**Встроенные операторы SQL.**

Как было отмечено ранее SQL (Structured Query Language) - структурированный язык запросов, позволяет оперировать данными в реляционных базах данных. Стандарт SQL определен Американским национальным институтом стандартов и ISO в качестве международного стандарта. Целью данного издания не является полное и всеобъемлющее освещение синтаксиса SQL, для этого есть специализированные справочники и документация [1-10], мы же постараемся на нескольких простых конкретных примерах показать Вам всю элегантность и мощь SQL. Все примеры, приведенные ниже даны, применительно к ER-диаграмме ДОКТОР-ПАЦИЕНТ, приведенной на рис.10.

Что же из себя представляет SQL - программа? Чаще всего это оформленная в виде отдельного файла программная конструкция, написанная в любом текстовом редакторе с учетом требований синтаксиса языка SQL. Такая форма представления SQL программы - называется скриптом и предназначена для выполнения на сервере, например с помощью специальной терминальной программы SQL+ (строка запуска скрипта в SQL+: @<путь>/<имя скрипта>.sql). Считается хорошим тоном наличие в скрипте комментариев. Для выделения строчных комментариев используется следующий набор символов: --.

Перед тем как перейти непосредственно к рассмотрению использования основных SQL операторов еще несколько слов об организации проектирования БД. Процесс создания БД - это сложный многоэтапный процесс, причем как правило в нем принимают участие большое число разработчиков, поэтому очень важным является правильная организация внесения изменений в БД. Для этих целей очень часто используется технология "РАЗДЕЛЕНИЯ ЗАДАЧ", которая заключается в следующем: каждый разработчик, выполняя конкретную часть создания или модификации БД, оформляет все производимые им изменения в виде скриптов (т.е. отдельных файлов), архивные версии которых перед запуском на сервере размещаются на отдельном, специально выделенном, носителе (диске сервера), причем каждое такое изменение БД оформляется в виде отдельной задачи, имеющей свой уникальный номер. Например, в задаче 000001/VER001/ (физически это просто каталог на диске) находятся все скрипты (файлы) по первоначальному созданию БД. Такой подход позволяет максимально удобно решать задачи "Контроля версий", производить миграцию созданной БД на другой сервер (достаточно на новом сервере выполнить все задачи в последовательности следования номеров задач), обеспечивает достаточно высокий уровень безопасности, возможности отката на любую предыдущую позицию (этап разработки БД) и многое другое. В дальнейшем при выполнении практических примеров приведенных в данном издании советуем вам придерживаться именно этой технологии. Если вы на каком-то из этапов допустили ошибку (которую выявили на этапе выполнения скрипта в БД) не надо исправлять текст непосредственно этого скрипта, оформьте новую версию выполняемой задачи, в которой разместите исправленный скрипт. Это позволит Вам всегда отслеживать все Ваши ошибки. Последняя рекомендация, которую хотелось бы дать, заключается в том, что если в задачу входит несколько скриптов, то целесообразно оформить один дополнительный запускающий скрипт, например start.sql, в который поместить запуск всех остальных скриптов. Поверьте на слово - это значительно съэкономит ваше время в дальнейшем. Например, если в задачу 000001/VER001/ входят файлы: db1.sql., db2.sql, db3.sql, то файл start.sql может быть представлен следующим способом:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* start.sql \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Spool 000001.log

@db1.sql

@db2.sql

@db3.sql

spool off

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

При этом необходимо помнить, что все файлы должны быть в кодировке той среды, из которой вы собираетесь запускать SQL+ (KOI8, Win1251, DOS).

**ОПЕРАТОРЫ СОЗДАНИЯ ОБЪЕКТОВ БД.**

Перед тем как работать с данными с БД ее надо создать, для этих целей используется специальная группа операторов, предназначенных для создания объектов базы данных, все операторы данной группы начинаются с ключевого слова CREATE.

**CREATE DATABASE**

Создает базу данных. Задает и определяет максимальное число экземпляров файлов данных и журнальных файлов, устанавливает режим архивирования.

**Filespec** (журнальные файлы)::=

-- Скрипт создания БД CLINICS

spool clinic.log

connect internal

startup nomount pfile=/oracle/dbs/initclinic.ora

-- создаем базу данных с именем clinics

create database "clinics"

maxinstances 1

maxlogfiles 10

character set "RU8PC866"

national character set "RU8PC866"

datafile

'/oracle/db/system01.dbf' size 100M

logfile

'/oracle/db/lo g01.dbf' size 1M,

'/oracle/db/log02.dbf' size 1M;

disconnect

spool off

**CREATE TABLESPASE**

Создает в базе данных область для хранения таблиц, сегментов и индексов, определяет файлы базы данных, параметры хранения по умолчанию и режим табличного пространства (автономный или оперативный).

**CREATE USER**

Создает пользователя базы данных. (синтаксис команды приведен упрощенно, за дополнительной информацией обратитесь к документации).

**CREATE SCHEMA**

Создает несколько таблиц и представлений и предоставляет некоторые привилегии в одной транзакции.

**CREATE TABLE**

Создает новую таблицу БД, определяя ее столбцы, правила целостности и параметры хранения.

Пример:

|  |  |
| --- | --- |
| create table | DOCTORS( |
| DC\_NNN | NUMBER(12,0) |
| , DC\_DC\_NNN | NUMBER(12,0) |
| , DC\_NAME | VARCHAR2(255) |
| , DC\_CS\_NNN | NUMBER(12,0) |
| , DC\_DIPLOMA\_NUMBER | NUMBER(12,0) |
| , DC\_SPECIALTY\_NNN | NUMBER(12,0) |
| , DC\_SHAT\_NNN | NUMBER(12,0) |
| , DC\_CALENDAR | NUMBER(12,0) |

) tablespace users;

**CREATE SYNONYM**

Создает синоним для таблицы, представления, последовательности, хранимой процедуры или функции, пакетной процедуры, моментальной копии или другого синонима.

**Пример: Сначала удаляем публичный синоним для таблицы DOCTORS, а потом его заново создаем.**

drop public synonym DOCTORS;

create public synonym DOCTORS for DOCTORS;

**CREATE INDEX**

Создает индекс для заданных столбцов таблицы или кластера.

**Пример:**

create UNIQUE index I\_DOCTORS on DOCTORS (

DC\_NNN

) tablespace users;

**CREATE TRIGGER**

Создает триггер базы данных.

**CREATE SEQUENCE**

Создает новую последовательность для генерации первичных ключей.

**Пример:**

create sequence S\_DOCTORS;

**Пример: Приведем полный текст скрипта для создания триггера для таблицы DOCTORS, который будет следить за автоматическим увеличением значения поля DC\_NNN на единицу при добавлении каждой новой записи в таблицу DOCTORS, что потребует от нас использование и такой конструкции - как сиквенс (генератор последовательностей).**

-- Сначала удаляем предыдущие изменения в базе (если конечно они были сделаны)

drop sequence S\_DOCTORS;

drop trigger tr\_DC\_NNN;

drop public synonym DOCTORS;

-- создаем таблицу

-- назначаем права доступа

-- создаем публичный синоним

CREATE PUBLIC SYNONYM DOCTORS FOR DOCTORS;

-- создаем сиквенс и устанавливаем его начальное значение в единицу

create sequence S\_DOCTORS

start with 1;

-- создаем индексы

create unique index i\_dc\_nnn on doctors(dc\_nnn);

create index i\_dc\_name on doctors(dc\_name);

-- создаем триггер

create trigger tr\_dc\_nnn

before insert

on doctors

for each row

begin

select dc\_nnn.nextval into :new.dc\_nnn from dual;

end;