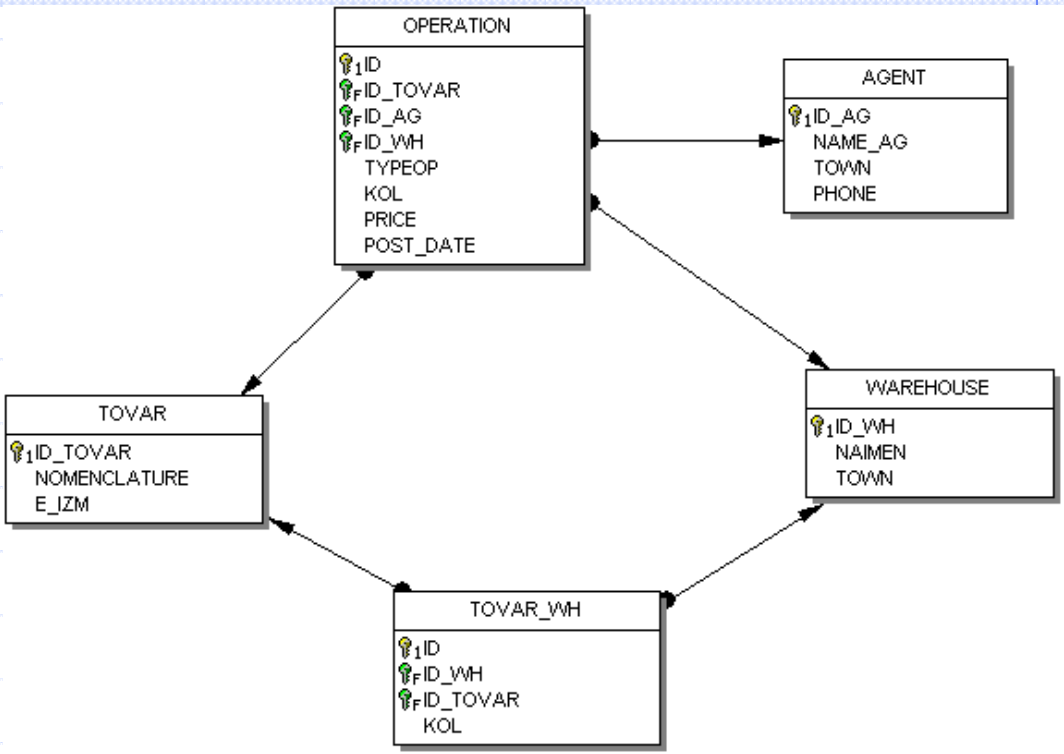
Хранимые процедуры

ЦЕЛИ

- Совместное хранение данных и алгоритмов
- Программирование на стороне сервера
- Параметризация




```
select  a.name_ag
from agent a
where not exists
    (select 1
        from operation op join  warehouse w
                    on op.id_wh=w.id_wh
        where a.id_ag=op.id_ag and w.naimen=
'Склад 1')
```

```
select  a.name_ag
from agent a
where not exists
    (select 1
        from operation op join  warehouse w
                    on op.id_wh=w.id_wh
        where a.id_ag=op.id_ag and
  w.naimen= ? )
```

```
create or alter procedure P1 (  NAME char(20) )
                        returns ( AGENT char(20) )
as
begin
 for select  a.name_ag
          from agent a
          where not exists     (select 1
                    from operation op join  warehouse w
                              on op.id_wh=w.id_wh
                    where a.id_ag=op.id_ag and
                              w.naimen=:NAME)
into :AGENT do
 suspend;
end
```

Синтаксис с учетом работы в командном окне

```
SET TERM ^ ;

{CREATE | RECREATE | ALTER } PROCEDURE
    <имя>
[(<список входных параметров>)]
[RETURNS (<список выходных параметров>)]
AS
[<объявления локальных переменных>]
BEGIN
<один или несколько блоков операторов>
END ^
COMMIT^

SET TERM ; ^
```

Смешанный синтаксис

CREATE OR ALTER PROCEDURE

Объявление локальных переменных

DECLARE VARIABLE <*имя*> < *тип*>
[=<*начальное значение*>];

И для переменных и для параметров возможно в качестве типа использовать домен или тип колонки существующей таблицы

```
CREATE OR ALTER procedure P1
        ( NAME type of column
    WAREHOUSE.NAIMEN)
        returns ( AGENT type of column
    AGENT.NAME_AG)
```

Использование локальных переменных

Особенность в операторах DML

(select, insert, update, delete, execute procedure)

должны предваряться префиксом «:».

Пример

```
declare variable min_price numeric(15,2);
begin
  min_price = 0.1;
  update OPERATION
        set price=price*1.5
  where price< :min_price;
end
```

Структурные операторы PSQL

BEGIN … END	Блок
<переменная> = <выражение>	Оператор присваивания
/*……*/	Многострочный комментарий
-- …	Комментарий до конца строки
IF … THEN … [ELSE …]	Оператор ветвления
WHILE … DO …	Оператор цикла
EXCEPTION …	Вызвать исключение

Структурные операторы PSQL

LEAVE	Завершить цикл
EXIT	Завершить процедуру
WHEN ….	Обработать исключение
SUSPEND	Для процедур выбора, передает одну строку результата в кэш, приостанавливая выполнение процедуры до тех пор, пока из кэша эту строку не заберет вызвавшее приложение. Для выполняемых процедур эквивалентен EXIT

Одиночный оператор SELECT

SELECT

<спецификация одиночного оператора SELECT>

INTO

<список локальных переменных>;

Можно считать оператором присваивания для кортежа

Допустимо использовать в случае корректности

<var> = (select ... from ...);

Пример

```
SELECT SUM(BUDGET), AVG(BUDGET)
    FROM  DEPARTMENT
WHERE HEAD_DEPT = :head_dept
INTO :tot_budget, :avg_budget;

max_salary = (select max(salary)
                        from employee
                        where job_country= :cntr);
```

Многострочный оператор SELECT

FOR SELECT
    <*спецификация оператора* SELECT>
INTO <*список переменных*> DO
BEGIN
  <*блок обработки*>
 [SUSPEND;]
 [WHEN … <*обработка исключения* >]
END

```
sum =0;
for  select salary from employee
       where dept_no = :DN
  into :SL  do
begin
  sum = sum +SL;
  suspend;
end
```

Курсоры

DECLARE <имя> CURSOR FOR
(<*спецификация оператора* SELECT>);

OPEN <имя>;

FETCH <имя>
INTO <*список переменных*>;

ROW_COUNT - 1/0 вернул ли FETCH строку

CLOSE <имя>;

Пример

```
CREATE PROCEDURE SHOW_TABL_NAMES
            RETURNS ( RNAME CHAR(31)) AS
DECLARE C CURSOR FOR
( SELECT  RDB$RELATION_NAME  FROM  RDB$RELATIONS );
BEGIN
   OPEN C;
   WHILE (1 = 1) DO
   BEGIN
         FETCH C INTO :RNAME;
         IF (ROW_COUNT = 0) THEN  LEAVE;
         SUSPEND;
   END
   CLOSE C;
END
```

Редактирование по курсору

UPDATE <имя таблицы>
SET <список изменений>
WHERE CURRENT OF <имя курсора>;

DELETE FROM <имя таблицы>
WHERE CURRENT OF <имя курсора>;

Выполнить оператор/процедуру

EXECUTE STATEMENT <строка>;
EXECUTE PROCEDURE <процедура>;

Для процедур, возвращающих результат

SELECT <. . .>
FROM <процедура>
[. . .]

Пример

```
CREATE PROCEDURE     T_KVADR (n
     INTEGER)
RETURNS (i INTEGER, kv INTEGER)
AS
BEGIN
  /* ТЕЛО ПРОЦЕДУРЫ */
  i=1;
  WHILE (i<=n) DO
  BEGIN
       kv=i*i;
      SUSPEND;
      i=i+1;
  END
END
```

Вызов

В командном окне

```
select * from  T_KVADR(10);
```

В другой процедуре

```
for select * from T_KVADR(10)
          into :x, :kvx
do
begin
. . .  -- что-то делаем с x и kvx
end
```

Примеры из БД employee

ADD_EMP_PROJ (emp_no SMALLINT, proj_id CHAR(5))

execute procedure ADD_EMP_PROJ (145,'GUIDE');

Обработка ошибки – выбрасывает исключение

DEPT_BUDGET (dno CHAR(3))

По номеру отдела определяет его суммарный бюджет (включая все подотделы). Рекурсивная.

select * from DEPT_BUDGET ('000');

Формирование SQL-оператора

```
create procedure TAB_RANG (name_t char(31))
                          returns      (KOL integer)
as
declare variable OPER varchar(200);
begin
OPER = 'select count(*) from  ' || name_t || '   into :KOL ';
 execute statement OPER;
suspend;
end
```

Опасно при использовании строковых параметров!!!

Возможный синтаксис

EXECUTE STATEMENT *<select-statement>*
INTO *<var>* [, *<var>* ...] ;

FOR EXECUTE STATEMENT *<select-statement>*
INTO *<var>* [, *<var>* ...]
DO *<psql-statement>;*

Позволяет также выполнять операторы DDL (create, alter, drop)

Исключения

EXCEPTION [*<exception-name>* [*custom-message*]];

```
exception ex_bad_id
                'Wrong id for input'|| emp_id;
```

Сообщение

Bad id exception Wrong id for input 1

Обработка ошибки/исключения

```
begin

when { <имя исключения>| SQLCODE  <код>|
        GDSCODE <код>|ANY}
        do <оператор>
[when… do <оператор>]
. . .
end
```

```
when any do
begin
insert into error_log (...)
        values (sqlcode, ...);
exception;
end
```

Выполнимые блоки

Начиная с версии 2.0

Аналог хранимой процедуры, но не имеет имени и хранится на клиенте, может быть встроен в программный код на ЯВУ

Позволяет отлаживать будущие процедуры «на лету»

Синтаксис

EXECUTE BLOCK [(*<inparams>*)]
 [RETURNS (*<outparams>*)]
AS
[*<declarations>*]
BEGIN
[*<PSQL statements>*]
 END

<inparams> ::= <paramname> <type> = ?
[, <inparams>]

<outparams> ::= paramname type
[, <outparams>]

Пример

```
execute block
as
declare i int = 0;
begin
 while (i < 128) do
   begin
        insert into AsciiTable
          values (:i, ascii_char(:i)); i = i + 1;
    end
end
```

Пример

```
execute block (x double precision = ?,
                y double precision = ?)
        returns (gmean double precision)
as
begin
gmean = sqrt(x*y);
suspend;
end
```

Процедуры и транзакции

- Не допускается выполнения операций старта/фиксации/отмены транзакции
- Процедура выполняется в рамках какой-то транзакции

Автономные транзакции

В случае успешного завершения фиксируется немедленно не дожидаясь завершения процедуры

Выполняется с теми же параметрами, что и транзакция, в которой выполняется процедура

Не зависит от родительской транзакции, поэтому может входить с ней в блокировку

Автономные транзакции

IN AUTONOMOUS TRANSACTION
DO *<psql-statement>;*

[FOR] EXECUTE STATEMENT
sql-statement
WITH {AUTONOMOUS|COMMON} TRANSACTION
[...other options...]
[INTO *<variables>*]
[DO *psql-statement*]