Методы представления сложных объектов во внешней памяти

И.Ю.Капризкина
Санкт-Петербургский университет
198904 Санкт-Петербург, Библиотечная пл. 2
E-mail: ban@hq.math.lgu.spb.su

The paper contains a survey of methods for complex object clustering in the secondary storage. Two main strategies, called vertical and horizontal scan order, are emphasized. Some variations of this strategies are presented.

Статья содержит описание основных методов представления иерархических сложных объектов во внешней памяти. Выделяются две основные стратегии такого представления и для каждой из них рассматриваются возможные вариации.

1 Введение

Понятие сложного объекта встречается в литературе в разных контекстах, однако многие модели данных используют это понятие сходным образом. Для многих моделей данных характерно представление сложного объекта в виде иерархически вложенных множеств атомарных объектов. Исторические корни такой структуры лежат в развитии оригинальной реляционной модели [2] к моделям вложенных отношений ([6, 9, 10] и во внедрении некоторых понятий объектно-ориентированного программирования в теорию моделей данных ([11, 5, 4]).

Атомарный объект содержит данные, сложный объект позволяет усложнить семантику данных. Для построения сложных объектов в большинстве моделей используются операторы создания кортежа и создания множества. На рисунке 1 приведен пример сложного объекта. Квадратные скобки указывают на кортеж; фигурные скобки соответствуют множеству.

Для отображения иерархически структурированного сложного объекта во внешнюю память

```
Departments
   {[ Dep_No  : INTEGER,
      Mgr_No  : INTEGER,
      Projects:
         {[ Pro_No  : INTEGER,
            Pro_Name: STRING(50),
            Members :
               {[Emp_No   : INTEGER,
                 Function : STRING(30)
               ]}
         ]},
      Budget    : INTEGER
   ]};
```

Рис. 1: Сложный объект

необходимо преобразовать его структуру к линейной, возможно с использованием промежуточных логических уровней. Существуют две основных стратегии для такого отображения, которые я буду называть горизонтальным и вертикальным сканированием.

При вертикальном сканировании разбор сложного объекта выполняется покомпонентно. Так как компонента также является сложным объектом, к нему может быть применено вертикальное сканирование и т.д. Такая стратегия позволяет сохранить логическую структуру объекта, без использования дополнительной памяти.

Для объекта, схема которого приведена на рисунке 1 такое сканирование будет означать, что для каждого подразделения все сведения логически размещаются в одном месте. Т.е., для каждого подразделения вначале будет выписан номер подразделения, регистрационный номер руководителя, затем будут сканироваться все проекты, выполняемые в подразделении и, наконец, будет выписан бюджет подразделения. В работе [12] такое размещение названо прямой моделью представле-

```
level 1: [Dep_No, Mgr_No, Budget]

level 2: [Dep_Id, Pro_No, Pro_Name]

level 3: [Pro_Id, Emp_No, Function]
```

Рис. 2: Горизонтальное сканирование

ния сложного объекта.

Отметим, что при таком сканировании информация для каждой компоненты составного объекта кластеризуется *логически*, в то время как физически она может быть размещена в нескольких блоках, разбросанных по диску.

При горизонтальном сканировании дерево сложного объекта разбирается по уровням иерархии, информация, относящаяся к отдельному уровню, кластеризуется и для её представления используются логически отдельный набор блоков внешней памяти. Для объекта с рисунка 1 будет создано несколько логических уровней, первый из которых будет относиться к подразделению, второй – к информации о проектах, третий – к работникам, занятым в проекте. На рисунке 2 приведена информация, которая будет хранится на каждом из уровней.

На уровне 2 появляется дополнительное поле данных `Dep_Id`, соответствующее уникальному идентификатору подразделения. Таким образом, зная уникальный идентификатор подразделения, можно получить информацию обо всех проектах данного подразделения. На уровне 3 появляется дополнительное поле данных `Pro_Id`, содержащее уникальный идентификатор проекта, среди всех проектов, выполняемых во всех подразделениях.

Такая стратегия дополнительно требует хранения информации об отношениях вложенности между отдельными уровнями и позволяет сохранить структуру сложного объекта косвенным образом при помощи использования уникальных идентификаторов. В работе [12] такая стратегия названа декомпозиционной моделью представления сложного объекта.

В чистом виде эти стратегии не встречаются почти нигде. Для повышения эффективности представления во многих работах используются дополнительные логические и физические структуры, стратегии совмещаются. Данная работа содержит обзор встречающихся в литературе способов представления сложных объектов с точки зрения их эффективности, по отношению к чисто горизонтальному или чисто вертикальному сканированию. Такой подход позволяет лучше выделить недостатки и достоинства рассматриваемых моделей.

2 Анализ эффективности горизонтального и вертикального сканирования

При вертикальном сканировании максимальное число доступов к внешней памяти, необходимое для извлечения сложного объекта, начиная с какого-либо уровня иерархии, определяется следующей формулой:

$$R_{max} = \sum_{n=1}^{N} r_n \tag{1}$$

Здесь R_{max} — максимальное число доступов к диску, необходимое для чтения объекта. Переменная r_n обозначает число доступов к диску при чтении информации, расположенной на уровне иерархии n сложного объекта. Для фактического числа доступов R выполнено неравенство:

$$1 \leq R \leq R_{max} \tag{2}$$

Так, может оказаться, что весь сложный объект верхнего уровня помещается в один физический блок. В таком случае потребуется ровно один доступ к файлу.

При горизонтальном сканировании число доступов, необходимое для извлечения сложного объекта, начиная с определенного уровня определяется похожей формулой:

$$R = \sum_{n=1}^{N} r_n \tag{3}$$

Однако, следует заметить, что в этом случае равенство строгое, в то время как в предыдущем случае число доступов к диску существенно зависит от размеров сложного объекта. Кроме того горизонтальное сканирование накладывает определенные ограничения на возможность моделирования списков, использующихся в некоторых моделях данных в качестве структурных операторов наряду с множествами и кортежами [6], т.к. при удалении сложного объекта из списка меняются значения всех последующих ключей.

Поиск по значению полей для горизонтального и вертикального сканирования выполняется по разному. В случае горизонтального сканирования необходимо вести поиск по значению в отдельном

логическом файле. Затем последовательно применяется такой же тип поиска в файлах, вплоть до верхнего уровня, для получения полной информации о сложном объекте. При вертикальном сканировании поиск по значению требует полного просмотра сложного объекта.

3 Повышение эффективности горизонтального и вертикального сканирования

При анализе эффективности горизонтального сканирования выясняется, что наибольшее число доступов требуется при чтении сложного объекта как единого целого. При этом требуется проход логической структуры объекта сверху вниз вплоть до листьев. В литературе предложено несколько возможных методов уменьшения числа доступов в этом случае. Так, в работе [7, 8] рассматривается возможность совмещения всех логических уровней в одном файле.

В этих работах рассматривается следующая модель данных. Абстрактная база данных состоит из объектов. Объект может быть конкретным (число или строка) или абстрактным. Каждый объект может принадлежать одной или нескольким категориям, которые являются абстрактными объектами. Между объектами могут быть установлены связи. Каждый объект или связь имеют уникальный на уровне базы данных целочисленный идентификатор.

Для представления абстрактной базы данных формируется файл, состоящий из логических записей следующей структуры.

xC Записи такого типа хранят информацию об объектах, принадлежащих категориям. Абстрактный объект с идентификатором x принадлежит категории с идентификатором C.

$\bar{C}$x Информация обо всех объектах, принадлежащих категории C. $\bar{C}$ представляет собой идентификатор, соответствующий идентификатору C, но не совпадающий с ним. Например, C может быть целым положительным числом и тогда в качестве $\bar{C}$ выбирается такое же по модулю отрицательное число.

xRv Здесь x представляет собой абстрактный объект, v – конкретный объект, R – идентификатор связи. Неформально записи такого типа хранят информацию о значениях абстрактных объектов.

$\bar{R}$vx $\bar{R}$ представляет собой идентификатор, связанный с идентификатором R, v и x имеют тот же смысл, что и в предыдущем случае. Неформально записи такого типа хранят информацию обо всех значениях абстрактного объекта.

xRy Здесь x, y представляют собой абстрактные объекты, R – идентификатор связи. Неформально записи такого типа хранят информацию о связях между абстрактными объектами.

y$\bar{R}$x Здесь x, y представляют собой абстрактные объекты, $\bar{R}$ – инвертированный идентификатор связи. Неформально записи такого типа хранят инвертированную информацию о связях между абстрактными объектами.

Файл, сформированный из записей такого типа имеет структуру B-дерева с не уникальными ключами. Кроме того, данная методология предполагает экономичное представление ключей и конкретных объектов во внешней памяти, позволяющих уменьшить высоту B-дерева.

Описанная выше структура файла была разработана для представления семантических моделей данных. При применении данной методологии к нашему случаю, множество и кортеж могут быть представлены в виде абстрактных объектов. Связи могут использоваться для назначения значений атрибутам и сопоставления значений, уникальным идентификаторам, если значение не является элементом кортежа.

При считывании значения сложного объекта необходимо считать значения всех его компонент и повторить операцию чтения для каждой из них. Формула 3 преобразовывается к следующему виду.

$$R \leq r_B * N \tag{4}$$

Здесь r_B обозначает число доступов к диску при чтении информации из B-дерева, N – число уровней в дереве иерархии сложного объекта. Равенство преобразуется в неравенство за счет возможного размещения информации с разных уровней в одном физическом блоке, что позволяет уменьшить общее число считанных блоков.

Файл такого типа позволяет эффективно выполнять запросы на поиск значений на определенном уровне. Структура B-дерева поддерживает упорядоченность ключей. Таким образом, эффективно выполняются разного рода запросы на поиск в определенных границах. Так может оказаться, что информация о всех абстрактных объектах, принадлежащих определенной категории, размещается в отдельном физическом блоке. Тогда запрос

```
Departments
   {[ Dep_No  : INTEGER,
      Mgr_No  : INTEGER,
      Projects:
         {[ Pro_No  : INTEGER,
            Pro_Name: STRING(50),
            Members :
               {[Emp_No   : INTEGER,
                 Function : STRING(30)
               ]}
         ]},
      Budget    : INTEGER
   ]};
```

Рис. 3: Иерархический индекс

на поиск записей вида $\bar{C}x$ будет выполнен за один доступ к диску. Однако, может оказаться, что эта информация размещается в двух соседних узловых блоках, хотя по размерам могла бы войти в один блок.

Отметим, что данная методология не допускает длинных конкретных объектов, т.е. объектов, размер которых превышает размер физического блока.

В работе [12] вводится понятие иерархических индексов соединения. В то время как основная информация хранится в отдельных файлах, полученных в результате горизонтального сканирования, дополнительный файл содержит структуру сложного объекта. В дополнительном файле хранятся только уникальные идентификаторы, он формируется про помощи вертикального сканирования и содержит как бы скелет сложного объекта. За счет отсутствия данных в этом файле и относительно малых размеров идентификаторов, структура отдельно взятого сложного объекта почти всегда должна вмещаться в один физический блок.

На рисунке 3 приведен пример иерархического индекса для сложного объекта с рисунка 1. При выполнении запроса на считывание всего сложного объекта вначале анализируется файл, содержащий иерархические индексы, для получения идентификаторов кортежей, содержащих данные. Затем выполняется поиск в соответствующих файлах. Такого рода стратегия представляет собой своеобразное дополнение горизонтального сканирования вертикальным и позволяет уменьшить число доступов к диску.

В работе [3] предложена методология, реализующая черты вертикального сканирования. Каждый отдельный сложный объект занимает определенныый набор блоков внешней памяти. Внутренняя структура объекта реализуется при помощи ссылок, как внутри одного блока, так и между блоками. Структура сложного объекта хранится в первом блоке адресного пространства, отведенного под объект. Кроме того, в этом блоке хранится заголовок объекта, содержащий информацию о номерах страниц, занятых данным сложным объектом.

На рисунке 3 приведен пример реализации сложного объекта с рисунка 1, использующий описываемую методологию.

Сложный объект занимает четыре физических блока, помеченные как p1, p2, p3, p4. В заголовке сложного объекта содержится информация о занимаемых блоках и указатель на начало описания структуры. Множество подразделений представляется в этом случае единственной записью, содержащей массив указателей на элементы множества. На рисунке это множество содержит два подразделения, обозначенные как D1 и D2, соответственно. Для каждого из подразделений записывается указатель на запись с данными, содержащую номер подразделения, личный номер руководителя и бюджет. Затем следует массив указателей на проекты, выполняемые в подразделении.

В подразделении D1 выполняются два проекта P11 и P12. В подразделении D2 выполняется один проект, помеченный на рисунке P21. Для каждого проекта первая ссылка указывает на запись данных для проекта, содержащую номер и имя проекта. Далее идет массив указателей, реализующий множество участников данного проекта.

В проекте P11 занято трое участников, в проектах P12 и ¶21 – по два человека. Записи для участников проектов кластеризуются в отдельных физических блоках. Из примера видно, что зная месторасположение объекта в иерархии мы можем извлечь информацию о нем, использую данные о его структуре, собранные в первом блоке.

При выполнении запроса на чтение информации из подобъекта, используются два доступа к диску. Один – для чтения первого блока, второй – для чтения информации на основании полученного значения указателей. При выполнении запросов, относящихся ко всему сложному объекту, требуется один доступ для чтения заголовка. Затем выполняется подкачка в оперативную память страниц, содержащих сложный объект.

Описанная методология не предоставляет возможности для поиска значеия на определенном уровне. Возможна реализация такого поиска при помощи полного просмотра сложного объекта, что может оказаться не эффективным при больших

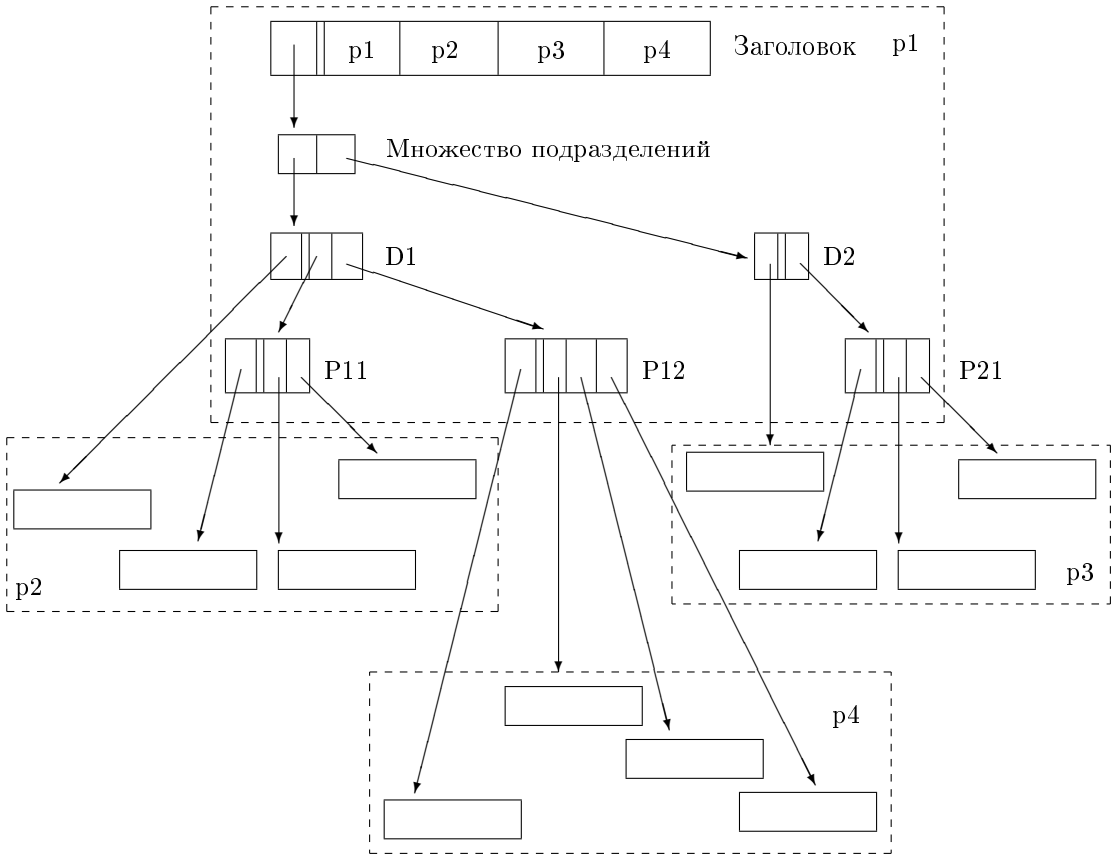

Рис. 4: Экземпляр сложного объекта

его размерах.

Кроме того, неясна стратегия обработки очень большого сложного объекта, структура которого не может быть размещена внутри одного физического блока.

При вертикальном сканировании эффективность теряется при выполнении запросов на поиск значений, находящихся на определенном уровне иерархии. Основным методом снижения потерь является метод ведения индексов. Отдельно к основному логическому файлу данных формируются файлы индексов, содержащие значения на определенном уровне и уникальные идентификаторы значений.

Такая методология используется в [1]. Таким образом, эффективность выполнения запроса на поиск определенного скалярного значения полностью определяется эффективностью реализации индекса, в то время как эффективность чтения объекта сверху вниз по уровням иерархии остается оптимальной. Недостаток данной стратегии состоит в использовании дополнительных объемов внешней памяти для реализации индексов.

4 Заключение

Предложенные в литературе методы отображения иерархически структурированных сложных объектов во внешнюю память можно разделить на две большие группы, в зависимости от используемой стратегии сканирования.

При вертикальном сканировании эффективно выполняются запросы на поиск объектов небольшого размера, т.к. такой объект может быть размещен в одном физическом блоке. Горизонтальное сканирование позволяет эффективно выполнять запросы на поиск значений промежуточных уровней иерархии.

Некоторые методы, используемые в литературе и описанные в статье позволяют повысить эффективность горизонтального и вертикального сканирования. В дальнейшем предполагается провести

статистическое сравнение эффективности данных методов с использованием возможностей, предоставляемых системой хранения, описанной в работе [1].

Список литературы

[1] Б.А. Новиков. Реализация сложных объектов в системе хранения. *УСuM*, 7, 1991, 46–52 с.

[2] E. F. Codd. A relational model of data for large shared data banks. *Communications of the ACM*, 6(13), 1970.

[3] U. Deppisch, H.-D. Paul, and H.-J. Schek. A storage system for complex objects. In *Int. workshop on object-oriented database systems*, pages 183–195, 1986.

[4] O. Deux et al. The O_2 system. *Communications of the ACM*, 34(10):34–48, October 1991.

[5] Charles Lamb, Gordon Landis, Jack Orenstein, and Dan Weinreb. The ObjectStore database system. *Communications of the ACM*, 34(10):50–63, October 1991.

[6] P. Pistor and P.Dadam. The advanced information management prototype. In *Nested relations and complex objects in databases*, volume 361 of *Lect. Notes in Comp. Sci.*, pages 3–26, 1989.

[7] N. Rishe. Efficient organization of semantic databases. In *Foundations of Data Organization and Algorithms. Proceedings 3rd Int. Conf., FODO 1989*, volume 367 of *Lect. Notes in Comp. Sci.*, pages 114–127, Paris, France, June 1989.

[8] N. Rishe. A file structure for semantic databases. *Inf. Syst. (Oxford)*, pages 375–385, 1991.

[9] H.-J. Schek and M.H. Scholl. The two roles of nested relations in DASDBS project. In *Nested relations and complex objects in databases*, volume 361 of *Lect. Notes in Comp. Sci.*, pages 50–68, 1989.

[10] M. Scholl, S. Abiteboul, F. Banchilhon, N. Bidoit, S. Gamerman, D. Plateau, P. Richard, and A. Verroust. VERSO: A database machine based on nested relations. In *Nested relations and complex objects in databases*, volume 361 of *Lect. Notes in Comp. Sci.*, pages 27–49, 1989.

[11] Michael Stonebraker and Gred Kemnitz. The POSTGRES next generation database management system. *Communications of the ACM*, 34(10):78–92, October 1991.

[12] P. Valduriez, S. Khoshafian, and G. Copeland. Implementation technique of complex objects. In *Proc. 12 conf. VLDB*, pages 101–110, 1986.