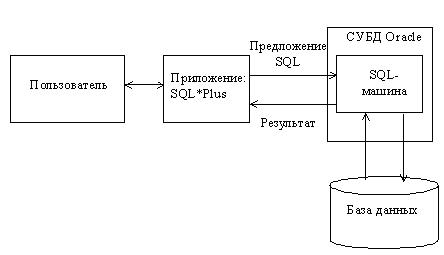
**Работа с СУБД Oracle XE 18c**

*Фирма* *Oracle* предоставляет два основных инструмента для общения с *БД* в диалоге посредством *SQL*: SQL\*Plus и SQL Developer. Дальнейшие примеры в тексте, как правило, предполагаются для исполнения в SQL\*Plus, однако с разной степенью корректировки исполняемы и в *SQL* *Developer*.

SQL\*Plus — *программа* из обычного комплекта *ПО* *Oracle* для диалогового общения с *БД* путем ввода пользователем (или, возможно, из сценарного файла-"скрипта") текстов на *SQL* и *PL/SQL* и предъявления на экране компьютера результата, полученного от *СУБД*:



#### Схемы данных

В дальнейших примерах чаще всего будут использоваться две таблицы, хранимые в "схеме данных" с именем SCOTT: это EMP и DEPT.

В Oracle и большинстве других подобных систем **схемой данных** называется специально оформляемое подмножество хранимых объектов, с которыми конкретные *прикладные программы имеют право работать в данный момент*. Понятие схемы данных задумано ради удобства работы с содержимым БД и защиты доступа. **Объектами схемы** в Oracle могут быть не только таблицы, но и прочие элементы, такие как индексы, процедуры и другие (в том числе так называемые "ограничения целостности"). Слово "схема" иногда употребляется и применительно к таблице. Под "схемой таблицы" понимают ее структуру вместе с некоторыми элементами описания.

#### Пользователи и полномочия

В отличие от схемы, понятие пользователя в базах данных служит для моделирования возможностей прикладной программы совершать те или иные действия в БД. Что касается действий с объектами БД, то специального разрешения на такие действия Oracle не требует в двух случаях:

* когда объект доступа принадлежит самому пользователю,
* когда действие с объектом объявлено "публичным".

В остальных случаях для выполнения действия пользователь должен иметь специально выданное ему полномочие ("привилегию", по терминологии Oracle). Примерами таких полномочий могут служить полномочие создавать таблицу (в собственной схеме!), полномочие обращаться запросом SELECT к чужой таблице.

В БД Oracle самостоятельным является понятие пользователя, а понятие схемы подчинено ему. Так, по команде CREATE USER СУБД создаст в БД *пользователя* и одновременно автоматически — *схему* с тем же именем, к которой будет относить все объекты "этого пользователя", и только их. Схема будет удалена из БД также автоматически при удалении пользователя.

**Основные команды SQL**

Язык SQL состоит из операторов языка, называемых иногда командами языка SQL, типов данных, набора встроенных функций.

По своему логическому назначению операторы языка SQL часто разбиваются на следующие группы:

* **Язык определения данных** (ЯОД). Он же Data Definition Language (DDL). Определяет структуру базы, задает пользователей, хранимые объекты и привилегии доступа к ним.
* **Язык манипулирования данными** (ЯМД). Он же Data Manipulation Language (DML). Вставляет, обновляет и удаляет данные и выполняет запросы к ним.
* **Язык управления данными (транзакциями)** Data Control Language (DCL).

**Язык определения данных**

Для каждого типа хранимых объектов базы (таблица, представление, последовательность, триггер, пользователь, но не курсор) существует “малый джентльменский” набор инструкций CREATE, ALTER, DROP (СОЗДАТЬ, ИЗМЕНИТЬ, УДАЛИТЬ), например:

CREATE TABLE - создать таблицу

ALTER TABLE - изменить таблицу

DROP TABLE - удалить таблицу

или

CREATE VIEW - создать представление

DROP VIEW - удалить представление

ALTER VIEW – изменить представление

Замечание: В стандарте предусмотрены еще инструкции для схем и доменов. Здесь они не приведены, так как домены в СУБД обычно не реализуются, а схемы иногда определяют косвенным образом, через пользователей, которые ими владеют.

**Часть синтаксиса инструкции создания таблицы:**

CREATE TABLE имя\_таблицы

(столбец

[,{столбец|именованное\_ограничение\_целостности}] .... )

где

столбец ::= имя\_столбца тип [неимен\_огр\_целостности] DEFAULT значение\_по\_умолчанию

неимен\_огр\_целостн ::= NULL | NOT NULL | UNIQUE | PRIMARY KEY

Замечание: Неименованные ограничения целостности не имеют имени заданного пользователем, но СУБД называет их своими именами.

Именованные ограничения целостности называются ещё ограничениями уровня таблицы. В простейшем варианте их синтаксис:

именованное\_ограничение\_целостности::= CONSTRAINT имя\_ограничения определение\_ограничения

Пример простой инструкции create table:

CREATE TABLE qq (c1 NUMBER(3) PRIMARY KEY, c2 CHAR(5))

Другой вариант:

CREATE TABLE qq (c1 NUMBER(3), c2 CHAR(5),

CONSTRAINT myPK PRIMARY KEY (с1))

Основные типы данных ORACLE используемые в SQL:

|  |  |
| --- | --- |
| **Тип данных** | **Описание типа данных** |
| **Blob** | Тип данных BLOB, содержащий двоичные данные максимального размера 4 гигабайта. |
| **Byte** | Целый тип для представления 8-разрядных целых чисел без знака, находящихся в диапазоне от 0 до 255. |
| **Char** | Тип данных Oracle CHAR, содержащий строку знаков фиксированной длины с максимальным размером 2000 байтов. |
| **Clob** | Тип данных CLOB, содержащий символьные данные, основан на стандартном наборе знаков на сервере, максимальный размер которого — 4 гигабайта. |
| **Date** | Тип данных DATE хранит значения в виде точек времени (т.е. дату и время). Тип данных DATE запоминает год (включая век), месяц, день, часы, минуты и секунды (после полуночи). ORACLE может хранить даты в диапазоне от 1 января 4712 года до н.э. до 31 декабря 4712 года нашей эры. Стандартный формат даты ORACLE для ввода и вывода имеет вид DD-MON-YY |
| **LongRaw** | Тип данных LONGRAW, содержащий двоичные данные переменной длины с максимальным размером 2 гигабайта. |
| **NChar** | Тип данных NCHAR, содержащий строку знаков фиксированной длины для хранения в национальном наборе знаков в базе данных максимального размера 2000 байтов (а не знаков). |
| **Number** | Тип данных NUMBER, содержащий числовые данные переменной длины с максимальной точностью и масштабом 38. |
| **Timestamp** | Тип данных TIMESTAMP, содержащий дату и время (включая секунды), с размером в диапазоне от 7 до 11 байтов. |
| **TimestampLocal** | Тип данных TIMESTAMP WITH LOCAL TIMEZONE содержащий дату, время и ссылку на исходный часовой пояс, с размером в диапазоне от 7 до 11 байтов |
| **VarChar2** | Тип данных VARCHAR2, содержащий строку знаков переменной длины с максимальным размером 4000 байтов |

Виды ограничений целостности:

* NOT NULL | NULL — ограничитель NOT NULL запрещает вводить и хранить пустые значения;
* UNIQUE - определяет уникальный ключ; формат ограничения уровня таблицы [CONSTRAINT имя\_ограничения] UNIQUE (столбец1, столбец2, ....)
* PRIMARY KEY --- обеспечивает уникальность набора значений перечисленных полей; естественно, пустые значения в отличие от UNIQUE запрещены; формат ограничения для уровня таблицы:

[CONSTRAINT имя\_ограничения] PRIMARY KEY (столбец1, столбец2, ....)

* FOREIGN KEY --- указывает, что перечисленные столбцы составляют внешний ключ; с каждым внешним ключом связаны первичный или уникальный ключи (для них заданы ограничения типа UNIQUE или PRIMARY KEY); формат на уровне таблицы

CONSTRAINT имя\_ограничения FOREIGN KEY (столбец1, столбец2, ....) REFERENCES таблица (столбец1, [столбец2], .....)

* CHECK --- задает условие, которому должны удовлетворять значения столбцов в каждой строке; формат

[CONSRAINT имя\_ограничения] CHECK (условие)

Еще примеры:

Первичный и внешний ключи:

CREATE TABLE example5 (

table\_id NUMBER(10) PRIMARY KEY,

example\_table\_id NUMBER(10) REFERENCES example4(table\_id),

first\_name VARCHAR2(50),

last\_name VARCHAR2(200) );

Ограничения целостности и значения по умолчанию:

CREATE TABLE projx (

projno NUMBER (4) NOT NULL

, pname VARCHAR2 (14) CHECK (SUBSTR(pname,1,1) BETWEEN 'A' AND 'Z')

, bdate DATE DEFAULT TRUNC ( SYSDATE )

, budget NUMBER (10,2) );

Версия с запросом AS SELECT:

CREATE TABLE dept\_copy AS SELECT \* FROM dept;

CREATE TABLE emps ( name, department )

AS SELECT ename, dname FROM emp, dept WHERE emp.deptno = dept.deptno;

Запрос AS SELECT передающий структуру таблицы, но не данные:

CREATE TABLE dept\_new

AS (SELECT \* FROM dept WHERE 1=0);

**КОМАНДА ИЗМЕНЕНИЯ ТАБЛИЦЫ – ALTER TABLE**

Оператор ALTER TABLE позволяет добавлять в таблицу столбцы и ограничения, изменять размер и тип данных существующих столбцов, изменять установки NULL/NOT NULL, удалять ограничения, изменять схему хранения таблицы.

**Синтаксис изменения таблицы:**

ALTER TABLE имя\_пользователя.таблица

ADD [CONSTRAINT] описание\_столбца | ограничение\_таблицы,

MODIFY [RENAME| CONSTRAINT] описание\_столбца, описание\_столбца,...

DROP COLUMN|CONSTRAINT ограничение, ограничение,...

Рассмотрим примеры использования команды ALTER применительно к таблице. В нижеприведенном примере мы производим добаление столбца «sem\_pol» с типом данных NUMBER. При добавлении столбца можно использовать дополнитльные опции, как при создании таблицы (DEFAULT, NOT NULL и т.д.), кроме командой ALTER можно добавить несколько столбцов одной командой, для этого нужно их перечислить через запятую.

|  |
| --- |
| ALTER TABLE s\_fiz\_lic3 ADD sem\_pol NUMBER(1); |

В следующий пример проиллюстрировано, как произвести перемименование столбца. В этом прмере меняем имя созданного столбца с «sem\_pol» на «sem\_poloj».

|  |
| --- |
| ALTER TABLE s\_fiz\_lic3 RENAME COLUMN sem\_pol TO sem\_poloj; |

В третьем примере мы производим учеличение размерности «sem\_poloj» с 1 до 10. Небходимо отметить, что если столбец таблицы содержит данные, то мы можем только увеличить значение размерности, уменьшить размерность мы можем в том случаи, если все строки по этому столбу содержат значение NULL.

|  |
| --- |
| ALTER TABLE s\_fiz\_lic3 MODIFY sem\_poloj NUMBER(10); |

В данный пример показывает как проивзодить удаление столбца таблицы.

|  |
| --- |
| ALTER TABLE s\_fiz\_lic3 DROP COLUMN sem\_poloj; |

Для индентификации строк и оптимизации таблицы используют первичный ключ. Первичный ключ может состоять из одного столбца или быть составным (несколько столбцов). Такой уникальный столбец (или уникальная группа столбцов), используемый для идентификации каждой строки и хранения всех строк по отдельности, называется первичным ключом таблицы.

Первичные ключи таблицы - важный элемент в структуре базы данных. Первичный ключ гарантируют, что ваши данные имеют определенную целостность. Если первичный ключ правильно используется и поддерживается, вы будете знать, что нет пустых строк таблицы, и что каждая строка отличается от любой другой строки.

Следующий пример показывает, как нужно добавлять перчиный ключ таблицы. Служебные слова USING INDEX используются опцинально, указывая эти слова, производится автоматическое создание системного индекса по данному столбцу таблицы.

|  |
| --- |
| ALTER TABLE s\_fiz\_lic ADD PRIMARY KEY (KOD) USING INDEX; |

В следующем примере мы производим удаление вторичного (внешнего) ключа.

|  |
| --- |
| ALTER TABLE s\_fiz\_lic3 DROP CONSTRAINT fk\_dolj; |

В следующем примере мы производим создание вторичного ключа, которые представляет собой ограничение целостности для изменяемой таблицы. Вторичный ключ определяет перечень тех значений, которые будет принимать столбец изменяемой таблицы. Общий синтаксис:

**ALTER** **TABLE** имя\_таблицы **ADD** **CONSTRAINT** имя\_вторичного\_ключа **FOREIGN** **KEY** (список\_столбцов) **REFERENCES** имя\_родит\_таблицы (список\_столбцов);

При создании вторичного (внешенго) ключа необходимо учесть, что тип данных и размерность столбцов обеих таблиц должны совпадать. У родительской таблицы (таблица, которая определяет перечень значений) по этому столбцу должен быть создан первичый ключ, данный столбец является родительским по отношению к столбцу другой таблицы.

|  |
| --- |
| ALTER TABLE s\_fiz\_lic3 ADD CONSTRAINT fk\_dolj FOREIGN KEY (kod\_dolj) REFERENCES dolj (KOD); |

Два следующих примера иллюстрируют, как производить включение и отключение вторичного ключа. Часто требуется производить отключением проверки вводимых значений, для этого необходимо произвести отключение вторичного ключа.

|  |
| --- |
| ALTER TABLE s\_fiz\_lic3 MODIFY CONSTRAINT fk\_dolj DISABLE;  ALTER TABLE s\_fiz\_lic3 MODIFY CONSTRAINT fk\_dolj ENABLE; |

***КОМАНДА УДАЛЕНИЯ – DROP***

Оператор DROP производит удаление объекта из БД.

**Синтаксис удаления объекта:**

DROP тип\_объекта имя\_объекта

Рассмотрим примеры удаления объектом БД, а так же опции, которые могут применяться при удалении таблицы. В первом примере произовдится удаление таблицы с именем «s\_fiz\_lic2» при этом таблица отправляется в спецмальную область для хранения удаленных объектов, называемой «Корзина».

|  |
| --- |
| DROP TABLE s\_fiz\_lic2; |

Для удаления таблицы полностью без отправки ее в корзину применяется опция PURGE. Если производится удаление таблицы, на которую ссылается другая таблица, то есть существует ограничение целостности, то такую таблицу нельзя удалить без опции CASCADE CONSTRAINTS, данная опция удаляет в месте с таблицей все ссылающиеся на нее ограничения целостности.

|  |
| --- |
| DROP TABLE dolj CASCADE CONSTRAINTS PURGE; |

В следующем примере мы производим удаление ранее созданного нами представления.

|  |
| --- |
| DROP VIEW v\_fiz\_lic; |

В этом примере производим удаление последовательности с именем «my\_seq».

|  |
| --- |
| DROP SEQUENCE my\_seq; |

Пересоздадим удаленные объекты

|  |
| --- |
| CREATE TABLE dolj  (kod NUMBER(10) NOT NULL PRIMARY KEY,  naimen VARCHAR2(50));  ALTER TABLE s\_fiz\_lic3 ADD CONSTRAINT fk\_dolj FOREIGN KEY (kod\_dolj) REFERENCES dolj (KOD); |