**Практическая работа № 16**

**Генераторы и их использование**

Генератор – это специальный объект базы данных, который генерирует уникальные последовательные числа. Эти числа могут быть использованы в качестве идентификаторов (например код клиента, номер счета и т.п.). Для создания генератора необходимо использовать оператор DDL

Создать генератор можно командой:

**CREATE GENERATOR generatorname;**

После создания генератора его значения можно получать при помощи функции

**GEN\_ID(generatorname, inc\_value)**

где inc\_value – число, на которое необходимо прирастить значение генератора.

Генераторы возвращают значения (и сохраняют свои значения на диске) вне контекста транзакции пользователя. Это означает, что если генератора было увеличено с 10 до 11 (инкремент 1), то даже при откате транзакции (ROLLBACK) значение генератора не вернется к предыдущему. Вместе с этим гарантируется что каждому пользователю будет возвращено уникальное значение генератора.

При выборке значения генератора запросом вида select gen\_id(genname, x) from ... следует учитывать буферизацию выборки на клиенте. Т.е. в многопользовательской среде при выполнении двух таких запросов значения генератора будут увеличиваться "пачками", а не на величину x для каждой выбираемой записи.

1. Использование генераторов в триггерах и хранимых процедурах

Пример триггера, автоматически присваивающего уникальное значение ключевому полю таблицы:

создадим генератор для уникальной идентификации клиентов:

**CREATE GENERATOR NEWCLIENT;**

создадим триггер для таблицы CLIENTS :

**CREATE TRIGGER TBI\_CLIENTS FOR CLIENTS**

**ACTIVE BEFORE INSERT POSITION 0**

**AS**

**BEGIN**

**NEW.CLIENT\_ID = GEN\_ID(NEWCLIENT, 1);**

**END**

В результате при создании новой записи полю CLIENT\_ID будет автоматически присваиваться новое значение.

Однако при использовании генератора в триггере возникает проблема на клиентской стороне (например в BDE, используемом в Delphi, C++Builder ...), когда клиентское приложение пытается перечитать только-что вставленную запись. Поскольку триггер меняет значение первичного ключа вставляемой записи, BDE "теряет" такую запись и чаще всего выдает сообщение "Record/Key deleted". Поскольку SQL-сервер не может сообщить клиентскому приложению о новом значении ключевого поля, необходимо сначала запросить уникальное значение с сервера, и только затем использовать его во вставляемой записи. Сделать это можно при помощи хранимой процедуры

**CREATE PROCEDURE GETNEWCLIENT**

**RETURNS (NID INTEGER)**

**AS**

**BEGIN**

**NID = GEN\_ID(NEWCLIENT, 1);**

**END**

1. Изменение значения генератора

Значение генератора можно переустановить при помощи оператора DDL

**SET GENERATOR generatorname TO value;**

Однако вы не сможете использовать такое выражение в теле триггера или хранимой процедуры, т.к. там можно использовать только операторы DML (а не DDL).

Если вы хотите обнулить генератор, или присвоить ему определенное значение в теле хранимой процедуры, то вы можете это сделать используя функцию GEN\_ID: (В данном примере генератор NEWCLIENT увеличивается на свое же значение с отрицательным знаком.)

**...**

**TEMPVAR = GEN\_ID(NEWCLIENT, -GEN\_ID(NEWCLIENT, 0);**

**...**

Будьте внимательны при выполнении таких операций в многопользовательских средах. Приложения, процедуры и триггеры, которые в данный момент используют этот генератор, могут предполагать что он не будет "обнулен". Обязательно проверяйте "обнуление" генератора на возникновение конфликтных ситуаций при работе 2-х и более пользователей.

1. Получение текущего значения генераторов

Текущее значение генератора можно получить, вызвав функцию GEN\_ID с нулевым увеличением значения генератора. Это можно сделать не только в триггере или хранимой процедуре, но и оператором SELECT

**SELECT GEN\_ID(NEWCLIENT, 0) FROM RDB$DATABASE**

Результатом выполнения запроса будет одна запись с одним полем, содержащим текущее значение генератора. Таблица RDB$DATABASES выбрана как содержаща в большинстве случаев одну запись, хотя использовать можно и любую другую таблицу.

При работе в многопользовательских средах будьте внимательны - в то время как вы получили "текущее" значение генератора, другое приложение может его изменить, и таким образом "текущее" значение окажется устаревшим. Тем более не рекомендуется использовать "текущее" значение генератора для его последующего изменения.

1. Удаление генераторов

В языке DDL Borland Interbase нет оператора для удаления генератора. Неизвестно, чем это вызвано, но серьезной проблемы не представляет. В самом начале статьи было упомянуто, что запись о генераторе создается в таблице RDB$GENERATORS. Эту запись, безусловно, можно удалить. Однако место, распределенное на странице генераторов, освобождено не будет. Оно будет освобождено только после того, как вы сделаете вашей БД BACKUP/RESTORE.

**Обзор операторов**

Имя генератора должно являться обычным идентификатором метаданных базы данных: максимум 31 символ, без специальных символов за исключением символа подчеркивания «\_» (если вы не используете регистрозависимые идентификаторы в кавычках).

CREATE GENERATOR <name>;

SET GENERATOR <name> TO <value>;

DROP GENERATOR <name>;

Операторы DML (Data Manipulation Language – язык манипуляции данными) в клиентском SQL:

SELECT GEN\_ID(<GeneratorName>, <increment>) FROM RDB$DATABASE;

Операторы DML в PSQL (Procedural SQL – процедурный SQL – расширение языка, используемое в хранимых процедурах и триггерах):

<intvar> = GEN\_ID(<GeneratorName>, <increment>);

Получение текущего значения («Select»)

Клиентский SQL

SELECT GEN\_ID(<GeneratorName>, 0) FROM RDB$DATABASE;

Генерация следующего значения («Update» + «Select»)

Так же, как и в случае получения текущего значения, это выполняется с помощью GEN\_ID, но в этом случае вы используете значение шага, равное 1

*SELECT GEN\_ID(<GeneratorName>, 1) FROM RDB$DATABASE;*

Прямое указание определенного значения генератора («Update»)

*SET GENERATOR <GeneratorName> TO <NewValue>;*

Удаление генератора («Delete»)

*DROP GENERATOR <GeneratorName>;*