ПРЕОБРАЗОВАНИЕ ВЫРАЖЕНИЙ РЕЛЯЦИОННОЙ АЛГЕБРЫ В ЯЗЫК SQL

ЛЯЛИН Сергей, САВИЦКИЙ Максим
Coordonator: САРАНЧУК Дориан

Технический Университет Молдовы

Аннотация: *В работе приведены исследования и сравнения между реляционной алгеброй и языком запросов SQL. Приведены примеры некоторых преобразований выражений реляционной алгебры в SQL. Разработано приложение для автоматизации преобразований.*

Ключевые слова: *реляционная алгебра, язык запросов SQL, отношение, атрибут, бинарное дерево*.

1. Введение

Реляционная алгебра (РА) — замкнутая система операций над отношениями в реляционной модели данных. Операции РА также называют реляционными операциями. Первоначальный набор из 8 операций был предложен Э. Коддом в 1970-е годы и включал как операции, которые до сих пор используются (проекция, соединение и т.д.) [1].

Язык запросов РА не получил широкого распространения в современных технологиях СУБД. Языком практического применения в БД является SQL, однако знания РА наравне с SQL является необходимым для лучшего понимания сути операции над отношениями в реляционных БД.

Для преобразования выражения РА в SQL разработано приложение, которое принимает на входе выражение на РА, а на выходе выдаёт команду на языке SQL.

2. Описание реляционной алгебры

РА описывает выполняемые над отношениями операции. РА является замкнутой относительно понятие отношения, то есть она использует отношение в качестве аргументов и выдает отношение в качестве результата [2]. Ниже описаны операции РА.

Объединение

Результатом объединения отношений *r* и *s* (*r*U*s*) является отношение с тем же заголовком, что и у совместимых по типу отношений *r* и *s*, и телом, состоящим из кортежей, принадлежащих *r* или *s*.

Представление операции объединения на языке SQL:

```
SELECT Имя, Возраст FROM студенты  UNION
SELECT Имя, Возраст FROM преподаватели;
```

Пересечение

Результатом пересечения отношений *r* и *s* (*r*∩*s*) является отношение на схеме R или S и телом, состоящим из кортежей, принадлежащих одновременно обоим отношениям *r* и *s*.

Представление операции пересечения на языке SQL:

```
SELECT Фамилия FROM студенты  INTERSECT
SELECT Фамилия FROM преподаватели;
```

Разность

Результатом разности совместимых по типу отношений *r* и *s* (*r**s*) является отношение на схеме R и телом, состоящим из кортежей, принадлежащих отношению *r* и не принадлежащих отношению *s*.

Представление операции разность на языке SQL:

```
SELECT Фамилия FROM студенты  EXCEPT
SELECT Фамилия FROM преподаватели;
```

Декартово произведение

При выполнении декартово произведения двух отношений *r* и *s* (*r* ? *s*) получим отношение, кортежи которого являются конкатенацией (сцеплением) кортежей первого и второго операндов.

Представление операции декартово произведение на языке SQL:

```
SELECT * FROM читатели, книги;
```

Активное дополнение. Пусть задано отношение *r(A,B).* Активное дополнение ˜*r*– это все возможные комбинации (декартово произведение) активных доменов всех атрибутов из схемы отношения *r* за вычетом исходного отношения *r*:

˜*r*=atup(*r*)*r*, где atup(*r*)=adom(A)?adom(B).

Итак, для выполнения данной операции необходимо знать активные домены всех атрибутов схемы отношения *r*, а для этого нужно подключится к базе данных, где находится отношение и узнать имена всех атрибутов.

Представление операции активного дополнения для ˜*r* на языке SQL:

```
SELECT DISTINCT r1.A, r2.B
FROM r AS r1, r AS r2
EXCEPT SELECT A, B FROM r;                                    (1)
```

Данный запрос (1) является обширным и медленным. При переводе из РА в SQL этот оператор будет постоянно просить подключение к базе данных.

Селекция (Выборка)

Операция селекции над отношением *r* — унарный оператор, записываемый как $\sigma_{F(t)}(r)$, где:

- *F(t)=AθB* – условие выборки, которое может быть атомарной или составной формулой, содержащей логические операторы AND, OR, NOT.
- *A, B* — имена атрибутов (B так же может быть константой)
- *θ* — оператор сравнения из множества *{*<, > ,>=,<=,=,<>*}.*

Селекция $\sigma_{F(t)}(r)$ выдаёт все кортежи из *r*, для которых формула *AθB* истинна. Представление селекции на языке SQL:

```
SELECT * FROM персоны WHERE Возраст = Вес;
```

Здесь формула для выборки представлена после оператора WHERE.

Проекция

Операция проекции — унарный оператор, записываемый как $\pi_{A1\ldots An}$ где *A1…An* — список атрибутов, подлежащих выборке. Результатом операции будет множество кортежей отношения *r*, определенных на тех атрибутах, которые указаны в списке *A1…An*. Потенциально возникающие кортежи-дубликаты удаляются из схемы.

Представление операции проекции на языке SQL:

```
SELECT DISTINCT Возраст, Вес FROM персоны;
```

Заметка: Отсутствие DISTINCT в команде SELECT может привести к появлению кортежей дубликатов в результате.

Тэта-соединение

Операция тэта-соединения $r\triangleright\!\lhd_{A\theta B}s$ есть результат последовательного применения операций декартово произведения и выборки $r\triangleright\!\lhd_{A\theta B}s=\sigma_{A\theta B}(r ? s)$. Если в отношениях и имеются одноименные атрибуты, тогда перед выполнением соединения такие атрибуты необходимо переименовать.

Представление операции тэта-соединение на языке SQL:

```
SELECT * FROM мультфильмы, каналы
WHERE мультфильмы.Название_канала = каналы.Код_канала;
```

Естественное соединение

Естественным соединением отношений *r* и *s* ($r\triangleright\!\lhd s$) является отношение, определенное на схеме RS. Кортеж t принадлежит результирующему отношению, если существуют кортежи $t_r\in r$ и $t_s\in s$ и соответственно выполняются условия: $t[R]=t_r$ и $t[S]=t_s$. Появляющиеся при этом атрибуты-дубликаты удаляются из схемы.

Операция естественного соединения может быть представлена на языке SQL тремя способами:

```
SELECT * FROM студенты NATURAL JOIN факультеты;
SELECT * FROM студенты JOIN факультеты USING Код_студента;
SELECT * FROM студенты INNER JOIN факультеты ON
студенты.Код_студента=факультеты.Код_студента;
```

Деление

Пусть *r, s* – отношения и $S\subseteq R$. Обозначим через Q=R\S. Деление отношения *r* на *s* ($r\div s$), есть отношение, определенное на схеме Q:

$r\div s=\{t|\forall t_s\in s(S)\ \exists\ t_r\in r(R)\ \&\ t_r[Q]=t\ \&\ t_r[S]=t_s\}$.

Деление может быть выражено через другие операции реляционной алгебры:

$r\div s=\pi_Q(r)\setminus \pi_Q((\pi_Q(r)?s)\setminus r)$.

3. Преобразование из РА в SQL

Для преобразования выражений РА в SQL было разработано приложение [3]. Приложение написано на язык C#, так как данный язык является одним из самых популярных языков и имеет большие функциональные возможности. Предпочтение C# было сделано также из-за наличия в нём библиотеки **Spart**, которая используется в проекте. На языке C++ есть схожая библиотека – **Boost.Spirit**. Во многом функционал этих библиотек схож, так как их основная задача - создание парсеров. Парсинг — это процесс сопоставления линейной последовательности лексем (слов) естественного или формального языка с его формальной грамматикой. Данные парсеры необходимы для проверки корректности записи выражения на языке РА.

Библиотека **Spart** позволяет составлять грамматику и правила, которые в процессе могут быть легко изменены. Также данная библиотека даёт возможность следить за ходом операции, когда строка проходит через созданную грамматику. [4]

Ниже представлена формальная грамматика (2), где показывается какой вид должна иметь входная строка на языке РА.

$$SP \rightarrow atc\ (< \mid > \mid = \mid <>)\ atc$$
(2)
$$SP \rightarrow atc$$
$$atc \rightarrow attribute$$
$$atc \rightarrow const$$
$$PP \rightarrow attribute, PP$$
$$PP \rightarrow attribute$$
$$OperN \rightarrow si\{SP\}$$
$$OperN \rightarrow pi\{PP\}$$
$$Operator \rightarrow OperN(TableName|Operator)$$

Грамматика является несложной, так как представляет операции выборки (si - δ) и проекции (pi - π). С помощью данной грамматики также можно выразить и вложенные запросы (3):

$$[\sigma_{\{A>2\}}(\sigma]_{\{A<2\}}(\pi_{\{A,B,C\}}(table1)))$$ (3)

Благодаря данной библиотеке можно легко составить дерево разбора при каждом удачном прохождении правила или выражения. Выражения могут быть представлены в виде унарных или бинарных операций. После построения дерева разбора можно рекурсивно получать значения каждой вершины и переводить в язык SQL.

4. Примеры построения дерева

Входное выражение имеет следующий вид (4):

$$\sigma_{\{a<2\}}(\pi_{\{a,b,c\}}(Table1))$$ (4)

После ввода выражения производится пошаговая проверка (парсинг). Вначале выполняется селекция по условию (рис.1a) и данная операция записывается как отдельная вершина. После этого идёт операция проекция на определенные атрибуты для заданного отношения. S выступает в роли стартовой вершиной дерева. Атрибуты операторов сохраняются в той же вершине. После селекции выполняется проекция (рис.1b). Так как данная вершина находится внутри селекции, то вершина с проекцией будет исходить от селекции.

Затем следует название отношения (рис.1c). Название отношения является окончанием цепи в дереве. Так как бинарные операторы отсутствуют – деления в вершинах не будет, и текущее состояние вернётся в первоначальное состояние **S**. Если хотя бы один бинарный оператор присутствовал бы в строке, тогда у одной вершины было бы два выхода – левый для выражения перед оператором и правый, соответственно, после оператора. Скобки используются при построении, но опускаются при обработке древа. После построения дерева начинается его обход. Обход выполняется всё время по левой стороне дерева, но когда доходит до последней вершины, действие возвращается на вершину выше и проверяет, возможно ли пройти в правую сторону. Если возможно – тогда обход снова начинается по левой стороне, и так будет продолжаться, пока не дойдёт обратно до первоначальной вершины.

Каждая вершина возвращает свою часть на языке SQL. В данном случае последняя вершина – это имя отношения и в этом случае вершина вернёт просто имя таблицы: *table1*. Если бы данная таблица находилась под знаком бинарного оператора – тогда возвращала бы не имя таблицы, а полный запрос SELECT * FROM *table1*.

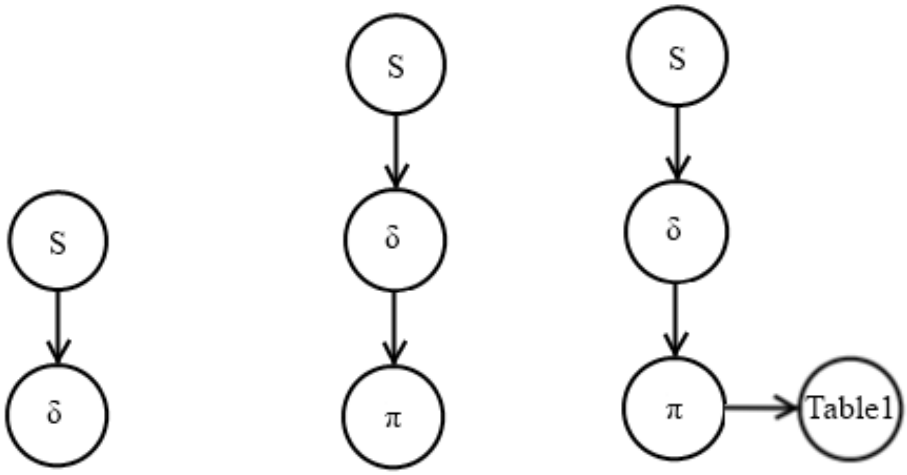

Рисунок 1 – Добавление новой вершины

В вершине проекции возвращает SQL выражение:

SELECT %атрибуты% FROM %предыдущие значение%,

где предыдущие значение – это то, что вернула вершина снизу.

В данном случае это имя таблицы, и полный запрос будет выглядеть так:

SELECT A, B, C FROM table1.

Хотя вместо имени таблицы мог быть любой другой запрос, как показано далее с селекцией. Селекция возвращает нам запрос:

SELECT * FROM %предыдущие значение% WHERE %условие%.

В конечном итоге получим выражение:

SELECT * FROM (SELECT A, B, C FROM table1) WHERE A<2.

Заключение

Язык реляционной алгебры используется реже, в отличие от языка SQL. Реляционная алгебра чаще всего используется только в учебных целях – для более лёгкого понимания последующих языков запросов и понятие принципа работы с отношениями. Реляционная алгебра выполняет операции над отношениями и поэтому каждую операцию можно представить в виде: *Select % from % where %.* На место имени отношения пишется другой подобный запрос, который вернёт отношение, но уже от другой операции. Сложности могут появиться в случае, когда возвращается имя отношения – нужно проверять предыдущий оператор, чтобы знать, что именно возвращать – только имя отношения или полный запрос. Также сложности могут возникнуть, когда возвращается активное дополнение – здесь необходимы дополнительные параметры, такие как атрибуты данного отношения. Данные команды на языке РА могут быть обратно не конвертируемыми. Так же выражение на языке SQL может быть не оптимизированным.

Библиография

1. Codd. E.F. *Relational Completeness of Data Base Sublanguages*. Data Base Systems, Courant Computer Science Symposia Series 6. — Englewood Cliffs, N.J.: Prentice-Hall, 1972.
2. Archae Development. *Реляционный алгебра. Базы данных*. URL: http://archae-dev.com/l6.htm
3. Саранчук Дориан, Опря Дмитрий, Войков Александр. *Преобразование выражений алгебры А в различные языки запросов*. Conferinţa tehnico-ştiinţifică a colaboratorilor, doctoranzilor şi studenţilor, Universitatea Tehnică a Moldovei, 22 octombrie 2013, Chişinău, Republica Moldova.
4. Jonathan de Halleux. *Spart, a parser generator framework C#.* URL: http://www.codeproject.com/Articles/5676/Spart-a-parser-generator-framework-100-C
5. Эрик Липперт. *Генерация всех бинарных деревьев.* URL: http://habrahabr.ru/post/93506