Создание триггеров, хранимых процедур и исключений в СУБД Interbase/Firebird

Создание триггера

Общий синтаксис создания триггера

```
set term ^;
create trigger имя for {таблица | просмотр} [active |
inactive]
{before | after}   {delete |   insert | update}
[position
число] as
[declare variable переменная тип-данных;
[declare variable переменная тип-данных;]…]
begin
оператор; [оператор; …]
end; ^
set term;
```

Событие/фаза

Вид триггера	Описание
BEFORE INSERT	Вызывается до создания новой строки. Позволяет изменять входные значения
AFTER INSERT	Вызывается после создания новой строки. Не позволяет изменять входные значения. Обычно используется для модификации других таблиц
BEFORE UPDATE	Вызывается до создания новой версии записи. Позволяет изменять входные значения
AFTER UPDATE	Вызывается после создания новой версии записи. Не позволяет изменять входные значения. Обычно используется для изменения других таблиц
BEFORE DELETE	Вызывается до удаления существующей строки. Не принимает изменений никаких столбцов в строке
AFTER DELETE	Вызывается после удаления строки. Не принимает изменений никаких столбцов в строке. Обычно используется для модификации других таблиц

Последовательность

Создание триггеров

```
active before update position 5 as ... create trigger
       bu_acccunt5 for account
active before update position 0 as ... create trigger
       bu_acccunt for account
active after update position 5 as ... create trigger
       au_account5 for account
active after update position 3 as ... create trigger
       au_account3 for account
```

Запрос на изменение

```
update account set c='canceled' where c2=5;
```

Последовательность вызовов

Выполняется триггер bu_account.
Выполняется триггер bu_account5.
Новая версия записи записывается на диск.
Выполняется триггер au_account3.
Выполняется триггер au_account5.

Переменные NEW и OLD используются для

- ✓ Получения допустимых значений по умолчанию в некоторых условиях
- ✓ Проверка и, при необходимости, преобразования входных данных пользователя

Значения переменных **NEW** и **OLD**

- ✓ переменные **new.*** имеют значения в событиях **insert** и **update**
- ✓ переменные **old.*** имеют значения в событиях **update** и **delete**

переменные **new.*** в событиях удаления и **old. *** в событиях добавления имеют значение **null**

применимые значения **new** и **old** доступны для всех столбцов таблицы или просмотра, даже если сами столбцы не указаны в операторе DML

- ✓ значения **old.***, если доступны, могут использоваться в триггерах как переменные, но изменение значения не влияет на сохраненное старое значение
- ✓ значения **new.***, если доступны, могут использоваться для чтения/записи в фазе **before** и только для чтения в фазе **after**

Создание автоинкрементных ключей

Создание генератора и триггера для автозаполнения поля **customer_id** таблицы **CUSTOMER**:

```
// Создание генератора
create generator customer_id_gen;
// Создание триггера
set term ^;
create trigger set_customer_id for CUSTOMER;
active before insert position 0 as
begin
if (new.customer_id is null) then
new.customer_id = gen_id(customer_id_gen, 1);
end; ^
set term ; ^
commit;
```

Хранимые процедуры

```
create procedure имя-процедуры
[(аргумент тип-данных [,аргумент тип-данных]… ) ]
[returns  ([аргумент тип-данных [,аргумент тип-данных]…
)] as
[declare variable  переменная тип-данных;
[declare variable  переменная тип-данных;]…]
begin
оператор; [оператор;…]
exit;
end
```

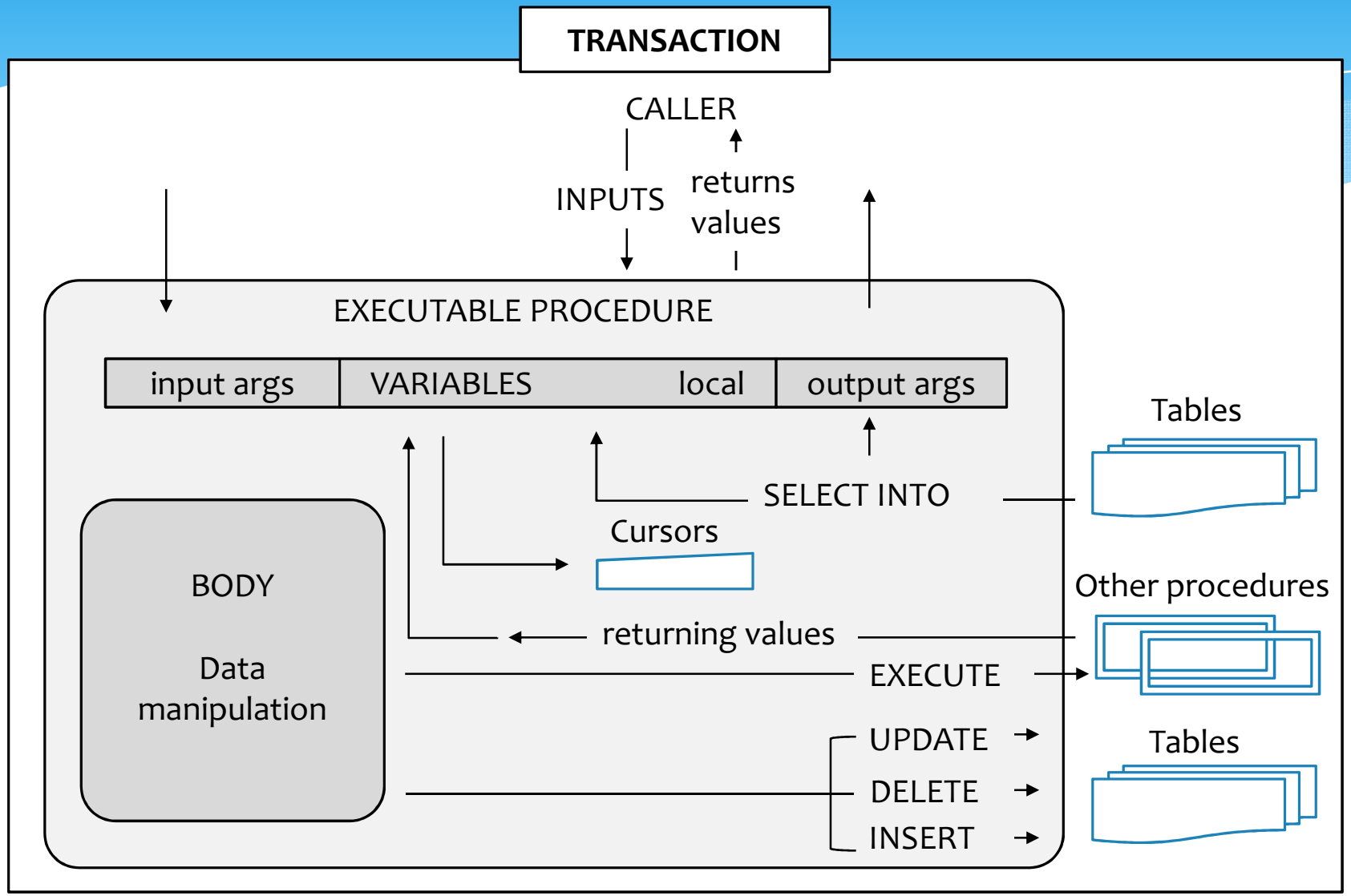
Получение результатов запроса в переменные

```
select {поле|выражение|функция} from {таблица|просмотр}
into :переменная [, :переменная …]
```

Вызов хранимых процедур

```
execute procedure имя [()[аргумент [,аргумент]… [)];
```

Работа хранимой процедуры



Исключения

Создание исключения

```
create exception имя сообщение;
```

Обработчик кода исключения

```
when {имя | any } do оператор;
```

Пример обработчика исключения

```
create exception nocustomers 'Нет записей';
create procedure get_last_cust returns (last_cust
numeric) as
begin
select max(customer_id) from customers into :last_cust;
if (:last_cust is null) then exception nocustomers;
exit;
when exception nocustomers do last_cust=0;
end;
```