Лекция #7

Базы данных

Михаил Моисеев

Procedural SQL

Представления

Представление – именованный SELECT – запрос, хранимый в БД.

CREATE VIEW <имя_представления>
[(<имя_столбца>,...)] **AS** <запрос>

```
CREATE VIEW studs (fname, lname, number) AS
SELECT fname, lname, number from students, groups
WHERE students.group_id = groups.id
```

Особенности представлений

- Доступно только для чтения
- Автоматическое обновление данных
- Может использоваться как обычная таблица
- Виртуальная таблица, копии данных не создается
- Поддерживаются вложенные представления

Представления #2

```
CREATE VIEW gsize(gid, scount) AS
SELECT groups.id AS gid, COUNT(students.id) AS s_count
FROM groups, students
WHERE students.group_id = groups.id GROUP BY groups.id
ORDER BY s_count desc;

CREATE VIEW bgroup(number) AS
SELECT groups.number AS number FROM gsize, groups
WHERE groups.id = gsize.gid ROWS 1;
```

Использование представлений

- Подготовка данных для клиентского приложения
- Выполнение часто используемых запросов
- Декомпозиция сложных запросов

SQL- программирование

Поддерживаются следующие возможности:

- Процедуры, триггеры, генераторы, …
- Входные, выходные и внутренние переменные
- Несколько операторов SQL
- Операторы ветвления и перехода
- Операторы цикла
- Операторы прекращения цикла и выхода из процедуры
- Оператор возвращающий результат выполнения
- Исключения
- Внешние пользовательские функции (UDF)

Оператор SUSPEND

Возврат результатов выполнения процедуры

SUSPEND

Особенности выполнения:

- В процедуре может быть несколько SUSPEND
- В процедуре может не быть SUSPEND
- Не прерывает процедуру
- Один SUSPEND может срабатывать несколько раз

Оператор ветвления

IF (<условие>) **THEN**
BEGIN
  …
END **ELSE**
BEGIN
  …
END;

```
IF (A > 10) THEN
BEGIN
    A = 0;
END
```

Оператор FOR

FOR <выборка> **INTO** <:переменная1, …> **DO**
BEGIN
  …
END

Выбираемые переменные помещаются в список после INTO

```
FOR SELECT fname FROM students WHERE group_id IS NOT NULL
ORDER BY fname INTO :out_fname do
BEGIN
    IF (CHAR_LENGTH(out_fname) > 5 ) THEN
            SUSPEND;
END
```

Оператор WHILE

WHILE (<условие>) **DO**
BEGIN
  …
END

```
WHILE (A < 100) DO
BEGIN
    INSERT INTO students(fname, lname)  VALUES ('Иван', 'Иванов');
    A = A + 1;
END;
```

Операторы BREAK и EXIT

BREAK - выход из цикла

```
for select age, fname from students into :var, :name do begin
    if (var > 20) then begin
       suspend;
       break;
    end
 end
```

EXIT – выход из процедуры

```
if (var > 10) then exit;
```

Генераторы

Генератор – именованный счетчик

GENERATOR <gen_name>

GEN_ID (<gen_name>, <increment_value>)

SET GENERATOR <gen_name> **TO** <value>

```
CREATE GENERATOR gen_stud_id
SET GENERATOR gen_stud_id TO 45
GenValue = GEN_ID(gen_stud_id, 1)
```

Использование генераторов

- В триггерах, процедурах
- Для заполнения автоинкрементых полей
- Для выдачи уникального номера клиентскому приложению

Тригеры

Триггер - процедура обработки определенных событий на сервере.

CREATE TRIGGER <trigger_name> **FOR** <table_name> <trigger_type>
POSITION <pos_value> **AS**
BEGIN
… // Тело триггера – набор SQL выражений разделяемых «;»
END

События – при операциях INSERT / DELETE / UPDATET.

Типы триггеров / событий:

- BEFORE DELETE; BEFORE INSERT; BEFORE UPDATE;
- AFTER DELETE; AFTER INSERT; AFTER UPDATE;

Триггеры #2

NEW и **OLD** – строка таблицы, над которой выполняется операция

```
CREATE TRIGGER StudTrigger FOR Students BEFORE  INSERT
POSITION 0  AS
BEGIN
    NEW.ID = GEN_ID(gen_stud_id, 1);
END
```

Особенности триггеров

- Триггер работает для одной таблицы
- Таблица может иметь несколько триггеров
- Можно обращаться к другим таблицам
- Можно выполнять несколько запросов
- Можно вызывать процедуры

	NEW	OLD
BI	RW	-
AI	R	-
BU	RW	R
AU	R	R
BD	-	R
AD	-	R

Триггеры #3

```
CREATE TRIGGER StudTrigger FOR Students BEFORE INSERT
POSITION 0  AS
BEGIN
   NEW.group_id = (SELECT FIRST 1 id FROM groups
                        GROUP BY id
END
```

Использование триггеров

- Для заполнения полей при INSERT / UPDATE
- Для контроля целостности
- В целях отладки

Исключения

Исключение – для обработки ошибок и выхода из процедур.

CREATE EXCEPTION <имя_искл> <сообщение>

EXCEPTION <имя_искл >

WHEN EXCEPTION < имя_искл > **DO**
BEGIN
  … /* Обработка исключения*/
END

Исключения #2

Системные исключения

- Ошибки SQL
- Ошибки Firebird

WHEN {GDSCODE | SQLCODE} <код_ошибки> **DO**
BEGIN
  …
END

WHEN ANY DO
BEGIN
  …
END

Хранимые процедуры

ХП – позволяют выполнять один или несколько операторов SQL, могут иметь входные и выходные параметры

```
CREATE PROCEDURE <имя_ХП> ([<вх_переменная1>, …])
RETURNS [<вых_переменная1>, …]
AS
DECLARE VARIABLE  [<внутр_переменная>];
BEGIN
   …
END
```

Хранимые процедуры #2

Особенности процедур

- Несколько операторов SQL разделенных “;”
- SET TERM
- Процедура может возвращать один или несколько наборов переменных
- Вызов процедур внутри процедуры, рекурсия

```
CREATE PROCEDURE group_size (number VARCHAR(10))
RETURNS (scount INTEGER) AS
BEGIN
  SELECT COUNT(*) FROM groups, students WHERE
  students.group_id= groups.id AND groups.number= :number
  GROUP BY groups.id INTO :scount;
  SUSPEND;
END
```

Хранимые процедуры #3

Использование процедур

- Для реализация бизнес-логики на стороне сервера
- Для реализации бизнес-логики не описываемой одним запросом
- Вызов из процедур, триггеров и клиентского приложения

EXECUTE PROCEDURE <имя_проц> ([<параметр1,…>])

```
CREATE PROCEDURE fill_avg_mark AS
FOR SELECT id FROM students INTO :sid DO
BEGIN
    UPDATE students SET avg_mark =
        (SELECT avg(cast(mark as float)) FROM stud_results
         WHERE student_id = :sid)
    WHERE id = :sid;
END
```

Хранимые процедуры #4

```
CREATE PROCEDURE check_recurse  RETURNS (recurse integer) AS
declare variable layer integer;
declare variable curr_sbj integer;
declare variable root_sbj integer;
BEGIN
    recurse = 0;
    FOR SELECT subjects.id, ness_subject_id FROM subjects, subject_rel
    WHERE subjects.id = subject_rel.subject_id  INTO :root_sbj, :curr_sbj do
    BEGIN
        layer = 1;
        EXECUTE PROCEDURE check_subject(root_sbj, curr_sbj, layer);
        WHEN EXCEPTION is_recurse DO  BEGIN
          recurse = 1;   break;
    END
  END
```

Хранимые процедуры #5

```
CREATE PROCEDURE check_subject(
      root_sbj integer, curr_sbj integer,  layer integer) AS
declare variable nid integer;
BEGIN
    FOR SELECT ness_subject_id FROM subject_rel
    WHERE subject_id = :curr_sbj INTO :nid DO
    BEGIN
        IF ((nid = root_sbj) AND (layer < 10)) THEN
        BEGIN
            ECEPTION is_recurse;
        END
        layer = layer + 1;
        EXECUTE PROCEDURE check_subject(:root_sbj, :nid, :layer);
  END
  SUSPEND;
END
```

Хранимые процедуры

ХП может возвращать множество значений

```
CREATE PROCEDURE mvalues_proc(number varchar(10))
RETURNS ( id INT, fname varchar(20), lname varchar(20)) AS
BEGIN
  FOR SELECT id, fname, lname FROM students, groups
  WHERE students.group_id = groups.id INTO :id, :fname, :lname DO
  BEGIN
    …
    SUSPEND;
  END
END
```

Хранимые процедуры #2

Выбор одного(первого) набора данных:

```
EXECUTE PROCEDURE  mvalues_proc('4081/1')
RETURNING_VALUES :id, :fname, :lname
```

Выбор множества наборов данных:

```
FOR SELECT * FROM mvalues_proc('4081/1') INTO :fname, :lname DO
BEGIN
    SUSPEND;
END
```

Вызов процедуры из запроса:

```
SELECT * FROM mvalues_proc('4081/1') WHERE id > 10;
```

События

Событие – сообщение, которое БД посылаемые клиентам.

POST_EVENT <текст сообщения>

```
POST_EVENT  ‘It`s an event’;

DECLARE VARIABLE Message1 VARCHAR(20);
Message1 = ‘It`s an event’;
POST_EVENT  :Message1;
```

Выполнение действий по инициативе БД

UDF

Функции пользователя (UDF) – для расширения возможностей SQL

UDF = DLL + SQL

DECLARE EXTERNAL FUNCTION <имя функции>
<список типов входных параметров>
RETURNS
<тип возвращаемого значения>
ENTRY_POINT <имя функции в DLL>
MODULE_NAME <имя DLL>;

Стандартные функции поставляемые вместе с Firebird

UDF #2

```
function strlength(s: PChar): Integer; cdecl; export;
begin
  Result := StrLen(s);
end;

DECLARE EXTERNAL FUNCTION strlength  CSTRING(100)
RETURNS
  INTEGER BY VALUE
ENTRY_POINT 'strlength' MODULE_NAME 'mydll.dll';
```

Вопросы

- Для чего используются представления ?
- Чем представления отличаются от таблиц ?
- Перечислите основные процедурные возможности SQL ?
- Для чего используется оператор SUSPEND ?
- В чем различие операторов FOR и WHILE?
- Какие операторы выхода из цикла/процедуры есть в SQL ?
- Для чего нужны генераторы ?
- Чем отличаются триггеры от процедур ?
- Для чего предназначены OLD и NEW, когда они могут применяться ?
- Для чего можно использовать триггеры ?
- Чем отличается процедура от SQL-запроса ?
- Для чего используются исключения ?
- Как можно получить множественный набор данных из процедуры ?
- Для чего используются события ?
- Для чего предназначены UDF ?