Санкт-Петербургский государственный университет

Кафедра системного программирования

Группа 24.М41-мм

Подзапросы в PosDB

Кузин Яков Сергеевич

Отчёт по учебной практике
в форме «Сравнение»

Научный руководитель:
ассистент кафедры информационно-аналитических систем Чернышев Г. А.

Санкт-Петербург
2025

Оглавление

Введение

Как известно, колоночные базы данных хранят данные на диске в виде столбцов. Они не являются идеальными во всех сценариях их использования, однако в некоторых из них позволяют добиться высокой эффективности [1]. К примеру, колоночные базы данных в значительной степени выигрывают у традиционных, когда речь идет об аналитических запросах. Если подобный запрос требует сканирования больших объемов данных, но при этом затрагивает лишь небольшую часть колонок рассматриваемых таблиц, хранение данных в виде столбцов позволяет избежать избыточных операций ввода-вывода. Это и приводит к сокращению времени выполнения запроса, ведь, как известно [11], подобные операции являются одними из самых дорогостоящих.

Со стороны пользователя колоночные системы ничем не отличаются от реляционных: он вводит привычные для него SQL запросы и получает результат в стандартном его представлении. Однако сторона реализации несет в себе значительные отличия и, порой, технические сложности. Одной из важнейших составляющих любого хоть сколько-нибудь сложного запроса являются подзапросы. Это очень мощный инструмент, которым активно пользуются многие специалисты и простые пользователи. Поэтому их поддержка очень важна в любой системе.

PosDB [8] — это распределенная колоночная СУБД, предназначенная для эффективного выполнения аналитических запросов. На начало учебной практики PosDB не имеет поддержку всех типов подзапросов, что и является отправной точкой для написания данной работы.

Перед внедрением новой функциональности необходимо научиться классифицировать подзапросы таким образом, чтобы их можно было идентифицировать по местоположению во внешнем запросе. Это позволит более точно анализировать структуру запросов и выявлять зависимости между основным запросом и подзапросами. Подобная классификация вместе с обзором предметной области и анализом существующих решений и будут представлены в данной работе.

1. Постановка задачи

Целью настоящей работы является обзор подзапросов в реляционных системах управления базами данных (СУБД), а также анализ их применения. Для ее достижения были поставлены следующие задачи:

1. Изучить теоретические основы подзапросов, их типы и синтаксис в различных реляционных СУБД.
2. Провести анализ существующих систем на предмет реализации в них разных типов подзапросов.
3. Рассмотреть практические примеры использования подзапросов в различных сценариях работы с базами данных.
4. Разработать и систематизировать несколько классификаций подзапросов по таким критериям, как взаимосвязь с окружением, тип возвращаемого значения и контекст использования.

2. Обзор

В настоящее время реляционные системы управления базами данных играют ключевую роль в хранении, обработки и организации данных [6]. Одним из важнейших инструментов этих систем являются подзапросы. Они представляют собой запросы, вложенные в другие запросы, и позволяют выполнять сложные задачи, обеспечивая высокую гибкость при работе с данными.

Использование подзапросов имеет как сильные, так и слабые стороны. К преимуществам можно отнести уменьшение количества объединений, улучшение читаемости и модульности SQL-кода, а также упрощение его дальнейшего сопровождения. Однако подзапросы имеют и ряд недостатков. Во-первых, их использование может привести к увеличению времени получения искомых данных. Во-вторых, избыточное использование подзапросов может уменьшить ясность основного запроса, особенно если имеется многократная их вложенность. Наконец, в отличие от простых запросов, результаты подзапросов могут кешироваться не так эффективно, что также может негативно сказаться на производительности.

Смотря на подзапросы с формальной стороны, можно заметить, что они отсутствуют в реляционной алгебре. Это связано с тем, что она ориентирована на выполнение операций над множествами данных, а не на их иерархическую структуру. Для борьбы с отсутствием подзапросов в реляционной алгебре разработаны различные расширения и методы, основанные на преобразовании запросов в их эквивалентные формы при помощи соединений и иных операций. Например, в книге [10] авторы показывают, что при помощи операций полусоединения и антисоединения можно переписать подзапросы, использующие предикаты EXISTS и NOT EXISTS. А для преобразования подзапросов с оператором сравнения с множеством (таких как, например, $= \text{ALL}$ или $> \text{SOME}$) уже используются специальные соединения с маркировкой.

В данном разделе будет представлен обзор подзапросов в реляционных СУБД, включая их типы и синтаксис. Кроме того, будут рассмот-

рены практические примеры использования подзапросов в различных сценариях работы с базами данных. Также будет показано, как они могут быть использованы для повышения читаемости и модульности запросов в различных сценариях работы с базами данных.

2.1. Синтаксис подзапросов

Вложенные запросы могут быть применены практически в любых компонентах внешнего запроса. Однако возможность их использования в определенных частях зависит от специфики конкретной системы управления базами данных. В связи с этим не существует общей схемы запроса, содержащего подзапросы.

Синтаксически подзапрос может быть представлен как SELECT-запрос, находящийся в теле любого другого запроса и обернутый в круглые скобки. Чаще всего его можно встретить в связке с оператором WHERE, как показано в листинге 1. При этом использование вложенных запросов в других частях внешнего запроса аналогично.

Листинг 1: Синтаксис подзапроса

```
SELECT column_name
    FROM table_name
    WHERE column_name expression operator
        (SELECT ...);
```

2.2. Типы подзапросов

Как уже говорилось ранее, подзапрос может располагаться практически в любых частях внешнего запроса. Можно выделить следующие места:

- После оператора IN (листинг 2).
- После оператора SOME (листинг 3).
- После оператора ALL (листинг 4).

- После оператора EXISTS (листинг 5).
- После предиката UNIQUE (листинг 6).
- Внутри блока SELECT (листинг 7).
- Внутри блока FROM (листинг 8).
- Внутри блока HAVING (листинг 9).
- Внутри обобщенного табличного выражения (листинг 10).

В зависимости от расположения вложенного запроса, контекста его использования, а также типа возвращаемого из него значения подзапросы могут быть классифицированы следующим образом:

CL1 Тип возвращаемого значения.

- SQ1 Скалярные подзапросы.
- SQ2 Табличные подзапросы.

CL2 Взаимосвязь с окружением.

- SQ3 Коррелированные подзапросы.
- SQ4 Некоррелированные подзапросы.

CL3 Взаимосвязь с блоком FROM.

- SQ5 Подзапросы, ссылающиеся на столбцы из предыдущих элементов блока FROM.

CL4 Контекст использования.

- SQ6 Подзапросы после операторов IN, SOME, ALL, EXISTS.
- SQ7 Подзапросы после предиката UNIQUE.
- SQ8 Подзапросы внутри блоков SELECT, FROM, HAVING.
- SQ9 Подзапросы внутри обобщенного табличного выражения.

Данная классификация (без типа SQ9) может помочь присвоить подзапросу строку, идентифицирующую его положение во внешнем запросе без учета вложенности. В общем виде она выглядит следующим образом: “CL1[SQ1|SQ2] – CL2[SQ3|SQ4] – CL3[SQ5|∅] – CL4[SQ6(m)|SQ7|SQ8(n)]”, где вместо m должно стоять название соотвутствующего оператора, а вместо n — название соответствующего блока. При этом CL_N могут быть упущены для краткости. К примеру, подзапросу из листинга 2 соответствует строка “SQ2-SQ4-SQ6(IN)”, поскольку он является табличным, некоррелированным, не ссылающимся на столбцы из предыдущих элементов блока FROM, а также располагается после оператора IN. Подзапросу из листинга 12, в свою очередь, соответствует строка “SQ2-SQ3-SQ5-SQ8(FROM)” из аналогичных соображений.

Каждая система управления базами данных имеет свой список поддерживаемых типов подзапросов. В таблицах 1, 2, 3, 4 приведено сравнение этих списков по предложенным классификациям у наиболее популярных СУБД, таких как MariaDB, PostgreSQL, SQLite, Oracle и Microsoft SQL Server.

По таблицам видно, что все рассматриваемые СУБД поддерживают подзапросы, имеющие типы SQ1, SQ2, SQ3, SQ4, SQ8 и SQ9. При этом SQLite не поддерживает операторы SOME и ALL, соответственно, в этой СУБД не получится использовать часть SQ6-подзапросов.

Таблица 1: Поддерживаемые типы подзапросов по классификации CL1

Тип подзапроса	MariaDB	PostgreSQL	SQLite	Oracle	Microsoft SQL Server
SQ1	+	+	+	+	+
SQ2	+	+	+	+	+

Таблица 2: Поддерживаемые типы подзапросов по классификации CL2

Тип подзапроса	MariaDB	PostgreSQL	SQLite	Oracle	Microsoft SQL Server
SQ3	+	+	+	+	+
SQ4	+	+	+	+	+

Нельзя не заметить, что ни одна из рассмотренных СУБД не поддерживает подзапрос типа SQ7. Дело в том, что предикат UNIQUE

имеет особенности, связанные с обработкой пустого множества, а также строк, содержащих null-значения. Зачастую вместо него рекомендуют использовать связку “1 >= (SELECT COUNT(*) ...)”, которая будет возвращать идентичный результат за исключением особых случаев, описанных ранее. Кроме того, Transact-SQL и вовсе не содержит этот предикат [4]. Однако некоторые СУБД все же поддерживают UNIQUE в данном контексте, поскольку он интересен в исследовательском плане. Примером такой системы может служить HyPer [7] — гибридная высокопроизводительная OLTP&OLAP система, разрабатываемая немецкими учеными. В своей статье [5] они показали, как предикат UNIQUE может быть представлен в виде соединений.

Подзапрос типа SQ5 не поддерживается такими СУБД, как MariaDB и SQLite, однако его поддержка есть у PostgreSQL и Oracle. Microsoft SQL Server тоже поддерживает этот тип подзапроса, но вместо классического оператора LATERAL, который входит в стандарт SQL, в нем используется собственный оператор APPLY. Он идентичен LATERAL, однако не является стандартизированным.

Таблица 3: Поддерживаемые типы подзапросов по классификации CL3

Тип подзапроса	MariaDB	PostgreSQL	SQLite	Oracle	Microsoft SQL Server
SQ5	–	+	–	+	*

Таблица 4: Поддерживаемые типы подзапросов по классификации CL4

Тип подзапроса	MariaDB	PostgreSQL	SQLite	Oracle	Microsoft SQL Server
SQ6	+	+	–	+	+
SQ7	–	–	–	–	–
SQ8	+	+	+	+	+
SQ9	+	+	+	+	+

2.3. Примеры подзапросов

В данном разделе будут приведены примеры запросов, охватывающие широкий спектр сценариев, включая агрегацию, фильтрацию и объединение. Это позволит продемонстрировать гибкость и функцио-

нальность подзапросов, а также то, как они могут быть использованы для решения различных задач.

Первым примером будет служить следующий запрос: хотим получить имена всех сотрудников, которые работают в отделах, расположенных в России. Для этого была использована фильтрация в связке с оператором IN. Код этого запроса можно увидеть в листинге 2.

Листинг 2: Подзапрос после оператора IN

```
SELECT name
    FROM employees
    WHERE department_id IN (
        SELECT department_id
            FROM departments
            WHERE location = 'Russia'
    );
```

Теперь рассмотрим следующую ситуацию: хотим получить названия всех акционных товаров, цена которых больше хотя бы одной из цен продуктов, относящихся к категории “meat products”. Данная информация может быть получена при помощи запроса из листинга 3, в котором используется подзапрос после оператора SOME.

Листинг 3: Подзапрос после оператора SOME

```
SELECT name
    FROM promotional_goods
    WHERE price > SOME (
        SELECT price
            FROM products
            WHERE category = 'meat␣products'
    );
```

Следующий пример демонстрирует, как подзапрос может быть использован после оператора ALL: находим сотрудников, зарплата которых выше среднего жалования во всех отделах. Его код может быть найден в листинге 4.

Листинг 4: Подзапрос после оператора ALL

```
SELECT *
    FROM employees
    WHERE salary > ALL (
        SELECT AVG(salary)
            FROM employees
            GROUP BY department_id
    );
```

В листинге 5 приведен запрос, в котором происходит поиск всех продуктов, которые содержались хотя бы в одном заказе, набравшим менее 4 звезд.

Листинг 5: Подзапрос после оператора EXISTS

```
SELECT name
    FROM products AS P
    WHERE EXISTS (
        SELECT 1
        FROM orders AS O
        WHERE O.product_id = P.product_id AND O.stars < 4
    );
```

Далее рассмотрим запрос из листинга 6. В нем происходит поиск всех товаров, которые были заказаны в 2024 году не более одного раза.

Листинг 6: Подзапрос после оператора UNIQUE

```
SELECT name
    FROM products AS P
    WHERE UNIQUE (
        SELECT order_id
            FROM orders AS O
            WHERE O.product_id = P.product_id AND O.year = 2024
    );
```

Следующий пример показывает, как подзапросы могут быть применены внутри блока SELECT. Его код может быть найден в листинге 7, а идея заключается в следующем: выводим информацию о сотруднике, его заработную плату, а также среднее жалование по отделу, в котором он работает.

Листинг 7: Подзапрос внутри блока SELECT

```
SELECT E.name, E.salary, (
    SELECT AVG(I.salary) AS avg_department_salary
        FROM employees AS I
        WHERE I.department_id = E.department_id
    )
    FROM employees as E;
```

Подзапросы внутри блока FROM помогают избавиться от наличия HAVING. Например, если мы хотим найти средние зарплаты сотрудников в тех отделах, где средняя зарплата ниже 50000, мы можем написать запрос, отраженный в листинге 8.

Листинг 8: Подзапрос внутри блока FROM

```
SELECT department_name, avg_salary
    FROM (
        SELECT department_name, AVG(salary) AS avg_salary
            FROM employees
            GROUP BY department_name
    )
    WHERE avg_salary < 50000;
```

Пусть теперь мы хотим получить список клиентов, которые сделали заказы на сумму больше средней суммы заказов по всем клиентам. Для этого мы может использовать подзапрос внутри блока HAVING, как показано в листинге 9.

Листинг 9: Подзапрос внутри блока HAVING

```
SELECT customer_id, SUM(price)
    FROM orders
    GROUP BY customer_id
    HAVING SUM(price) > (
        SELECT AVG(price) FROM orders
    );
```

Обобщенные табличные выражения (CTE) позволяют создавать временные наборы данных (таблицы), которые могут быть использованы в последующих запросах. Их главное преимущество заключается в том, что они упрощают написание и чтение сложных запросов. Пример за-

проса, использующего CTE, отражен в листинге 10. В нем происходит поиск отделов, имеющих максимальный показатель эффективности.

Листинг 10: Подзапрос в виде CTE

```
WITH max_efficiency(value) AS (
    SELECT MAX(efficiency) FROM departments
)
SELECT name, efficiency
    FROM departments, max_efficiency
    WHERE departments.efficiency = max_efficiency.value;
```

Следует отметить, что подзапрос может располагаться внутри другого подзапроса. Как правило, системы управления базами данных поддерживают до 32 уровней вложенности, хотя этот предел может варьироваться в зависимости от сложности других выражений в запросе и доступной оперативной памяти. Пример вложенного подзапроса изображен в листинге 11. В нем происходит извлечение средней заработной платы по всем профессиям. При этом налагается условие, что она должна быть меньше максимального значения средней минимальной заработной платы по всем профессиям, которыми владеют сотрудники из отделов с идентификаторами, меньшими 10.

Листинг 11: Вложенные подзапросы

```
SELECT AVG(salary)
    FROM employees
    GROUP BY job_id
    HAVING AVG(salary) < (
        SELECT MAX(AVG(min_salary))
            FROM jobs
            WHERE job_id IN (
                SELECT job_id
                    FROM job_history
                    WHERE department_id < 10
            )
            GROUP BY job_id
    );
```

Последним примером служит запрос из листинга 12. Он показывает, как подзапрос может ссылаться на столбцы из предыдущих элементов

блока FROM. Идея данного запроса в следующем: хотим получить имена сотрудников и среднюю зарплату на их кафедре.

Листинг 12: Подзапрос с LATERAL

```
SELECT name, avg_salary
    FROM employees AS E1, LATERAL (
        SELECT AVG(salary) AS avg_salary
            FROM employees AS E2
            WHERE E1.department_id = E2.department_id
    );
```

3. Реализация подзапросов в PosDB

В основе PosDB лежит итераторная модель Volcano [2]. При этом данные передаются поблочно, а для их чтения используются отдельные сущности — считыватели, которые позволяют решить проблему ромбовидных шаблонов в потоках данных [9]. Кроме того, PosDB представляет собой распределённую СУБД, обладающую механизмами поддержки параллельной обработки данных, что способствует достижению высокой эффективности ее работы.

В системе используются два типа операторов: кортежные и позиционные. Первый тип подразумевает передачу данных между операторами в виде кортежей, а второй в виде позиций. Внутри кортежей находятся конкретные значения атрибутов. Позиции же описывают не сами значения, а их местоположение в таблице, что позволяет добиться более эффективного использования ресурсов. В начале своей работы PosDB использует позиции для передачи данных, однако в определенный момент происходит переход на кортежи. Для достижения максимальной эффективности система стремится осуществить этот переход как можно позже. Данный процесс обозначается термином “ультра поздняя материализация” [3].

Таблица 5: Поддерживаемые типы подзапросов в PosDB

Тип подзапроса	Поддержка в PosDB
Классификация CL1	
SQ1	+
SQ2	+
Классификация CL2	
SQ3	+
SQ4	+
Классификация CL3	
SQ5	–
Классификация CL4	
SQ6	–
SQ7	+
SQ8	–
SQ9	+

Для реализации указанного подхода необходимо поддерживать два типа операторов на уровне плана запроса. Подзапросы не являются исключением. Их поддержка была обеспечена разработчиками путем внедрения новых операторов в PosDB, а также расширения существующего парсера запросов. В результате была создана система, обеспечивающая поддержку подзапросов в объеме, представленном в таблице 5.

Как можно заметить, на данный момент в PosDB отсутствует поддержка подзапросов, имеющих типы SQ5, SQ6[1] и SQ8. В связи с этим следующие работы будут посвящены расширению данной функциональности.

[1]Отсутствует поддержка подзапросов после оператора IN

Заключение

Был проведен обзор подзапросов в реляционных системах управления базами данных (СУБД), а также анализ их применения. Как результат, были выполнены следующие задачи:

1. Изучены теоретические основы подзапросов, их типы и синтаксис в различных реляционных СУБД.

2. Проведен анализ существующих систем на предмет реализации в них разных типов подзапросов.

3. Рассмотрены практические примеры использования подзапросов в различных сценариях работы с базами данных.

4. Разработаны и систематизированы несколько классификаций подзапросов по таким критериям, как взаимосвязь с окружением, тип возвращаемого значения и контекст использования.

Список литературы

[1] Abadi Daniel J., Boncz Peter A., Harizopoulos Stavros. Column-oriented database systems // Proc. VLDB Endow. — 2009. — Aug.. — Vol. 2, no. 2. — P. 1664–1665. — URL: https://doi.org/10.14778/1687553.1687625.

[2] Graefe G. Volcano — An Extensible and Parallel Query Evaluation System // IEEE Trans. on Knowl. and Data Eng. — 1994. — Feb.. — Vol. 6, no. 1. — P. 120–135. — URL: https://doi.org/10.1109/69.273032.

[3] Hybrid Materialization in a Disk-Based Column-Store / Evgeniy Klyuchikov, Michael Polyntsov, Anton Chizhov et al. // Proceedings of the 7th Joint International Conference on Data Science & Management of Data (11th ACM IKDD CODS and 29th COMAD). — CODS-COMAD '24. — New York, NY, USA : Association for Computing Machinery, 2024. — P. 164–172. — URL: https://doi.org/10.1145/3632410.3632422.

[4] Microsoft. Search condition (Transact-SQL). — 2024. — URL: https://learn.microsoft.com/en-us/sql/t-sql/queries/search-condition-transact-sql?view=sql-server-ver16&redirectedfrom=MSDN (дата обращения: 18.12.2024).

[5] Neumann Thomas, Leis Viktor, Kemper Alfons. The Complete Story of Joins (in HyPer) // Datenbanksysteme für Business, Technologie und Web (BTW 2017), 17. Fachtagung des GI-Fachbereichs „Datenbanken und Informationssysteme" (DBIS), 6.-10. März 2017, Stuttgart, Germany, Proceedings / Ed. by Bernhard Mitschang, Daniela Nicklas, Frank Leymann et al. — Vol. P-265 of LNI. — GI, 2017. — P. 31–50. — URL: https://dl.gi.de/handle/20.500.12116/657.

[6] Nordeen A. Learn DBMS in 24 Hours. — Guru99, 2022. — URL: https://books.google.ru/books?id=aPh7EAAAQBAJ.

[7] One DBMS for all: the brawny few and the wimpy crowd / Tobias Mühlbauer, Wolf Rödiger, Robert Seilbeck et al. // Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data. — SIGMOD '14. — New York, NY, USA : Association for Computing Machinery, 2014. — P. 697–700. — URL: https://doi.org/10.1145/2588555.2594527.

[8] PosDB: A Distributed Column-Store Engine / George A. Chernishev, Viacheslav Galaktionov, Valentin D. Grigorev et al. // Perspectives of System Informatics - 11th International Andrei P. Ershov Informatics Conference, PSI 2017, Moscow, Russia, June 27-29, 2017, Revised Selected Papers / Ed. by Alexander K. Petrenko, Andrei Voronkov. — Vol. 10742 of Lecture Notes in Computer Science. — Springer, 2017. — P. 88–94. — URL: https://doi.org/10.1007/978-3-319-74313-4_7.

[9] PosDB: An Architecture Overview / George A. Chernishev, Vyacheslav Galaktionov, Valentin D. Grigorev et al. // Program. Comput. Softw. — 2018. — Vol. 44, no. 1. — P. 62–74. — URL: https://doi.org/10.1134/S0361768818010024.

[10] Silberschatz Abraham, Korth Henry, Sudarshan S. Database Systems Concepts. — 5 edition. — USA : McGraw-Hill, Inc., 2005. — ISBN: 0072958863.

[11] zHyperLink: Low-latency I/O for Db2 on IBM Z and DS8880 storage / D. Craddock, A. Hoshikawa, J. Josten et al. // IBM Journal of Research and Development. — 2018. — Vol. 62, no. 2/3. — P. 13:1–13:10.