ЯЗЫКИ МАНИПУЛИРОВАНИЯ РЕЛЯЦИОННЫМИ ДАННЫМИ: РЕЛЯЦИОННАЯ АЛГЕБРА, РЕЛЯЦИОННОЕ ИСЧИСЛЕНИЕ И SQL

САРАНЧУК Дориан, СТРАТИЛА Дмитрий

Технический Университет Молдовы

***Аннотация**: В данной статье рассматриваются различные языки доступа к реляционным данным. Описаны основные принципы реляционного исчисления в своих двух разновидностях: реляционное исчисление кортежей и реляционное исчисление доменов. Проделан сравнительный анализ реляционного исчисления, реляционной алгебры и SQL.*

***Ключевые слова:** реляционное исчисление, реляционное исчисление кортежей, реляционное исчисление доменов, реляционная алгебра, SQL.*

1. Введение

При проектировании баз данных на логическом уровне возможно использование многих моделей данных. Широкое распространение получила реляционная модель, основанная на отношениях. Для доступа к данным эта модель предполагает использование реляционного исчисления или эквивалентной ему реляционной алгебры.

Ни реляционная алгебра, ни реляционное исчисление сейчас не используются в чистом виде в конкретных реализациях СУБД. Язык SQL (Structured Query Language), который стал стандартом де-факто, обеспечивает доступ к реляционным данным.

2. Реляционное исчисление

Реляционное исчисление базируется на одном из разделов математической логики – исчислении предикатов. Впервые идея использования предикатов в теории баз данных была предложена Э. Коддом в 1972 году.

Переменная кортежа является основным понятием реляционного исчисления. Главная задача реляционного исчисления состоит в нахождении кортежей, удовлетворяющих заданному условию (предикату). Таким образом, если *P* — предикат, то множество всех значений переменной *x*, при которых суждение *P* является истинным, символически записывается следующим образом:

$$\{X \mid P(x)\}$$

Предикаты могут соединяться с помощью логических операций $\wedge$ (AND), $\vee$ (OR) и ~ (NOT), образуя сложные выражения.

Существует и другая версия реляционного исчисления, которая называется исчислением доменов. Здесь переменные кортежа изменяются на доменах, а не на отношениях. Эта версия реляционного исчисления была предложена Лакруа (Lacroix) и Пиротте (Pirotte).

2.1 Реляционное исчисление кортежей

В реляционном исчислении кортежей областью определения переменной (кортежа) является указанное отношение. Например, для указания того, что областью определения кортежа *S* служит отношение *Staff*, используется следующая форма записи:

$$Staff(S)$$

Тогда множество кортежей, для которых предикат *F* принимает истинное значение, записывается:

$$\{S \mid F(S)\}$$

В математической логике первого порядка предикат *F* называется формулой или правильно построенной формулой (Well Formed Formula, WFF). Таж же как не все буквы алфавита образуют слова, так и не каждая составленная формула является правильной. Выражение в реляционном исчислении кортежей имеет следующую общую форму:

$$\{S_1 \cdot a_1, S_2 \cdot a_2, \ldots, S_n \cdot a_n \mid F(S_1, S_2, \ldots, S_m)\};\ m \geq n,$$

где $S_1, S_2, \ldots, S_n, \ldots, S_m$ - переменные кортежа, a_i - атрибуты отношения, в котором определено значение переменной S_i, F — формула.

Правильно построенная формула состоит из элементарных выражений, которые имеют одну из следующих форм:

- $R(S_i)$, где S_i — переменная кортежа, а R — отношение;
- $S_i \cdot a_1 \ \Theta \ S_j \cdot a_2$, где S_i и S_j — переменные кортежа, a_1 — атрибут отношения, в котором определено значение переменной S_i, a_2 — атрибут отношения, в котором определено значение переменной S_j, и Θ — одна из операций сравнения ($<, \leq, >, \geq, =, \neq$);
- $S_i \cdot a_1 \ \Theta \ c$, где S_i — переменная кортежа, a_1 — атрибут отношения, в котором определено значение переменной S_i, c — константа из области определения атрибута a_1 и Θ — одна из операций сравнения.

Также возможно построение рекурсивных формул из элементарных выражений. При этом необходимо соблюдать следующие правила:

- любое элементарное выражение рассматривается как формула;
- если выражения F_1 и F_2 являются формулами, то выражения, полученные в результате их конъюнкции $(F_1 \wedge F_2)$, дизъюнкции $(F_1 \vee F_2)$ и отрицания $(\sim F_1)$, также являются формулами;
- если выражение F является формулой со свободной переменной X, то выражения $(\exists X)\ (F)$ и $(\forall X)\ (F)$ также являются формулами.

Довольно часто в реляционном исчислении используются кванторы. Существует два типа кванторов: квантор существования и квантор общности. Первый из них $(\exists)$ указывает на существование хотя бы одного кортежа, удовлетворяющего заданному условию. Квантор общности $(\forall)$ относится ко всем экземплярам кортежей.

При составлении логических выражений возможно применение следующих правил эквивалентности:

$$(\exists X)\ (F(X)) \equiv \ \sim (\forall X)\ (\sim (F(X)))$$
$$(\forall X)\ (F(X)) \equiv \ \sim (\exists X)\ (\sim (F(X)))$$
$$(\exists X)\ (F_1(X) \equiv F_2(X)) \equiv \ \sim (\forall X)\ (\sim (F_1(X)) \vee \sim (F_2(X)))$$
$$(\forall X)\ (F_1(X) \equiv F_2(X)) \equiv \ \sim (\exists X)\ (\sim (F_1(X)) \vee \sim (F_2(X)))$$

Рассмотрим пример формулирования запроса к базе данных на основе синтаксиса реляционного исчисления. Текст запроса: «Создать список всех менеджеров, зарплата которых превышает 25 000». Одна из возможных реализаций данного запроса имеет вид:

$$(S.fName, S.lName \mid Staff(S) \wedge S.position = \ 'Manager' \wedge S.salary > 25000)$$

2.2 Реляционное исчисление доменов

В реляционном исчислении доменов областью определения переменных являются атрибуты отношения, а не кортежи. Общая форма записи для выражений реляционного исчисления доменов имеет следующий вид:

$$\{d_1, d_2, \ldots, d_n \mid F(d_1, d_2, \ldots, d_m)\};\ m \geq n,$$

где $d_1, d_2, \ldots, d_n, \ldots, d_m$ - переменные области определения (домена), а $F(d_1, d_2, \ldots, d_m$ - формула.

Элементарные выражения, входящие в состав правильно построенной формулы, могут иметь одну из следующих форм:

- $R(d_1, d_2, \ldots, d_n)$, где R — отношение степени n и d_i — переменная домена;
- $d_i \ \Theta \ d_j$, где d_i и d_j — переменные домена и Θ — одна из операций сравнения ($<, \leq, >, \geq, =, \neq$), переменные d_i и d_j должны иметь области определения, для сравнения элементов которых применение операции Θ является допустимым;
- $d_i \ \Theta \ c$, где d_i — переменная домена, c — константа из области определения переменной домена d_i и Θ — одна из операций сравнения.

В реляционном исчислении доменов также возможно использование рекурсивных формул. Правила составления рекурсивных формул такие же, как и в реляционном исчислении кортежей.

Рассмотрим пример записи запроса к базе данных, сформулированного на основе синтаксиса реляционного исчисления доменов. Условие запроса: «Найти имена всех менеджеров, зарплата

которых превышает 25 000». Выражение реляционного исчисления доменов, эквивалентное данному запросу, выглядит так:

$$\{fN, lN \mid (\exists sN,\ posn,\ sex,\ DOB,\ sal,\ bN)\ (Staff(sN,\ fN,\ lN,\ posn,\ sex,\ DOB,\ sal,\ bN) \wedge$$
$$\wedge\ posn = 'Manager'\ \wedge\ sal > 25000)\}$$

В отличие от аналогичного запроса, записанного с помощью синтаксиса реляционного исчисления кортежей, здесь атрибутам присваивается имя (переменной). Условие *Staff (sN, fN, . . ., bN)* ограничивает переменные домена атрибутами того же самого кортежа. Поэтому возможно использование формулы *posn = 'Manager'* вместо формулы *Staff.position = 'Manager'* . Следует отметить различия в использовании квантора существования. В реляционном исчислении кортежей применение этого квантора к некоторой переменной кортежа *posn* в форме $\exists posn$ равносильно связыванию этой переменной с отношением *Staff* с использованием выражения *Staff(posn)*. С другой стороны, в реляционном исчислении доменов переменная *posn* ссылается на одно из значений в домене и на нее не налагаются ограничения до тех пор, пока она не появится в субформуле, такой как $Staff(sN,\ fN,\ lN,\ posn,\ sex,\ DOB,\ sal,\ bN)$, после чего ее значения ограничиваются значениями *position*, которые присутствуют в отношении *staff.*

Выражения, сформулированные с использованием синтаксиса реляционного исчисления доменов, безопасны, так как реляционное исчисление доменов ограничивается значениями атрибутов отношения.

3. Сравнительный анализ реляционного исчисления, реляционной алгебры и SQL

Реляционное исчисление и реляционная алгебра по сути своей являются эквивалентными. Э. Кодд доказал, что любое выражение реляционной алгебры может быть записано на языке реляционного исчисления и наоборот. Реляционная алгебра представляет собой набор операций, которые используются для определения процедуры извлечения данных из отношений. Поэтому операции реляционной алгебры носят предписывающий характер, тогда как выражения реляционного исчисления – описательный. Реляционное исчисление дает общее описание сути проблемы, но не определяет, каким именно образом данные извлекаются из таблиц. Тем временем реляционная алгебра ближе к процедурным языкам программирования и позволяет выявить конкретный алгоритм решения некоторой задачи. Но все эти различия, по большому счету, только кажущиеся, хотя и считается, что реляционная алгебра обладает большей вычислительной мощностью.

Язык SQL содержит элементы и реляционного исчисления, и реляционной алгебры. Для данного языка характерен синтаксис, близкий к фразам английского языка, расширяющий возможности реляционного исчисления и реляционной алгебры. SQL является реляционно полным языком, так как он не уступает по выразительной силе реляционной алгебре или исчислению.

Любая операция реляционной алгебры может быть выражена средствами языка SQL. Для каждой из операций над множествами (объединение, пересечение, разность, декартово произведение) в SQL существуют соответствующие эквиваленты. К этим операторам добавляются специальные реляционные операции: проекция и селекция. В совокупности все шесть операторов позволяют выразить и другие операции из теории множеств (дополнение, активное дополнение), а также различные виды соединений (тета-соединение, экви-соединение, естественное соединение, полусоединение, левое внешнее соединение, правое внешнее соединение, полное внешнее соединение). Кроме того, язык SQL позволяет пренебречь основным свойством множества (каждый элемент множества уникален), давая возможность выборки всех кортежей, включая дубликаты.

Как и в реляционном исчислении, в языке SQL существует возможность применения кванторов. Квантору существования на языке SQL соответствует оператор EXISTS, а квантору общности – оператор ALL.

Реализация основных операторов реляционной алгебры в реляционном исчислении и SQL представлена в таблице 1.

Таблица 1 – Реализация основных операторов

Операция	**Реляционное исчисление**	**Реляционная алгебра**	**SQL**
Объединение	$\{t \mid r1(t) \vee r2(t)\}$	$r_1 \cup r_2$	SELECT * FROM r1 UNION SELECT * FROM r2;
Пересечение	$\{t \mid r1(t) \wedge r2(t)\}$	$r_1 \cap r_2$	SELECT * FROM r1 INTERSECT SELECT * FROM r2;
Разность	$\{t \mid r1(t) \wedge (\sim(\exists t))r2(t)\}$	$r_1 \setminus r_2$	SELECT * FROM r1 EXCEPT SELECT * FROM r2;
Декартово произведение	$\{t \mid (\exists t1)(\exists t2)(r1(t1) \wedge r2(t2) \wedge$ $t.A1 = t1.A1 \wedge \ldots \wedge t.An = t1.An \wedge$ $t.B1 = t2.B1 \wedge \ldots \wedge t.Bm = t2.Bm)\}$	$r_1 \times r_2$	SELECT * FROM r1, r2; SELECT * FROM r1 CROSS JOIN r2;
Селекция	$\{t \mid r(t) \wedge t.A \Theta a\}$	$\sigma_{A \Theta a}(r)$	SELECT * FROM r WHERE A = ‘a’;
Проекция	$\{t.A1, t.A2, \ldots, t.An \mid r(t)\}$	$\pi_X(r)$	SELECT A FROM r;

Заключение

Реляционное исчисление является весьма сложным с точки зрения освоения и использования. Несмотря на это, его непроцедурная природа считается довольно перспективной и это стимулирует поиск других, более простых в употреблении непроцедурных методов. Подобные исследования вызвали появление двух других категорий реляционных языков: трансформационных (SQUARE, SQL) и графических (QBE).

ЛИТЕРАТУРА

1. Codd E.F. Relational completeness of data base sublanguages. IBM Research Laboratory, San Jose, California. КО 987 (#170041), March 6, 1972, Computer Sciences.
2. Ковальски Р. Логика в решении проблем: Пер. с англ. – М.: Наука. Гл. ред. Физ.-мат. Лит., 1990. – (Пробл. Искусств. Интеллекта.) – 280 с.
3. Дейт, К., Дж. Введение в системы баз данных, 7-е издание. : Пер. с англ. – М. : Издательский дом «Вильямс», 2001. – 1072 с.
4. Гарсиа-Молина Г., Ульман Д., Уидом Д. Системы баз данных. Полный курс. М.: Вильямс, 2002. – 1088 с.
5. Конноли Т., Бегг К. Базы данных: проектирование, реализация и сопровождение: Теория и практика. 3-е изд.: – М.: «Вильямс», 2003. – 1440 с.