УПРАВЛЕНИЕ ДАННЫМИ

Лекция 3. ЯЗЫКИ МАНИПУЛИРОВАНИЯ БД.

Язык SQL

Язык SQL (*structured query language* — «язык структурированных запросов») --является языком манипулирования для реляционной БД.

В SQL реализовано три основных функции манипулирования данными:

1) определение;
2) выборка;
3) обновление.

Типы данных

Типы данных используются в случае:

- определения столбца в таблице базы данных в операторе CREATE TABLE или для его изменения с использованием ALTER TABLE (динамическое изменение характеристик ранее созданной таблицы);
- при объявлении и редактировании домена оператором CREATE DOMAIN/ALTER DOMAIN;
- при объявлении локальных переменных в хранимых процедурах, PSQL-блоках и триггерах, при указании аргументов хранимых процедур;
- при описании внешних функций (UDF – функций, определённых пользователем) для указания аргументов и возвращаемых значений;
- при явном преобразовании типов данных в качестве аргумента для функции CAST.

Типы данных Firebird

Название	Размер	Диапазон и точность	Описание
BLOB	Переменный	Нет. Размер сегмента BLOB ограничивается 64К. Максимальный размер поля BLOB 4 Гб. Для размера страницы 4096 максимальный размер BLOB поля несколько ниже 2 Гб.	Тип данных с динамически изменяемым размером для хранения больших данных, таких как графика, тексты, оцифрованные звуки. Базовая структурная единица — сегмент. Подтип Blob описывает содержимое.
CHAR(*n*) CHARACTER(*n*)	*n* символов (размер в байтах зависит от кодировки, кол-во байт на символ)	от 1 до 32 767 байт	Символьный тип данных фиксированной длины. При извлечении данных, строка дополняется пробелами справа до указанной длины. Если количество символов *n* не указано, то по умолчанию принимается 1.
DATE	32 бита	От 01.01.0001 н.э. до 31.12.9999 н.э.	Сохраняет дату в 32-битовом длинном слове

Название	Размер	Диапазон и точность	Описание
DECIMAL (*precision*, *scale*)	Переменный (16, 32 или 64 бита)	*precision* = от 1 до 18, указывает, по меньшей мере, количество цифр для хранения; *scale* = от 0 до 18. Задаёт количество знаков после разделителя	*scale* <= *precision*. Число с десятичной точкой, имеющей после точки *scale* разрядов. Пример: DECIMAL(10,3) содержит число точно в следующем формате: ppppppp.sss.
DOUBLE PRECISION	64 бита	$2{,}225 \times 10^{-308}$ .. $1{,}797 \times 10^{308}$	Вещественные данные двойной точности, 15 цифр, размер зависит от платформы
FLOAT	32 бита	$1{,}175 \times 10^{-38}$ .. $3{,}402 \times 10^{38}$	Вещественные данные одинарной точности, 7 цифр
INTEGER INT	32 бита	–2 147 483 648 .. 2 147 483 647	4-байтные целочисленные данные.

Название	Размер	Диапазон и точность	Описание
NUMERIC (*precision*, *scale*)	Переменный (16, 32 или 64 бита)	*precision* = от 1 до 18, указывает, по меньшей мере, количество цифр для хранения; *scale* = от 0 до 18. Задаёт количество знаков после разделителя.	*scale* <=precision. Число с десятичной точкой, имеющей после точки scale разрядов. Пример: NUMERIC(10,3) содержит число точно в следующем формате: ppppppp.sss.
SMALLINT	16 бит	–32 768 .. 32 767	2-байтные целочисленные данные
TIME	32 бита	От 0:00 до 23:59:59.9999	Беззнаковое целое типа InterBase ISC_TIME. Время дня в единицах 0.0001 секунды после полуночи.
TIMESTAMP	64 бита	От 01.01.0001 н.э. до 31.12.9999 н.э.	Включает информацию и о времени

Название	Размер	Диапазон и точность	Описание
VARCHAR(*n*) CHAR VARYING CHARACTER VARYING	*n* символов (размер в байтах зависит от кодировки, кол-ва байт на символ)	От 1 до 32 765 байтов	Размер символов в байтах с учётом их кодировки не может быть больше 32765. Для этого типа данных, в отличие от CHAR (где по умолчанию предполагается количество символов 1), количество символов n обязательно должно быть указано.

1. Определение

Синтаксис оператора CREATE DATABASE

CREATE {DATABASE | SCHEMA} '<спецификация файла>'

[USER '<имя пользователя>' [PASSWORD '<пароль>']]

[PAGE_SIZE [=] <целое>]

[LENGTH [=] <целое> [PAGE[S]]]

[DEFAULT CHARACTER SET <набор символов>]

[<вторичный файл>]...;

<вторичный файл> ::= FILE '<спецификация файла>'

[LENGTH [=] <целое> [PAGE[S]]]

[STARTING [AT [PAGE]] <целое>]

Описание конструкций оператора CREATE DATABASE

'<спецификация файла>‘ - задает полный путь к создаваемому файлу базы данных и имя файла, включая его расширение. Сам файл должен отсутствовать на внешнем носителе.

USER '<имя пользователя>‘ - Имя пользователя-владельца базы данных. Может содержать до 31 символа. Нечувствительно к регистру.

PASSWORD '<пароль>‘- Пароль пользователя-владельца базы данных. Может содержать до 64 символов, однако только первые 8 имеют значение. Чувствителен к регистру.

PAGE_SIZE - Задает размер страницы базы данных. Допустимы значения 4096 (значение по умолчанию), 8192 и 16384. Если вы зададите неправильное значение размера страницы, то система не выдаст сообщения об ошибке, а установит размер до ближайшего меньшего числа. Если указать значение меньше чем 4096, то будет выбрано значение по умолчанию — 4096.

LENGTH - Задает количество страниц в первичном файле базы данных.

DEFAULT CHARACTER SET - Задает набор символов по умолчанию для строковых типов данных в базе данных.

STARTING AT PAGE - Страница, с которой начинается вторичный файл базы данных.

Описание конструкций оператора CREATE TABLE

CREATE TABLE <имя таблицы>(<определение столбца>
[,<определение столбца>...
|<ограничения>]);

<определение столбца>::=<имя столбца> <тип данных> [NOT NULL];

<тип данных>::=INTEGER | SMALLINT | DECIMAL(p,q) | CHAR(n) | VARCHAR(n)|FLOAT

Обязательно надо указать имя таблицы и определить как минимум один столбец.

CREATE TABLE поддерживает следующие возможности для определения столбцов:

- Столбцы определяют имя и тип данных для данных, вводимых в столбец.
- Основанные на доменах столбцы наследуют все характеристики домена.
- На столбец можно определить значение по умолчанию, атрибут NOT NULL, дополнительные ограничения CHECK или порядок сортировки.

Ограничение целостности CHECK

Синтаксис:

CHECK (<условие столбца>);

В условии можно пользоваться операторами сравнения, операторами NOT, BETWEEN, LIKE, IN, IS [NOT] NULL и прочими. У столбца может быть только одно условие CHECK.

Условие (<условие столбца>) должно принимать значение истинно, для того чтобы разрешить операцию добавления или изменения данных. <условие столбца> может проверять несколько значений, введенных в другие столбцы. Это ограничение на уровне таблицы.

Пример. Рассмотрим таблицу «Служащий» (EMPLOYEE), структура.

Emp_Num – номер служащего, целочисленное, не может принимать значение NULL;

L_Name – фамилия, строка переменной длины с максимальной длиной 20 символов, не может принимать значение NULL;

F_Name – имя, строка переменной длины с максимальной длиной 10 символов, не может принимать значение NULL;

Dep_Num – номер отдела, целочисленное, не может принимать значение NULL;

Job_Cod – должность, строка переменной длины с максимальной длиной 10 символов;

Phone Ext – внутренний номер телефона, целочисленное, находится в переделах от 222 до 444;

Salary – оклад, значение от 0 до 9999,99.

Следующая инструкция создает указанную таблицу:

```
CREATE TABLE EMPLOYEE
        (EMP_NUM INTEGER NOT NULL,
         L_NAME VARCHAR(20) NOT NULL,
         N_NAME VARCHAR(10) NOT NULL,
         DEP_NUM INTEGER NOT NULL,
         JOB_CODE VARCHAR(10),
         PHONE_EXT INTEGER CHECK(PHONE_EXT
                                 BETWEEN 222 AND 444),
         SALARY DECIMAL(6,2) DEFAULT 0 );
```

INSERT добавляет одну или более новых строк данных к существующей таблице.

Значения вставляются в столбцы по порядку их следования, если не задан факультативный список столбцов.

Если список столбцов задан на множестве всех доступных столбцов, то во все не перечисленные столбцы автоматически вставляется или значение по умолчанию, или NULL.

Если факультативный список столбцов упущен, предложение VALUES должно содержать значения для всех столбцов таблицы.

Чтобы вставить одну строку данных, должно присутствовать предложение VALUES и содержать определенный список значений.

Чтобы вставить несколько, строк данных, определите <select_expr>, которое возвращает уже существующие данные из другой таблицы. Выбранные строки должны соответствовать списку столбцов.

Предупреждение: Допустимо выбирать из той же таблицы, в которую строки вставляются, но эта практика не рекомендуется, так как подобные действия могут привести к бесконечным вставкам строк (зацикливанию).

Синтаксис оператора INSERT

INSERT INTO <object> [(col [, col ...])]

{VALUES (<val> [, <val> ...]) | <select_expr>}

[RETURNING <column_list> [INTO <variable_list>]];

<object> = tablename | viewname

val = {:variable | constant | expr| function | udf ([val [, val …]])

| NULL | USER | RDB$DB_KEY | ?

} [COLLATE collation]

<constant> = num | 'string' | charsetname 'string‘

<expr> = Допустимое выражение SQL, которое возвращает в одиночное значение столбца.

<function> = {

CAST (<val> AS <datatype>)

| UPPER (<val>)

| GEN_ID (generator, <val>)

}

<select_expr> = SELECT возвращающий ноль или более строк, где число столбцов в каждой строке такое же, как число элементов, которые должны быть вставлены.

Пример. Следующая инструкция добавляет строку в таблицу «Служащий», присваивает значения шести столбцам:

```
INSERT INTO EMPLOYEE
(EMP_NUM,L_NAME,N_NAME,DEP_NUM,
SALARY,JOB_CODE)
VALUES (112,‘Petrov’,’Stepan’,10,1456.96,‘admin’);
```

2. Выборка

SELECT [DISTINCT] <элементы>

FROM<таблица(ы)>

WHERE <предикат>

[GROUP BY <поле(ля)>

[HAVING <предикат>]]

[PLAN <список полей>]

[ORDER BY <поле(ля)>];

Назначение предложений, входящих в оператор SELECT

- SELECT -- Список столбцов, которые возвращаются.
- FROM -- Определяет таблицы, в которых ищутся значения.
- WHERE -- Определенное условие поиска, которое используется, чтобы выбрать необходимые стоки из множества всех строк. Предложение WHERE может содержать инструкцию SELECT, которая упоминается, как подзапрос.
- GROUP BY -- Группирует возвращенные строки, основываясь на общих значениях столбцов. Используется совместно с HAVING.
- HAVING -- Определяет условие поиска для группы записей. Строки отбираются из значений столбцов, указанных в GROUP BY и значений агрегатных функций, вычисленных для каждой группы, образованной GROUP BY.
- UNION -- Комбинирует результаты двух или более инструкций SELECT, создавая одиночную динамическую таблицу, исключая повторяющиеся строки.
- ORDER BY -- Определяет порядок сортировки строк возвращенных SELECT, по умолчанию в возрастающем порядке (ASC), или в убывающем порядке (DESC).
- PLAN -- Определяет план запроса, который будет использоваться оптимизатором запроса вместо обычного выбора.

3. Обновление

UPDATE <имя таблицы>

SET <имя столбца> = <выражение> [,<имя столбца> =

<выражение>...]

[WHERE <предикат>];

<выражение>::=<константа>|<арифметическое

выражение> | NULL

СПАСИБО ЗА ВНИМАНИЕ!