Полтавцев А.А. Иерархические схемы в РСУБД для структур типа “Перечень материалов”. // Проблемы информатики в образовании, управлении, экономике и технике: Сб. статей XVI Междунар. научно-техн. конф. – Пенза: ПДЗ, 2016. – С. 194-199.

УДК 004.652.4

# ИЕРАРХИЧЕСКИЕ СХЕМЫ В РСУБД ДЛЯ СТРУКТУР ТИПА «ПЕРЕЧЕНЬ МАТЕРИАЛОВ»

А.А. Полтавцев

# RDB HIERARCHICAL SCHEMES FOR STRUCTURS OF TYPE «THE LIST OF MATERIALS»

A.A. Poltavtsev

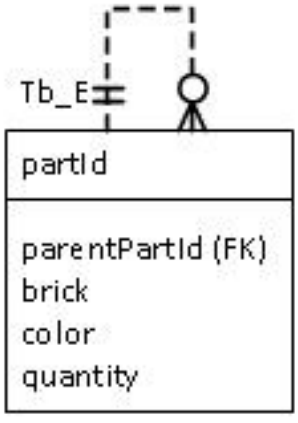
**Аннотация.** В предыдущих статьях автора рассматривались проблемы представления структурированной информации в реляционных базах данных. Рассматривались иерархические структуры, осложненные дополнительными требованиями к структурированию их информации. Изложение велось с использованием структуры «перечень материалов». В данной работе показано, как сформировать схему хранения с использованием рекурсивной ссылки и обработать с помощью рекурсивных компонентов стандарта SQL99.

**Ключевые слова:** базы данных, реляционная модель данных, реляционная схема, рекурсия.

**Abstract.** In previous author articles the problems of structured information representation in relational databases were considered. The hierarchical structures complicated by additional requirements to their information structurization were examined. The statement was conducted with use «list of materials « structure. In the given paper is shown storage scheme with using recursive link and handling with standard SQL99recursive components use.

**Keywords:** databases, relational data model, relational scheme, recursion.

Наиболее распространенным решением для хранения иерархий является использование рекурсивной ссылки [1, 2]. Для примера о сборках букв/слов из блоков получим:




Соответствующее реляционное отношение будет

| partId | parentPartId | brick | color | quantity |
|---|---|---|---|---|
| 50 | NULL | E | NULL | 1 |
| 1 | 50 | 1×1 | yellow | 4 |
| 2 | 50 | 1×3 | Black | 1 |
| 3 | 50 | 1×4 | Blue | 2 |

Исследуемая структура отличается от обычной иерархии прежде всего тем, что в ней процесс ассемблирования многоэтапен: одна сборка может быть частью другой, более общей. В примере сборки E и F могут быть частями сборка (слова) EF. В этом случае получаем реляционное отношение следующего вида:

| partId | parentPartId | brick | color | quantity |
|---|---|---|---|---|
| 5060 | NULL | EF | NULL | 1 |
| 50 | 5060 | E | NULL | 1 |
| 1 | 50 | 1×1 | yellow | 4 |
| 2 | 50 | 1×3 | black | 1 |
| 3 | 50 | 1×4 | blue | 2 |
| 60 | 5060 | F | NULL | 1 |
| 4 | 60 | 1×1 | yellow | 5 |
| 5 | 60 | 1×3 | black | 1 |
| 6 | 60 | 1×4 | blue | 1 |

Содержимое данного отношения представляет собой то, что в производстве называют деталировкой продукта (по частям). Для того чтобы деталировка имела прикладной смысл, необходимо ее показать с указанием уровней ассемблирования, т.е. указав иерархический уровень каждого компонента. Для модельного примера это

| Уровень | Компонент | Цвет | Количество |
|---|---|---|---|
| 1 | EF | NULL | 1 |
| 2 | E | NULL | 1 |
| 3 | 1×1 | yellow | 4 |
| 3 | 1×3 | black | 1 |
| 3 | 1×4 | blue | 2 |
| 3 | F | NULL | 1 |
| 3 | 1×1 | yellow | 5 |
| 3 | 1×3 | black | 1 |
| 3 | 1×4 | blue | 1 |

Для данной схемы хранения наиболее рациональным способом реализации SQL запроса является использование рекурсивного CTE (commontableexpression) для СУБД, поддерживающих это понятие стандарта SQL99 (Microsoft SQL Server), или конструкций разработчика СУБД, заменяющих его (connect by Oracle).

В CTE необходимо определить атрибут уровень. Он необходим, чтобы соотнести подсборки и исходные компоненты с их уровнем иерархии. Первый уровень иерархии задается в якорном подзапросе, а остальные вычисляются в рекурсивной части.

```
--WITH BOMcte(partId, Name, Color, Quantity, Lvl)
--AS
--(
--SELECT AdjacencyListBOM.partId,
--       CAST(AdjacencyListBOM.Name as nvarchar(20) ),
--       AdjacencyListBOM.Color,
```

```
--      AdjacencyListBOM.Quantity,
--      0
--FROM AdjacencyListBOM
--WHERE AdjacencyListBOM.parentPartId IS NULL -- AND AdjacencyListBOM.PartId = 5060
--
--UNION ALL
--
--SELECT AdjacencyListBOM.partId,
--CAST( (REPLICATE ('| ', Lvl + 1) + AdjacencyListBOM.Name) as nvarchar(20)),
--AdjacencyListBOM.Color,
--AdjacencyListBOM.Quantity,
--Lvl + 1
--FROM AdjacencyListBOM
--JOIN BOMcteAScte
--ON AdjacencyListBOM.ParentPartId = cte.PartId
--)
--SELECT Lvl, Name, Color, Quantity
--FROM BOMcte;
```

Результат запроса:

| Lvl | Name | Color | Quantity |
|---|---|---|---|
| 0 | EF | NA | 1 |
| 1 | \|E | NA | 1 |
| 1 | \|F | NA | 1 |
| 2 | \|\|1x1 | yellow | 5 |
| 2 | \|\|1x3 | black | 1 |
| 2 | \|\|1x4 | blue | 1 |
| 2 | \|\|1x1 | yellow | 4 |
| 2 | \|\|1x3 | black | 1 |
| 2 | \|\|1x4 | blue | 2 |

Эту же технику можно применить, если необходимо получить полные иерархические пути сборки (т.е. построить порядок сборки изделия). Для модельного примера это может выглядеть следующим образом:

```
\EF
\EF\E
\EF\E\1×1 yellow 4
\EF\E\1×3 black  1
\EF\E\1×4 blue   2
\EF\F
\EF\F\1×1 yellow 5
\EF\F\1×3 black  1
\EF\F\1×4 blue   1
```

SQL запрос:

```
--WITH BOMcte(PartId, Name, Color, Quantity, Lvl, ParentPartId, Sort)
```

```
--AS
--(
--SELECT AdjacencyListBOM.PartId,
--CAST(AdjacencyListBOM.Name as nvarchar(20)),
--AdjacencyListBOM.Color,
--AdjacencyListBOM.Quantity,
--0,
--AdjacencyListBOM.ParentPartId,
---- hierarchical relationships (sort key)
---- sort key – полный путь от корня до данного узла
--CAST('\' + AdjacencyListBOM.Name as nvarchar(30))
--FROM AdjacencyListBOM
--WHERE AdjacencyListBOM.ParentPartId IS NULL
--
--UNION ALL
--
--SELECT AdjacencyListBOM.PartId,
--CAST(REPLICATE('| ', Lvl + 1) + AdjacencyListBOM.Name as nvarchar(20)),
--AdjacencyListBOM.Color,
--AdjacencyListBOM.Quantity,
--Lvl + 1,
--AdjacencyListBOM.ParentPartId,
--CAST(cte.Sort + '\' + AdjacencyListBOM.Name as nvarchar(30))
--FROM AdjacencyListBOM
--JOIN BOMcteAScte
--ON AdjacencyListBOM.ParentPartId = cte.PartId
--)
--SELECT Lvl, Name, Color, Quantity, Sort
--FROM BOMcte
--ORDER BY Sort
```

Результат запроса:

| Lvl | Name | Color | Quantity | Sort | | |
|---|---|---|---|---|---|---|
| 0 | EF | NA | 1 | \EF | | |
| 1 | \|E | NA | 1 | \EF | \E | |
| 2 | \|\| 1x1 | yellow | 4 | \EF | \E | \1x1 |
| 2 | \|\| 1x3 | black | 1 | \EF | \E | \1x3 |
| 2 | \|\| 1x4 | blue | 2 | \EF | \E | \1x4 |
| 1 | \|F | NA | 1 | \EF | \F | |
| 2 | \|\| 1x1 | yellow | 5 | \EF | \F | \1x1 |
| 2 | \|\| 1x3 | black | 1 | \EF | \F | \1x3 |
| 2 | \|\| 1x4 | blue | 1 | \EF | \F | \1x4 |

Таким образом, может быть сформирована схема хранения данных с использованием рекурсивных ссылок, допускающая обработку данных с использованием рекурсивных компонентов стандарта SQL99.

Библиографический список

1. Дейт К. Введение в системы баз данных. М.: Диалектика, 1998. 784 с.
2. Полтавцева М.А. Хранение сложных структур данных в реляционной базе данных. Тверь: Изд-во ТвГТУ, 2013. 184 с.

**Полтавцев Анатолий Алексеевич**
Тверской государственный
технический университет,
г. Тверь, Россия
E-mail: aapolt@gmail.com

**Poltavtsev A.A.**
Tver State Technical University,
Tver, Russia