МЕТОДЫ ОРГАНИЗАЦИЙ ДОСТУПА К ЗАПИСЯМ БАЗ ДАННЫХ

Аканов А.Д.[1] – докторант 1 – курса
Сагындыков К.М.[2] – кандидат технический наук, доцент

[1] Казахский гуманитарно- юридический инновационный университет г. Семей
[2] Евразиский национальный университет имени Л.Т. Гумилева г. Астана

С развитием информационных систем все чаще встает вопрос о стоимости информации, в них хранящейся. Большая часть таких систем использует в своей основе БД с клиент-серверной технологией доступа. Изначально для распределения доступа в БД использовалось описание доступа к отдельным таблицам базы. Основывалось такое представление на том, что таблица представляет набор сущностей (записей), которые имеют одинаковую структуру, содержат в себе однотипные данные, а потому равноправны с точки зрения предоставляемого к ним доступа.

Однако при росте БД в объеме в них стали появляться сущности, которые имеют один тип (размещаются в одной таблице), но доступ к ним должен предоставляться различный. Классическая теория распределения доступа к таблицам [1] в данном случае не имеет возможности реализации, так как она может разграничить доступ «по горизонтали» (к таблицам и атрибутам), но не «по вертикали» (к отдельным записям таблиц).

Исходя из вышесказанного встает вопрос о разработке иного подхода к проектированию защищенных БД.

Рассмотрим варианты разложения отношения в целях предоставления доступа пользователям к конкретным частям отношения. Как известно из теории реляционных баз данных [2], отношение может быть разложено на более простые отношения без потерь:

$$r = \pi_{R1}(r) \vee \pi_{R2}(r) \ldots \vee \pi_{Rn}(r) \qquad (1)$$

Здесь знаком $\vee$ обозначен оператор соединения. Представим *i*-ю проекцию отношения *r* в виде r_i. Тогда разложение без потерь (1) можно будет представить в виде, указанном в выражении (2).

$$r = r_1 \vee r_2 \vee \ldots \vee r_i \vee \ldots \vee r_n \qquad (2)$$

Здесь r_i представляет собой проекцию *r* на некоторый набор атрибутов. Наборы атрибутов, на которые производится проекция таковы что пользователи могут иметь или не иметь доступ к каждому из наборов атрибутов, но не к отдельным атрибутам из наборов. Назовем классом атрибутов совокупность атрибутов, при осуществлении проекции на которые получим отношение r_i.

Каждое отношение, в том числе полученное результате разложения исходного на более простые, может быть представлено в виде совокупности (объединения) записей [3]:

$$r_{ij} = \sigma_{\text{условие класса записи } j}(r_i), \qquad (3)$$

$$r_i = r_{i1} \cup r_{i2} \cup ... \cup r_{ij} \cup ... \cup r_{im} \ . \quad (4)$$

Пусть разбиение на записи таково, что пользователи могут иметь или не иметь доступ к конкретным наборам записей, но не к отдельным записям из наборов. Назовем такие наборы записей классами записей.

Как видно из вышесказанного, каждая сущность может содержать в себе несколько классов атрибутов и классов записей. Соответственно в ИС должны присутствовать классы пользователей, которым предоставляется доступ к определенным наборам классов записей и атрибутов. Назовем такие классы пользовательскими ролями. Тогда каждая пользовательская роль имеет доступ к определенным классам атрибутов определенных классов записей.

Для реализации модели предоставления прав доступа к классам записей существуют различные методы. Рассмотрим наиболее очевидные из них.

1. Использование сервера приложений, через который происходит работа с БД.
2. Использование специального интерфейса, стоящего между клиентом и сервером, который ограничивает доступ.
3. Создание отдельной таблицы для каждого класса записей сущности.
4. Создание отдельного представления пользователя для каждого класса записей в сущности.
5. Создание «динамического» представления пользователя, которое позволяет пользователю просматривать и редактировать только записи, выбранные по специальному алгоритму.

Рассмотрим эти методы более подробно.

Технология сервера приложений часто используется на различных Web-сайтах. Суть технологии заключается в том, что пользователь отправляет запросы приложению (например, по протоколу HTTP), а обрабатывает их программный код серверного приложения (написанного, например, на языках PHP или Perl). Серверное приложение само осуществляет разделение доступа к отдельным записям таблиц. Положительными свойствами этой технологии можно считать то, что клиенту не требуется иметь ничего, кроме браузера для того, чтобы работать с ИС, а распределение доступа программируется с помощью обычного алгоритмического языка программирования.

К минусам этой реализации доступа можно отнести следующие:

1) при работе по подходящим для такой передачи информации протоколам клиент и сервер при общении передают друг другу большое количество служебной информации (код HTML), которое занижает скорость работы с ИС;

2) скорость работы серверных приложений, как правило, ниже, чем у обычных клиентских, так как для них используются языки скриптового типа (PHP, Perl) или виртуальные машины(Java). Стандартное клиентское приложение, как правило, скомпилировано на обычном языке и представляет собой бинарный исполняемый код

Второй вариант - создание интерфейсного ПО для взаимодействия клиента с сервером. В таком случае клиент обращается к интерфейс-ному ПО,

которое определяет доступ клиента и выполняет либо не выполняет на сервере запросы клиента в зависимости от прав доступа. На данный момент существуют стандартные протоколы, которые можно довольно удобно использовать в данном случае, например XML-RPC.

Недостатками данной реализации доступа можно считать следующие:

1) сложность реализации интерфейсного ПО;

2) если API интерфейсного ПО не будет сов-падать со стандартные API для работы с БД, то большинство уже разработанных на данный момент модулей для работы с БД использовать будет невозможно. Это осложняет разработку клиентского ПО.

Третий метод - создание отдельной таблицы для каждого класса записей сущности. Для реализации этого подхода каждая реальная сущность разделяется на множество таблиц, в каждой из которых хранятся только те записи, которые относятся к одному определенному классу записей:

$$r = r_1 \vee r_2 \vee ... \vee r_n \ . \quad (5)$$

Такой подход делает возможным распределять доступ классическими средствами по таблицам и атрибутам. Однако разложение одной сущности на множество таблиц неудобно по ряду причин.

1. При изменении структуры сущности возникает необходимость производить соответствующие изменения во всех таблицах, хранящих ее данные.

2. Большие трудности представляет создание уникальных ключей, так как эта задача переходит из области стандартных функций СУБД в область функций, реализуемых пользователем искусственно.

3. При организации ссылочной целостности БД внешние ключи должны будут указывать не на одну таблицу, а на несколько. Такие внешние ключи можно реализовать не во всех СУБД.

4. Такой подход противоречит классическим методам проектирования БД, где одна таблица представляет собой одну сущность.

5. При необходимости добавить классы записей возникает необходимость добавления новых таблиц, относящихся к сущности триггеров, изменения организации ключей и т.д.

Четвертый метод - создание отдельного представления для каждого класса записей. Для его реализации следует выделить условие, по которому возможно определить принадлежность записи к тому или иному классу. На основе этого условия формируются представления пользователей:

$$\begin{gathered} {}^{s}i = {}^{\sigma}\textit{условие класса записи } i^{\,r} \\ r = s_1 \cup s_2 \cup ... \cup s_n \ . \end{gathered} \quad (6)$$

По сравнению с предыдущим методом в данном случае все данные одной сущности хранятся в одной таблице, что позволяет организовать ключи и триггеры значительно проще. Распределение прав доступа пользователей осуществляется по отношению к набору представлений $\{s_1\ s_2\ ...\ s_n\}$. Однако при

добавлении нового класса записей остается необходимость производить изменение структуры БД (добавление и удаление представлений пользователей). Следует отметить то, что при необходимости выбора записей из разных классов пользователю придется прибегать к объединению (union) вместо простого выбора данных из одной таблицы. Существует возможность создания таких классов записей, которые будут включать в себя несколько других классов записей и предоставлять к ним доступ соответствующим пользовательским ролям. Однако это может привести к значительному возрастанию количества представлений пользователей, задействованных в системе безопасности.

В большем числе случаев данный метод организации доступа имеет применение. Но, как правило, оно ограничено тем, что набор классов записей не изменяется.

К недостаткам этого метода можно отнести возрастание затрачиваемых ресурсов серверной ЭВМ при работе с большим количеством представлений, которые требовательны к оперативной памяти.

Пятый метод - создание «динамического» представления пользователя, основанного на возможности создания такого специально спроектированного представления пользователя, в котором выбираются только те записи, к которым пользователь имеет доступ.

Пусть P – набор всех прав доступа, имеющихся в системе. Пусть P_i – набор прав доступа, необходимых для доступа к i–ой записи таблицы. Пусть Q_j – набор прав доступа пользователя j. P_i, Q_j и P относятся, как показано в формулах (7).

$$\begin{aligned} &P_i \in P, \\ &Q_j \in P. \end{aligned} \qquad (7)$$

Тогда условие доступа пользователя j к записи i может быть записано в виде выражения (8).

$$P_i \subset Q_j . \qquad (8)$$

В таком случае «динамическое» представление пользователя будет вычисляться по следующей формуле

$$s = \sigma_b r = \sigma_{Q_j \supset P_i} r . \qquad (9)$$

Для организации такого доступа необходима совокупность служебных таблиц, хранящих данные о правах пользователей Q_i и правах, необходимых для доступа к классам записей P_i. Пользователям, не имеющие доступа к данным определенной таблицы, не должен быть предоставлен доступ к соответствующему предоставлению пользователя. К положительным чертам данного метода можно отнести следующие:

1) все записи сущности хранятся в одной таблице, что позволяет легко манипулировать ключами и триггерами;

2) существует одно представление пользователя, к которому обращаются все пользователи, что значительно упрощает разработку клиентского приложения;

3) одно представление пользователя тратит не столь много оперативной памяти сервера.

К отрицательным чертам этой реализации доступа можно отнести сложность реализации такого представления и взаимодействия с ним служебных таблиц. Однако это возможно даже на самых простых СУБД, таких как mySQL.

Среди рассмотренных выше методов реализации различного доступа к классам записей в СУБД на практике возможно использовать 3 из них. При этом надо отметить, что при выборе метода реализации доступа следует уменьшить количество объектов БД, к которым должен быть предоставлен доступ, так как увеличение количества таких объектов приводит к увеличению непреднамеренных ошибок администратора БД

В случаях некритичности скорости работы ИС возможно использование технологии серверных приложений. Если классы записей заранее определены и имеют несколько различающийся физический смысл, возможно использование варианта с созданием множества представлений пользователей.

В случае, если данные имеют физически полностью идентичный смысл, а количество классов записей заранее не определено, наиболее удобным будет создание динамического представления пользователя.

Список использованной литературы:

1. Агальцов, В.П. Базы данных. В 2-х т. Т. 2. Распределенные и удаленные базы данных: Учебник / В.П. Агальцов. - М.: ИД ФОРУМ, НИЦ ИНФРА-М, 2013.
2. Агальцов, В.П. Базы данных. В 2-х т. Т. 1. Локальные базы данных: Учебник / В.П. Агальцов. - М.: ИД ФОРУМ, НИЦ ИНФРА-М, 2013.
3. Голицына, О.Л. Базы данных: Учебное пособие / О.Л. Голицына, Н.В. Максимов, И.И. Попов. - М.: Форум, 2012.
4. Карпова, И.П. Базы данных: Учебное пособие / И.П. Карпова. - СПб.: Питер, 2013.
5. Кириллов, В.В. Введение в реляционные базы данных. Введение в реляционные базы данных / В.В. Кириллов, Г.Ю. Громов. - СПб.: БХВ-Петербург, 2012.
6. Кузин, А.В. Базы данных: Учебное пособие для студ. высш. учеб. заведений / А.В. Кузин, С.В. Левонисова. - М.: ИЦ Академия, 2012.
7. Советов, Б.Я. Базы данных: теория и практика: Учебник для бакалавров / Б.Я. Советов, В.В. Цехановский, В.Д. Чертовской. - М.: Юрайт, 2013.
8. Т.Кайт. Oracle для профессионалов. Архитектура, методики программирования и особенности версий 9i, 10g и 11g (Expert Oracle Database Architecture: Oracle Database 9i, 10g, and 11g: Programming Techniques and Solutions), Изд-во: Вильямс, 2011.

9. К. Дж. Дейт. SQL и реляционная теория. Как грамотно писать код на SQL (SQL and Relational Theory: How to Write Accurate SQL Code).- Изд-во: Символ-Плюс, Серия: High Tech, 2010.
10. Джейсон С. Каучмэн, Ульрике Швинн. Подготовка администраторов баз данных (Oracle Certified Professional DBA: Certification Exam Guide). - Изд-во: Лори, 2009.
11. Р.Гринвальд, Р. Стаковьяк, Дж. Стерн. Oracle 11g. Основы. Oracle Essentials: Oracle Database 11g. - Изд-во: Символ-Плюс, 2009.
12. Джейсон Прайс.SQL для Oracle 10g (Oracle Database 10g SQL).-Изд-во: Лори, 2010.