К ВОПРОСУ ОПТИМИЗАЦИИ АЛГОРИТМОВ НОРМАЛИЗАЦИИ БАЗ ДАННЫХ

ЧЕБОТАРЬ Сергей, ОЛЕЙНИК Сергей
Coordonator: САРАНЧУК Дориан

Технический Университет Молдовы

Аннотация: *В работе описаны два алгоритма нормализации реляционных баз данных на основе функциональных зависимостей методом декомпозиции: алгоритм приведения к третьей нормальной форме и нормальной форме Бойса-Кодда. Данные алгоритмы имеют экспоненциальную сложность. Приведены предложения по оптимизации некоторых частей алгоритмов, которые привносят экспоненциальный характер.*

Ключевые слова: реляционная база данных, нормальная форма, избыточность, аномалии, алгоритм нормализации, экспоненциальная сложность.

1. Введение

В наше время существует множество систем управления базами данных (СУБД). Это обусловлено тем, что любая современная информационная система использует для хранения необходимой информации базу данных (БД). Перед созданием и последующим использованием базы данных её необходимо спроектировать. Процесс проектирования БД может сопровождаться трудностями. Не смотря на это, ни одна СУБД не обладает функционалом по нормализации баз данных.

Неправильно составленные отношения при проектировании БД могут повлечь за собой аномалии и избыточность данных, что является негативным фактором, поэтому обязательным шагом при нормализации баз данных является их приведение к нормальным формам.

Следствием некорректно составленной схемы БД являются дополнительные затраты памяти на хранение избыточных данных, а также пониженное быстродействие БД, что связано с необходимостью выполнения обновлений избыточных копий. В свою очередь это может привести к более глубоким последствиям, а именно к противоречивости информации в базе данных, связанной с нахождением избыточных копий на разных уровнях обновления, аномалиями добавления и удаления данных.

Приведение к нормальным формам осуществляется для минимизации избыточности данных и предотвращения аномалий в БД. Однако, процесс нормализации БД является очень трудоемким этапом, а некоторые алгоритмы обладают экспоненциальной сложностью, поэтому оптимизация и упрощение алгоритмов являются важной задачей для проектировщиков БД.

2. Экспоненциальная сложность

В теории сложности алгоритмов экспоненциальная сложность или экспоненциальное время означает время решения задачи, ограниченное экспонентой от размерности задачи. Другими словами, если размерность задачи возрастает линейно, время её решения возрастает экспоненциально.

Принято считать, что алгоритмы с полиномиальной сложностью являются «быстрыми», в то время как алгоритмы, сложность которых больше полиномиальной, - «медленными». С этой точки зрения алгоритмы с экспоненциальной сложностью являются медленными. Однако, это предположение не совсем точное. Дело в том, что время работы алгоритма зависит от значения n (размерности задачи) и сопутствующих констант скрытых в О-нотации. В некоторых случаях для малых значений n полиномиальное время может превосходить экспоненциальное. Однако, для больших значений n время работы алгоритма с экспоненциальной сложностью существенно больше.

3. Третья нормальная форма

Схема отношения находится в третьей нормальной форме (3НФ) относительно множества функциональных зависимостей (МФЗ) F, если она находится в первой нормальной форме (1НФ) и ни один не первичный атрибут не зависит транзитивно от какого-либо ключа схемы.

Пусть дано отношение r(R), $X,Y \subseteq R$, $A \in R$. Атрибут A транзитивно зависит от множества

атрибутов X через Y, если выполняются следующие условия:

1). $X \to Y$, 2). $Y \to A$, 3) $Y \not\to X$, 4). $A \notin \{X,Y\}$.

Другими словами, схема находится в 3НФ относительно МФЗ F, если для любой нетривиальной зависимости $X \longrightarrow A \in F^+$ имеет место: X – суперключ для схемы R, или A – первичный атрибут.

Алгоритм 3NF используется для декомпозиции универсальной схемы на схемы, удовлетворяющие требованиям третьей нормальной формы, на основании множества функциональных зависимостей. Псевдокод алгоритма [1]:

Algorithm 3NF(R, F, Db)

Input: R – универсальная схема; F – множество функциональных зависимостей.

Output: Db – схема базы данных в третьей нормальной форме.

```
Begin
    k := 1
    Rk := R
    for i := 1 to k do begin
            GetKey(Ri, F, K)
            AttrNP := Ri\(UKj)
            while (X→Y∈F+) and (X↛ Ri) and (X∩Y = Ø) and (XY ⊆ Ri) and (Y ⊆ AttrNp) do begin
                    k := k + 1
                    Rk := XY
                    Ri := Ri\Y
            end
    end
    Db := {R1, R2, ... , Rk}
    return Db
end.
```

4. Нормальная форма Бойса-Кодда

Отношение находится в нормальной форме Бойса-Кодда (НФБК) относительно МФЗ F, если в нем полностью отсутствуют транзитивные зависимости от ключа, то есть данная форма не допускает никаких транзитивных зависимостей. Для приведения отношения к НФБК можно использовать и другую формулировку: схема R находится в НФ Бойса-Кодда относительно МФЗ F, если для любой нетривиальной функциональной зависимости $X \to A \notin F^+$ детерминант X является суперключом для схемы R (все схемы $R_1, R_2, \ldots, R_n$).

Алгоритм BCNF [1] используется для декомпозиции универсальной схемы на схемы, удовлетворяющие требованиям нормальной формы Бойса-Кодда, на основании множества функциональных зависимостей. Псевдокод алгоритма [1]:

Algorithm BCNF(R, F, Db)

Input: R – универсальная схема; F – множество функциональных зависимостей.

Output: Db – схема базы данных в третьей нормальной форме.

```
Begin
    k := 1
    Rk := R
    for i := 1 to k do begin
            while (X→Y ∈ F+) and (X↛ Ri) and (X∩Y = Ø ) and (XY ⊆ Ri) do begin
                    k := k + 1
                    Rk := XY
                    Ri := Ri\Y
            end
    end
    Db := {R1, R2, ... , Rk}
    return Db
end.
```

5. Оптимизация алгоритмов

Проанализировав два вышеприведенных алгоритма, можно сделать вывод, что в обоих есть необходимость построения замыкания множества функциональных зависимостей F^+. Алгоритм

построения замыкания множества функциональных зависимостей имеет экспоненциальную сложность, из-за чего и алгоритмы нормализации решаются за экспоненциальное время.

В нашем конкретном случае время работы алгоритма напрямую зависит от количества атрибутов в исходной схеме базы данных. То есть, при линейном увеличении количества атрибутов в схеме время нахождения всех функциональных зависимостей из F^+ увеличивается экспоненциально.

Для избежания экспоненциальных вычислений в обоих алгоритмах можно заменить строку, содержащую построение замыкания множества функциональных зависимостей. В исходных алгоритмах данная строчка действует по следующему принципу:

1. Для введенного множества атрибутов методом полного перебора находятся все функциональные зависимости, которые можно построить.
2. Для каждой функциональной зависимости проверяется, если она принадлежит замыканию. Если зависимость удовлетворяет данному условию, следует переход на следующий шаг, иначе берется следующая зависимость.
3. Для выбранной функциональной зависимости, принадлежащей замыканию, проверяется, удовлетворяет ли она всем необходимым условиям. Если все условия выполняются, происходит декомпозиция отношения на основании выбранной функциональной зависимости. Иначе происходит возврат к предыдущему пункту и выбор другой функциональной зависимости.

Для оптимизации данных алгоритмов было решено отказаться от полного перебора всех функциональных зависимостей и их проверки на соответствие условиям. Вместо этого происходит генерация зависимостей в соответствии с условиями конкретного алгоритма нормализации.

Псевдокод генерации функциональных зависимостей для алгоритма приведения к третьей нормальной форме:

GetDependencyFor3NF(R, AttrNP, F, f)

Input: R – универсальная схема, AttrNP – множество непервичных атрибутов, F – множество функциональных зависимостей.

Output: f – функциональная зависимость.

```
Begin
    for i:=1 to R.length
            X = GetCombinations(R, i)
            for j:=1 to X.length
                    if not closure(X[i], F).IsSupersetOf(R)
                            Rmod = R – X[i]
                            for k:=1 to Rmod.length
                                    Y = GetCombinations(Rmod, k)
                                    for n:=1 to Y.length
                                            if Y[n].IsSubsetOf(AttrNP)
                                                    if membership(X[i]→Y[n], F)
                                                            return X[i]→Y[i]
                                    end
                            end
            end
    end
    return null
end.
```

Псевдокод генерации функциональных зависимостей для алгоритма приведения к нормальной форме Бойса-Кодда приведен ниже:

GetDependencyForBCNF(R, F, f)

Input: R – универсальная схема , F – множество функциональных зависимостей.

Output: f – функциональная зависимость.

```
Begin
    for i:=1 to R.length
            X = GetCombinations(R, i)
            for j:=1 to X.length
                    if not closure(X[i], F).IsSupersetOf(R)
                            Rmod = R – X[i]
```

```
                    for k:=1 to Rmod.length
                        Y = GetCombinations(Rmod, k)
                        for n:=1 to Y.length
                            if membership(X[i]→Y[n], F)
                                return X[i]→Y[i]
                        end
                    end
            end
        end
        return null
    end.
```

Алгоритм генерации функциональных зависимостей для третьей нормальной формы полностью включает в себя алгоритм генерации функциональных зависимостей для нормальной формы Бойса-Кодда, поэтому следует описать действие первого алгоритма.

Так как генерация зависимостей происходит на основании введенного множества атрибутов, то для обеспечения того, чтобы правая и левая часть функциональной зависимости являлись подмножеством схемы разбиваемого отношения ($XY \subseteq Ri$), на вход алгоритма следует подавать не множество атрибутов универсального отношения, а множество атрибутов разбиваемого отношения.

В начале действия алгоритма происходит выбор всех комбинаций атрибутов длины i из введенного множества атрибутов. Полученные комбинации являются потенциальными левыми частями функциональной зависимости. Затем для каждой полученной левой части происходит построение замыкания и его сравнение с введенным множеством атрибутов, чтобы проверить, что левая часть не определяет множество атрибутов схемы разбиваемого отношения ($X \nrightarrow Ri$). Если левая часть удовлетворяет всем данным условиям, то можно подбирать для неё правую часть зависимости.

Правая часть функциональной зависимости находится также путем получения комбинаций различной длины. Но для того, чтобы правая часть не имела общих атрибутов с левой частью ($X \cap Y = \varnothing$), её следует строить из разницы всех введенных атрибутов и атрибутов правой части ($Rmod = R - X[i]$). Для каждой найденной комбинации атрибутов следует проверить, чтобы она являлась подмножеством множества непервичных атрибутов ($Y \subseteq AttrNp$), иначе происходит выбор другой правой части.

После подбора левой и правой части функциональной зависимости, которые удовлетворяют всем условиям, следует проверить принадлежность сгенерированной зависимости замыканию множества функциональных зависимостей, действительных в универсальном отношении ($X \rightarrow Y \in F^{+}$). Если зависимость удовлетворяет данному условию, то на её основании происходит декомпозиция отношения.

Предложенные алгоритмы для оптимизации экспоненциальных вычислений были реализованы практически на C# [2] в системе для логического проектирования реляционных баз данных [3,4].

Заключение

В результате оптимизации алгоритмов была уменьшена сложность некоторых алгоритмов нормализации баз данных. Этот факт является очень важным для проектировщиков БД, так как результат оптимизации привел к:

- снижению количества необходимых операций для получения результата;
- уменьшению количества времени на получение ответа.

Построение множества непервичных атрибутов AttrNP предполагает построение множества ключей схемы, что по сути, является задачей с экспоненциальной сложностью. Снижение сложности вычисления ключей есть одно из возможных направлений улучшения разработанной системы [4].

Библиография

1. V. Cotelea. *Baze de date relaţionale: proiectare logică*.- Chişinău, ASEM, 1997.
2. Эндрю Троелсен. *Язык программирования C# и платформа .Net 4.* ООО “И.Д. Вильямс”, 2011
3. Саранчук Дориан, Чеботарь Сергей, Олейник Сергей. *Реализация алгоритмов нормализации баз данных.* International Conference on Information Technologies, Systems and Networks ITSN-2012, 15 octombrie 2012, ULIM, Chişinău, Republica Moldova.
4. Олейник Сергей, Саранчук Дориан, Чеботарь Сергей. *Система логического проектирования реляционных БД с вожможностью использования в обучающих целях*. Conferinţa Ştiinţifică a Colaboratorilor, Doctoranzilor şi Studenţilor UTM, 23 noiembrie 2013, UTM, Chişinău, Republica Moldova.