**Триггеры базы данных.**

***Триггер*** базы данных – это оформленный специальным образом именованный блок PL/SQL, хранящийся в базе данных. Каждый триггер связан с определенной таблицей и автоматически запускается ORACLE при выполнении одного из DML-операторов  (INSERT, DELETE, UPDATE)  или их совокупности над этой таблицей.

***Назначение триггеров.*** Триггеры могут быть использованы:

1. для реализации сложных ограничений целостности данных, которые не могут быть осуществлены стандартным образом при создании таблицы;
2. предотвращения неверных транзакций;
3. выполнения процедур комплексной проверки прав доступа и секретности данных;
4. генерации некоторых выражений на основе значений, имеющихся в столбцах таблиц;
5. реализации сложных бизнес-правил для обработки данных (возможность отследить «эхо», т.е. возможность при изменении одной таблицы, обновлять данные связанных с ней таблиц).

***Создание и включение триггеров.*** Для создания и автоматического включения триггера применяется следующий общий синтаксис:

CREATE  [OR  REPLACE]  TRIGGER  имя\_триггера

{BEFORE  |  AFTER}

{INSERT  |  DELETE  |  UPDATE  [OF  список\_столбцов]}

ON  имя\_таблицы   [FOR  EACH  ROW]   [WHEN  условие]

< PL/SQL\_блок >

При наличии ключевых слов OR REPLACE триггер создается заново, если он уже существует.

Конструкция BEFORE | AFTER указывает на момент запуска триггера. Вариант BEFORE означает, что триггер будет запускаться перед выполнением активизирующего DML-оператора; вариант AFTER означает, что триггер будет запускаться после выполнения активизирующего DML-оператора.

Конструкция INSERT | DELETE | UPDATE [OF список\_столбцов] указывает тип активизирующего триггер DML-оператора. Разрешается, используя логическую операцию OR, задать совокупность активизирующих операторов, например: INSERT OR DELETE. Если при использовании варианта UPDATE указан список столбцов, то триггер будет запускаться при модификации одного из указанных столбцов; если список столбцов отсутствует, то триггер будет запускаться при изменении любого из столбцов связанной с триггером таблицы.

Конструкция FOR EACH ROW указывает на характер воздействия триггера: строковый или операторный. Если конструкция FOR EACH ROW присутствует, то триггер является строковым; при отсутствии ее триггер является операторным. Операторный триггер запускается один раз до или после выполнения активизирующего триггер DML-оператора независимо от того, сколько строк в связанной с триггером таблице подвергается модификации. Строковый триггер запускается один раз для каждой из строк, которая подвергается модификации DML-оператором, активизирующим триггер.

С помощью ключевого слова WHEN можно задать дополнительное ограничение на строки связанной с триггером таблицы, при модификации которых может быть запущен триггер.

Конструкция PL/SQL\_блок представляет блок PL/SQL, который ORACLE запускает при активизации триггера.

***Классификация триггеров.*** В основном различают двенадцатьтипов триггеров. Тип триггера определяется сочетанием следующих трех параметров:

1. характером воздействия триггера на строки связанной с ним таблицы (строковый или операторный);
2. моментом запуска триггера: до (BEFORE) или после (AFTER) исполнения активизирующего триггер DML-оператора;
3. типом активизирующего триггер DML-оператора (INSERT, DELETE, UPDATE);

***Порядок активизации триггеров.*** Если у таблицы имеется несколько типов триггеров, то они активизируются по следующей схеме:

1. выполняется операторный триггер BEFORE (если их несколько, то ничего о порядке их выполнения сказать нельзя);
2. выполняется строковый триггер BEFORE;
3. выполняется активизирующий триггер DML-оператор с последующей проверкой всех ограничений целостности данных;
4. выполняется строковый триггер AFTER с последующей проверкой всех ограничений целостности данных;
5. выполняется операторный триггер AFTER.

***Триггерные предикаты.*** Если в триггере указывается совокупность активизирующих триггер DML-операторов (например, INSERT OR DELETE), то для распознавания того, какой конкретно из DML-операторов выполняется над связанной с триггером таблицей, используются триггерные предикаты: INSERTING, DELETING, UPDATING. Они представляют собой логические функции, возвращающие TRUE, если тип активизирующего оператора совпадает с типом предиката, и FALSE – в противном случае. Для задания одних и тех же действий в случае выполнения различных DML-операторов в условном операторе триггерные предикаты объединяются с помощью логических операций.

***Псевдозаписи.*** Для строковых триггеров существуют специальные конструкции, которые позволяют при выполнении DML-операторов над строкой таблицы, обращаться как к старым значениям, которые находились в ней до модификации, так и к новым, которые появятся в строке после ее модификации. Эти конструкции называются псевдозаписями и обозначаются old и new. Структура этих псевдозаписей идентична структуре строки модифицируемой таблицы, но оперировать можно только отдельными полями псевдозаписи. Обращение к полям псевдозаписи происходит по следующей схеме: перед old или new ставится символ «:», далее через точку указывается название поля. Значения, которые принимают поля псевдозаписи при выполнении активизирующих DML-операторов, определяются следующим образом.

Оператор INSERT – псевдозапись :new  эквивалентна вставляемой строке, а псевдозапись :old во всех полях имеет значение NULL.

Оператор DELETE – псевдозапись :old эквивалентна удаляемой строке, а псевдозапись :new во всех полях имеет значение NULL.

Оператор UPDATE – псевдозапись :new  эквивалентна строке, полученной в результате модификации, а псевдозапись :old во всех полях имеет исходное значение строки.

***Включение, выключение триггеров.*** Хранящийся в базе данных триггер можно временно отключить, не удаляя его из базы данных. Для этого используется следующая команда:

ALTER  TRIGGER  имя\_триггера  DISABLE;

Включить триггер через некоторый промежуток времени можно, используя команду

ALTER  TRIGGER  имя\_триггера  ENABLE;

Запретить или разрешить запуск всех триггеров, связанных с некоторой таблицей, можно с помощью команды

ALTER  TABLE  имя\_таблицы  {DISABLE | ENABLE}  ALL  TRIGGERS;

где вариант DISABLE используется для отключения, а вариант ENABLE – для включения всех триггеров данной таблицы.

***Удаление триггеров из базы данных.*** Уничтожение триггера, т. е. удаление триггера из базы данных осуществляется с помощью следующей команды:

DROP  TRIGGER  имя\_триггера;

***Получение информации о триггерах.*** Триггеры хранятся в базе данных, поэтому информацию о них можно получить из представления словаря данных USER\_TRIGGERS, например, следующей командой:

SELECT  \*  FROM  USER\_TRIGGERS;

***Примеры:триггеры.***

**1.** Создать триггер, который перед вставкой очередной строки в таблицу KNIGA\_POSTAVKA проверяет наличие указанного кодакниги в таблице KNIGA. При отсутствии указанного кода книги в таблице KNIGA должно генерироваться исключение с выдачей соответствующего сообщения.

Добавление новых строк в таблицу KNIGA\_POSTAVKA выполняется оператором INSERT. Поскольку триггер должен запускаться перед выполнением каждого оператора INSERT, следовательно, он должен быть строковым BEFORE-триггером. Для сохранения целостности данных необходимо проверить, имеются ли вносимые коды книг и в таблице KNIGA. Для этого с помощью однострочного оператора SELECT осуществляется выборка информации из таблицы KNIGA, где в условии выборки используется поле КОД\_КНИГИ псевдозаписи :new. Если количество строк с данным кодом книги в таблице KNIGA окажется равным нулю, будет сгенерировано исключение и выдано соответствующее сообщение.

Создание триггера TR1 выполняется вводом следующего оператора:

CREATE  OR  REPLACE  TRIGGER  TR1

     BEFORE  INSERT  ON  KNIGA\_POSTAVKA

     FOR  EACH  ROW

DECLARE

     KOL  NUMBER(4);

BEGIN

     SELECT  COUNT(\*)  INTO  KOL  FROM  KNIGA

          WHERE  КОД\_КНИГИ = :NEW.КОД\_КНИГИ;

         IF KOL =0 THEN

                RAISE\_APPLICATION\_ERROR

                      (-20212,'В таблице отсутствует информация о книге');

            END IF;

END  TR1;/

Действие триггера TR1 может быть проверено выполнением следующего оператора, осуществляющего вставку строки в таблицу KNIGA\_POSTAVKA и тем самым вызывающих активизацию триггера TR2:

INSERT  INTO  KNIGA\_POSTAVKA  VALUES(21,15,’Иванов’,15,

’20-JAN-2006’);

Поскольку код книги 15 отсутствует в таблице KNIGA, то будет сгенерировано исключение и выдано соответствующее сообщение.

**2.** Создать триггер, который запрещает вносить в таблицу KNIGA строки со значением поля ЦЕНА больше, чем 5000 рублей, а также осуществлять увеличение цены книг, информация о которых хранится в таблице KNIGA, более чем на 20 %. При нарушении данного требования должно генерироваться исключение с выдачей соответствующего сообщения.

Так как внесение новых строк в таблицу KNIGA осуществляется в результате выполнения оператора INSERT, а значение поля ЦЕНА в таблице KNIGA, содержащего цену книги, может быть изменено в результате выполнения оператора UPDATE, то в триггере указывается совокупность запускающих DML-операторов. Поскольку триггер должен запускаться перед выполнением каждого из указанных DML-операторов, следовательно, он является строковым BEFORE-триггером. Так как действия, выполняемые триггером, различны для каждого из запускающих DML-операторов, модифицирующих таблицу KNIGA, то для распознавания типа DML-оператора используются соответствующие триггерные предикаты INSERTING и UPDAITING. Вследствие того что при вставке новых строк проверке должно быть подвергнуто новое значение поля ЦЕНА, а при модификации значения поля ЦЕНА новое значение должно сравниваться со старым значением, необходимо использовать псевдозаписи :new и :old.

Создание триггера TR2 выполняется вводом следующего оператора:

CREATE OR REPLACE TRIGGER TR2

  BEFORE INSERT OR UPDATE OF ЦЕНА ON KNIGA

  FOR EACH ROW

BEGIN

  IF INSERTING THEN

    IF :NEW.ЦЕНА > 5000 THEN

      RAISE\_APPLICATION\_ERROR(-20102,

                              'В таблицу KNIGA нельзя вносить записи с ценой книги > 5000');

    END IF;

  END IF;

  IF UPDATING THEN

    IF :NEW.ЦЕНА > :OLD.ЦЕНА \* 1.2 THEN

      RAISE\_APPLICATION\_ERROR(-20103,

                              'В таблице KNIGA нельзя изменять цену книги более чем на 20 %');

    END IF;

  END IF;

END TR2;

/

Действие триггера TR2 может быть проверено выполнением следующих операторов, которые, осуществляя вставку строк в таблицу KNIGA и обновление строк в таблице KNIGA, тем самым вызывают его активизацию.

Оператор вставки строк в таблицу KNIGA, вызывающие активизацию триггера TR2:

INSERT  INTO  KNIGA  VALUES( 21, 'Дюна', 'Герберт', 5268, 'Аст',

 'Фантастика');

Оператор обновления строк в таблице KNIGA, вызывающие активизацию триггера TR2:

UPDATE  KNIGA  SET ЦЕНА=6000;

Поскольку эти операторы нарушают требования, предъявляемые к значению и модификации цены книг, то во всех случаях будет сгенерировано исключение и выдано соответствующее сообщение.

**3.** Создать триггер, который в созданную таблицу STAT, содержащую столбцы:

имя издательства – IZD,

количество книг в жанре «Роман» – KOL\_ROM,

количество книг в жанре «Фантастика» – KOL\_FAN,

при каждой модификации таблицы KNIGA формирует и заносит в соответствующие столбцы таблицы STAT суммарное количество книг по каждому из издательств в разрезе указанных тематик: «Роман» и «Фантастика».

Модификация таблицы KNIGA осуществляется выполнением следующих DML-операторов: INSERT, DELETE или оператора UPDATE, модифицирующего значения столбца ЖАНР в таблице KNIGA. Так как действия по формированию информации таблицы STAT выполняются после выполнения каждого из модифицирующих таблицу KNIGA операторов, то по типу это операторный AFTER-триггер. Поскольку действия, выполняемые триггером, одинаковы для всех типов активизирующих его операторов, то триггерные предикаты не используются. Перед созданием триггера должна быть создана таблица STAT.

Создание таблицы STAT может быть выполнено вводом следующей совокупности операторов:

DROP  TABLE  STAT;

CREATE  TABLE  STAT

( IZD  VARCHAR2(15),

  KOL\_ROM  NUMBER(7),

  KOL\_FAN  NUMBER(7)

);

Создание триггера TR3 выполняется вводом следующего оператора:

CREATE  OR  REPLACE  TRIGGER  TR3

     AFTER  INSERT  OR  DELETE  OR  UPDATE  OF  ЖАНР

     ON  KNIGA

DECLARE

CURSOR  V1  IS  SELECT  ИЗДАТЕЛЬСТВО,

COUNT(НАЗВАНИЕ)  KOL1

          FROM  KNIGA  WHERE  ЖАНР = 'Роман'

          GROUP  BY  ИЗДАТЕЛЬСТВО;

     CURSOR  V2  IS  SELECT  ИЗДАТЕЛЬСТВО,

     COUNT(НАЗВАНИЕ)  KOL2

          FROM  KNIGA  WHERE  ЖАНР = 'Фантастика'

          GROUP  BY  ИЗДАТЕЛЬСТВО;

BEGIN

DELETE  FROM  STAT;

     FOR  Z1  IN  V1  LOOP

          INSERT  INTO  STAT  VALUES(Z1.ИЗДАТЕЛЬСТВО,

          Z1.KOL1, 0);

     END  LOOP;

     FOR  Z1  IN  V2  LOOP

          UPDATE  STAT  SET  KOL\_FAN = Z1.KOL2

          WHERE  IZD = Z1.ИЗДАТЕЛЬСТВО;

          IF  SQL%NOTFOUND  THEN

              INSERT  INTO  STAT  VALUES(Z1.ИЗДАТЕЛЬСТВО, 0,

              Z1.KOL2);

          END IF;

     END LOOP;

END  TR3;

/

Действие триггера может быть проверено выполнением следующих операторов, которые, осуществляя вставку строк в таблицу KNIGA, удаление строк и обновление строк в таблице KNIGA, тем самым вызывают активизацию триггера TR3.

Операторы вставки строк в таблицу KNIGA, вызывающие активизацию триггера TR3:

INSERT  INTO  KNIGA  VALUES( 46, 'Еретики Дюны', 'Герберт',368,

'Аст', 'Фантастика');

INSERT  INTO  KNIGA  VALUES( 42, 'Ингвар и Ольха',

 'Никитин',168, 'Аст', 'Роман ');

Операторы удаления строк из таблицы KNIGA, вызывающие активизацию триггера TR3:

DELETE  KNIGA  WHERE  АВТОР = 'Герберт';

DELETE  KNIGA  WHERE  НАЗВАНИЕ = 'Казаки';

Операторы модификации строк в таблице KNIGA, вызывающие активизацию триггера TR3:

UPDATE  KNIGA  SET  ЖАНР='Фантастика'  WHERE  НАЗВАНИЕ =

 'Ингвар и Ольха';

Просмотр информации в таблице STAT можно выполнить следующим оператором:

SELECT  \*  FROM  STAT;

**4.**  Создать триггер, запрещающий обновлять данные о сотрудниках в таблице EMP в выходные дни.

     CREATE OR REPLACE TRIGGER d\_emp

           BEFORE INSERT OR UPDATE OR DELETE ON emp

     BEGIN

           IF TO\_CHAR (SYSDATE,'DY') = 'СБТ'   OR

               TO\_CHAR (SYSDATE,'DY') = 'ВСК'   THEN

                         RAISE\_APPLICATION\_ERROR

                                (-20102,'Таблицу emp в выходные дни изменять нельзя.');

           END IF;

     END d\_emp;

**5.**  Создать триггер,  сохраняющий информацию, удаленную из таблицы EMP,  в таблице H\_EMP.

    CREATE OR REPLACE TRIGGER h\_emp

         BEFORE DELETE ON emp

         FOR EACH ROW

    BEGIN

         INSERT INTO h\_emp

             VALUES (seq\_nemp.nextval, :old.ename, :old.empno, :old.job ,:old.deptno, sysdate);

     END h\_emp;

**3.** Создать триггер, который при изменении номера отдела в таблице DEPT вносит соответствующие изменения в таблицу EMP.

    CREATE OR REPLACE TRIGGER  ch\_dept

         BEFORE UPDATE ON dept

         FOR EACH ROW

    BEGIN

           IF UPDATING  AND  :old.deptno!=:new.deptno  THEN

                UPDATE emp SET   emp.deptno=:new.deptno

                     WHERE   emp.deptno=:old.deptno;

           END IF;

    END ch\_dept;

Перед изменением номера отдела в таблице DEPT необходимо отключить ограничение целостности FK\_DEPTNO:

ALTER TABLE emp DISABLE CONSTRAINT fk\_deptno;

UPDATE dept SET deptno=50  WHERE deptno=10;

ALTER TABLE emp ENABLE CONSTRAINT fk\_deptno;

**6.** Создать триггер,  который при добавлении  заказа в таблицу SITEM проверяет наличие  информации о данном товаре в таблице  SPROD.

     CREATE OR REPLACE TRIGGER   ins\_sitem

           BEFORE INSERT ON   sitem

           FOR EACH ROW

     DECLARE

           x   number(4);

     BEGIN

           SELECT COUNT(\*)   INTO x FROM  sprod

                  WHERE   prodid=:new.prodid;

           IF X=0 THEN

                RAISE\_APPLICATION\_ERROR

                      (-20212,'В таблице sprod отсутствует информация о товаре');

            END IF;

     END   ins\_sord;

**7.**  Создать триггер, обновляющий количество в таблице SPROD при изменении таблицы SITEM.

     CREATE OR REPLACE TRIGGER  ch\_sprod

         AFTER  INSERT  OR  UPDATE  OF  qty   OR DELETE   ON   sitem

         FOR EACH ROW

     BEGIN

        IF INSERTING THEN

              UPDATE sprod

                  SET p\_qty=p\_qty-:new.qty

                      WHERE   prodid=:new.prodid;

         ELSIF  UPDATING THEN

              IF   :new.qty>:old.qty   THEN

                  UPDATE   sprod

                     SET   p\_qty=p\_qty-(:new.qty-:old.qty)

                        WHERE   prodid=:new.prodid;

              ELSE

                   UPDATE   sprod

                       SET   p\_qty=p\_qty+(:old.qty-:new.qty)

                           WHERE   prodid=:new.prodid;

           END IF;

        ELSIF DELETING THEN

               UPDATE    sprod

                     SET   p\_qty=p\_qty+:old.qty

                          WHERE   prodid=:old.prodid;

        END IF;

     END ch\_sprod;

**8.** Создать триггер, который при поступлении нового заказа в таблицу SITEM  вносит соответствующие изменения для данного заказчика в таблицу CR\_SCUST  (снять со счета заказчика или отпустить товар в кредит).

 CREATE OR REPLACE TRIGGER cr\_scust

        BEFORE INSERT ON siteM

        FOR EACH ROW

  DECLARE

            ncustid number(4);

            s1 number(7);

            x1 number (7);

 BEGIN

       SELECT custid INTO ncustid FROM sord

                WHERE   ordid=:new.ordid;

       SELECT  debet  INTO  s1  FROM  cr\_scust

                  WHERE    custid=ncustid;

        x1:= s1-(:new.price\*:new.qty);

        IF   x1>=0   THEN

              UPDATE   cr\_scust SET debet=x1

                     WHERE   custid=ncustid;

        ELSE

               UPDATE   cr\_scust SET debet=0, credit=-x1

                     WHERE custid=ncustid;

        END IF;

 END   cr\_scust;

**Курсоры. Задания для самостоятельной работы.**

1. Составить программу, которая увеличивает оклад в 1.2 раза всем сотрудникам, информация о которых находится в таблице EMP и у которых суммарная заработная плата (сумма оклада и комиссионных) меньше 1200$.

1. Составить программу, которая начисляет премию сотрудникам компании, информация о которых находится в таблице EMP. Размер премии не должен превышать оклад сотрудника. Сумма оклада и премии не должна превышать 5000. Поместить в таблицу ТЕМР табельный номер, размер премии и фамилию сотрудника.

1. Составить программу, вычисляющую суммарную зарплату служащих 20-го отдела, информация о которых находится в таблице EMP. Подсчитать и вставить в таблицу ТЕМР количество служащих, которые имеют оклад выше 2000$ , количество служащих, у которых комиссионные больше оклада и суммарную зарплату.

1. Таблица SKLAD содержит данные о количестве деталей в ячейках склада. Составить программу, которая просматривает ячейки склада, содержащие детали с номером 5469, подсчитывает количество недостающих деталей и дополняет ячейки до 500 штук. Номер детали передать в качестве параметра. В таблицу ТЕМР заносится суммарное количество деталей, вновь поступивших на склад.

1. Составить программу, которая  проверяет наличие средств у заказчика на счету 106 для покупки 5 телевизоров Panasonic 37 и осуществляет снятие денег со счета, в противном случае вносит запись в таблицу temp, содержащую номер банковского счета (код заказчика), остаток средств на счете и текст сообщения о том, что средств недостаточно. Используются таблицы DPROD И SCUST.

1. При покупке телевизоров предприятие предоставляет систему скидок: 5% - покупка более 10 штук; 10% - более 100 штук. Составить программу, которая при оформлении заказа на покупку партии телевизоров снимает деньги со счета заказчика, учитывая систему скидок. Использовать в программе курсор с параметрами. В качестве параметров передать: номер счета заказчика, количество единиц и код товара (102, 15, 100063). Используются таблицы DPROD и SCUST.

**Процедуры. Задания для самостоятельной работы.**

**1.** Составить процедуру, вносящую изменения в таблицу SPROD при поступлении очередной партии товара. В качестве параметров передавать код товара, наименование, цену и количество. Количество суммируется, если в таблице SPROD уже имеется аналогичный товар. Таблица SPROD приведена ниже.

**2.** Составить процедуру, обеспечивающую изменение цены при переоценке товара в таблице SPROD. В качестве параметров передавать код товара и изменение цены. Поместить в таблицу TEMP сообщение, если код товара отсутствует в таблице SPROD. Таблица SPROD приведена ниже.

**3.** Составить процедуру, добавляющую новые записи в таблицу EMP при оформлении на работу новых служащих компании. В качестве параметров передавать имя, должность, табельный номер начальника, оклад, размер комиссионных,  номер отдела.

**4.** Составить процедуру, удаляющую запись из таблицы EMP при увольнении сотрудника. В качестве параметра передавать табельный номер уволенного сотрудника.

**5.** Составить процедуру, увеличивающую оклад служащего в таблице EMP. В качестве параметров передавать табельный номер сотрудника и заданную в процентах величину, на которую необходимо увеличить оклад.

**6.** Составить процедуру, начисляющую премию сотрудникам компании в таблице EMP. В качестве параметров передавать величину (в процентах), определяющую размер премии от оклада, и величину (в процентах), определяющую размер премии от комиссионных. Размер начисленной inи должен равняться сумме премии от оклада и комиссионных. Информацию о премировании занести в таблицу P\_EMP.

**7.** Составить процедуру, обеспечивающую загрузку партии деталей в ячейки склада в таблице SKLAD. В качестве параметров передавать номер партии и количество деталей, а также максимальное количество деталей, которое можно поместить в одну ячейку склада. Таблица SKLAD приведена ниже.

1. Составить функцию, подсчитывающую общее количество деталей одного вида на складе в таблице SKLAD. При вызове функция должна принимать в качестве аргумента номер партии деталей. Таблица SKLAD приведена ниже.

**Триггеры. Задания для самостоятельной работы.**

**1.**  Создать триггер, запрещающий обновлять данные о сотрудниках в таблице EMP в нерабочее время.

**2.**  Создать триггер, фиксирующий изменения таблицы SCUST в протокольной таблице H\_SCUST.

**3.** Создать триггер,  который при добавлении нового заказа в таблицу SORD проверяет наличие  информации о данном заказчике в таблице SCUST.

**4.** Создать триггер, который формирует дополнительный заказ в таблице ORD\_PROD  на поставку товара, если количество этого товара  в таблице SPROD становится меньше 10 единиц.

**5.** Создать триггер,  который  запрещает устанавливать границу минимальной зарплаты в таблице SALS  ниже 600.

**6.**  Создать триггер,  который при добавлении нового служащего в таблицу EMP или при изменении должности или оклада какого-либо служащего обеспечивает, что зарплата попадает в пределы, установленные для данной должности в таблице SALS.

**7.** Создать триггер,  который при добавлении нового заказчика в таблицу SCUST вносит соответствующие изменения в таблицу CR\_SCUST.

**8.** Создать триггер,  который при добавлении нового заказа  в таблицу SITEM подсчитывает сумму заказа в таблице SORD.

**1.  Таблицы, используемые в лабораторной работе**

Таблицы EMP и DEPT из демонстрационной базы данных ORACLE, KNIGA,SCUST, SORD, SITEM, SPROD, SALS, CR\_SCUST, H\_EMP, H\_SCUST, ORD\_PROD.

Таблицы CR\_SCUST, H\_EMP, H\_SCUST, ORD\_PROD необходимо создать самостоятельно.

**Входная таблица KNIGA:**

CREATE  TABLE  KNIGA (

КОД\_КНИГИ  NUMBER(5) CONSTRAINT  PK\_KN  PRIMARY  KEY,

НАЗВАНИЕ VARCHAR2(25)  NOT  NULL, АВТОР  VARCHAR2(20),

ЦЕНА  NUMBER(7)  CONSTRAINT  CEN\_KN  CHECK(ЦЕНА>100),

ИЗДАТЕЛЬСТВО  VARCHAR2(15),  ЖАНР  VARCHAR2(15));

INSERT INTO KNIGA VALUES(1,'Дюна','Герберт',268,'Аст', 'Фантастика');

INSERT INTO KNIGA VALUES(2,'Ингвар и Ольха', 'Никитин', 168, 'Аст', 'Роман');

INSERT INTO KNIGA VALUES(3,'Гибель Богов','Перумов',345,'Аст', 'Фантастика');

INSERT INTO KNIGA VALUES(4,'Мифы','Асприн',266,'Аст','Детектив');

INSERT INTO KNIGA VALUES(5,'Казаки','Толстой',5568,'Нова','Роман');

INSERT INTO KNIGA VALUES(6,'Еретики Дюны', 'Герберт', 368, 'Аст', 'Фантастика');

INSERT INTO KNIGA VALUES(7,'Князь Рус','Никитин',1168,'Аст', 'Роман');

INSERT INTO KNIGA VALUES(8,'Страсть','Перумов',1385,'Аст', 'Детектив');

INSERT INTO KNIGA VALUES(9,'Мифотолкование', 'Асприн', 2265, 'Нова', 'Детектив');

INSERT INTO KNIGA VALUES(10,'Война и мир', 'Толстой', 4588, 'Нова', 'Роман');

INSERT INTO KNIGA VALUES(11,'Еретики Дюны', 'Герберт', 2668, 'Нова', 'Фантастика');

INSERT INTO KNIGA VALUES(12,'Владимир','Никитин',568,'Аст', 'Детектив');

INSERT INTO KNIGA VALUES(13,'Гибель Титана', 'Кристи', 2345, 'Аст', 'Роман');

**Входная таблица EMP:**

DROP TABLE EMP cascade constraints;

CREATE TABLE **EMP** (

 EMPNO               NUMBER(4) NOT NULL,         --личный номер

 ENAME               VARCHAR2(10),                       --имя и фамилия

 JOB                 VARCHAR2(9),                              --должность

 MGR                NUMBER(4) CONSTRAINT EMP\_SELF\_KEY REFERENCES EMP (EMPNO),   --номер начальника

 HIREDATE            DATE,                           --дата поступления на работу

 SAL                 NUMBER(7,2),                    --зарплата

 COMM                NUMBER(7,2),                --комиссионные

 DEPTNO              NUMBER(2) NOT NULL,     --номер отдела

 CONSTRAINT EMP\_FOREIGN\_KEY FOREIGN KEY (DEPTNO) REFERENCES DEPT (DEPTNO),

 CONSTRAINT EMP\_PRIMARY\_KEY PRIMARY KEY (EMPNO));

INSERT INTO EMP VALUES (7839,'KING','PRESIDENT',NULL,'17.11.1981',5000,NULL,10);

INSERT INTO EMP VALUES (7698,'BLAKE','MANAGER',7839,'1.05.1981',2850,NULL,30);

INSERT INTO EMP VALUES (7782,'CLARK','MANAGER',7839,'9.06.1981',2450,NULL,10);

INSERT INTO EMP VALUES (7566,'JONES','MANAGER',7839,'2.04.1981',2975,NULL,20);

INSERT INTO EMP VALUES (7654,'MARTIN','SALESMAN',7698,'28.09.1981',1250,1400,30);

INSERT INTO EMP VALUES (7499,'ALLEN','SALESMAN',7698,'20.02.1981',1600,300,30);

INSERT INTO EMP VALUES (7844,'TURNER','SALESMAN',7698,'8.09.1981',1500,0,30);

INSERT INTO EMP VALUES (7900,'JAMES','CLERK',7698,'3.12.1981',950,NULL,30);

INSERT INTO EMP VALUES (7521,'WARD','SALESMAN',7698,'22.02.1981',1250,500,30);

INSERT INTO EMP VALUES (7902,'FORD','ANALYST',7566,'3.12.1981',3000,NULL,20);

INSERT INTO EMP VALUES (7369,'SMITH','CLERK',7902,'17.12.1980',800,NULL,20);

INSERT INTO EMP VALUES (7788,'SCOTT','ANALYST',7566,'09.12.1982',3000,NULL,20);

INSERT INTO EMP VALUES (7876,'ADAMS','CLERK',7788,'12.01.1983',1100,NULL,20);

INSERT INTO EMP VALUES (7934,'MILLER','CLERK',7782,'23.01.1982',1300,NULL,10);

**Входная таблица DEPT:**

DROP TABLE DEPT cascade constraints;

CREATE TABLE **DEPT** (

 DEPTNO              NUMBER(2) NOT NULL,                 --номер отдела

 DNAME               VARCHAR2(14),                               --название отдела

 LOC                 VARCHAR2(13),                                    --местоположение отдела

 CONSTRAINT DEPT\_PRIMARY\_KEY PRIMARY KEY (DEPTNO));

INSERT INTO DEPT VALUES (10,'ACCOUNTING','NEW YORK');

INSERT INTO DEPT VALUES (20,'RESEARCH','DALLAS');

INSERT INTO DEPT VALUES (30,'SALES','CHICAGO');

INSERT INTO DEPT VALUES (40,'OPERATIONS','BOSTON');

DROP TABLE EMP cascade constraints;

**Входная таблица EMP:**

DROP TABLE EMP cascade constraints;

CREATE TABLE **EMP** (

 EMPNO               NUMBER(4) NOT NULL,         --личный номер

 ENAME               VARCHAR2(10),                       --имя и фамилия

 JOB                 VARCHAR2(9),                              --должность

 MGR                 NUMBER(4) CONSTRAINT EMP\_SELF\_KEY REFERENCES EMP (EMPNO),   --номер начальника

 HIREDATE            DATE,                           --дата поступления на работу

 SAL                 NUMBER(7,2),                    --зарплата

 COMM                NUMBER(7,2),                --комиссионные

 DEPTNO              NUMBER(2) NOT NULL,     --номер отдела

 CONSTRAINT EMP\_FOREIGN\_KEY FOREIGN KEY (DEPTNO) REFERENCES DEPT (DEPTNO),

 CONSTRAINT EMP\_PRIMARY\_KEY PRIMARY KEY (EMPNO));

INSERT INTO EMP VALUES (7839,'KING','PRESIDENT',NULL,'17.11.1981',5000,NULL,10);

INSERT INTO EMP VALUES (7698,'BLAKE','MANAGER',7839,'1.05.1981',2850,NULL,30);

INSERT INTO EMP VALUES (7782,'CLARK','MANAGER',7839,'9.06.1981',2450,NULL,10);

INSERT INTO EMP VALUES (7566,'JONES','MANAGER',7839,'2.04.1981',2975,NULL,20);

INSERT INTO EMP VALUES (7654,'MARTIN','SALESMAN',7698,'28.09.1981',1250,1400,30);

INSERT INTO EMP VALUES (7499,'ALLEN','SALESMAN',7698,'20.02.1981',1600,300,30);

INSERT INTO EMP VALUES (7844,'TURNER','SALESMAN',7698,'8.09.1981',1500,0,30);

INSERT INTO EMP VALUES (7900,'JAMES','CLERK',7698,'3.12.1981',950,NULL,30);

INSERT INTO EMP VALUES (7521,'WARD','SALESMAN',7698,'22.02.1981',1250,500,30);

INSERT INTO EMP VALUES (7902,'FORD','ANALYST',7566,'3.12.1981',3000,NULL,20);

INSERT INTO EMP VALUES (7369,'SMITH','CLERK',7902,'17.12.1980',800,NULL,20);

INSERT INTO EMP VALUES (7788,'SCOTT','ANALYST',7566,'09.12.1982',3000,NULL,20);

INSERT INTO EMP VALUES (7876,'ADAMS','CLERK',7788,'12.01.1983',1100,NULL,20);

INSERT INTO EMP VALUES (7934,'MILLER','CLERK',7782,'23.01.1982',1300,NULL,10);

**Входная таблица SCUST:**

DROP TABLE SCUST;

CREATE TABLE SCUST

(    CUSTID NUMBER(4) NOT NULL                --номер счета заказчика

     CONSTRAINT PK\_CUSTID PRIMARY KEY,

     NAME VARCHAR2(100),                               --название

     ZIP VARCHAR2(6),                                       --почтовый индекс

     CITY VARCHAR2(50),                                  --город

     ADR VARCHAR2(40),                                   --адрес

     PHONE VARCHAR2(6),                                 --телефон

     PREDOPLATA NUMBER(9));                        --размер предоплаты

insert into scust (custid, name, zip, city, adr, phone, predoplata)

values (100, 'ДЮСШ ЮНОСТЬ', '220100', 'Минск',

        'ул.Кедышко,35', '641222', 4000);

insert into scust (custid, name, zip, city, adr, phone, predoplata)

values (101, 'СШ N110', '220025', 'Минск',

        'ул.Чкалова,27', '271222', 2400);

insert into scust (custid, name, zip, city, adr, phone, predoplata)

values (102, 'Лицей при БГУ', '220006', 'Минск',

        'ул.Маяковского,96', '250063', 1500);

insert into scust (custid, name, zip, city, adr, phone, predoplata)

values (103, 'Гимназия N54', '220088', 'Минск',

        'ул.Захарова,58', '322244', 2600);

insert into scust (custid, name, zip, city, adr, phone, predoplata)

values (104, 'Гимназия N111', '220107', 'Минск',

        'ул.Васнецова,10', '450044', 1500);

insert into scust (custid, name, zip, city, adr, phone, predoplata)

values (105, 'Гимназия N199', '220037', 'Минск',

        'ул.Менделеева,14', '382233', 2000);

insert into scust (custid, name, zip, city, adr, phone, predoplata)

values (106, 'Гимназия N1', '220101', 'Минск',

        'пр.Рокоссовского,138', '456402', 3200);

insert into scust (custid, name, zip, city, adr, phone, predoplata)

values (107, 'ДЮСШ ДИНАМО', '210035', 'Витебск',

        'ул.Черняховского,34', '350618', 4900);

insert into scust (custid, name, zip, city, adr, phone, predoplata)

values (108, 'СДЮШОР СПАРТАК', '210038', 'Витебск',

        'ул.Свердлова,5', '320806', 2900);

insert into scust (custid, name, zip, city, adr, phone, predoplata)

values (109, 'СШ N2', '210001', 'Витебск',

        'ул.Кирова,16', '372463', 1000);

insert into scust (custid, name, zip, city, adr, phone, predoplata)

values (110, 'СДЮСШ СМЕНА', '212008', 'Могилев',

        'ул.Первомайская,24', '207455', 2000);

insert into scust (custid, name, zip, city, adr, phone, predoplata)

values (111, 'Лицей N1', '212004', 'Могилев',

        'пр.Мира,22', '240018', 1100);

insert into scust (custid, name, zip, city, adr, phone, predoplata)

values (112, 'СДЮСШ ДИНАМО', '212024', 'Могилев',

        'ул.Космонавтов,18', '280010', 4200);

insert into scust (custid, name, zip, city, adr, phone, predoplata)

values (113, 'ДЮСШ N1', '230004', 'Брест',

        'ул.Московская,32', '330084', 3000);

insert into scust (custid, name, zip, city, adr, phone, predoplata)

values (114, 'Институт туризма', '230084', 'Брест',

        'пр.Строителей,28', '289604', 3000);

insert into scust (custid, name, zip, city, adr, phone, predoplata)

values (115, 'СДЮСШ СПАРТАК', '231020', 'Гродно',

        'ул.Васнецова,56', '360001', 1100);

insert into scust (custid, name, zip, city, adr, phone, predoplata)

values (116, 'Институт физкультуры', '231004', 'Гродно',

        'ул.Замковая,12', '385432', 3100);

insert into scust (custid, name, zip, city, adr, phone, predoplata)

values (117, 'СДЮСШОР N1', '200004', 'Гомель',

        'ул.Металлистов,22', '254563', 1200);

**Входная таблица SORD:**

DROP TABLE SORD;

CREATE TABLE SORD(

ORDID         NUMBER(4) CONSTRAINT PK\_ORDERID PRIMARY KEY,

ORDERDATE      DATE   DEFAULT SYSDATE,

CUSTID          NUMBER(6),

SHIPDATE         DATE,

SUMMA         NUMBER(9));

 insert into sord  values(601,to\_date('07.02.1995','dd.mm.yyyy'),106,to\_date('10.02.1995','dd.mm.yyyy'),0);

 insert into sord  values(602,to\_date('27.02.1995','dd.mm.yyyy'),102,to\_date('01.03.1995','dd.mm.yyyy'),0);

 insert into sord  values(603,to\_date('01.03.1995','dd.mm.yyyy'),102,to\_date('02.03.1995','dd.mm.yyyy'),0);

 insert into sord  values(604,to\_date('10.03.1995','dd.mm.yyyy'),106,to\_date('11.03.1995','dd.mm.yyyy'),0);

 insert into sord  values(605,to\_date('27.03.1995','dd.mm.yyyy'),106,to\_date('29.03.1995','dd.mm.yyyy'),0);

insert into sord  values(606,to\_date('17.04.1995','dd.mm.yyyy'),100,to\_date('19.04.1995','dd.mm.yyyy'),0);

 insert into sord  values(607,to\_date('18.04.1995','dd.mm.yyyy'),112,to\_date('20.04.1995','dd.mm.yyyy'),0);

 insert into sord  values(608,to\_date('25.04.1995','dd.mm.yyyy'),112,to\_date('27.04.1995','dd.mm.yyyy'),0);

 insert into sord  values(609,to\_date('02.05.1995','dd.mm.yyyy'),115,to\_date('08.05.1995','dd.mm.yyyy'),0);

 insert into sord  values(610,to\_date('07.01.1996','dd.mm.yyyy'),101,to\_date('10.01.1996','dd.mm.yyyy'),0);

 insert into sord  values(611,to\_date('10.01.1996','dd.mm.yyyy'),105,to\_date('12.01.1996','dd.mm.yyyy'),0);

 insert into sord  values(612,to\_date('15.01.1996','dd.mm.yyyy'),104,to\_date('16.01.1996','dd.mm.yyyy'),0);

 insert into sord  values(613,to\_date('20.02.1996','dd.mm.yyyy'),108,to\_date('22.02.1996','dd.mm.yyyy'),0);

 insert into sord  values(614,to\_date('25.02.1996','dd.mm.yyyy'),116,to\_date('28.02.1996','dd.mm.yyyy'),0);

 insert into sord  values(615,to\_date('04.03.1996','dd.mm.yyyy'),110,to\_date('10.03.1996','dd.mm.yyyy'),0);

 insert into sord  values(616,to\_date('12.03.1996','dd.mm.yyyy'),107,to\_date('18.03.1996','dd.mm.yyyy'),0);

 insert into sord  values(617,to\_date('20.03.1996','dd.mm.yyyy'),113,to\_date('25.03.1996','dd.mm.yyyy'),0);

 insert into sord  values(618,to\_date('27.03.1996','dd.mm.yyyy'),112,to\_date('31.03.1996','dd.mm.yyyy'),0);

insert into sord  values(619,to\_date('01.04.1996','dd.mm.yyyy'),103,to\_date('03.04.1996','dd.mm.yyyy'),0);

 insert into sord  values(620,to\_date('02.04.1996','dd.mm.yyyy'),100,to\_date('04.04.1996','dd.mm.yyyy'),0);

insert into sord  values(621,to\_date('04.04.1996','dd.mm.yyyy'),116,to\_date('08.04.1996','dd.mm.yyyy'),0);

insert into sord  values(622,to\_date('05.04.1996','dd.mm.yyyy'),117,to\_date('09.04.1996','dd.mm.yyyy'),0);

**Входная таблица SITEM:**

DROP TABLE SITEM;

CREATE TABLE SITEM (ORDID NUMBER(4),

                   PRODID NUMBER(6),

                   PRICE NUMBER(9),

                   QTY NUMBER(8),

                   SUMMA NUMBER(9));

insert into sitem (ordid, prodid, price, qty, summa)

values (610, 100001, 120, 10,1200);

insert into sitem (ordid, prodid, price, qty, summa)

values (611, 100050, 60, 30,1800);

insert into sitem (ordid, prodid, price, qty, summa)

values (612, 100061, 295, 1,295);

insert into sitem (ordid, prodid, price, qty, summa)

values (601, 100002, 160, 20,3200);

insert into sitem (ordid, prodid, price, qty, summa)

values (602, 100067, 486, 1,486);

insert into sitem (ordid, prodid, price, qty, summa)

values (604, 100070, 500, 2,1000);

insert into sitem (ordid, prodid, price, qty, summa)

values (604, 100069, 335, 1,335);

insert into sitem(ordid, prodid, price, qty, summa)

values (604, 100064, 300, 1,300);

insert into sitem (ordid, prodid, price, qty, summa)

values (605, 100002, 160, 20,3200);

insert into sitem (ordid, prodid, price, qty, summa)

values (606, 100004, 60, 10,600);

insert into sitem (ordid, prodid, price, qty, summa)

values (607, 100002, 160, 10,1600);

insert into sitem (ordid, prodid, price, qty, summa)

values (608, 100002, 160, 10,1600);

insert into sitem(ordid, prodid, price, qty, summa)

values (603, 100070, 500, 2,1000);

insert into sitem (ordid, prodid, price, qty, summa)

values (610, 100010, 6, 100,600);

insert into sitem (ordid, prodid, price, qty, summa)

values (610, 100064, 300, 2,600);

insert into sitem (ordid, prodid, price, qty, summa)

values (613, 100052, 60, 20,1200);

insert into sitem (ordid, prodid, price, qty, summa)

values (614, 100041, 60, 10,600);

insert into sitem (ordid, prodid, price, qty, summa)

values (614, 100040, 80, 10,800);

insert into sitem (ordid, prodid, price, qty, summa)

values (612, 100062, 340, 2,680);

insert into sitem(ordid, prodid, price, qty, summa)

values (612, 100060, 230, 1,230);

insert into sitem (ordid, prodid, price, qty, summa)

values (620, 100003, 80, 20,1600);

insert into sitem (ordid, prodid, price, qty, summa)

values (620, 100004, 60, 20,1200);

insert into sitem (ordid, prodid, price, qty, summa)

values (620, 100012, 6, 100,600);

insert into sitem (ordid, prodid, price, qty, summa)

values (613, 100050, 60, 10,600);

insert into sitem (ordid, prodid, price, qty, summa)

values (613, 100042, 50, 10,500);

insert into sitem (ordid, prodid, price, qty, summa)

values (613, 100041, 60, 10,600);

insert into sitem (ordid, prodid, price, qty, summa)

values (619, 100067, 486, 2,972);

insert into sitem (ordid, prodid, price, qty, summa)

values (617, 100071, 890, 1,890);

insert into sitem (ordid, prodid, price, qty, summa)

values (617, 100070, 500, 1,500);

insert into sitem (ordid, prodid, price, qty, summa)

values (614, 100066, 330, 2,660);

insert into sitem (ordid, prodid, price, qty, summa)

values (616, 100007, 50, 10,500);

insert into sitem (ordid, prodid, price, qty, summa)

values (616, 100004, 60, 20,1200);

insert into sitem (ordid, prodid, price, qty, summa)

values (616, 100012, 6, 100,600);

insert into sitem (ordid, prodid, price, qty, summa)

values (616, 100013, 6, 100,600);

insert into sitem (ordid, prodid, price, qty, summa)

values (616, 100020, 400, 5,2000);

insert into sitem (ordid, prodid, price, qty, summa)

values (619, 100064, 300, 4,1200);

insert into sitem (ordid, prodid, price, qty, summa)

values (619, 100065, 350, 1,350);

insert into sitem (ordid, prodid, price, qty, summa)

values (615, 100030, 60, 4,240);

insert into sitem (ordid, prodid, price, qty, summa)

values (618, 100005, 100, 10,1000);

insert into sitem (ordid, prodid, price, qty, summa)

values (615, 100021, 300, 5,1500);

insert into sitem (ordid, prodid, price, qty, summa)

values (617, 100002, 160, 10,1600);

insert into sitem (ordid, prodid, price, qty, summa)

values (621, 100007, 50, 20,1000);

insert into sitem (ordid, prodid, price, qty, summa)

values (609, 100007, 50, 10,500);

insert into sitem (ordid, prodid, price, qty, summa)

values (609, 100012, 6, 100, 600);

insert into sitem (ordid, prodid, price, qty, summa)

values (622, 100041, 60, 10, 600);

insert into sitem (ordid, prodid, price, qty, summa)

values (622, 100050, 60, 10, 600);

**Входная таблица SPROD:**

DROP TABLE SPROD;

CREATE TABLE SPROD (PRODID NUMBER(6) --код товара

                      CONSTRAINT PK\_PRODID PRIMARY KEY,

                      DESCRIP CHAR(140), --название товара

                      PRICE NUMBER(9), --цена товара

                      P\_QTY NUMBER(9)); --количество товара

insert into sprod (prodid,  descrip, price,p\_qty)

values (100001,'Теннисная ракетка SPALDING',120,1000);

insert into sprod (prodid,  descrip, price,p\_qty)

values (100002,'Теннисная ракетка PRINCE',160,1000);

insert into sprod (prodid,  descrip, price,p\_qty)

values (100003,'Теннисная ракетка MAJOR',80,1000);

insert into sprod (prodid,  descrip, price,p\_qty)

values (100004,'Теннисная ракетка PANTERA',60,1000);

insert into sprod (prodid,  descrip, price,p\_qty)

values (100005,'Теннисная ракетка SHIRMA',100,1000);

insert into sprod (prodid,  descrip, price,p\_qty)

values (100006,'Теннисная ракетка VALMIERA',40,1000);

insert into sprod (prodid,  descrip, price,p\_qty)

values (100007,'Теннисная ракетка IDOL',50,1000);

insert into sprod (prodid,  descrip, price,p\_qty)

values (100010,'Набор теннисных мячей SPALDING 1',6,100);

insert into sprod (prodid,  descrip, price,p\_qty)

values (100011,'Набор теннисных мячей SPALDING 2',7,100);

insert into sprod (prodid,  descrip, price,p\_qty)

values (100012,'Набор теннисных мячей PRINCE 1',6,100);

insert into sprod (prodid,  descrip, price,p\_qty)

values (100013,'Набор теннисных мячей PRINCE 2',6,0);

insert into sprod (prodid,  descrip, price,p\_qty)

values (100014,'Набор теннисных мячей PENN',5,100);

insert into sprod (prodid,  descrip, price,p\_qty)

values (100015,'Набор теннисных мячей NOVA',5,100);

insert into sprod (prodid,  descrip, price,p\_qty)

values (100016,'Набор теннисных мячей LENINGRAD',2,100);

insert into sprod (prodid,  descrip, price,p\_qty)

values (100017,'Набор теннисных мячей TIGER',6,100);

insert into sprod (prodid,  descrip, price,p\_qty)

values (100020,'Чехол для тенниса REEBOK',400,100);

insert into sprod (prodid,  descrip, price,p\_qty)

values (100021,'Чехол для тенниса TENNIS',300,100);

insert into sprod (prodid,  descrip, price,p\_qty)

values (100030,'Сетка для тенниса',60,100);

insert into sprod (prodid,  descrip, price,p\_qty)

values (100040,'Кроссовки REEBOK',80,10);

insert into sprod (prodid,  descrip, price,p\_qty)

values (100041,'Кроссовки ADIDAS',60,100);

insert into sprod (prodid,  descrip, price,p\_qty)

values (100042,'Кроссовки PUMA',50,100);

insert into sprod (prodid,  descrip, price,p\_qty)

values (100043,'Кроссовки HEADE',70,100);

insert into sprod (prodid,  descrip, price,p\_qty)

values (100050,'Костюм спортивный PUMA',60,10);

insert into sprod (prodid,  descrip, price,p\_qty)

values (100051,'Костюм спортивный ADIDAS',50,100);

insert into sprod (prodid,  descrip, price,p\_qty)

values (100052,'Костюм спортивный HEADE',60,100);

insert into sprod (prodid,  descrip, price,p\_qty)

values (100053,'Костюм спортивный HAWAI',40,100);

insert into sprod (prodid,  descrip, price,p\_qty)

values (100060,'Телевизор DAEWOO 37',230,1000);

insert into sprod (prodid,  descrip, price,p\_qty)

values (100061,'Телевизор DAEWOO 51',295,1000);

insert into sprod (prodid,  descrip, price,p\_qty)

values (100062,'Телевизор DAEWOO 54',340,1000);

insert into sprod (prodid,  descrip, price,p\_qty)

values (100063,'Телевизор ColdStar 37',240,1000);

insert into sprod (prodid,  descrip, price,p\_qty)

values (100064,'Телевизор GoldStar 51',300,1000);

insert into sprod (prodid,  descrip, price,p\_qty)

values (100065,'Телевизор GoldStar 54',350,1000);

insert into sprod (prodid,  descrip, price,p\_qty)

values (100066,'Телевизор SONY 37',330,1000);

insert into sprod (prodid,  descrip, price,p\_qty)

values (100067,'Телевизор SONY 54',486,1000);

insert into sprod (prodid,  descrip, price,p\_qty)

values (100068,'Телевизор SONY 63',650,1000);

insert into sprod (prodid,  descrip, price,p\_qty)

values (100069,'Телевизор Panasonic 37',335,5);

insert into sprod (prodid,  descrip, price,p\_qty)

values (100070,'Телевизор Panasonic 54',500,5);

insert into sprod (prodid,  descrip, price,p\_qty)

values (100071,'Телевизор Panasonic 63',890,5);

**Входная таблица SALS  включает столбцы:**

             JOBS         -  должность;

             MINSAL   -  минимальная зарплата;

             MAXSAL  -  максимальная зарплата.

DROP TABLE SALS;

CREATE TABLE SALS (

 JOBS               VARCHAR2(12),

 MINSAL               NUMBER,

 MAXSAL               NUMBER);

INSERT INTO SALS VALUES ('CLERK',700,1400);

INSERT INTO SALS VALUES ('SALESMAN',1200,1500);

INSERT INTO SALS VALUES ('ANALYST',1500,3000);

INSERT INTO SALS VALUES ('MANAGER',2000,4000);

INSERT INTO SALS VALUES ('PRESIDENT',5000,9999);

**Выходная таблица  CR\_SCUST включает столбцы:**

            CUSTID    -  код заказчика;

            DEBET      -  сумма на счету заказчика;

            KREDIT   -   сумма кредита, предоставленного заказчику.

 Name                                       Null?                     Type

 ------------------------------- ------------------ -------------------

 CUSTID                                NOT NULL       NUMBER(4)

 DEBET                                                            NUMBER(9)

 CREDIT                                                           NUMBER(7)

**Выходная таблица H\_EMP включает столбцы:**

             ENAME    -  имя;

             EMPNO    -  табельный номер;

             JOB            -  должность;

             DEPTNO  -  номер отдела.

SDATE   -системная дата.

 Name                                        Null?                      Type

 ------------------------------- ---------------- ------------------------

ENAME                                                            VARCHAR2(10)

 EMPNO                                                            NUMBER(4)

 JOB                                                                   VARCHAR2(9)

 DEPTNO                                                          NUMBER(2)

 SDATE                                                             DATE

**Выходная таблица  H\_SCUST включает столбцы:**

NOM          -    порядковый номер;

S\_DATE     -    дата внесения изменений;

D\_TAB       -    наименование таблицы;

D\_TIP         -    название команды, по которой вносятся изменения;

CUSTID     -    код заказчика.

 Name                                            Null?                     Type

 ------------------------------- ----------------- ------------------------------

 NOM                                                                          NUMBER(3)

 S\_DATE                                                                     DATE

 D\_TAB                                                                       VARCHAR2(16)

 D\_TIP                                                                         VARCHAR2(16)

 CUSTID                                                                      NUMBER(4)

**Выходная таблица  ORD\_PROD включает столбцы:**

             PRODID      -  код товара;

             DESCRIP    -  описание товара;

             P\_QTY        -  количество единиц товара.

Name                            Null?    Type

 ------------------------ ------------ ------------------

 PRODID                                       NUMBER(6)

 DESCRIP                                     CHAR(40)

 QTY                                              NUMBER(5)

**Информация входной таблицы SKLAD: создается самостоятельно.**

Структура записи:

S\_NUM --номер ячейки

PART\_NUM --номер партии

S\_DETAL --количество деталей в партии

    S\_NUM           PART\_NUM    S\_DETAL

    --------- --------- ------------------------

                1                 5469               500

                2                 7243               450

                3                 5469               500

                4                 5469               500

                5                 6085               415

                6                 5469               500

                7                 8159               619