22. Язык хранимых процедур и триггеров. Назначение и область применения

Хранимая процедура представляет собой программу, расположенную на сервере и вызываемую из приложения клиента. Использование этих объектов увеличивает скорость доступа к БД по следующим причинам:

1) вместо текста запроса, который может быть достаточно длинным, серверу передается по сети относительно короткие обращения к хранимой процедуре;

2) хранимая процедура, в отличие от запроса, не требует предварительной синтаксической проверки.

Еще одним преимуществом при обращении к хранимым процедурам является то, что, будучи общими, для всех приложений-клиентов, они реализуют единые для них правила работы с БД.

*Создание*

CREATE PROCEDURE имя\_проц

[(имя\_параметра <тип> [,имя\_параметра <тип> ...])]

[RETURNS имя\_параметра <тип> [,имя\_параметра <тип> ...])]

AS <тело\_процедуры> [terminator]

имя\_проц Имя процедуры.

param <datatype> Входные параметры, используемые процедурой и тип данных

RETURNS param <datatype> Выходные параметры, используемые процедурой. Допустимые типы данных перечислены в Типы данных.

<тело\_процедуры> Тело процедуры. Включает: ·Объявления локальных переменных. · Блок инструкций на языке процедур и триггеров.

terminator Терминатор, определенный для SQL командой SET TERM, указывающий конец тела процедуры

*Изменение*

ALTER PROCEDURE имя\_проц

[(имя\_параметра <тип> [,(имя\_параметра <тип> ...])]

[RETURNS ((имя\_параметра <тип> [,(имя\_параметра <тип> ...])]

AS <тело\_процедуры> [terminator]

*Удаление*

DROP PROCEDURE имя\_проц;

*Примеры*

SET TERM ^ ;

CREATE PROCEDURE "MyProc"(

"ID" INTEGER)

AS

BEGIN EXIT; END ^

ALTER PROCEDURE "MyProc"

( "LID" INTEGER,

"LFAM" CHAR(30),

"LYEARS" INTEGER,

"LSTATUS" CHAR(10))

AS

BEGIN

INSERT INTO sotrudnik(id\_sotr, fam, years, status)

VALUES(:lid, :lfam, :lyears, :lstatus);

END^

SET TERM ; ^

Чтобы выполнить эту процедуру необходимо выполнить команду:

execute procedure MyProc(123, 'Ленин', 35, 'Вождь');

commit;

А теперь удалим процедуру:

DROP PROCEDURE MyProc;

Еще один пример в котором используются циклы FOR:

CREATE PROCEDURE “SET\_DEF\_USER”( “NewDefValue” CHAR(10), “OldDefValue” CHAR(10))

RETURNS ( “Number” INTEGER) AS DECLARE VARIABLE Count INTEGER;

DECLARE VARIABLE TempID INTEGER; BEGIN :Count = 0;

FOR SELECT “Status”, “ID\_Status” FROM “StatusTable” WHERE “Status” = :OldDefValue

INTO :NewDefValue, :TempID DO BEGIN UPDATE “StatusTable”

SET “Status” = :NewDefValue WHERE “ID” = :TempID; :Count = :Count + 1;

END; :Number = :Count; END^

### 6.5.4 Триггеры

Триггер представляет собой процедуру, которая находится на сервере БД и вызывается автоматически при модификации записей БД, т. е. при изменении столбцов или при их удалении и добавлении. В отличие от хранимых процедур, триггеры нельзя вызывать из приложения клиента, а также передавать им параметры и получать от них результаты.

Триггеры обычно используются для программной реализации так называемых бизнес-правил. С их помощью удобно реализовать различные ограничения, например, ограничения на значения столбцов или ограничение ссылочной целостности, а также выполнять такие действия, как накапливание статистики работы БД или резервное копирования записей.

*Создание*

CREATE TRIGGER имя\_триггера FOR имя\_табл

[ACTIVE | INACTIVE]

{BEFORE | AFTER}

{DELETE | INSERT | UPDATE}

[POSITION number]

AS <тело\_триггера> terminator

имя\_триггера Имя существующего триггера.

ACTIVE Определяет, что действие триггера дает эффект, когда срабатывает (по умолчанию).

INACTIVE Определяет, что действие триггера не дает эффекта.

BEFORE Определяет, что триггер срабатывает перед ассоциированной операцией.

AFTER Определяет, что триггер срабатывает после ассоциированной операцией.

DELETE | INSERT | UPDATE Определяет операцию над таблицей, с которой связывает триггер to fire.

POSITION число Определяет порядок в котором срабатывают триггеры перед или после того же самого действия. *число* должно быть целым от 0 до 32767. Триггер с меньшим номером срабатывает первым. Триггеры для того же самого действия, с тем же самым позиционным номером, буду fire в случайном порядке.

тело\_триггера Тело триггера, блок инструкций на языке процедур и триггеров.

*Изменение*

ALTER TRIGGER имя\_триггера

[ACTIVE | INACTIVE]

[{BEFORE | AFTER} {DELETE | INSERT | UPDATE}]

[POSITION number]

[AS <тело\_триггера>] [terminator]

*Удаление*

DROP TRIGGER имя\_триггера;

*Примеры*

SET TERM ^ ;

CREATE TRIGGER "ADDSTREET" FOR "STREET"

ACTIVE BEFORE INSERT POSITION 0

as

begin

street.id\_street=gen\_id(gnstreetid, 1);

end

^

CREATE TRIGGER "DELSTREET" FOR "STREET"

ACTIVE BEFORE DELETE POSITION 0 as begin delete from person

where person.id\_street = street.id\_street;

end

^

SET TERM ;^

Определение переменных

DECLARE VARIABLE var <тип\_данных>;

[DECLARE VARIABLE var <datatype>; ...]

<тип\_данных> может быть числового, строкового типа или дата.

Используются переменный обычным образом. Перед именем переменной ставится «:»

*Пример использования инструкций:*

Следующая процедура возвращает количество сотрудников, работающих в должности переданной строковым параметром (но не ее идентификатором), например «Доцент» и общее количество денег выплачиваемых им в качестве зарплаты.

SET TERM ^ ;

ALTER PROCEDURE num\_sotr( dolg CHAR(10))

RETURNS( num integer, sum\_money integer)

AS

DECLARE VARIABLE id\_d INTEGER;

BEGIN num=0;

FOR SELECT id\_d

FROM dolgnost

WHERE dolg=:dolg

INTO :id\_d

DO

BEGIN SELECT COUNT(\*) FROM a

WHERE a.id\_d=:id\_d INTO num;

END

sum\_money=0;

IF (num>0) THEN

BEGIN

sum\_money = sum\_money \* num;

END

END ^

SET TERM ; ^