SQL-ПОДОБНЫЙ ДЕКЛАРАТИВНЫЙ ЯЗЫК ДЛЯ N-МОДЕЛИ ДАННЫХ.

Ольховик Олег Владимирович
Белых Александр Валерьевич

Введение.

В работе предлагается SQL-подобный язык N-Declarative Language (NDL) для N-модели данных, представленной в [1]. Для выбора в качестве образца языка SQL есть, по крайней мере, две причины: во-первых, специалисты, практически занимающиеся базами данных, привыкли к таким языкам; во-вторых, предлагаемый язык, как и SQL, является декларативным.

На NDL могут быть описаны структуры и отношения N-модели данных, а так же может производиться манипулирование данными и администрирование системы в целом. Язык обладает простым и лаконичным синтаксисом. Он не перегружен различными конструкциями расширения и достаточно прост в изучении. NDL позволяет создавать информационные системы и обслуживать их. Он в полной мере раскрывает потенциал и возможности N-модели данных.

В этой работе задается синтаксис декларативного ядра языка NDL, а процедурное расширение выходит за ее рамки. Для определения синтаксиса языка NDL используется синтаксический метаязык Extended BNF, описанный в Международном стандарте [2].

Операторы определения данных

(* Первая часть лексического синтаксиса определяет литеры 7-битового набора Международного стандарта [3], который представляет каждую терминальную-литеру и разделитель в N-SQL. *)

буква
= 'a' | 'b' | 'c' | 'd' | 'e' | 'f' | 'g' | 'h' | 'i' | 'j' | 'k' | 'l' | 'm' | 'n' | 'o' | 'p' | 'q' | 'r' | 's' | 't' | 'u' | 'v' | 'w' | 'x' | 'y' | 'z' | 'A' | 'B' | 'C' | 'D' | 'E' | 'F' | 'G' | 'H' | 'I' | 'J' | 'K' | 'L' | 'M' | 'N' | 'O' | 'P' | 'Q' | 'R' | 'S' | 'T' | 'U' | 'V' | 'W' | 'X' | 'Y' | 'Z' | 'а' | 'б' | 'в' |'г' |'д' | 'е' | 'ё' | 'ж' | 'з' | 'и'| 'й' | 'к' | 'л' | 'м' | 'н' | 'о' | 'п' | 'р' | 'с' | 'т' | 'у' | 'ф' | 'х' | 'ц' | 'ч' | 'ш' | 'щ' | 'ъ' | 'ы' | 'ь' | 'э' | 'ю' | 'я' | 'А' | 'Б' | 'В' |'Г' | 'Д' | 'Е' | 'Ё' | 'Ж' | 'З' | 'И' | 'Й' | 'К' | 'Л' | 'М' | 'Н' | 'О' | 'П' | 'Р' | 'С' | 'Т' | 'У' | 'Ф' | 'Х' | 'Ц' | 'Ч' | 'Ш' | 'Щ' | 'Ъ' | 'Ы' | 'Ь' | 'Э' | 'Ю' | 'Я' ;

десятичная цифра
= '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9' ;

символ апострофа = """ ;

символ двойной кавычки = "" ;

символ начала комментария = '(*' ;

символ конца комментария = '*)' ;
символ завершения = ';' ;
литера пробела = ' ' ;
литера горизонтальной табуляции
= ? литера Горизонтальная Табуляция [4] ? ;
новая строка
= { ? литера Возврат Каретки [4] ? }, ? литера Перевод Строки [4] ?, { ? литера Возврат Каретки [4]? };
литера вертикальной табуляции
= ? литера Вертикальная Табуляция [4] ? ;
перевод страницы
= ? литера Перевод Страницы [4] ? ;
другая литера
= буква | десятичная цифра |',' | '|' | '/' | '!' | '*)' | ')' | ']' | '/)' | '}' | '-' | """ | '*' | "" | '?' | '(' | '[' | '{' | ';' | '.' | ' ' | ':' | '+' | '_' | '%' | '@' | '&' | '#' | '$' | '\' | '/' || '^' | '`' | '~' ;
натуральное число = десятичная цифра, { десятичная цифра } ;
целое число = ['-'], натуральное число ;
вещественное число = целое число, ['. ', натуральное число] ;
оператор сравнения = '=' | '<>' | '>' | '<' | '=>' | '<=' ;
идентификатор = буква, {буква | десятичная цифра | '_'} ;
значение = целое число | вещественное число | терминальная строка ;

(* Вторая часть синтаксиса определяет удаление ненужных непечатных литер из синтаксиса. *)
терминальная литера
= буква | десятичная цифра | другая литера | символ апострофа | символ двойной кавычки | символ начала комментария | символ конца комментария | символ завершения | литера пробела | литера горизонтальной табуляции | новая строка | литера вертикальной табуляции | перевод страницы ;
неразрывный символ
= терминальная литера - (символ апострофа | символ двойной кавычки) | терминальная строка ;
терминальная строка
= символ апострофа, первая терминальная литера, {первая терминальная литера}, символ апострофа | символ двойной кавычки, вторая терминальная литера, {вторая терминальная литера}, символ двойной кавычки ;
первая терминальная литера
= терминальная литера - символ апострофа;
вторая терминальная литера
= терминальная литера - символ двойной кавычки ;
разделитель
= литера пробела

| литера горизонтальной табуляции
| новая строка
| литера вертикальной табуляции
| перевод строки ;

(* Третья часть синтаксиса определяет удаление заключенных в скобки текстовых комментариев из неразрывных символов, образующих синтаксис. *)
символ без комментария
= терминальная литера - (буква | десятичная цифра | символ апострофа | символ двойной кавычки | символ начала комментария | символ конца комментария | другая литера) | целое | терминальная строка ;
символ комментария
= заключенный в скобки текстовый комментарий | другая литера | символ без комментария;
заключенный в скобки текстовый комментарий
= символ начала комментария, {символ комментария}, символ конца комментария ;
синтаксис комментария
= {заключенный в скобки текстовый комментарий}, символ без комментария, {заключенный в скобки текстовый комментарий}, {символ без комментария, {заключенный в скобки текстовый комментарий}} ;

синтаксис языка N-SQL
= {разделитель}, {синтаксис комментария}, {разделитель}, неразрывный символ, {разделитель}, {синтаксис комментария}, {разделитель}, {неразрывный символ, {разделитель}, {синтаксис комментария}, {разделитель},} ;

(* Каждый оператор пишется по правилам <синтаксис языка N-SQL>.
Ключевые слова в правилах операторов языка <оператор> пишутся заглавными буквами, а в синтаксисе ключевое слово представляет собой строку, в которой каждая буква может быть написана как в верхнем, так и в нижнем регистре *)

Для реализации вышеописанных замечаний к синтаксису метаязыка Extended BNF [2] следует добавить следующие дополнительные правила, определяющие ключевое слово языка:

ключевое слово = заглавная буква, {символ ключевого слова} ;
символ ключевого слова = заглавная буква | литера пробела | '_'} ;
заглавная буква
= 'A' | 'B' | 'C' | 'D' | 'E' | 'F' | 'G'
| 'H' | 'I' | 'J' | 'K' | 'L' | 'M' | ’N’

| 'O' | 'P' | 'Q' | 'R' | 'S' | 'T' | 'U'
| 'V' | 'W' | 'X' | 'Y' | 'Z' ;

(*Операторы языка NDL *)
оператор
=создание базы данных | создание домена | изменение домена | удаление домена | создание класса | изменение класса | удаление класса | создание категории | изменение категории | удаление категории | создание объекта | изменение объекта | удаление объекта | выборка | создание индекса | изменение индекса | удаление индекса | создание пользователя | изменение прав пользователя | удаление пользователя ;

(* Оператор создания базы данных *)
создание базы данных
= CREATE DATABASE, путь к файлу,
USER, имя пользователя, PASSWORD, пароль,
PAGE_SIZE, размер страницы БД,
CHARACTER SET, набор символов, символ завершения;
набор символов
= идентификатор ? идентификатор допустимой на компьютере кодировки? ;
имя пользователя = идентификатор ;
путь к файлу
= (символ апострофа | символ двойной кавычки), идентификатор директории, ('\' | '/'), идентификатор файла, (символ апострофа | символ двойной кавычки) ;
идентификатор директории
= ? Идентификатор директории файловой системы NTFS ? | ? Идентификатор директории [5] ?;
имя файла
= ? Идентификатор файла файловой системы NTFS ?| ? Идентификатор файла [5] ? ;
пароль
= терминальная литера – (символ апострофа | символ двойной кавычки | символ начала комментария | символ конца комментария | символ завершения | литера пробела | литера горизонтальной табуляции | новая строка | литера вертикальной табуляции | перевод страницы) ;
размер страницы БД = '1024' | '2048' | '4096' | '8192' | '16384' ;

При создании базы данных (БД) указывается имя и пароль пользователя, который будет восприниматься как первый администратор со всеми правами. Регистр символов в имени пользователя несущественен, но учитывается в пароле. Только администратор может создавать новых пользователей и наделять их правами. После слов CHARACTER SET

указывается набор символов, который будет использоваться в базе данных (БД). Например, для русскоязычных баз это – WIN1251.

(* Операторы работы с доменом *)
создание домена =
CREATE DOMAIN, имя домена, AS, тип данных,
[CHECK, ограничение домена];
изменение домена
= ALTER DOMAIN, имя домена,
(ADD, CONSTRAINT, ограничение домена | DROP, CONSTRAINT) ;
удаление домена = DROP, DOMAIN, имя домена;
имя домена = идентификатор ;
ограничение домена
= VALUE, оператор сравнения, значение |
VALUE, [NOT], BETWEEN, значение, AND, значение |
VALUE, [NOT], IN, '(', значение, {',', значение}')' |
VALUE, [NOT], STARTING, значение |
VALUE, [NOT], CONTAINING, значение |
NOT, ограничение домена |
ограничение домена, OR, ограничение домена |
ограничение домена, AND, ограничение домена ;

Имя домена должно быть уникальным в пределах одной базы данных. Ограничения на значения в домене могут задаваться операциями сравнения через <оператор сравнения>, где:

- = -равно;
- <> - неравно;
- > - больше;
- < - меньше;
- >= - больше либо равно;
- <= - меньше либо равно.

Предложение BETWEEN позволяет указать нижнюю и верхнюю границы домена (сами границы включаются в домен). Предложение IN позволяет перечислить значения домена. Конструкции STARTING и CONTAINING могут использоваться только для строковых доменов. В предложении STARING указывается подстрока, с которой могут начинаться значения домена. В CONTAINING указывается подстрока, которая должна содержаться в значениях домена.

В операторе изменения домена конструкция ADD CONSTRAINT позволяет добавить новое ограничение, а DROP CONSTRAINT снимает все ограничения на домене.

(* Операторы работы с классами *)
создание класса

= CREATE, CLASS, тип класса, имя класса,
[PARENT, '(', имя класса, ')']
[ATRIBUTS, описание атрибута, {',', описание атрибута}]
[METODS, описание метода, {',', описание метода}]
[UNIQUE, описание ограничения, {',', описание ограничения}] ;
тип класса = ABSTRACT | CONCEPT | ENTITY | STATE ;
изменение класса
= ALTER, CLASS, имя класса,
[UPDATE, PARENT, имя класса]
[(ADD | ALTER), ATRIBUTS, описание атрибута, {',', описание атрибута}]
[DROP, ATRIBUTS, имя атрибута, {',', имя атрибута}]
[(ADD | ALTER), METODS описание метода, {',', описание метода}]
[DROP, METODS, имя метода, {',', имя метода}]
[(ADD | ALTER), UNIQUE, описание ограничения, {',', описание ограничения}]
[DROP, UNIQUE, имя ограничения, {',', имя ограничения}] ;
удаление класса = DROP, CLASS, имя класса, [POSTERITY] [CATEGORYS | CLASSES] ;
имя класса = идентификатор ;

Класс имеет тип CONCEPT, если не имеет собственных экземпляров. Если класс определен как ABSTRACT, ENTITY или STATE, то экземплярами этого класса являются, соответственно, абстрактные объекты, конкретные объекты или объекты-состояния. В разделе PARENT оператора создания класса указывается имя непосредственного родительского класса. В разделе ATRIBUTS задаются атрибуты класса. В разделе METODS описываются заголовки методов класса.

(* Синтаксис определения атрибута *)
описание атрибута
= имя атрибута, ('=', определение атрибута | ': ', ограничение атрибута) ;
имя атрибута = идентификатор ;
определение атрибута
= ('[', значение, {',', значение}']' | операционное задание, [HIDE]) ;
ограничение атрибута
= (имя домена | тип данных | EXT, '(', имя класса, ')')
[ONE |SET, '(', натуральное число, ')'], ['(' PK, ')'] ;
тип данных
= CHAR, '(', натуральное число, ')' | VARCHAR, '(', натуральное число, ')' | INTEGER | TIMESTAMP | DOUBLE | BLOB;
операция = + | - | * | / | CONCAT ;
унарная операция
= INV | GROUP | REC | SQRT | SQR | ABS | ROUND | COUNT | MIN | MAX | SUM | AVG ;

бинарная операция = AND | NOT | OR | '.' | ':' | MUL | операция ;
операционное задание
= имя функции |
унарная операция, ([операция], [имя класса'.'], операционное задание)
['('], операционное задание, [')'], бинарная операция, ['('], операционное задание, [')'] |
['('], операционное задание, [')'], WHERE, условия ;
условие
= операционное задание, оператор сравнения, операционное задание |
значение | VOID |
BETWEEN, значение, AND, значение |
IN, '(', значение, {',', значение}, ')' |
STARTING, значение |
CONTAINING, значение ;
условия
= ['('], [NOT], условие, [')'], {(AND | OR), ['('], [NOT], условие, [')']} ;

Типы данных, которые могут использоваться при описании атрибутов в языке NDL определены в таблице 1.

Таблица 1.

Тип данных	**Размер, байт**	**Описание**
INTEGER	8	Целое
DOUBLE	16	Вещественное
CHAR(n)	0-32767	Фиксированная строка
VARSHAR(n)	0-32767	Переменная строка
TIMESTAMP	8	Дата/время
BLOB	-	Двоичный объект

К типу данных BLOB неприменимы операции над атрибутивными функциями.

Ограничение атрибута задается для атрибутов, которые определяются путем фактического задания значений в экземплярах класса, или вычисляются операционно в категориях или класса. Если атрибут является элементарным, то он ограничивается доменом или типом данных. Если атрибут объектный, то он ограничивается экстенсионалом некоторого класса. Атрибут может иметь единственное значение для каждого экземпляра (ONE или по умолчанию). Кроме того, атрибут может иметь множество значений (SET), мощность которого должна быть ограничена. Атрибут идентифицирует экземпляры класса, т.е. является первичным ключом, если указано слово PK.

Атрибут класса может быть задан фактическими значениями. В этом случае все экземпляры класса имеют такое же значение атрибута. Кроме того, атрибут может быть задан операционно.

Для операций введены следующие обозначения:

- INV – операция инволюции;

- GROUP – операция группировки;
- REC – операция рекурсии;
- SQR – возведение в квадрат (унарное преобразование);
- SQRT – взятие квадратного корня (унарное преобразование);
- SQR – возведение в квадрат (унарное преобразование);
- ABS – взятие значения по модулю (унарное преобразование);
- ROUND – округление до целого (унарное преобразование);
- COUNT – количество значений (операция агрегации);
- MIN – минимальное значение (операция агрегации);
- MAX – максимальное значение (операция агрегации);
- SUM – сумма значений (операция агрегации);
- AVG – среднее значение (операция агрегации);
- AND – операция пересечения;
- NOT – операция взятия разности;
- OR – операция объединения;
- '.' – операция композиции;
- ':' - операция элементарной композиции;
- MUL – операция произведения;
- '+' - сложение (бинарное преобразование);
- '-' - вычитание (бинарное преобразование);
- '*' - произведение (бинарное преобразование);
- '/' - деление (бинарное преобразование);
- CONCAT – конкатенация строк (бинарное преобразование);
- WHERE – операция ограничения.

(* Синтаксис определения метода *)
описание метода
= имя метода, '(', (IN, параметр, {', ', параметр} |
OUT, параметр, {',', параметр}), ')' ;
имя метода = идентификатор ;
параметр = имя функции | имя переменной, ':', имя домена ;
имя функции = идентификатор ;
имя переменной = идентификатор ;

(* Синтаксис определения ограничения *)
описание ограничения
= имя ограничения, '(' имя атрибута, {',', имя атрибута}, ')' ;
имя ограничения = идентификатор ;

Под ограничением понимается набор атрибутов, значения которых должны быть уникальными для всех объектов класса.

(* Операторы работы с категориями *)

создание категории
= CREATE, CATEGORY, имя категории,
[PARENT, имя класса]
[NEGATIONS, имя категории, {',', имя категории}]
CONDITION, условия,
[ATRIBUTS, описание атрибута, {',', описание атрибута}]
[METODS, описание метода, {',', описание метода}] ;
изменение категории
= ALTER, CATEGORY, имя категории
[UPDATE, PARENT, имя класса]
[UPDATE, CONDITION, условия]
[(ADD|ALTER), ATRIBUTS, описание атрибута, {',', описание атрибута}]
[DROP, ATRIBUTS, имя атрибута, {',', имя атрибута}]
[(ADD|ALTER), METODS, описание метода {',', описание метода}]
[DROP, METODS, имя метода, {',', имя метода}] ;
удаление категории = DROP, CATEGORY, имя категории, [POSTERITY] ;
имя категории = идентификатор ;

В разделе PARENT оператора создания категории указывается имя родительского класса. В разделе NEGATIONS задаются категории, экстенсионалы которых не должны пересекаться с экстенсионалом данной категории. В разделе CONDITION записывается условие, удовлетворяя которое объект попадает в данную категорию.

Операторы манипулирование данными
(* Операторы работы с объектами *)
создание объекта
= INSERT, INTO, имя класса, ['(', имя атрибута {',', имя атрибута}')']
[PARENT, имя класса],
VALUES, '(', имя атрибута, '=', (значение, {',', значение} | '[', значение {',', значение}, ']'), {',', имя атрибута, '=', (значение, {',', значение} | '[', значение, {',', значение}']')} ;
изменение объектов
= UPDATE, OBJECT, (имя класса | имя категории)
[SET, имя атрибута, '=', (операционное задание | '[', значение, {',', значение}, ']'), {',', имя атрибута, '=', (операционное задание | '[', значение, {',', значение}, ']')}]
[ADD, имя атрибута, '=' (операционное задание | '[', значение, {',', значение}, ']')
{',', имя атрибута, '=', (операционное задание | '[', значение, {',', значение}, ']')}
[DROP, имя атрибута, {',', имя атрибута}]
[WHERE, условия] ;
удаление объектов

= DELETE, OBJECT, (имя класса | имя категории)
[WHERE, условия] ;

выборка
= SELECT, [DISTINCT]
[агрегатная функция, {',', агрегатная функция}]
[операционное задание, {',', операционное задание}]
FROM, (имя класса | имя категории),
[WHERE, условия]
[ORDER, BY, [DESC] операционное задание, {',', операционное задание}] ;

агрегатная функция
= (SUM | AVG | MIN | MAX | COUNT), '(', операционное задание, ')',
[AS, имя] //имя атрибута (в селекте)

Изменению подвергаются все объекты класса или категории, удовлетворяющие условию после слова WHERE. В операторе выборки, в отличие от принятого в SQL оператора SELECT отсутствует раздел группировки, возможность реализации подзапросов и соединения. Таким образом, синтаксис оператора упрощен. Тем не менее, функциональные возможности данного оператора, по крайней мере, не беднее, чем у оператора SELECT в SQL.

Операторы администрирования базы данных

(* Операторы работы с индексами *)
создание индекса
= CREATE, [UNIQUE] [(ASC | DESC)] INDEX, имя индекса,
ON, имя класса, '(', имя атрибута, {',', имя атрибута}, ')' ;
изменение индекса
= ALTER, INDEX, имя индекса, (DEACTIVATE | ACTIVATE) ;
удаление индекса = DROP, INDEX, имя индекса ;
имя индекса = идентификатор ;

(* Операторы для управления правами пользователей *)
создание пользователя
= CREATE, USER, имя пользователя [ADMIN] PASSWORD, пароль;
изменение прав пользователя
= ALTER, GRANT USER, имя пользователя, CLASS, имя класса,
(READ | WRITE | ALL | ADMIN) ;
удаление пользователя = DROP, USER, имя пользователя ;
имя пользователя = идентификатор ;

Заключение. Ограничения на объем работы не позволили иллюстрировать ее примерами и описать процедурное расширение декларативного языка NDL. Автор признателен специалистам, высказавшим

замечания по данной работе, надеется на сотрудничество в решении еще неразработанных вопросов и будет благодарен за конструктивную критику.

Литература

1. Ольховик О.В., Белых А.В. N-модель данных, базовые понятия //Инженерный вестник Дона, 2009. №4. http://www.ivdon.ru/magazine/archive/n4y2009/ (доступ свободный) — Загл. с экрана. — Яз. рус.
2. ISO/IEC 14977:1996, Information technology. Syntactic Metalanguage. Extended BNF;
3. ISO/IEC 646:1991, Information technology - ISO 7-bit coded character set for information interchange;
4. ISO/IEC 6429:1992, Information technology - Control functions for 7-bit and 8-bit coded character sets;
5. ISO/IEC 13346-2:1999, Information technology. Volume and file structure of write-once and rewritable media using non-sequential recording for information interchange. Part 2: Volume and boot block recognition;