# АНАЛИЗ СПОСОБОВ ОРГАНИЗАЦИИ ИЕРАРХИЧЕСКИХ СТРУКТУР В РЕЛЯЦИОННЫХ БАЗАХ ДАННЫХ

*Савёнок В.А.*

*Белорусский государственный университет информатики и радиоэлектроники*
*г. Минск, Республика Беларусь*

*Медведев С.А. – к.т.н.,доцент*

Одним из центральных понятий в объектно-ориентированном программировании является класс и, следовательно, иерархия классов выступает в качестве основополагающего типа связи классов, представленного в виде иерархии. Таким образом, наиболее характерной задачей для интерфейсных модулей взаимодействия с базами данных SQL типа на основе объектно-ориентированного интерфейса является организация эффективного хранения и работы с иерархическими структурами в реляционной базе данных.

Существует четыре основных способа представления иерархических структур в реляционных БД: хранение ссылки на родителя, метод накопления пути, модель вложенных множеств и таблица замыканий. Рассмотрим перечисленные методы на примере реализации дерева классов представленного на рисунке 1.

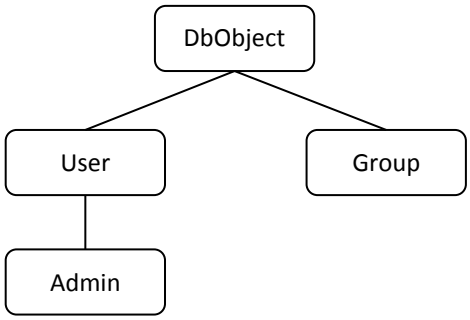


Рисунок 1 – Графическая представление иерархии классов

Самым простым в понимании способом организации иерархической структуры в реляционной БД является *хранение в каждом элементе иерархии ссылки на родителя*. В такой структуре каждый узел дерева знает своего непосредственного родителя. К плюсам этого метода относится простота хранения структуры, простота добавления и удаления узла из дерева, простота получения непосредственных потомков узла, а также обеспечение ссылочной целостности при помощи стандартных механизмов СУБД. К недостаткам стоит отнести сложность манипуляций с поддеревьями: в частности, получение всех потомков какого-либо элемента, т. к. для этого необходимо произвести столько операций соединения, сколько уровней содержится в интересующем нас поддереве.

Прямым развитием метода хранения ссылки на родителя является *метод накопления пути*. Суть его заключается в добавлении поля, сохраняющего полный путь к текущему элементу в виде строки. Этот способ позволяет проще работать с поддеревьями, однако не предоставляет возможности использования стандартных средств для поддержания ссылочной целостности. Более того, операции по удалению элементов из середины иерархии требует перебора всех элементов, находящихся на нижних уровнях, и обновления их путей, что негативно сказывается на производительности операции.

Совершенно другой подход реализуется в *модели вложенных множеств*. В ней дерево классов из примера будет иметь вид, изображенный на рисунке 2. Здесь связь «родитель – сын» представлена вложенностью сына в родителя. Для того, чтобы представить связь в виде таблицы необходимо завести два дополнительны поля, обозначающих левую и правую границы множества.

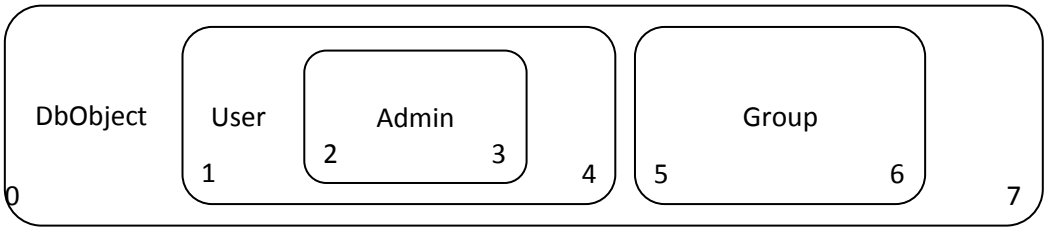


Рисунок 2 – Представление иерархии классов в модели вложенных множеств

Множества можно интерпретировать как одномерные отрезки, имеющие координаты начала и конца. Алгоритм получения поддерева в текущем подходе достаточно прост – необходимо запросить из базы все записи, границы которых находятся между границами родителя. При этом алгоритм не зависит от количества уровней в иерархии. Получение пути от листа к корню также не требует большого количества операций соединения. К недостаткам модели относится высокая трудоёмкость реализации операций добавления и удаления элементов иерархии, т. к. при этом необходимо производить пересчет границ всех множеств; отсутствует поддержка стандартных механизмов обеспечения ссылочной целостности в базе данных. Сложность восприятия описанного подхода также приводит к осуществлению выбора в пользу других способов организации иерархий.

Последний из рассматриваемых способов организации иерархических структур в реляционных базах данных – *таблица замыканий* – является более универсальным, т. к. позволяет представить в реляционной базе данных и более сложные структуры – графы. Особенностью этого способа является необходимость введения новой таблицы, содержащей связи каждого элемента иерархии со всеми его предками, а также ссылки каждого элемента на самого себя – собственно замыкание. Все связи изображены на рисунке 3.

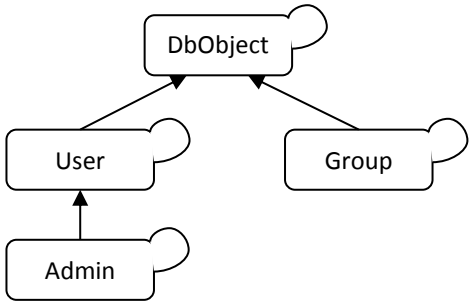


Рисунок 3 – Представление связей между элементами иерархии в методе замыканий

В описанном подходе исходная таблица не модифицируется. Поддержание таблицы замыканий в актуальном состоянии требует дополнительных операций вставки и удаления. С одной стороны, увеличивается расход памяти, а с другой – значительно упрощается работа со структурой данных. К плюсам можно отнести возможность использования стандартных механизмов обеспечения ссылочной целостности в БД. Более того, в СУБД, поддерживающих каскадные операции, существует возможность упрощения операции удаления элемента из дерева путем установки соответствующей каскадной операции. К минусам относится неэффективное добавление новых элементов в конец глубокой иерархии по сравнению со вставкой вблизи корня дерева. Однако, учитывая тот факт, что в реальных задачах редко встречается необходимость в использовании иерархий глубокой вложенности, его значимость следует оценивать в зависимости от реальных условий использования рассматриваемого подхода.

При сравнительном анализе описанных способов основными критериями для оценки были выбраны сложность реализации и поддержка стандартных механизмов обеспечения ссылочной целостности. Также следует отметить, что метод накопления пути и модель вложенных множеств не поддерживают стандартные механизмы обеспечения ссылочной целостности, что понижает их надежность и негативно сказывается на общей оценке. Из проведенного анализа следует, что наиболее гибкой, оптимальной по производительности и надежной с точки зрения целостности данных моделью организации иерархических структур в реляционных базах данных является таблица замыканий. Несмотря на необходимость создания дополнительной таблицы для хранения всех связей элементов и неэффективность вставки в глубину иерархии, выбор последнего подхода в качестве оптимального оправдан ввиду статистического преобладания операций чтения над операциями записи при работе с БД. С другой стороны, для работы с деревом, при которой не будут производиться операции с поддеревьями, хранение ссылок на непосредственного родителя в элементах иерархии является предпочтительнее ввиду простоты и скорости реализации.

***Список использованных источников:***

1. Ramez Elmasri, Smakant B. Navathe. Fundamentals of Database Systems, 6th Edition. – Addison-Wesley, 2011;
2. C.J. Date, Hugh Darwen. Databases, Types, and The Relational Model: The Third Manifesto. – Pearson, 2006;
3. Object Relational Mapping (OR Mapping) Definition [Электронный ресурс]. – Режим доступа: http://www.service-architecture.com/articles/object-relational-mapping/object-relational_mapping_or_mapping_definition.html Дата доступа: 20.02.2019;
4. Bill Karwin. Models for hierarchical data [Электронный ресурс]. – Режим доступа: https://www.slideshare.net/billkarwin/models-for-hierarchical-data Дата доступа: 03.03.2019;
5. Managing Hierarchical Data in MySQL [Электронный ресурс]. – Режим доступа: http://mikehillyer.com/articles/managing-hierarchical-data-in-mysql/ Дата доступа: 03.03.2019.