Решение всех вопросов теста 1С:Профессионал по технологическим вопросам (Раздел №6)

Опубликовано 19.12.2018 автором Виталий Онянов

Ниже приводится решение всех вопросов для подготовки к аттестации 1С:Профессионал по технологическим вопросам. Текстов самих вопросов и вариантов ответов нет. Предполагается, что у вас имеется книга «Комплект вопросов сертификационного экзамена «1С:Профессионал» по технологическим вопросам с примерами решений». Я ни в комем случае не призываю заучивать ответы, а рекомендую прорешивать и анализировать каждый вопрос, ведь сдача данного экзамена, это лишь первый шаг к сертификации 1С:Эксперт по технологическим вопросам.

Все решения авторские, потому любые замечания, предложения и критика только приветствуется. Все ответы проверены на сайте учебного тестирования.

В данной стате представлены решения раздела №6

«Запросы и методы их оптимизации».

06.01 — 5

План запроса, формируемый SQL Server можно получить как с помощью SQL Server Profiler, так с помощью технологического журнала и ЦУП. А вот центр контроля качества, который входит в ЦУП не предоставляет такой возможности.

Источники:

- https://its.1c.ru/db/v8doc/content/26/1/
- https://its.1c.ru/db/kip#content:3
- https://its.1c.ru/db/kip#content:49

Смотрите также:

- http://tavalik.ru/poluchenie-trassirovki-v-sql-server-profiler/

06.02 — 2

Если элемент **plansql** присутствует, то будет включен сбор планов запросов, которые генерируют СУБД при выполнении запросов «1С:Предприятия». Сами планы запросов расположены в свойстве **planSQLText** событий, связанных с исполнением запросов конкретной СУБД (см. здесь).

Источник:

- https://its.1c.ru/db/v8doc/content/26/1/issogl3_3.14.1.7

06.03 — 1

Чтобы получить текст запроса так, как его получает SQL Server, и увидеть план запроса, нужно сделать следующее:

1. Запустить SQL Server Profiler
2. Создать трассировку. Можно использовать стандартный шаблон.
3. В свойствах трассировки:
   3.1. Убедиться, что в нее входят события SQL:BatchStarted, SQL:BatchCompleted, RPC:Completed.
   3.2. Установить галочку «все события».
   3.3. Добавить события **Showplan Statistics Profile** и **Showplan XML StatisticsProfile** из узла **Performance**.

Поиск

Рубрики:

- 1С (81)
  - 1С 7 (7)
  - 1С 8 (66)
    - Лицензирование (5)
    - Работа в 1С (4)
    - Разработка в 1С (10)
    - Системные требования (3)
    - Хранилище конфигурации (3)
    - Эксперт 1С (25)
  - OneScript (5)
- Microsoft Windows (52)
  - Windows 10 (3)
  - Windows 7 (8)
  - Windows 8 (14)
  - Windows Server 2008 R2 (22)
  - Windows Server 2012 R2 (13)
- SQL (34)
  - Microsoft SQL Server 2008 (12)
  - Microsoft SQL Server 2012 (22)
  - Microsoft SQL Server 2014 (1)
  - Oracle MySQL (1)
- Без рубрики (1)
- Видео (17)
- Виртуализация (22)
  - ESXi (6)
  - Hyper-V (6)
    - Hyper-V в Windows 8 (2)
    - Hyper-V в Windows Server 2008 R2 (2)
  - VirtualBox (6)
  - VMware Workstation (6)
- Психология (3)
- Разное (39)
  - cmd (5)
  - Exchange 2010 (3)
  - Железо (8)
  - Избавляемся от рекламы (4)
  - Конференции (3)
  - Юмор (9)
- Сайт своими руками (20)
  - Drupal (1)
  - WordPress (8)
  - Копипаст не пройдет (2)
  - Первые шаги (4)
  - Хостинг на своем компьютере (7)
- Софт сисадмину (37)
  - «Облачные» приложения (2)
  - Антивирусная защита (2)
  - Жесткий диск (4)
  - Работа с драйверами (2)
  - Резервное копирование (4)
  - Удаленное управление (4)

3.4. Снять галочку «все события», останутся только выбранные.
3.5. Установить галочку «все столбцы».
3.6. Добавить столбец «DatabaseName».
3.7. Зайти в «фильтры столбцов», поставить фильтр «DatabaseName» «Похоже на (Like)» <написать_имя_своей_базы>
3.8. Убедиться, что у события Showplan Statistics Profile включен столбецBinaryData.
3.9. Снять галочку «все столбцы», останутся только выбранные.
3.10. …

Источник:

- Е. В. Филиппов «Настольная книга 1С:Эксперта по технологическим вопросам», стр. 81

…

Класс событий **Showplan Statistics Profile** происходит, когда Microsoft SQL Server выполняет инструкцию SQL. Включаемые в эти события сведения представляют собой часть данных, доступных в классе событий Showplan XML Statistics Profile.

Класс событий Showplan Statistics Profile выводит полные данные времени компиляции. Трассировки, содержащие профиль статистики инструкции Showplan, могут значительно снизить производительность. Чтобы свести их к минимуму, сделайте использование этого класса событий доступным только для трассировок, наблюдающих за отдельными проблемами в течение короткого промежутка времени.

Источник:

- https://docs.microsoft.com/ru-ru/sql/relational-databases/event-classes/showplan-statistics-profile-event-class

Класс событий **Showplan XML Statistics Profile** возникает, когда Microsoft SQL Server выполняет инструкцию SQL. Включите класс событий Showplan XML Statistics Profile для идентификации операторов Showplan в Microsoft SQL Server.

Класс событий Showplan XML Statistics Profile отображает полные данные этапа компиляции, поэтому трассировки, которые содержат этот класс событий, могут существенно увеличивать рабочую нагрузку. Чтобы минимизировать привносимые издержки, используйте этот класс событий только в трассировках, наблюдающих за определенными проблемами в течение непродолжительного периода времени.

Источник:

- https://docs.microsoft.com/ru-ru/sql/relational-databases/event-classes/showplan-xml-statistics-profile-event-class

06.04 — 3

Оператор **Index Scan** получает все записи некластерного индекса, указанного в столбце Argument. Index Scan является логическим и физическим оператором.

Источник:

- Е. В. Филиппов «Настольная книга 1С:Эксперта по технологическим вопросам», стр. 88

Шпаргалка:

- **Index Scan** — Логический и физический оператор, который извлекает **все строки** из **некластеризованного** индекса, указанного в колонке Argument.
- **Clustered Index Scan** — Логический и физический оператор, который **сканирует кластеризованный** индекс, определенный в колонке Argument.
- **Index Seek** — Логический и физический оператор, который использует возможность **поиска** индексов с целью извлечения строк из **некластеризованного** индекса.

- **Clustered Index Seek** — Логический и физическая оператор, использующий **поисковую способность** индексов извлекать хранимые строки из кластеризованного индекса.

06.05 — 1

Чтобы иметь возможность выполнять запросы, Компонент SQL Server Database Engine должен анализировать инструкцию и определять наиболее эффективный способ доступа к необходимым данным. Этот анализ обрабатывается компонентом, который называется оптимизатором запросов. Входные данные оптимизатора запросов включают сам запрос, схему базы данных (определения таблиц и индексов) и статистику базы данных. Выходные данные оптимизатора запросов — это **план выполнения запроса**, который иногда называется планом запроса или просто планом выполнения.

План выполнения запроса представляет собой определение следующего:

- Последовательности, в которой происходит обращение к исходным таблицам…
- Методы, используемые для извлечения данных из каждой таблицы…

Источник:

- https://docs.microsoft.com/ru-ru/sql/relational-databases/performance/execution-plans

06.06 — 1

Оператор **Index Seek** использует возможности поиска по индексам для получения строк из некластерного индекса, указанного в столбце Argument. Index Seek является логическим и физическим оператором.

Источник:

- Е. В. Филиппов «Настольная книга 1С:Эксперта по технологическим вопросам», стр. 89

Шпаргалка:

- **Index Scan** — Логический и физический оператор, который извлекает **все строки** из **некластеризованного** индекса, указанного в колонке Argument.
- **Clustered Index Scan** — Логический и физический оператор, который **сканирует кластеризованный** индекс, определенный в колонке Argument.
- **Index Seek** — Логический и физический оператор, который использует возможность **поиска** индексов с целью извлечения строк из **некластеризованного** индекса.
- **Clustered Index Seek** — Логический и физическая оператор, использующий **поисковую способность** индексов извлекать хранимые строки из кластеризованного индекса.

06.07 — 2

Оператор **Clustered Index Scan** сканирует кластерный индекс, указанный в столбце Argument. Clustered Index Scan является логическим и физическим оператором.

Источник:

- Е. В. Филиппов «Настольная книга 1С:Эксперта по технологическим вопросам», стр. 88

Шпаргалка:

- **Index Scan** — Логический и физический оператор, который извлекает **все строки** из **некластеризованного** индекса, указанного в колонке Argument.
- **Clustered Index Scan** — Логический и физический оператор, который **сканирует кластеризованный** индекс, определенный в колонке Argument.
- **Index Seek** — Логический и физический оператор, который использует возможность **поиска** индексов с целью извлечения строк из **некластеризованного** индекса.

- **Clustered Index Seek** — Логический и физическая оператор, использующий **поисковую способность** индексов извлекать хранимые строки из кластеризованного индекса.

06.08 — 4

Оператор **Clustered Index Seek** использует поисковые возможности индексов для получения строк из кластерного индекса, указанного в столбце Argument. Clustered Index Seek-это логический и физический оператор.

Источник:

- Е. В. Филиппов «Настольная книга 1С:Эксперта по технологическим вопросам», стр. 89

Шпаргалка:

- **Index Scan** — Логический и физический оператор, который извлекает **все строки** из **некластеризованного** индекса, указанного в колонке Argument.
- **Clustered Index Scan** — Логический и физический оператор, который **сканирует кластеризованный** индекс, определенный в колонке Argument.
- **Index Seek** — Логический и физический оператор, который использует возможность **поиска** индексов с целью извлечения строк из **некластеризованного** индекса.
- **Clustered Index Seek** — Логический и физическая оператор, использующий **поисковую способность** индексов извлекать хранимые строки из кластеризованного индекса.

06.09 — 4

Оператор **Table Scan** получает строки из таблицы, указанной в столбце Argument плана выполнения запроса. Table Scan является логическим и физическим оператором.

Источник:

- Е. В. Филиппов «Настольная книга 1С:Эксперта по технологическим вопросам», стр. 88

06.10 — 5

Сканирование — это получение всех записей. Разница состоит только в том, что именно проходится: кластерный индекс, некластерный индекс или сама таблица.

Источник:

- Е. В. Филиппов «Настольная книга 1С:Эксперта по технологическим вопросам», стр. 88

Clustered Index Scan сканирует кластерный индекс, а Table Scan возвращает все строки из таблицы, указанной в колонке Argument.

Clustered Index Scan используется для сканирования сортированных таблиц целиком, а Table Scan — несортированных.

06.11 — 4

Сканирование — это получение всех записей. Разница состоит только в том, что именно проходится: кластерный индекс, некластерный индекс или сама таблица.

Источник:

- Е. В. Филиппов «Настольная книга 1С:Эксперта по технологическим вопросам», стр. 88

Clustered Index Scan — операция, выполняемая над кластерным индексом, а Index Scan — операция, выполняющая сканирование некластерного индекса

06.12 — 2

Последовательность элементов **event** определяет условие, при выполнении которого событие будет помещено в журнал. В журнал помещаются только такие события, которые удовлетворяют условию. Иначе говоря, если условие, определяемое последовательностью элементов **event**, принимает значение Истина, то событие будет записано в журнал.

Доступны следующие имена групп событий:

- **DBMSSQL** — Исполнение операторов SQL СУБД Microsoft SQL Server.

Каждое событие имеет набор свойств. Каждое свойство имеет имя. Возможно присутствие в событии нескольких свойств с одинаковыми именами. Имена свойств могут использоваться для фильтрации событий и свойств. Большие и малые буквы при сравнении имен не различаются. Пустое условие в элементе property будет означать, что свойство будет выводиться при любом условии.

- **Duration** – длительность события в десятитысячных долях секунды.

Источник:

- https://its.1c.ru/db/v8doc/content/26/1/issogl3_3.14.1.2

…

Событие **DBMSSQL** позволит собрать все SQL запросы технологической платформы 1С к СУБД MS SQL Server с фильтром по одной базе db.

Источник:

- https://its.1c.ru/db/metod8dev#content:5808

06.13 — 2

Документального подтверждения найти не удалось, однако правильный ответ на сайте учебного тестирования говорит о том, что событие **НЕ БУДЕТ** записано в технологический журнал в момент своего начала (например, при начале выполнения запроса), что и подтверждается практически.

Кроме того, известно, что запись попадает в ТЖ в момент завершения события, при этом указывается время окончания и длительность (сам формат не предполагает записи в момент возникновения).

06.14 — 2

Предложение **ДЛЯ ИЗМЕНЕНИЯ** позволяет заблаговременно заблокировать некоторые данные (которые могут читаться транзакцией другого соединения) уже при считывании, чтобы исключить взаимные блокировки при записи. **ДЛЯ ИЗМЕНЕНИЯ** дает возможность указать в запросе таблицы, считываемые данные которых предполагается изменять. В этом случае другое соединение будет ожидать освобождения этих данных уже в момент считывания внутри транзакции, т.е. не сможет прочесть заблокированные данные до тех пор, пока не будет завершена транзакция, наложившая блокировку. Блокировка от изменения данных считываемых в транзакции выполняется независимо от предложения **ДЛЯ ИЗМЕНЕНИЯ**. Это значит, что если внутри какой-либо транзакции считаны некоторые данные, то из другого соединения эти данные не могут быть изменены до тех пор, пока блокировка не будет снята. Если запрос выполняется вне транзакции, то в нем могут быть считаны и заблокированные данные.

Блокировки устанавливаются в момент выполнения запроса, сбрасываются же при окончании транзакции. В случае если запрос выполняется вне транзакции предложение **ДЛЯ ИЗМЕНЕНИЯ** игнорируется.

Источник:

- https://its.1c.ru/db/metod8dev/content/2651/hdoc

06.15 — 5

Предложение **ДЛЯ ИЗМЕНЕНИЯ** позволяет заблаговременно заблокировать некоторые данные (которые могут читаться транзакцией другого соединения) уже при считывании, чтобы исключить взаимные блокировки при записи. ДЛЯ ИЗМЕНЕНИЯ дает возможность указать в запросе таблицы, считываемые данные которых предполагается изменять. В этом случае другое соединение будет ожидать освобождения этих данных уже в момент считывания внутри транзакции, т.е. не сможет прочесть заблокированные данные до тех пор, пока не будет завершена транзакция, наложившая блокировку.

Источник:

- https://its.1c.ru/db/metod8dev/content/2651/hdoc

Таким образом, конструкция ДЛЯ ИЗМЕНЕНИЯ используется для защиты от взаимоблокировки, которая возникает при повышении уровня блокировки в транзакциях с уровнем изоляции:

- **REPEATABLE READ** — обеспечивает повторяемость чтения данных. Если моя транзакция начинает *читать* данные, то другая транзакция не может их *изменить* до окончания моей транзакции.
- **SERIALIZABLE** — последовательное выполнение. Этот уровень изоляции является максимальным и обеспечивает полную изоляцию транзакций друг от друга. Решаются все рассмотренные проблемы, включая проблему «фантомов».

Источник:

- https://its.1c.ru/db/metod8dev#content:5839

06.16 — 4

Data Definition Language (**DDL**) (язык описания данных) — это семейство компьютерных языков, используемых в компьютерных программах для описания структуры баз данных.

Источник:

- https://ru.wikipedia.org/wiki/Data_Definition_Language

Шпаргалка:

- **DDL** — Язык определения данных, предназначенный для создания, удаления и модификации таблиц базы данных.
- **DCL** — Язык управления данными, предназначенный для обеспечения защиты базы данных.
- **DML** — Семейство компьютерных языков, используемых в компьютерных программах или пользователями баз данных для получения, вставки, удаления или изменения данных в базах данных.

06.17 — 2

Data Control Language (**DCL**) — подмножество языка управления базами данных SQL, предназначенное для осуществления административных операций, присваивающих или отменяющих право (привилегию) использовать базу данных, таблицы и другие объекты базы данных, а также выполнять те или иные операторы SQL.

Источник:

- https://ru.wikipedia.org/wiki/Data_Control_Language

Шпаргалка:

- **DDL** — Язык определения данных, предназначенный для создания, удаления и модификации таблиц базы данных.
- **DCL** — Язык управления данными, предназначенный для обеспечения защиты базы данных.
- **DML** — Семейство компьютерных языков, используемых в компьютерных программах или пользователями баз данных для получения, вставки, удаления или изменения данных в базах данных.

06.18 — 1

Data Manipulation Language (**DML**) (язык управления (манипулирования) данными) — это семейство компьютерных языков, используемых в компьютерных программах или пользователями баз данных для получения, вставки, удаления или изменения данных в базах

данных.

Источник:

- https://ru.wikipedia.org/wiki/Data_Manipulation_Language

Шпаргалка:

- **DDL** — Язык определения данных, предназначенный для создания, удаления и модификации таблиц базы данных.
- **DCL** — Язык управления данными, предназначенный для обеспечения защиты базы данных.
- **DML** — Семейство компьютерных языков, используемых в компьютерных программах или пользователями баз данных для получения, вставки, удаления или изменения данных в базах данных.

06.19 — 4

Чтобы иметь возможность выполнять запросы, Компонент SQL Server Database Engine должен анализировать инструкцию и определять наиболее эффективный способ доступа к необходимым данным. Этот анализ обрабатывается компонентом, который называется оптимизатором запросов. Входные данные оптимизатора запросов включают сам запрос, схему базы данных (определения таблиц и индексов) и статистику базы данных. Выходные данные оптимизатора запросов — это **план выполнения запроса**, который иногда называется планом запроса или просто планом выполнения.

План выполнения запроса представляет собой определение следующего:

- Последовательности, в которой происходит обращение к исходным таблицам…
- Методы, используемые для извлечения данных из каждой таблицы…

Источник:

- https://docs.microsoft.com/ru-ru/sql/relational-databases/performance/execution-plans

06.20 — 3

Чтобы получить текст запроса так, как его получает SQL Server, и увидеть план запроса, нужно сделать следующее:

1. Запустить SQL Server Profiler
2. Создать трассировку. Можно использовать стандартный шаблон.
3. В свойствах трассировки:
   3.1. Убедиться, что в нее входят события **SQL:BatchStarted**, **SQL:BatchCompleted**, **RPC:Completed**.
   3.2. Установить галочку «все события».
   3.3. Добавить события **Showplan Statistics Profile** и **Showplan XML StatisticsProfile** из узла **Performance**.
   3.4. Снять галочку «все события», останутся только выбранные.
   3.5. Установить галочку «все столбцы».
   3.6. Добавить столбец «DatabaseName».
   3.7. Зайти в «фильтры столбцов», поставить фильтр «DatabaseName» «Похоже на (Like)» <написать_имя_своей_базы>
   3.8. Убедиться, что у события Showplan Statistics Profile включен столбецBinaryData.
   3.9. Снять галочку «все столбцы», останутся только выбранные.
   3.10. …

Источник:

- Е. В. Филиппов «Настольная книга 1С:Эксперта по технологическим вопросам», стр. 81

…

Класс событий **SQL:BatchCompleted** указывает на завершение выполнения пакета языка Transact-SQL

Источник:

- https://docs.microsoft.com/ru-ru/sql/relational-databases/event-classes/sql-batchcompleted-event-class

Класс событий **RPC:Completed** указывает, что удаленный вызов процедуры завершен.

Источник:

- https://docs.microsoft.com/ru-ru/sql/relational-databases/event-classes/rpc-completed-event-class

Класс событий **Showplan Statistics Profile** происходит, когда Microsoft SQL Server выполняет инструкцию SQL. Включаемые в эти события сведения представляют собой часть данных, доступных в классе событий Showplan XML Statistics Profile.

Класс событий Showplan Statistics Profile выводит полные данные времени компиляции. Трассировки, содержащие профиль статистики инструкции Showplan, могут значительно снизить производительность. Чтобы свести их к минимуму, сделайте использование этого класса событий доступным только для трассировок, наблюдающих за отдельными проблемами в течение короткого промежутка времени.

Источник:

- https://docs.microsoft.com/ru-ru/sql/relational-databases/event-classes/showplan-statistics-profile-event-class

Класс событий **Showplan XML Statistics Profile** возникает, когда Microsoft SQL Server выполняет инструкцию SQL. Включите класс событий Showplan XML Statistics Profile для идентификации операторов Showplan в Microsoft SQL Server.

Класс событий Showplan XML Statistics Profile отображает полные данные этапа компиляции, поэтому трассировки, которые содержат этот класс событий, могут существенно увеличивать рабочую нагрузку. Чтобы минимизировать привносимые издержки, используйте этот класс событий только в трассировках, наблюдающих за определенными проблемами в течение непродолжительного периода времени.

Источник:

- https://docs.microsoft.com/ru-ru/sql/relational-databases/event-classes/showplan-xml-statistics-profile-event-class

06.21 — 2

События класса Showplan XML происходят, когда Microsoft SQL Server выполняет инструкцию SQL…
Showplan XML хранит план запроса, который создается при оптимизации запроса.

Источник:

- https://docs.microsoft.com/ru-ru/sql/relational-databases/event-classes/showplan-xml-event-class
- https://docs.microsoft.com/ru-ru/sql/relational-databases/event-classes/showplan-xml-statistics-profile-event-class

06.22 — 2

Showplan XML for Query Compile — это событие аналогично событию **SHOWPLAN XML** за исключением того, что выходные данные Showplan создаются только при компиляции (или перекомпиляции) запроса. Для последующих выполнений, если план запроса извлекается из

кэша планов, информация showplan не отображается. Это событие менее затратно с точки зрения производительности и лучше всего подходит для быстрого просмотра созданного плана запроса.

Источник:

- https://blogs.msdn.microsoft.com/sqlqueryprocessing/2006/10/17/showplan-trace-events/

06.23 — 1

Оператор **Nested Loops** в упрощённом виде сравнивает каждую строку одной таблицы (называемой внешней таблицей) с каждой строке другой таблицы (называемой внутренней таблицей), ища те строки, которые удовлетворяют предикату соединения.

Получившиеся вложенные циклы и представляют тот самый алгоритм, который дал название соединению вложенных циклов.
Все строки должны подвергнуться сравнению, и, таким образом, стоимость этого алгоритма пропорциональна размеру внешней таблицы, умноженному на размер внутренней таблицы. Поскольку при увеличении размеров соединяемых таблиц стоимость растет очень сильно, на практике её уменьшают путём сокращения выборки из внутренней таблицы, строки которой обрабатываются для каждой строки внешней таблицы.

Источники:

- http://www.sql.ru/articles/mssql/2007/051101nestedloopsjoin.shtml
- https://blogs.msdn.microsoft.com/craigfr/2006/07/26/nested-loops-join/

Merge Join в отличие от соединения вложенных циклов, которое поддерживает любые предикаты соединения, требует существования не менее одного предиката соединения по эквивалентности. Кроме того, получаемые соединением слиянием данные должны быть отсортированы по ключу соединения. Например, если мы имеем предикат соединения «T1.a = T2.b», таблица T1 должна быть отсортирована по T1.a, а таблица T2 должна быть сортирована по T2.b.

Соединение слиянием одновременно считывает и сравнивает два отсортированных входных потока, по одной строке за шаг. На каждом из этих шагов происходит сравнение со следующей строкой входного потока. Если строки равны, выводится присоединяемая строка, и процесс продолжается дальше. Если строки не равны, исключается меньшее из двух входных значений, и процесс продолжается. Так как входные потоки отсортированы, легко видно, что исключаемая строка будет меньше любой из оставшихся строк в любом из входных потоков и, таким образом, не должна участвовать в соединении.

Источники:

- http://www.sql.ru/articles/mssql/2007/051102mergejoin.shtml
- https://blogs.msdn.microsoft.com/craigfr/2006/08/03/merge-join/

Hash Join имеет много общего с соединением слиянием. Подобно соединению слиянием, для него требуются не менее одного предиката объединения по эквивалентности, оно поддерживает остаточные предикаты, а также все внешние соединения и полусоединения. В отличие от соединения слиянием, для него не требуется наличие упорядоченных входных потоков и для поддержки полного внешнего соединения требуется наличие предиката соединения по эквивалентности.

Hash Join выполняется в две стадии: компоновка и проба. Во время компоновки осуществляется чтение всех строк первого входного потока (часто, его называют левым потоком или потоком компоновки), хеширование строк по ключам соединения эквивалентности и создание в оперативной памяти хеш-таблицы. Во время пробы осуществляется чтение всех строк второго входного потока (часто его называют правым потоком или пробным потоком), хеширование строк этого потока по тем же ключам соединения эквивалентности, а потом

осуществляется просмотр или поиск соответствий строк в хеш-таблице. Так как хеш-функции могут быть подвержены коллизиям (два разных значения ключа имеют одинаковый хэш), приходится проверять каждое потенциальное соответствие, что необходимо для гарантии того, что в этом случае действительно совпадение обусловлено соединением, а не коллизией.

Источники:

- http://www.sql.ru/articles/mssql/2007/051103hashjoin.shtml
- https://blogs.msdn.microsoft.com/craigfr/2006/08/10/hash-join/

Смотрите также:

- Е. В. Филиппов «Настольная книга 1С:Эксперта по технологическим вопросам», стр. 91

06.24 — 1

Hash Join имеет много общего с соединением слиянием. Подобно соединению слиянием, для него требуются не менее одного предиката объединения по эквивалентности, оно поддерживает остаточные предикаты, а также все внешние соединения и полусоединения. В отличие от соединения слиянием, для него не требуется наличие упорядоченных входных потоков и для поддержки полного внешнего соединения требуется наличие предиката соединения по эквивалентности.

Hash Join выполняется в две стадии: компоновка и проба. Во время компоновки осуществляется чтение всех строк первого входного потока (часто, его называют левым потоком или потоком компоновки), хеширование строк по ключам соединения эквивалентности и создание в оперативной памяти хеш-таблицы. Во время пробы осуществляется чтение всех строк второго входного потока (часто его называют правым потоком или пробным потоком), хеширование строк этого потока по тем же ключам соединения эквивалентности, а потом осуществляется просмотр или поиск соответствий строк в хеш-таблице. Так как хеш-функции могут быть подвержены коллизиям (два разных значения ключа имеют одинаковый хэш), приходится проверять каждое потенциальное соответствие, что необходимо для гарантии того, что в этом случае действительно совпадение обусловлено соединением, а не коллизией.

Источники:

- http://www.sql.ru/articles/mssql/2007/051103hashjoin.shtml
- https://blogs.msdn.microsoft.com/craigfr/2006/08/10/hash-join/

06.25 — 3

Merge Join в отличие от соединения вложенных циклов, которое поддерживает любые предикаты соединения, требует существования не менее одного предиката соединения по эквивалентности. Кроме того, получаемые соединением слиянием данные должны быть отсортированы по ключу соединения. Например, если мы имеем предикат соединения «T1.a = T2.b», таблица T1 должна быть отсортирована по T1.a, а таблица T2 должна быть сортирована по T2.b.

Соединение слиянием одновременно считывает и сравнивает два отсортированных входных потока, по одной строке за шаг. На каждом из этих шагов происходит сравнение со следующей строкой входного потока. Если строки равны, выводится присоединяемая строка, и процесс продолжается дальше. Если строки не равны, исключается меньшее из двух входных значений, и процесс продолжается. Так как входные потоки отсортированы, легко видно, что исключаемая строка будет меньше любой из оставшихся строк в любом из входных потоков и, таким образом, не должна участвовать в соединении.

Источники:

- http://www.sql.ru/articles/mssql/2007/051102mergejoin.shtml
- https://blogs.msdn.microsoft.com/craigfr/2006/08/03/merge-join/

Смотрите также:

- Е. В. Филиппов «Настольная книга 1С:Эксперта по технологическим вопросам», стр. 91

06.26 — 1

Оператор **Nested Loops** в упрощённом виде сравнивает каждую строку одной таблицы (называемой внешней таблицей) с каждой строке другой таблицы (называемой внутренней таблицей), ища те строки, которые удовлетворяют предикату соединения.

Получившиеся вложенные циклы и представляют тот самый алгоритм, который дал название соединению вложенных циклов.
Все строки должны подвергнуться сравнению, и, таким образом, стоимость этого алгоритма

пропорциональна размеру внешней таблицы, умноженному на размер внутренней таблицы. Поскольку при увеличении размеров соединяемых таблиц стоимость растет очень сильно, на практике её уменьшают путём сокращения выборки из внутренней таблицы, строки которой обрабатываются для каждой строки внешней таблицы.

Источники:

- http://www.sql.ru/articles/mssql/2007/051101nestedloopsjoin.shtml
- https://blogs.msdn.microsoft.com/craigfr/2006/07/26/nested-loops-join/

Смотрите также:

- Е. В. Филиппов «Настольная книга 1С:Эксперта по технологическим вопросам», стр. 91

06.27 — 1

Оператор **Nested Loops** в упрощённом виде сравнивает каждую строку одной таблицы (называемой внешней таблицей) с каждой строке другой таблицы (называемой внутренней таблицей), ища те строки, которые удовлетворяют предикату соединения.

Получившиеся вложенные циклы и представляют тот самый алгоритм, который дал название соединению вложенных циклов.
Все строки должны подвергнуться сравнению, и, таким образом, стоимость этого алгоритма пропорциональна размеру внешней таблицы, умноженному на размер внутренней таблицы. Поскольку при увеличении размеров соединяемых таблиц стоимость растет очень сильно, на практике её уменьшают путём сокращения выборки из внутренней таблицы, строки которой обрабатываются для каждой строки внешней таблицы.

Источники:

- http://www.sql.ru/articles/mssql/2007/051101nestedloopsjoin.shtml
- https://blogs.msdn.microsoft.com/craigfr/2006/07/26/nested-loops-join/

Merge Join в отличие от соединения вложенных циклов, которое поддерживает любые предикаты соединения, требует существования не менее одного предиката соединения по эквивалентности. Кроме того, получаемые соединением слиянием данные должны быть отсортированы по ключу соединения. Например, если мы имеем предикат соединения «T1.a = T2.b», таблица T1 должна быть отсортирована по T1.a, а таблица T2 должна быть сортирована по T2.b.

Соединение слиянием одновременно считывает и сравнивает два отсортированных входных потока, по одной строке за шаг. На каждом из этих шагов происходит сравнение со следующей строкой входного потока. Если строки равны, выводится присоединяемая строка, и процесс продолжается дальше. Если строки не равны, исключается меньшее из двух входных значений, и процесс продолжается. Так как входные потоки отсортированы, легко видно, что исключаемая строка будет меньше любой из оставшихся строк в любом из входных потоков и, таким образом, не должна участвовать в соединении.

Источники:

- http://www.sql.ru/articles/mssql/2007/051102mergejoin.shtml
- https://blogs.msdn.microsoft.com/craigfr/2006/08/03/merge-join/

Hash Join имеет много общего с соединением слиянием. Подобно соединению слиянием, для него требуются не менее одного предиката объединения по эквивалентности, оно поддерживает остаточные предикаты, а также все внешние соединения и полусоединения. В отличие от соединения слиянием, для него не требуется наличие упорядоченных входных потоков и для поддержки полного внешнего соединения требуется наличие предиката соединения по эквивалентности.

Hash Join выполняется в две стадии: компоновка и проба. Во время компоновки

осуществляется чтение всех строк первого входного потока (часто, его называют левым потоком или потоком компоновки), хеширование строк по ключам соединения эквивалентности и создание в оперативной памяти хеш-таблицы. Во время пробы осуществляется чтение всех строк второго входного потока (часто его называют правым потоком или пробным потоком), хеширование строк этого потока по тем же ключам соединения эквивалентности, а потом осуществляется просмотр или поиск соответствий строк в хеш-таблице. Так как хеш-функции могут быть подвержены коллизиям (два разных значения ключа имеют одинаковый хэш), приходится проверять каждое потенциальное соответствие, что необходимо для гарантии того, что в этом случае действительно совпадение обусловлено соединением, а не коллизией.

Источники:

- http://www.sql.ru/articles/mssql/2007/051103hashjoin.shtml
- https://blogs.msdn.microsoft.com/craigfr/2006/08/10/hash-join/

Смотрите также:

- Е. В. Филиппов «Настольная книга 1С:Эксперта по технологическим вопросам», стр. 91

06.28 — 1

При написании запросов не следует использовать **соединения с вложенными запросами**. Следует соединять друг с другом только объекты метаданных или временные таблицы. Если запрос использует соединения с вложенными запросами, то его следует переписать с использованием временных таблиц (не важно с какой стороны соединения находится вложенный запрос), кроме случая, когда вложенный запрос сканирует мало записей.

Если запрос содержит соединения с вложенными запросами, то это может привести к следующим негативным последствиям:

- **Крайне медленное выполнение запроса при слабой загрузке серверного оборудования. Замедление запроса может быть очень значительным (до нескольких порядков);**
- Нестабильная работа запроса. При некоторых условиях запрос может работать достаточно быстро, при других — очень медленно;
- Значительная разница по времени выполнения запроса на разных СУБД;
- Повышенная чувствительность запроса к актуальности и полноте статистик. Сразу после полного обновления статистик запрос может работать быстро, но через некоторое время опять замедлиться.

Оптимизатор сервера СУБД (независимо от того, какую СУБД вы используете) не всегда может правильно оптимизировать подобный запрос. В данном случае, проблемой для оптимизатора является выбор правильного способа соединения. Существуют несколько алгоритмов соединения двух выборок. Выбор того или иного алгоритма зависит от того, сколько записей будет содержаться в одной и в другой выборке. В том случае, если вы соединяете две физические таблицы, СУБД может легко определить объем обоих выборок на основании имеющейся статистики. Если же одна из соединяемых выборок представляет собой вложенный запрос, то понять, какое количество записей она вернет, становится очень сложно. В этом случае СУБД может ошибиться с выбором плана, что приведет к катастрофическому падению производительности запроса.

Источник:

- https://its.1c.ru/db/v8std/content/-2145782992/hdoc

06.29 — 3

При написании запросов не следует использовать **соединения с вложенными запросами**. Следует соединять друг с другом только объекты метаданных или временные таблицы. Если запрос использует соединения с вложенными запросами, то его следует переписать с использованием временных таблиц (не важно с какой стороны соединения находится вложенный запрос), кроме случая, когда вложенный запрос сканирует мало записей.

Если запрос содержит соединения с вложенными запросами, то это может привести к следующим негативным последствиям:

- Крайне медленное выполнение запроса при слабой загрузке серверного оборудования. Замедление запроса может быть очень значительным (до нескольких порядков);
- Нестабильная работа запроса. При некоторых условиях запрос может работать достаточно быстро, при других — очень медленно;
- Значительная разница по времени выполнения запроса на разных СУБД;
- Повышенная чувствительность запроса к актуальности и полноте статистик. Сразу после полного обновления статистик запрос может работать быстро, но через некоторое время

опять замедлиться.

Оптимизатор сервера СУБД (независимо от того, какую СУБД вы используете) не всегда может правильно оптимизировать подобный запрос. В данном случае, проблемой для оптимизатора является выбор правильного способа соединения. Существуют несколько алгоритмов соединения двух выборок. Выбор того или иного алгоритма зависит от того, сколько записей будет содержаться в одной и в другой выборке. В том случае, если вы соединяете две физические таблицы, СУБД может легко определить объем обоих выборок на основании имеющейся статистики. Если же одна из соединяемых выборок представляет собой вложенный запрос, то понять, какое количество записей она вернет, становится очень сложно. В этом случае СУБД может ошибиться с выбором плана, что приведет к катастрофическому падению производительности запроса.

Источник:

- https://its.1c.ru/db/v8std/content/-2145782992/hdoc

06.30 — 3

При написании запросов не следует использовать **соединения с вложенными запросами**. Следует соединять друг с другом только объекты метаданных или временные таблицы. **Если запрос использует соединения с вложенными запросами, то его следует переписать с использованием временных таблиц** (не важно с какой стороны соединения находится вложенный запрос), кроме случая, когда вложенный запрос сканирует мало записей.

- Крайне медленное выполнение запроса при слабой загрузке серверного оборудования. Замедление запроса может быть очень значительным (до нескольких порядков);
- Нестабильная работа запроса. При некоторых условиях запрос может работать достаточно быстро, при других — очень медленно;
- Значительная разница по времени выполнения запроса на разных СУБД;
- Повышенная чувствительность запроса к актуальности и полноте статистик. Сразу после полного обновления статистик запрос может работать быстро, но через некоторое время опять замедлиться.

Оптимизатор сервера СУБД (независимо от того, какую СУБД вы используете) не всегда может правильно оптимизировать подобный запрос. В данном случае, проблемой для оптимизатора является выбор правильного способа соединения. Существуют несколько алгоритмов соединения двух выборок. Выбор того или иного алгоритма зависит от того, сколько записей будет содержаться в одной и в другой выборке. В том случае, если вы соединяете две физические таблицы, СУБД может легко определить объем обоих выборок на основании имеющейся статистики. Если же одна из соединяемых выборок представляет собой вложенный запрос, то понять, какое количество записей она вернет, становится очень сложно. В этом случае СУБД может ошибиться с выбором плана, что приведет к катастрофическому падению производительности запроса.

Источник:

- https://its.1c.ru/db/v8std/content/-2145782992/hdoc

06.31 — 4

Если в запросе используется **соединение с виртуальной таблицей** языка запросов 1С:Предприятия (например, РегистрНакопления.Товары.Остатки()) и запрос работает с неудовлетворительной производительностью, то рекомендуется вынести обращение к виртуальной таблице в отдельный запрос с сохранением результатов во **временной таблице**.

Источник:

- https://its.1c.ru/db/v8std/content/-2145782992/hdoc

06.32 — 3

В языке запросов возможно обращаться не только к полям исходных таблиц запроса, перечисленных в предложении ИЗ, но и к полям таблицы, на которую ссылается поле исходной таблицы запроса, если это поле имеет ссылочный тип. Имена полей при этом пишутся «через точку». Применение такой конструкции приводит к неявному соединению с дополнительными таблицами для получения значений полей «через точку».

Большое число исходных таблиц запроса приводит к его усложнению и может значительно увеличивать время его выполнения. Особенно это важно помнить в тех случаях, когда поле таблицы ссылочного типа имеет **составной тип** и может содержать ссылки на несколько таблиц. В таком случае, получение полей других таблиц «через точку» от такого поля составного

типа приведет к соединению со всеми таблицами, ссылки на которые могут оказаться в данном поле и в RLS к этим таблицам.

Для того чтобы избежать запросов с использованием большого числа исходных таблиц следует жертвовать компактностью хранения данных ради производительности и помещать соответствующие данные в исходную таблицу запроса.

При необходимости следует жертвовать компактностью и универсальностью кода ради производительности:

- Как правило, для выполнения конкретного запроса в данных условиях не нужны все возможные типы данной ссылки. В этом случае, следует ограничить количество возможных типов при помощи функции **ВЫРАЗИТЬ**.
- Если данный запрос является универсальным и используется в нескольких разных ситуациях (где типы ссылки могут быть разными), то можно формировать запрос динамически, подставляя в функцию **ВЫРАЗИТЬ** тот тип, который необходим при данных условиях.

Это увеличит объем исходного кода и, возможно, сделает его менее универсальным, но может существенно повысить производительность и стабильность работы запроса.

Источник:

- https://its.1c.ru/db/v8std#content:2149184303:hdoc

06.33 — 2

В языке запросов возможно обращаться не только к полям исходных таблиц запроса, перечисленных в предложении ИЗ, но и к полям таблицы, на которую ссылается поле исходной таблицы запроса, если это поле имеет ссылочный тип. Имена полей при этом пишутся «через точку». Применение такой конструкции приводит к неявному соединению с дополнительными таблицами для получения значений полей «через точку».

Большое число исходных таблиц запроса приводит к его усложнению и может значительно увеличивать время его выполнения. Особенно это важно помнить в тех случаях, когда поле таблицы ссылочного типа имеет **составной тип** и может содержать ссылки на несколько таблиц. В таком случае, получение полей других таблиц «через точку» от такого поля составного типа приведет к соединению **со всеми таблицами**, ссылки на которые могут оказаться в данном поле и в RLS к этим таблицам.

Для того чтобы избежать запросов с использованием большого числа исходных таблиц следует жертвовать компактностью хранения данных ради производительности и помещать соответствующие данные в исходную таблицу запроса.

При необходимости следует жертвовать компактностью и универсальностью кода ради производительности:

- Как правило, для выполнения конкретного запроса в данных условиях не нужны все возможные типы данной ссылки. В этом случае, следует ограничить количество возможных типов при помощи функции **ВЫРАЗИТЬ**.
- Если данный запрос является универсальным и используется в нескольких разных ситуациях (где типы ссылки могут быть разными), то можно формировать запрос динамически, подставляя в функцию **ВЫРАЗИТЬ** тот тип, который необходим при данных условиях.

Это увеличит объем исходного кода и, возможно, сделает его менее универсальным, но может существенно повысить производительность и стабильность работы запроса.

Источник:

- https://its.1c.ru/db/v8std#content:2149184303:hdoc

06.34 — 1

Основные причины неоптимальной работы запросов, диагностируемые на уровне кода конфигурации и структуры метаданных:

- соединения с подзапросами
- соединения с виртуальными таблицами
- несоответствие индексов и условий запроса
- использование логического ИЛИ в условиях
- использование подзапросов в условии соединения
- получение данных через точку от полей составного типа
- фильтрация виртуальных таблиц без использования параметров

Источник:

- https://its.1c.ru/db/metod8dev/content/5842/hdoc

06.35 — 4

Убедитесь в том, что для всех условий, использованных в запросе, имеются подходящие индексы.

Условия используются в следующих секциях запроса:

- ВЫБРАТЬ … ИЗ … ГДЕ <условие>
- СОЕДИНЕНИЕ … ПО <условие>
- ВЫБРАТЬ … ИЗ <ВиртуальнаяТаблица>(, <условие>)
- ИМЕЮЩИЕ <условие>

Для каждого условия должен существовать подходящий индекс. Подходящим является индекс, удовлетворяющий следующим требованиям:

1. Индекс содержит все поля перечисленные в условии;
2. Эти поля находятся в самом начале индекса;
3. Эти поля идут подряд, то есть между ними не «вклиниваются» поля, не участвующие в условии запроса;

Источник:

- https://its.1c.ru/db/metod8dev/content/5842/hdoc

06.36 — 6

Подтверждения не нашел, но очевидно, что подходят все представленные варианты ответов, а именно:

- Медленное выполнение запроса при слабой нагрузке на оборудование
- Возникновение избыточных блокировок.
- Значительная разница по времени выполнения запроса на разных СУБД.
- Повышенная чувствительность запроса к актуальности и полноте статистик.
- Нестабильная работа запроса. При некоторых условиях запрос может работать достаточно быстро, при других — очень медленно.

06.37 — 1

JOIN добавляет столбцы в результирующую таблицу, а UNION добавляет таблицу с тем же составом столбцов.

Источники:

- https://docs.microsoft.com/ru-ru/sql/t-sql/queries/from-transact-sql
- https://docs.microsoft.com/ru-ru/sql/t-sql/language-elements/set-operators-union-transact-sq

06.38 — 2

При UNION полностью одинаковые строки заменяются одной, на что затрачивается дополнительное время, даже в случаях, когда одинаковых строк в запросах заведомо быть не может.

Источник:

- https://docs.microsoft.com/ru-ru/sql/t-sql/language-elements/set-operators-union-transact-sq

06.39 — 1

```
select *
from dbo._document180
where _number like 'ТД00%'
order by _number
```

Источник:

- https://docs.microsoft.com/ru-ru/sql/t-sql/queries/select-transact-sql

06.40 — 3

```
delete
from dbo._document180
where _number = 'ТД00-000003'
```

Источник:

- https://docs.microsoft.com/ru-ru/sql/t-sql/statements/delete-transact-sql

06.41 — 1

```
delete
from dbo._document180
where _number = 'ТД00-000003'
```

Источник:

- https://docs.microsoft.com/ru-ru/sql/t-sql/queries/update-transact-sql

06.42 — 2

```
select _number, posted
from dbo._document180
where _number like 'ТД00%'

union all

select _number, posted
from dbo._document182 o

rder by _number
```

Источник:

- https://docs.microsoft.com/ru-ru/sql/t-sql/language-elements/set-operators-union-transact-sql

06.43 — 1

```
select *
from dbo.document180
    inner join dbo.document180_vt4131
    on dbo.document180._idrref = dbo.document180_vt4131._idrref
where dbo.document180._number like 'ТД00%'
```

Соединения outer join не существует.

Источник:

- https://docs.microsoft.com/ru-ru/sql/t-sql/queries/from-transact-sql

06.44 — 3

DBORACLE - исполнение операторов SQL СУБД Oracle Database

Источник:

- https://its.1c.ru/db/v8313doc/bookmark/adm/TI000000396

06.45 — 3

DB2 - исполнение операторов SQL СУБД DB2

Источник:

- https://its.1c.ru/db/v8313doc/bookmark/adm/TI000000396

06.46 — 3

DBPOSTGRS - исполнение операторов SQL СУБД PostgreSQL

Источник:

- https://its.1c.ru/db/v8313doc/bookmark/adm/TI000000396

06.47 — 4

DBV8DBENG - исполнение операторов SQL файловой СУБД

Источник:

- https://its.1c.ru/db/v8313doc/bookmark/adm/TI000000396

06.48 — 2

EDS - работа с внешними источниками данных

Источник:

- https://its.1c.ru/db/v8313doc/bookmark/adm/TI000000396

Помогла ли вам данная статья?

- Да, спасибо, помогла.
- Немного помогла.
- Совсем не помогла.
- Не то, что я искал(а).

Голосовать

Смотреть результаты

Смотрите также:

Решение всех вопросов теста 1С:Профессионал по технологическим вопросам (Раздел №5)

Ниже приводится решение всех вопросов для подготовки к аттестации 1С:Профессионал по технологическим вопросам. Текстов самих вопросов и вариантов ответов нет. Предполагается, что у вас имеется книга «Комплект вопросов…

Решение всех вопросов теста 1С:Профессионал по технологическим вопросам (Раздел №3)

Ниже приводится решение всех вопросов для подготовки к аттестации 1С:Профессионал по технологическим вопросам. Текстов самих вопросов и вариантов ответов нет. Предполагается, что у вас имеется книга «Комплект вопросов…

Решение всех вопросов теста 1С:Профессионал по технологическим вопросам (Раздел №1)

Ниже приводится решение всех вопросов для подготовки к аттестации 1С:Профессионал по технологическим вопросам. Текстов самих вопросов и вариантов ответов нет. Предполагается, что у вас имеется книга «Комплект вопросов…

Запись опубликована в рубрике Эксперт 1С с метками 1С:Профессионал, 1С:Эксперт. Добавьте в закладки постоянную ссылку.

← Решение всех вопросов теста 1С:Профессионал по технологическим вопросам (Раздел №5)

Песни разработчика: Don’t Worry Be Happy →

10 Responses to Решение всех вопросов теста 1С:Профессионал по технологическим вопросам (Раздел №6)

Concrete *говорит:*
28.01.2019 в 19:44

Скажите, а ответы на разделы с 7 по 14 будут выкладываться?

Ответить

Concrete *говорит:*
25.03.2019 в 18:55

Здравствуйте! Огромное спасибо за решения. Скажите, разделы 7 — 14 выкладываться будут?

Ответить

Дмитрий *говорит:*
10.06.2019 в 14:25

Отличная статья, помогло, будут ли статьи по разделам 7 — 14 ?

Ответить

CSiER *говорит:*
09.07.2019 в 11:24

06.13 — 2 — запись попадает в ТЖ в момент завершения события, при этом указывается время окончания и длительность (сам формат не предполагает записи в момент возникновения).

Ответить

Виталий Онянов *говорит:*
19.07.2019 в 12:53

Спасибо, добавил в статью.

Ответить

Виталий Онянов *говорит:*
19.07.2019 в 12:51

Вышел новый сборник задач (издание 2). Сначала эти статьи надо переписать, потом продолжу. Есть в планах, да.

Ответить

Аноним *говорит:*
30.07.2019 в 20:04

Спасибо, очень ждем

Ответить

Петр *говорит:*
28.10.2022 в 14:45

Виталий, я прошёл 6 разделов с комплектом вопросов редакции 2 и нашёл только одно отличие, отписал в комментариях раздела. То есть все эти статьи переписывать нет нужды, они по прежнему актуальны.
Далее я продолжаю двигаться по сборнику вопросов и в любом случае буду конспектировать информацию по ответам для себя. Если это поможет при наполнении сайта, то могу поделиться информацией.

Ответить

Виталий Онянов *говорит:*
01.11.2022 в 20:01

Да, конечно, я сам после сдачи экзамена к продолжению цикла статей интерес потерял. Если у вас будет готовый материал — с радостью опубликую.

Ответить

Антон *говорит:*
14.11.2022 в 14:52

Пётр, очень надо. Зачёт в карму, если победите этот вопрос.

Ответить

Добавить комментарий

Ваш адрес email не будет опубликован. Обязательные поля помечены *

Комментарий *

Уведомлять меня о новых комментариях по e-mail

Имя

Email

Сайт

Отправить комментарий