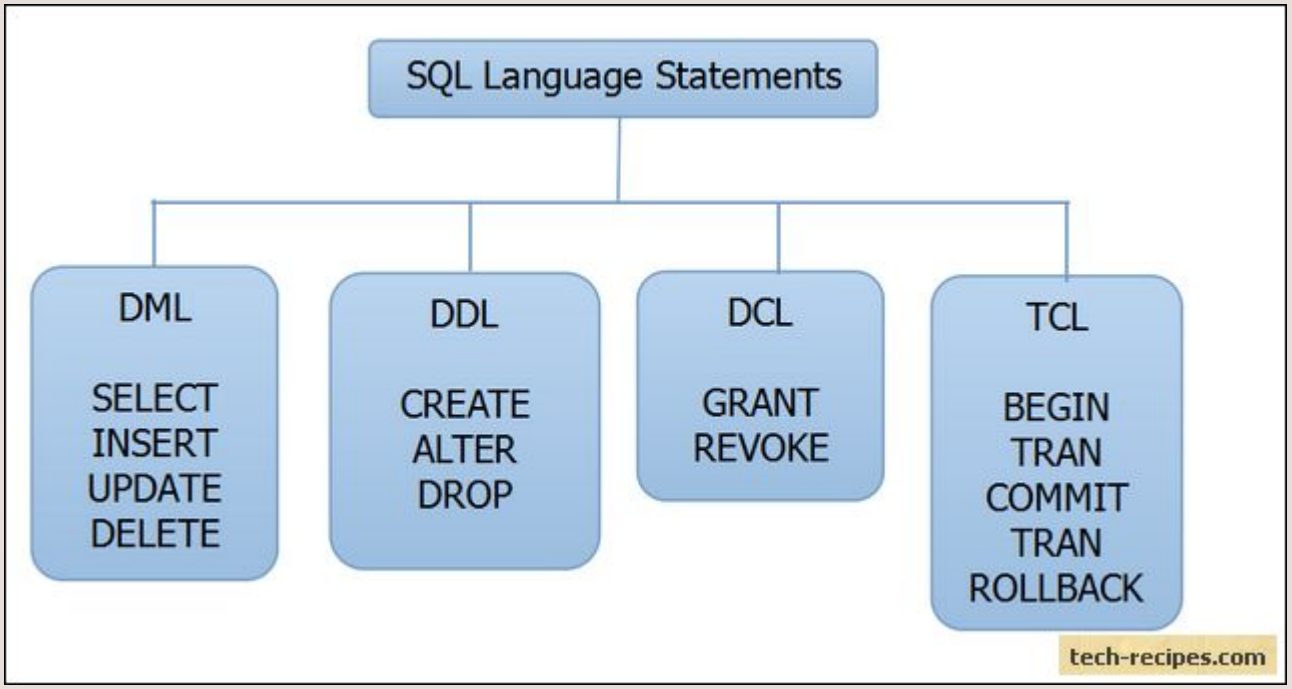
ЛЕКЦИЯ 4:
СОЗДАНИЕ ТАБЛИЦ SQL


SQL Language Statements
DML
SELECT
INSERT
UPDATE
DELETE
DDL
CREATE
ALTER
DROP
DCL
GRANT
REVOKE
TCL
BEGIN
TRAN
COMMIT
TRAN
ROLLBACK
tech-recipes.com

DDL – Data Definition Language

Data Definition Language (DDL) – это группа операторов определения данных. Другими словами, с помощью операторов, входящих в эту группы, мы определяем структуру базы данных и работаем с объектами этой базы, т.е. создаем, изменяем и удаляем их.

В эту группу входят следующие операторы:

- CREATE – используется для создания объектов базы данных;
- ALTER – используется для изменения объектов базы данных;
- DROP – используется для удаления объектов базы данных.

DCL – Data Control Language

Data Control Language (DCL) – группа операторов определения доступа к данным. Иными словами, это операторы для управления разрешениями, с помощью них мы можем разрешать или запрещать выполнение определенных операций над объектами базы данных.
Сюда входят:

- GRANT – предоставляет пользователю или группе разрешения на определённые операции с объектом;
- REVOKE – отзывает выданные разрешения;
- DENY– задаёт запрет, имеющий приоритет над разрешением.

Transaction Control Language (TCL)

Группа операторов для управления транзакциями. Транзакция – это команда или блок команд (инструкций), которые успешно завершаются как единое целое, при этом в базе данных все внесенные изменения фиксируются на постоянной основе или отменяются, т.е. все изменения, внесенные любой командой, входящей в транзакцию, будут отменены.

Группа операторов TCL предназначена как раз для реализации и управления транзакциями. Сюда можно отнести:

- BEGIN TRANSACTION – служит для определения начала транзакции;
- COMMIT TRANSACTION – применяет транзакцию;
- ROLLBACK TRANSACTION – откатывает все изменения, сделанные в контексте текущей транзакции;
- SAVE TRANSACTION – устанавливает промежуточную точку сохранения внутри транзакции.

DML – Data Manipulation Language

Data Manipulation Language (DML) – это группа операторов для манипуляции данными. С помощью этих операторов мы можем добавлять, изменять, удалять и выгружать данные из базы, т.е. манипулировать ими.

В эту группу входят самые распространённые операторы языка SQL:

SELECT – осуществляет выборку данных;

INSERT – добавляет новые данные;

UPDATE – изменяет существующие данные;

DELETE – удаляет данные.

ПРИМЕР (ТАБЛИЦА СОЗДАНИЕ):

Таблицы создаются командой **CREATE TABLE**. Эта команда создает пустую таблицу — таблицу без строк:

CREATE TABLE <table-name >

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

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

….);

Порядок столбцов в таблице определяется порядком, в котором они указаны. Имя столбца не должно разделяться при переносе строки, но отделяется запятыми.

```
CREATE TABLE PEOPLE
(       ID NUMBER,
        NM VARCHAR2(50),
        FAMIL VARCHAR2(50),
        OTCH VARCHAR2(50),
        DROG DATE
) /
```

ПРИМЕР (ТАБЛИЦА ЗАПОЛНЕНИЕ):

Заполняем ее тремя записями:

```
INSERT INTO PEOPLE(ID, NM, FAMIL, OTCH, DROG)
        VALUES(1, 'John', 'Godwin', 'Petrovich', TO_DATE('03-12-1967','DD-MM-YYYY'))
/
INSERT INTO PEOPLE(ID, NM, FAMIL, OTCH, DROG)
        VALUES(2, 'Bob', 'Doris', 'Martovich', TO_DATE('01-02-1960','DD-MM-YYYY'))
/
INSERT INTO PEOPLE(ID, NM, FAMIL, OTCH, DROG)
        VALUES(3, 'Frank', 'Black', 'Milleniumich', TO_DATE('03-07-1953','DD-MM-YYYY'))
/
COMMIT  /
```

ПРИМЕР(ТАБЛИЦА ПРОСМОТР):

```
CREATE TABLE NEWPEOPLE
(       NM VARCHAR2(50),
        FAMIL VARCHAR2(50),
        OTCH VARCHAR2(50)
)
/
COMMIT
/
```

Итак, новая табличка создана с применением оператора **DDL - CREATE TABLE**. Убедиться можно с помощью запроса:

```
SELECT a.OBJECT_NAME, a.OBJECT_TYPE, a.STATUS FROM USER_OBJECTS a
WHERE a.OBJECT_TYPE = 'TABLE'
/
```

- Получаем:

```
OBJECT_NAME          OBJECT_TYPE          STATUS
-------------        -------------        -------
CUSTOMERS    TABLE          VALID
NEWPEOPLE    TABLE          VALID
OFFICES              TABLE          VALID
ORDERS               TABLE          VALID
PEOPLE               TABLE          VALID
PRODUCTS             TABLE          VALID
SALESREPS            TABLE          VALID
```

ПРИМЕР(ТАБЛИЦА ИЗМЕНЕНИЕ):

Команда ALTER TABLE не часть стандарта ANSI; но это — широко доступная, и довольно содержательная форма. Обычно, она добавляет столбцы к таблице. Иногда она может удалять столбцы или изменять их размеры, а также в некоторых программах добавлять или удалять ограничения. Типичный синтаксис, чтобы добавить столбец к таблице:

ALTER TABLE <table name> ADD/DROP <column name> <data type> <size>;

ALTER TABLE "NEWPEOPLE" ADD "BloodGroup" number;

ПРИМЕР(ТАБЛИЦА УДАЛЕНИЕ):

- Для удаления таблиц воспользуемся оператором языка **DDL - DROP TABLE**:

```
DROP TABLE PEOPLE
/
DROP TABLE NEWPEOPLE
/
COMMIT
/
```

- После ввода в **SQL*Plus** получаем:

```
SQL> DROP TABLE PEOPLE
  2 /
Таблица удалена.
SQL> DROP TABLE NEWPEOPLE
  2 /
Таблица удалена.
SQL>
SQL>
COMMIT
  2 /
Фиксация обновлений завершена.
```

С оператором **DROP TABLE** необходимо обращаться осторожно, так же помнить о возможности воспользоваться откатом транзакции, оператор **ROLLBACK**

ПРИМЕР **ИНДЕКСЫ:**

- Индекс — это упорядоченный (буквенный или числовой) список столбцов или групп столбцов в таблице. Таблицы могут иметь большое количество строк, а, так как строки не находятся в каком-нибудь определенном порядке, их поиск по указанному значению может потребовать значительного времени. Когда вы создаете индекс в поле, ваша база данных запоминает соответствующий порядок всех значений этого поля в области памяти.

CREATE INDEX <index name> ON <table name>

(<column name> [,<column name>]...);

- Таблица, конечно, должна уже быть создана и должна содержать имя столбца. Имя индекса не может быть использовано для чего-то другого в базе данных (любым пользователем). Однажды созданный, индекс будет невидим пользователю. Сервер SQL сам решает, когда он необходим, чтобы ссылаться на него, и делает это автоматически.

CREATE INDEX Clientgroup ON PEOPLE (ID);

Этот синтаксис используется для удаления индекса:

DROP INDEX <Index name>;

ПРИМЕР **INSERT :**

INSERT INTO <имя_таблицы> [(<имя_столбца>,<имя_столбца>,...)

VALUES (<значение>,<значение>,..)

- Список столбцов в данной команде не является обязательным параметром. В этом случае должны быть указаны значения для всех полей таблицы в том порядке, как эти столбцы были перечислены в команде CREATE TABLE, например:

```
INSERT INTO PEOPLE VALUES(1, 'John', 'Godwin', 'Petrovich',
TO_DATE('03-12-1967','DD-MM-YYYY'))
/
```

С перечислением необходимых столбцов:

```
INSERT INTO PEOPLE(ID, NM, FAMIL, DROG)
    VALUES(2, 'Bob', 'Doris', TO_DATE('01-02-1960','DD-MM-YYYY'))
/
```

ПРИМЕР **UPDATE :**

- UPDATE <имя_таблицы> SET <имя_столбца>=<значение>,...
- [WHERE <условие>]
- Если задано ключевое слово WHERE и условие, то команда UPDATE применяется только к тем записям, для которых оно выполняется. Если условие не задано, UPDATE применяется ко всем записям.

Пример:

UPDATE PEOPLE SET DROG = SYSDATE - 1 WHERE ID < 17;

- **В качестве условия используются** логические выражения над константами и полями. В условиях допускаются: операции сравнения: > , < , >= , <= , = , <> , != . В SQL эти операции могут применяться не только к числовым значениям, но и к строкам ("<" означает раньше, а ">" позже в алфавитном порядке) и датам ("<" раньше и ">" позже в хронологическом порядке).
- оперции проверки поля на значение NULL: IS NULL, IS NOT NULL
- операции проверки на вхождение в диапазон: BETWEEN и NOT BETWEEN.
- операции проверки на вхождение в список: IN и NOT IN
- операции проверки на вхождение подстроки: LIKE и NOT LIKE
- отдельные операции соединяются связями AND, OR, NOT и группируются с помощью скобок.

ПРИМЕР **DELETE** :

DELETE FROM <имя_таблицы> [WHERE <условие>]

- Удаляются все записи, удовлетворяющие указанному условию. Если ключевое слово WHERE и условие отстутствуют, из таблицы удаляются все записи.

Пример:

DELETE FROM PEOPLE WHERE ID < 17 and DROG = SYSDATE - 1 ;

ПРИМЕР **SELECT** :

- Для извлечения записей из таблиц в SQL определен оператор SELECT. С помощью этой команды осуществляется не только операция реляционной алгебры "выборка" (горизонтальное подмножество), но и предварительное соединение (join) двух и более таблиц. Это наиболее сложное и мощное средство SQL, полный синтаксис оператора SELECT имеет вид:

```
SELECT [ALL | DISTINCT] <список_выбора>
        FROM <имя_таблицы>, ...
         [ WHERE <условие> ]
         [ GROUP BY <имя_столбца>,... ]
            [ HAVING <условие> ]
         [ORDER BY <имя_столбца> [ASC | DESC],... ]
```

- Порядок предложений в операторе SELECT должен строго соблюдаться (например, GROUP BY должно всегда предшествовать ORDER BY), иначе это приведет к появлению ошибок. Этот оператор всегда начинается с ключевого слова SELECT. В кострукции <список_выбора> определяется столбец или столбцы, включаемые в результат. Он может состоять из имен одного или нескольких столбцов, или из одного символа * (звездочка), определяющего все столбцы. Элементы списка разделяются запятыми.

ПРИМЕР **SELECT (ПРОДОЛЖЕНИЕ):**

SELECT * FROM authors;

SELECT title FROM titles WHERE yearpub > 1996;

SELECT title FROM titles WHERE yearpub>=1995 AND yearpub<=1997;

SELECT title FROM titles WHERE yearpub BETWEEN 1995 AND 1997;

SELECT title FROM titles WHERE yearpub IN (1995,1996,1997);

- LIKE

WHERE <имя_столбца> LIKE <образец> [ESCAPE <ключевой_символ>]

- Образец заключается в кавычки и должен содержать шаблон подстроки для поиска. Обычно в шаблонах используются два символа:

% (знак процента) - заменяет любое количество символов

_ (подчеркивание) - заменяет одиночный символ.

Примеры:

SELECT publiser, url FROM publishers WHERE publisher LIKE '%Wiley%';

SELECT title FROM titles WHERE title LIKE 'SQL%';

SELECT site, url FROM wwwsites WHERE url LIKE '%my@_works%' ESCAPE '@';

SELECT ИЗ НЕСКОЛЬКИХ ТАБЛИЦ.

- Очень часто возникает ситуация, когда выборку данных надо производить из отношения, которое является результатом слияния (join) двух других отношений. Например, нам нужно получить из базы данных publications информацию о всех печатных изданиях в виде следующей таблицы: |название_книги | год_выпуска | издательство |
- Для этого СУБД предварительно должна выполнить слияние таблиц titles и publishers, а только затем произвести выборку из полученного отношения.
- Для выполнения операции такого рода в операторе SELECT после ключевого слова FROM указывается список таблиц, по которым произвоится поиск данных. После ключевого слова WHERE указывается условие, по которому производится слияние. Для того, чтобы выполнить данный запрос, нужно дать команду:

```
SELECT titles.title,titles.yearpub,publishers.publisher
        FROM titles,publishers
        WHERE titles.pub_id=publishers.pub_id;
```

SELECT ИЗ НЕСКОЛЬКИХ ТАБЛИЦ.

- А вот пример, где одновременно задаются условия и слияния, и выборки (результат предыдущего запроса ограничивается изданиями после 1996 года):

```
SELECT titles.title,titles.yearpub,publishers.publisher
    FROM titles,publishers
    WHERE titles.pub_id=publishers.pub_id AND
        titles.yearpub>1996
```

- Следует обратить внимание на то, что когда в разных таблицах присутствуют одноименные поля, то для устранения неоднозначности перед именем поля указывается имя таблицы и знак "." (точка).

ВЫЧИСЛЕНИЯ ВНУТРИ SELECT..

- SQL позволяет выполнять различные арифметические операции над столбцами результирующего отношения. В конструкции <список_выбора> можно использовать константы, функции и их комбинации с арифметическими операциями и скобками. Например, чтобы узнать сколько лет прошло с 1992 года (год принятия стандарта SQL-92) до публикации той или иной книги можно выполнить команду:

```
SELECT title, yearpub-1992 FROM titles WHERE yearpub > 1992;
```

- В арифметических вражения допускаются операции сложения (+), вычитания (-), деления (/), умножения (*), а также различные функции (COS, SIN, ABS - абсолютное значение и т.д.). Также в запрос можно добавить строковую константу:

```
SELECT 'the title of the book is', title, yearpub-1992
```

- FROM titles WHERE yearpub > 1992;