МИНОБРНАУКИ РОССИИ

Передовые инженерные школы

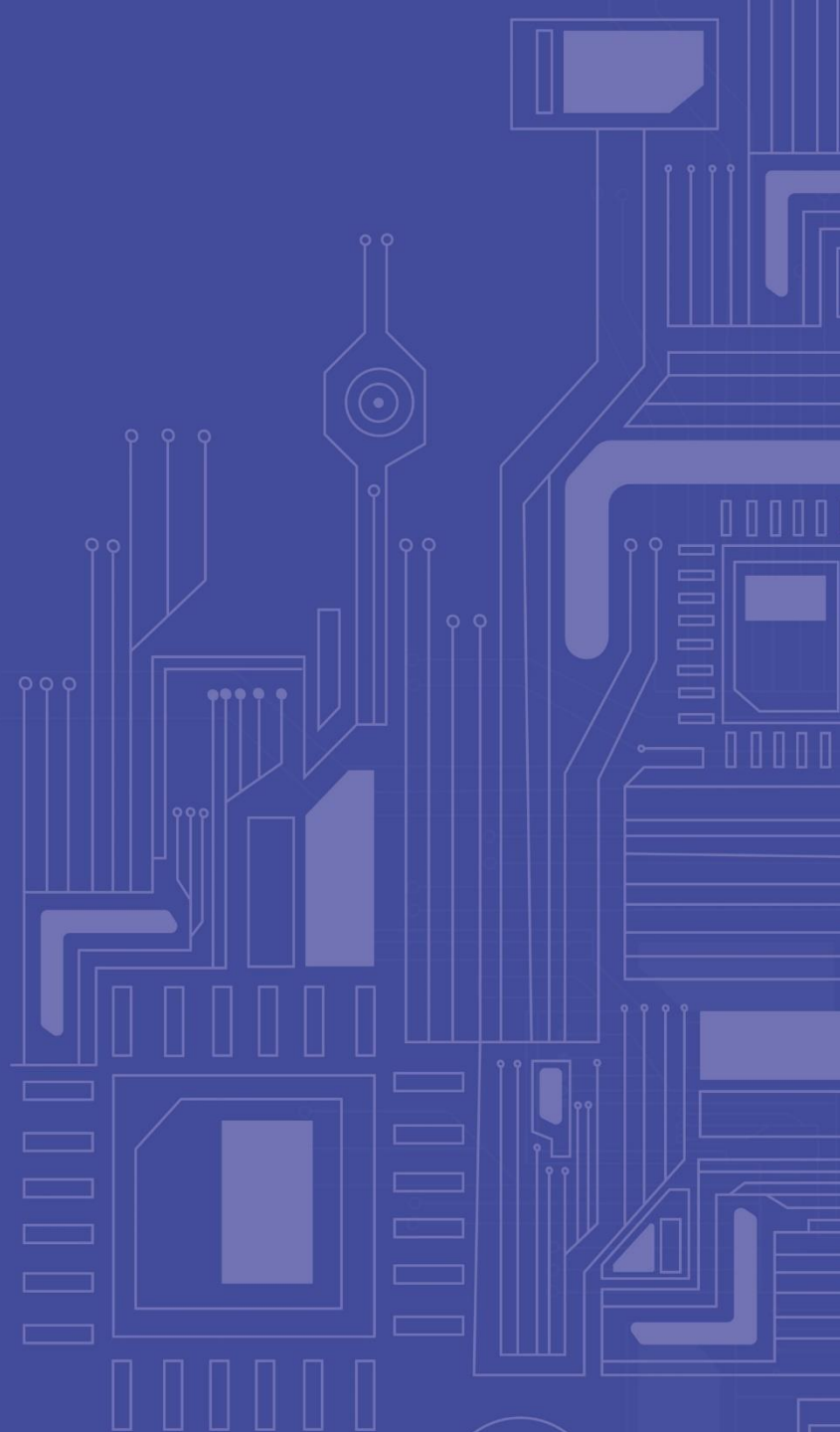
СИСТЕМЫ УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ

Лекция 6

ПЛАН ЛЕКЦИИ

- Цели СУБД
  - System R
    - Организация внешней памяти в System R
      - Интерфейс RSS
      - Общие принципы организации данных во внешней памяти в SQL-ориентированных СУБД

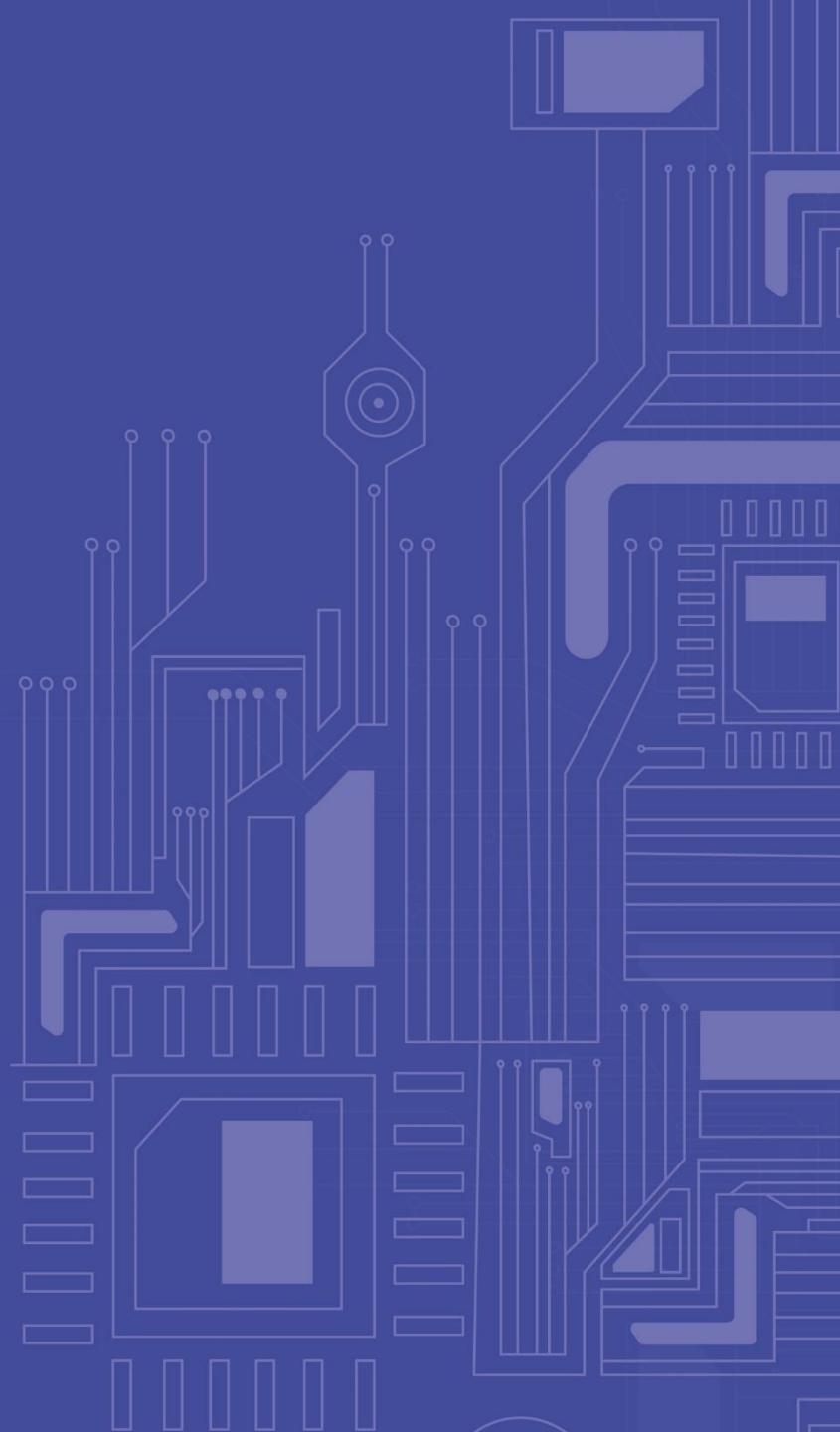
ЦЕЛИ СУБД



ЗАЧЕМ НУЖНЫ СУБД? ЗАДАЧИ.

- Обеспечить ненавигационный интерфейс высокого уровня пользователя с системой, позволяющий достичь независимости данных и дать возможность пользователям работать максимально эффективно;
- обеспечить многообразие допустимых способов использования СУБД, включая программируемые транзакции, диалоговые транзакции и генерацию отчетов;
- поддерживать динамически изменяемую среду баз данных, в которой таблицы, индексы, представления, транзакции и другие объекты могут легко добавляться и уничтожаться без приостановки нормального функционирования системы;
- обеспечить возможность параллельной работы с одной базой данных многих пользователей, допуская параллельную модификацию объектов базы данных при наличии необходимых средств защиты целостности базы данных;
- обеспечить средства восстановления согласованного состояния баз данных после разного рода сбоев аппаратуры или программного обеспечения;
- обеспечить гибкий механизм, позволяющий определять различные представления хранимых данных и ограничивать этими представлениями доступ пользователей к базе данных по выборке и модификации на основе механизма авторизации;
- обеспечить производительность системы при выполнении упомянутых функций, сопоставимую с производительностью существующих СУБД низкого уровня.

SYSTEM R



ИСТОРИЯ И ТЕРМИНОЛОГИЯ

1. В 1975-1979 г.г. в исследовательской лаборатории компании IBM разрабатывалась система управления реляционными базами данных System R.
2. Исключительно важен опыт, приобретенный при разработке этой системы. Практически во всех более поздних реляционных СУБД в той или иной степени используются методы, примененные в System R.
3. Несмотря на то, что при реализации System R использовался подход, несколько отличающийся от реляционного подхода Кодда (отсюда и пошли расхождения между реляционной моделью данных и моделью данных SQL), мы будем активно пользоваться терминами реляционной модели.
   - Таблицы и отношения
   - Кортежи и типы данных
4. Таблицы могут физически храниться в одном или нескольких сегментах, каждому из которых соответствует отдельный файл внешней памяти. Сегменты разбиваются на страницы, в которых располагаются кортежи, принадлежащие таблицам и вспомогательные служебные структуры данных – индексы. Соответственно, каждый сегмент содержит две группы страниц – страницы данных и страницы индексной информации. Страницы каждой группы имеют фиксированный размер, но страницы с индексной информацией меньше по размеру, чем страницы данных. В страницах данных могут располагаться кортежи более чем одной таблицы .

ТЕРМИНОЛОГИЯ - ПОЯСНЕНИЯ

ТАБЛИЦА

Базовым понятием System R является понятие *таблицы* (приближенный к реализации аналог основного понятия реляционного подхода *отношения*; иногда, в зависимости от контекста, мы будем использовать и этот термин). Таблица – это регулярная структура данных, состоящая из конечного набора однотипных записей – кортежей. Каждый кортеж одной таблицы состоит из конечного (и одинакового) числа полей кортежа, причем *i*-тое поле каждого кортежа одной таблицы может содержать данные только одного типа, и набор допустимых типов данных в System R предопределен и фиксирован.

КОРТЕЖ – ПОЛЕ ТАБЛИЦЫ

ДОМЕН – ТИП ДАННЫХ

В силу регулярности структуры таблицы понятие поля кортежа расширяется до понятия поля таблицы. Тогда *i*-тое поле таблицы можно трактовать как набор одноместных кортежей, полученных выборкой *i*-тых полей из каждого кортежа этой таблицы, т.е. в общепринятой терминологии как проекцию таблицы на *i*-тый атрибут. В терминологию System R не входит понятие домена, оно заменяется здесь понятием типа поля, т.е. типом данных, хранение которых в данном поле допускается (это не вполне эквивалентная замена, но такова реальность System R).

ЯЗЫК SEQUEL (SQL)

Основой System R является «реляционный» язык SEQUEL (который достаточно быстро был переименован в SQL). Заметим, что разработчики System R искренне считали созданный ими язык реляционным; однако, в этом языке в действительности нарушаются многие важные принципы реляционной модели данных.

ВОЗМОЖНОСТИ ЯЗЫКА SQL

1. Динамическая компиляция запросов (возможность диалогов)
2. Динамическая параметризация
3. Средствами SQL определяются все доступные пользователю объекты баз данных: таблицы, индексы, представления.
4. Имеются средства уничтожения любого такого объекта.
5. Соответствующие операторы языка могут выполняться в любой момент, и возможность выполнения операции данным пользователем зависит от ранее предоставленных ему прав.

ЦЕЛОСТНОСТЬ ДАННЫХ (1/2)

Под ***целостным состоянием*** базы данных понимается состояние, удовлетворяющее набору сохраняемых при базе данных предикатов целостности. Эти предикаты, называемые в System R *утверждениями целостности* (assertion), также задаются средствами языка SQL.

ТРАНЗАКЦИИ

Любой оператор языка выполняется в границах некоторой *транзакции* – последовательности операторов языка, неделимой в смысле состояния базы данных. Неделимость означает, что все изменения базы данных, произведенные в пределах одной транзакции, либо целиком отображаются в состоянии базы данных, либо полностью в нем отсутствуют.

ОТКАТ ТРАНЗАКЦИИ

Откат транзакции - удаление изменения базы данных, произведенного в пределах одной транзакции, который может произойти по инициативе пользователя (при выполнении соответствующего оператора SQL) или по инициативе системы. Одной из причин отката транзакции по инициативе системы является нарушение целостности базы данных в результате действий данной транзакции.

ТОЧКИ СОХРАНЕНИЯ

Язык SQL System R содержит средство установки так называемых *точек сохранения* (savepoint). При инициируемом пользователем откате транзакции можно указать номер точки сохранения, выше которого откат не распространяется. Инициируемый системой откат транзакции производится до ближайшей точки сохранения, в которой условие, вызвавшее откат, уже отсутствует.

ЦЕЛОСТНОСТЬ ДАННЫХ (2/2)

ЖУРНАЛ

Журнал – специальный набор данных, в который помещаются записи обо всех операциях всех транзакций, изменяющих состояние базы данных. Естественно, что для реального выполнения отката транзакции необходимо запоминать некоторую информацию о выполнении транзакции.

UNDO

При откате транзакции происходит процесс *обратного выполнения* транзакции (undo), в ходе которого в обратном порядке выполняются все изменения, запомненные в журнале.

ТРИГГЕРЫ

В языке SQL System R имеется средство определения так называемых *триггеров* (trigger), позволяющих автоматически поддерживать целостность базы данных при модификациях ее объектов. В SQL System R триггер – это каталогизированная операция модификации, для которой задано условие ее автоматического выполнения.

Возможна ситуация, когда пользователи просто не могут соблюдать целостность базы данных без автоматического выполнения условных воздействий, поскольку они просто «не видят» всей базы данных и, в частности, не могут представить всех ограничений ее целостности.

ПРЕДСТАВЛЕНИЯ И ДОСТУПЫ

АВТОРИЗАЦИЯ ДОСТУПА

Авторизация доступа к базе данных также основана на средствах SQL. При создании любого объекта базы данных пользователь, выполняющий эту операцию, становится полновластным владельцем этого объекта, т.е. может выполнять по отношению к этому объекту любую допустимую операцию SQL. Далее этот пользователь может выполнить оператор SQL, означающий передачу всех его прав на этот объект (или их подмножества) любому другому пользователю. В частности, этому пользователю может быть передано право на передачу всех переданных ему прав (или их части) третьему пользователю и т.д. Одним из прав пользователя по отношению к объекту является право на изъятие у других пользователей всех или некоторых прав, которые ранее им были переданы. Эта операция распространяется транзитивно на всех дальнейших наследников этих прав.

ПРЕДСТАВЛЕНИЕ (VIEW)

Представление – это каталогизированный именованный запрос на выборку данных (из одной или нескольких таблиц). Поскольку SQL – это «реляционный» язык, результатом выполнения любого запроса на выборку является таблица, и поэтому концептуально можно относиться к любому представлению как к таблице. В языке допускается использование ранее определенных представлений практически везде, где допускается использование таблиц. Наличие возможности определять представления в совокупности с развитой системой авторизации позволяет ограничить доступ некоторых пользователей к базе данных выделенным набором представлений.

НАДЕЖНОСТЬ И ЭФФЕКТИВНОСТЬ

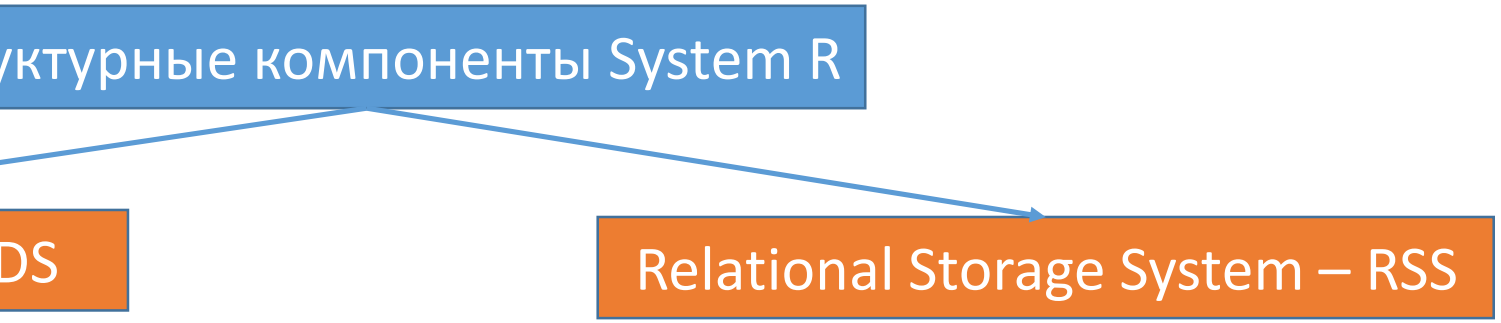
УСТОЙЧИВОСТЬ К СБОЯМ

Одним из основных требований к СУБД вообще и к System R в частности является обеспечение надежности баз данных по отношению к различного рода сбоям. К таким сбоям могут относиться программные ошибки прикладного и системного уровня, сбои процессора, поломки внешних носителей и т.д. В частности, к одному из видов сбоев можно отнести упоминавшиеся выше нарушения целостности базы данных и автоматический инициируемый системой откат транзакции – это системное средство восстановления базы данных после сбоев такого рода. Такое восстановление происходит путем обратного выполнения транзакции на основе информации о внесенных ею изменениях, запомненной в журнале. На информации журнала также основано восстановление базы данных и после сбоев другого рода.

ОСНОВЫ ЭФФЕКТИВНОСТИ

- Специфика физической организации баз данных во внешней памяти,
- использование техники индексированного доступа к данным,
- буферизация используемых страниц базы данных в основной памяти,
- развитая техника оптимизации SQL-запросов, производимой на стадии их компиляции.

СТРУКТУРА SYSTEM R: RSS И RDS

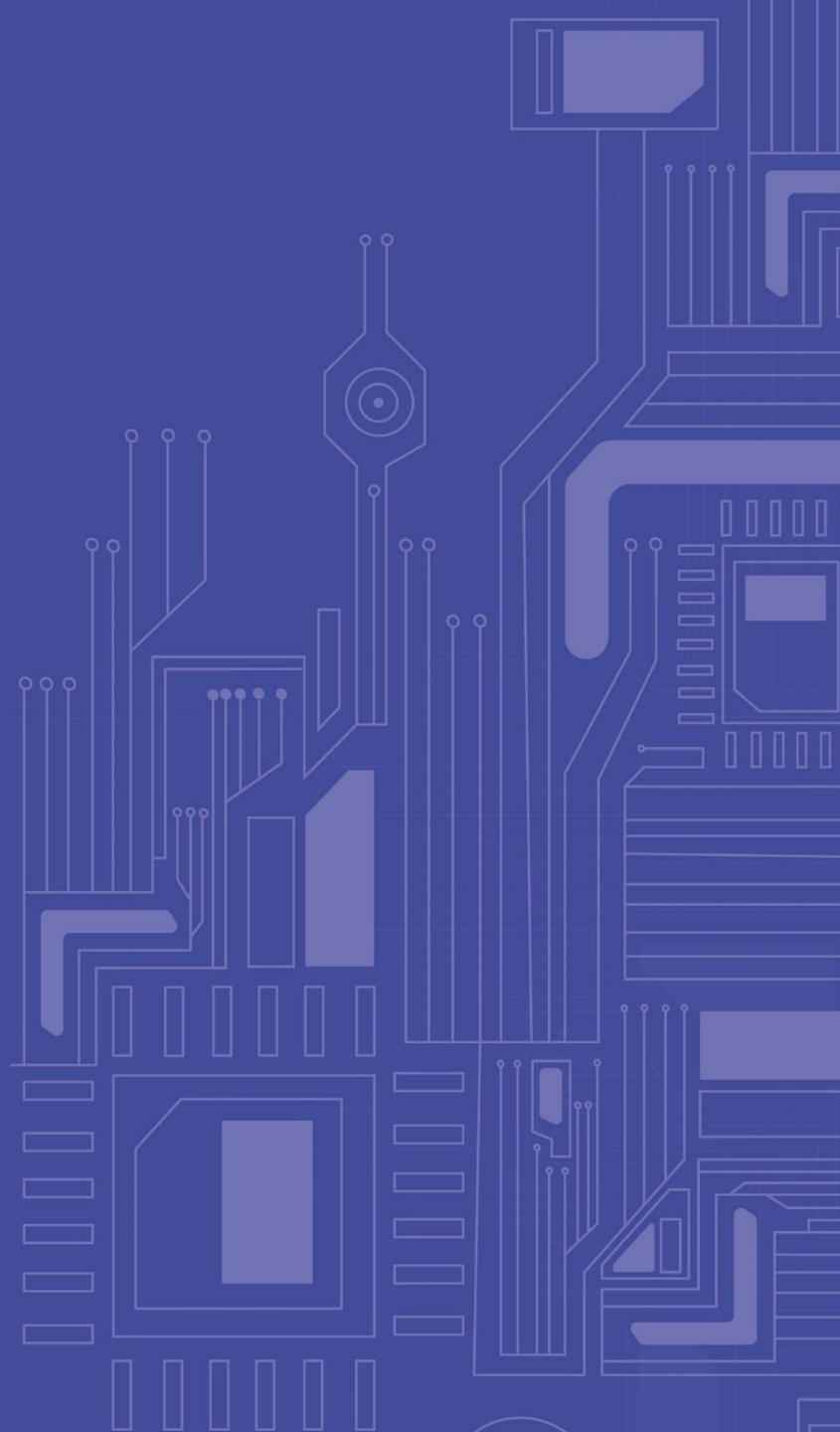


Система управления реляционными данными, состоящая, по существу, из компилятора языка SQL и подсистемы поддержки откомпилированных операторов. Компилятор запросов использует RSS для доступа к справочной информации (каталоги таблиц, индексов, прав доступа, условий целостности, и т.д.) и производит программы, выполняемые в дальнейшем также с использованием RSS.

Система управления реляционной памятью, которая обеспечивает интерфейс довольно низкого, но достаточного для реализации SQL уровня, для доступа к хранимым в базе данным. Синхронизация транзакций, журнализация изменений и восстановление баз данных после сбоев также относятся к числу функций RSS.

Таким образом, система естественно разделяется на два уровня – уровень управления памятью и синхронизацией, фактически, не зависящий от базового языка запросов системы, и языковый уровень (уровень SQL), на котором решается большинство проблем System R. Заметим, что язык SQL можно в принципе заменить каким-либо другим языком, но он должен обладать примерно такой же семантикой.

ОРГАНИЗАЦИЯ ВНЕШНЕЙ ПАМЯТИ В SYSTEM R



СТРАНИЦЫ ДАННЫХ

База данных System R располагается в одном или нескольких сегментах внешней памяти. Каждый сегмент состоит из страниц данных и страниц индексной информации. Размер страницы данных в сегменте может быть выбран равным либо 4, либо 32 килобайтам; размер страницы индексной информации равен 512 байтам. При работе RSS поддерживается дополнительный набор данных для ведения журнала. Для повышения надежности журнала этот набор данных дублируется на двух внешних носителях.

В каждой странице данных хранятся кортежи одной или нескольких таблиц. Фундаментальным понятием RSS является *идентификатор кортежа (tuple identifier – tid)*. Гарантируется неизменяемость tid'а во все время существования кортежа в базе данных независимо от перемещений кортежа внутри страницы и даже при перемещении кортежа в другую страницу.

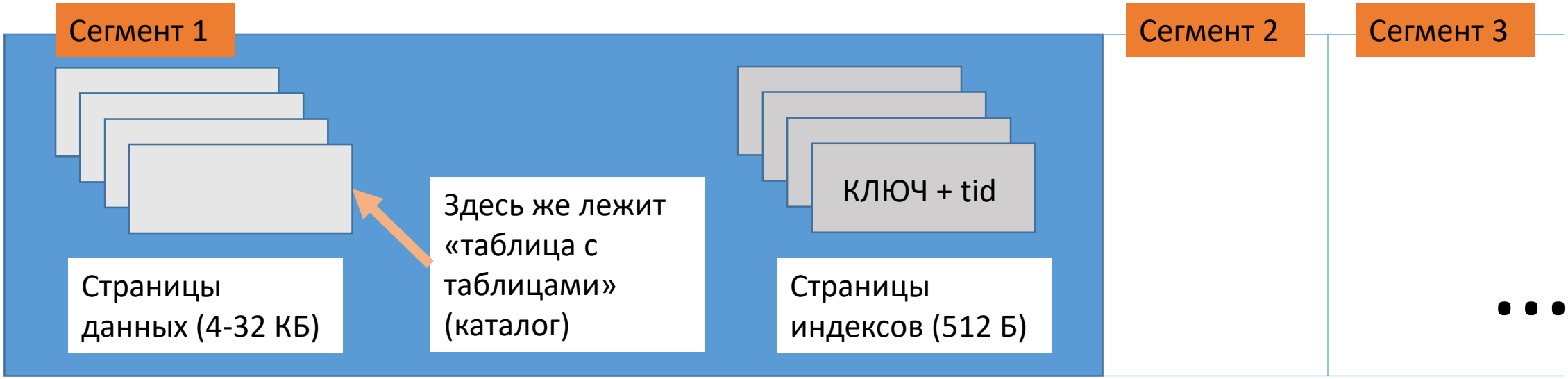
МОДИФИКАЦИЯ ДАННЫХ

СЛУЖЕБНАЯ ИНФОРМАЦИЯ В КОРТЕЖАХ

Поскольку допускается нахождение в одной странице данных кортежей разных таблиц, каждый кортеж должен, кроме содержательной части, включать служебную информацию, идентифицирующую таблицу, которой принадлежит данный кортеж. Требуется хранения при кортеже дополнительной служебной информации, определяющей реальное число полей в данном кортеже.

ДОБАВЛЕНИЕ ПОЛЕЙ В ТАБЛИЦЫ

В языке SQL допускается динамическое добавление полей к существующим таблицам. При этом реально происходит лишь модификация описателя таблицы в таблице-каталоге таблиц. В существующем кортеже таблицы новое поле возникает только при модификации этого кортежа, затрагивающей новое поле.



ИНДЕКСЫ

ИНДЕКСЫ

На основе наличия уникальных, обеспечивающих почти прямой доступ к кортежам и не изменяемых во время существования кортежей tid'ов в System R поддерживаются дополнительные управляющие структуры – *индексы*. Каждый индекс определяется на одном или нескольких полях таблицы, значения которых составляют его ключ, и позволяет производить прямой поиск по ключу кортежей (их tid'ов) и последовательное сканирование таблицы по индексу, начиная с указанного ключа, в порядке возрастания или убывания значений ключа.

КЛЮЧИ

Некоторые индексы при их создании могут обладать атрибутом уникальности. В таком индексе не допускаются дубликаты ключа. Это единственное средство SQL System R указания системе первичного ключа таблицы (фактически, набора первичного и всех возможных ключей таблицы).

B+ - ДЕРЕВЬЯ

Для организации индексов в System R применяется техника *B+-деревьев* . Каждый индекс занимает отдельный набор страниц, номер корневой страницы запоминается в описателе индекса. Использование B+-деревьев позволяет достичь эффективности при прямом поиске, поскольку они из-за своей *сильной ветвистости* обладают небольшой глубиной. Кроме того, B+-деревья сохраняют *порядок ключей* в листовых блоках иерархии, что позволяет производить последовательное сканирование таблицы в порядке возрастания или убывания значений полей, на которых определен индекс.

КЛАСТЕРИЗАЦИЯ ТАБЛИЦ

КЛАСТЕРИЗАЦИЯ КОРТЕЖЕЙ

Под кластеризацией кортежей понимается физически близкое расположение (в пределах одной страницы данных) логически связанных кортежей. Обеспечение соответствующей кластеризации позволяет добиться высокой эффективности системы при выполнении некоторого класса запросов.

КЛАСТЕРИЗАЦИЯ ТАБЛИЦ

Средство определения условий кластеризации таблицы – объявить до заполнения таблицы один (и только один) индекс, определенный на полях этой таблицы, кластеризованным. Тогда, если заполнение таблицы кортежами производится в порядке возрастания или убывания значений полей кластеризации (в зависимости от атрибутики индекса), система физически располагает кортежи в страницах данных в том же порядке.

ХРАНЕНИЕ КЛАСТЕРИЗОВАННЫХ ТАБЛИЦ

В каждой странице данных кластеризованной таблицы оставляется некоторое резервное свободное пространство. При последующих вставках кортежей в такую таблицу система стремится поместить каждый кортеж в одну из страниц данных, в которых уже находятся кортежи этой таблицы с такими же (или близкими) значениями полей кластеризации. Естественно, что поддерживать идеальную кластеризацию таблицы можно только до определенного предела, пока не исчерпается резервная память в страницах. Далее этого предела степень кластеризации таблицы начинает уменьшаться, и для восстановления идеальной кластеризации таблицы требуется физическая реорганизация таблицы (ее можно произвести средствами SQL).

ВЫПОЛНЕНИЕ ЗАПРОСОВ

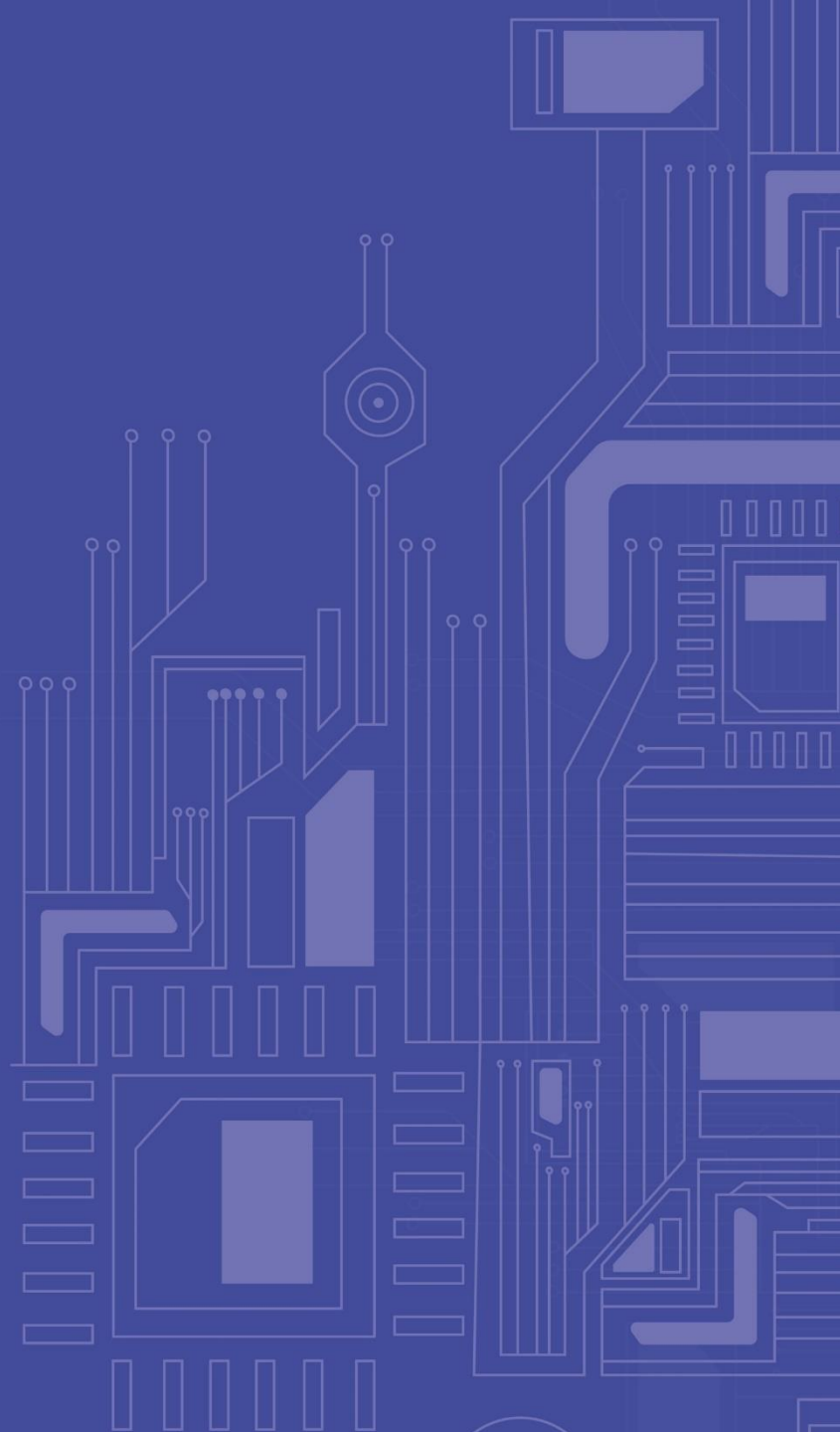
ЗАПРОСЫ НА КЛАСТЕРИЗОВАННЫХ ТАБЛИЦАХ

Очевидным преимуществом кластеризации таблицы является то, что при последовательном сканировании кластеризованной таблицы с использованием кластеризованного индекса потребуется ровно столько чтений страниц данных из внешней памяти, сколько страниц занимают кортежи этой таблицы. Следовательно, при правильно выбранных критериях кластеризации запросы, связанные с заданием условий на полях кластеризации можно выполнить почти оптимально.

СПИСКИ

Кроме таблиц и индексов при работе System R во внешней памяти могут располагаться еще и временные объекты – *списки (list)*. Список – это временная структура данных, создаваемая с целью оптимизации выполнения SQL-запроса, содержащая некоторые кортежи хранимой таблицы базы данных, не имеющая имени и, следовательно, не видимая на уровне интерфейса SQL. Кортежи списка могут быть упорядочены по возрастанию или убыванию полей соответствующей таблицы. Средства работы со списками имеются в интерфейсе RSS, но их, естественно, нет в SQL. Соответственно, эти средства используются только внутри системы при выполнении запросов (в частности, один из наиболее эффективных алгоритмов выполнения соединений основан на использовании отсортированных списков кортежей).

ИНТЕРФЕЙС RSS



ГРУППЫ ОПЕРАЦИЙ

На уровне RSS отсутствует именование объектов базы данных, употребляемое на уровне SQL. Вместо имен объектов используются их уникальные идентификаторы, являющиеся прямыми или косвенными адресами внутренних описателей объектов во внешней памяти для постоянных объектов или в основной памяти для временных объектов. Замена имен объектов базы данных на их идентификаторы производится компилятором SQL на основе информации, черпаемой им из системных таблиц-каталогов.

Можно выделить следующие группы операций:

- операции сканирования таблиц и списков;
- операции создания и уничтожения постоянных и временных объектов базы данных;
- операции модификации таблиц и списков;
- операция добавления поля к таблице;
- операции управления прохождением транзакций;
- операция явной синхронизации.

ОПЕРАЦИИ СКАНИРОВАНИЯ (1/2)

Операции группы сканирования позволяют последовательно, в порядке, определяемом типом сканирования, прочитать кортежи таблицы или списка, удовлетворяющие требуемым условиям. Группа включает операции **OPEN, NEXT** и **CLOSE**, означающие, соответственно, начало сканирования, требование чтения следующего кортежа, удовлетворяющего условиям, и конец сканирования.

ПРЯМОЕ СКАНИРОВАНИЕ

При прямом сканировании единственным параметром операции OPEN является идентификатор таблицы (включающий и идентификатор сегмента, в котором эта таблица хранится). По причине того, что в System R допускается размещение в одной странице данных кортежей нескольких таблиц, прямое сканирование предполагает последовательный просмотр всех страниц сегмента с выделением в них кортежей, входящих в данную таблицу; это очень дорогой способ сканирования таблицы.

СКАНИРОВАНИЕ ЧЕРЕЗ ИНДЕКС

При начале сканирования таблицы через индекс в число параметров операции OPEN входит идентификатор индекса, определенного ранее на полях этой таблицы. Кроме того, можно указать диапазон сканирования в терминах значений поля (полей), составляющего ключ индекса. При открытии сканирования через индекс производится начальная установка указателя сканирования в позицию листа B-дерева индекса, соответствующую левой границе заданного диапазона. Дальше перебор последовательно по ключу до окончания диапазона.

ОПЕРАЦИИ СКАНИРОВАНИЯ (2/2)

СКАНИРОВАНИЕ СПИСКА

При сканировании списка, как и при прямом сканировании таблицы, единственным параметром операции OPEN является идентификатор списка, но, в отличие от прямого сканирования таблицы это сканирование максимально эффективно: читаются только страницы, содержащие кортежи из данного списка, и порядок сканирования совпадает с порядком занесения кортежей в список или порядком списка, если он упорядочен.

ОПЕРАЦИЯ NEXT

Операция NEXT выполняет чтение следующего кортежа указанного сканирования, удовлетворяющего условию данной операции. Условие представляет собой дизъюнктивную нормальную форму простых условий, накладываемых на значения указанных полей таблицы. Простое условие – это условие вида номер-поля op константа, где op – операция сравнения <, <=, >, >=, = или !=. Общее условие является параметром операции NEXT.

Семантика операции NEXT следующая: начиная с текущей позиции сканирования выбираются кортежи таблицы в порядке, определяемом типом сканирования, до тех пор, пока не встретится кортеж, значения полей которого удовлетворяют указанному условию. Этот кортеж и является результатом операции.

ОПЕРАЦИЯ CLOSE

Операция CLOSE может быть выполнена в данной транзакции по отношению к любому ранее открытому сканированию независимо от его состояния. Параметром операции является идентификатор сканирования, и ее выполнение приводит к тому, что этот идентификатор становится недействительным (и уничтожаются служебные структуры памяти RSS, относящиеся к данному сканированию).

ОПЕРАЦИИ СОЗДАНИЯ (1/2)

Группа операций создания и уничтожения постоянных и временных объектов базы данных включает операции создания таблиц (**CREATE TABLE**), списков (**CREATE LIST**), индексов (**CREATE IMAGE**) и уничтожения любого из подобных объектов (**DROP TABLE**, **DROP LIST** и **DROP IMAGE**).

СОЗДАНИЕ ТАБЛИЦ

Входным параметром операций создания таблиц и списков является спецификатор структуры объекта, т.е. число полей объекта и спецификаторы их типов. Кроме того, при спецификации полей таблицы указывается разрешение или запрещение наличия неопределенных значений полей в кортежах этой таблицы или списка. Неопределенные значения кодируются специальным образом. Любая операция сравнения константы данного типа с неопределенным значением по определению вырабатывает значение *false*, кроме операции сравнения на совпадение со специальной литеральной константой NULL.

В результате выполнения этих операций заводится описатель в служебной таблице описателей таблиц или основной памяти (в зависимости от того, создается ли постоянный объект или временный), и вырабатывается идентификатор объекта, который служит входным параметром других операций, относящихся к соответствующему объекту (в частности, параметром операции OPEN при открытии сканирования объекта).

ОПЕРАЦИИ СОЗДАНИЯ (2/2)

СОЗДАНИЕ ИНДЕКСОВ

Входными параметрами операции CREATE IMAGE являются идентификатор таблицы, для которой создается индекс, список номеров полей, значения которых составляют ключ индекса, и признаки упорядочения по возрастанию или убыванию для всех полей, составляющих ключ. Кроме того, может быть указан признак уникальности индекса, т.е. запрещения наличия в данном индексе ключей-дубликатов.

Если операция выполняется по отношению к пустой в этот момент таблице, то выполнение операции такое же простое, как и для операций создания таблиц и списков: создается описатель в служебной таблице описателей индексов и возвращается идентификатор индекса (который, в частности, используется в качестве аргумента операции открытия сканирования таблицы через индекс).

Если же к моменту создания индекса соответствующая таблица не пуста (а это допускается), то операция становится существенно более дорогостоящей, поскольку при ее выполнении происходит реальное создание B-дерева индекса, что требует, по меньшей мере, одного последовательного просмотра таблицы. При этом, если создаваемый индекс имеет признак уникальности, то это контролируется при создании B-дерева, и если уникальность нарушается, то операция не выполняется (т.е. индекс не создается). Из этого следует, что хотя создание индексов в динамике не запрещается, более эффективно создавать все индексы на данной таблице до ее заполнения.

ОПЕРАЦИИ УНИЧТОЖЕНИЯ

УНИЧТОЖЕНИЕ ОБЪЕКТОВ

Операции DROP TABLE, DROP LIST и DROP IMAGE могут быть выполнены в любой момент независимо от состояния объектов. Выполнение операции приводит к уничтожению соответствующего объекта и, вследствие этого, недействительности его идентификатора.

Операции DROP TABLE, DROP LIST и DROP IMAGE могут быть выполнены в любой момент независимо от состояния объектов. Выполнение операции приводит к уничтожению соответствующего объекта и, вследствие этого, недействительности его идентификатора.

ОПЕРАЦИИ МОДИФИКАЦИИ (1/2)

Группа операций модификации таблиц и списков включает операции вставки кортежа в таблицу или список (INSERT), удаления кортежа из таблицы (DELETE) и обновления кортежа в таблице (UPDATE).

ВСТАВКА КОРТЕЖЕЙ

Параметрами операции вставки кортежа являются идентификатор таблицы или списка и набор значений полей кортежа. Среди значений полей могут быть литеральные неопределенные значения NULL. При выполнении операции контролируется допустимость неопределенных значений При занесении кортежа в кластеризованную таблицу поиск места в сегменте под кортеж производится с использованием кластеризованного индекса: система пытается вставить кортеж в страницу данных, уже содержащую кортежи с теми же или близкими значениями полей кластеризации. При вставке кортежа в список он помещается в конец списка.

При занесении кортежа в таблицу производится коррекция всех индексов, определенных на этой таблице. Реально это выражается во вставке новой записи во все B-деревья индексов. Если индекс определен с атрибутом уникальности, то проверяется соблюдение этого условия, и если оно нарушено, операция вставки считается невыполненной. Из этого видно, что операция вставки кортежа тем более накладна, чем больше индексов определено для данной таблицы (это относится и к операциям удаления и модификации кортежей).

В результате успешного выполнения операции вставки кортежа в таблицу вырабатывается идентификатор нового кортежа, который выдается в качестве результата операции и может быть в дальнейшем использован как прямой параметр операций удаления и модификации кортежей таблицы.

ОПЕРАЦИИ МОДИФИКАЦИИ (2/2)

УДАЛЕНИЕ И МОДИФИКАЦИЯ КОРТЕЖЕЙ

Операции удаления и модификации кортежей допускаются только для кортежей таблиц. Естественно, что для выполнения этих операций необходимо идентифицировать соответствующий кортеж. В интерфейсе RSS допускаются два способа такой идентификации: с помощью идентификатора кортежа (явная адресация) и с использованием идентификатора открытого к этому времени сканирования. Первый вариант возможен, поскольку идентификатор кортежа сообщается как ответный параметр операции занесения кортежа в постоянную таблицу. При идентификации кортежа с помощью идентификатора сканирования имеется в виду кортеж, прочитанный с помощью последней операции NEXT. Если при такой идентификации выполняется операция DELETE или операция UPDATE, задевающая порядок сканирования (т.е. сканирование ведется по индексу и операция модификации меняет поле кортежа, входящее в состав ключа этого индекса), то текущий кортеж сканирования теряется, и его идентификатор нельзя использовать для идентификации кортежа, пока не будет выполнена следующая операция NEXT.

Единственным параметром операции DELETE является идентификатор кортежа или идентификатор сканирования. Параметры операции UPDATE включают, кроме этого, спецификацию изменяемых полей кортежа (список номеров полей и их новых значений). Среди значений могут находиться литеральные изображения неопределенных значений, если соответствующие поля таблицы допускают хранение неопределенных значений. При выполнении операции DELETE производится коррекция всех индексов, определенных на данной таблице.

BUILDLIST

МАКРООПЕРАЦИЯ BUILDLIST

Кроме описанных «атомарных» операций сканирования и модификации таблиц и списков, интерфейс RSS включает одну «макрооперацию» BUILDLIST, позволяющую за одно обращение к RSS построить список, отсортированный в соответствии со значениями заданных полей. Эта операция включает сканирование заданной таблицы или списка, создание нового списка, в который включаются указанные поля выбираемых кортежей, и сортировку построенного списка в соответствии со значениями указанных полей. Идентификатор заново построенного отсортированного списка является ответным параметром операции.

Параметрами операции BUILDLIST являются набор параметров для открытия сканирования (допускается любой способ сканирования), список номеров полей, составляющих кортежи нового списка, и список номеров полей, по которым нужно производить сортировку (как и в случае создания нового индекса, можно отдельно для каждого из этих полей указать требование к сортировке по возрастанию или убыванию значений данного поля). Отдельным параметром операции BUILDLIST является признак, в соответствии со значением которого в новом списке допускаются или не допускаются кортежи-дубликаты.

ДОБАВЛЕНИЕ ПОЛЯ В ТАБЛИЦУ

ДОБАВЛЕНИЕ ПОЛЯ В СУЩЕСТВУЮЩУЮ ТАБЛИЦУ

Операция RSS добавления поля к существующей таблице позволяет в динамике изменять схему таблицы. Параметрами операции CHANGE являются идентификатор существующей таблицы и спецификация нового поля (его тип). При выполнении операции изменяется только описатель данной таблицы в служебной таблице описателей таблиц. До выполнения первой операции UPDATE, затрагивающей новое поле таблицы, реально ни в одном кортеже таблицы память под новое поле выделяться не будет. По умолчанию значения нового поля во всех кортежах таблицы, в которые еще не производилось явное занесение значения, считаются неопределенными. Тем самым, ни для одного поля, динамически добавленного к существующей таблице, не может быть запрещено хранение неопределенных значений.

ПРОХОЖДЕНИЕ ТРАНЗАКЦИЙ (1/2)

Каждая операция RSS выполняется в пределах некоторой транзакции. Интерфейс RSS включает набор операций управления прохождением транзакции: начать транзакцию (BEGIN TRANSACTION), закончить транзакцию (END TRANSACTION), установить точку сохранения (SAVE) и выполнить откат до указанной точки сохранения или до начала транзакции (RESTORE).

ИДЕНТИФИКАТОР ТРАНЗАКЦИИ

При вызове любой операции функции RSS, кроме BEGIN TRANSACTION, должен указываться еще один параметр – идентификатор транзакции. Этот идентификатор и вырабатывается при выполнении операции BEGIN TRANSACTION, которая сама входных параметров не требует.

ОТКАТ ТРАНЗАКЦИИ

В любой точке транзакции до выполнения операции END TRANSACTION может быть выполнен откат данной транзакции, т.е. обратное выполнение всех изменений, произведенных в данной транзакции, и восстановление состояния позиций сканирования. Откат может быть произведен до начала транзакции (в этом случае о восстановлении позиций сканирования говорить бессмысленно) или до установленной ранее в транзакции точки сохранения.

ТОЧКА СОХРАНЕНИЯ

Точка сохранения устанавливается с помощью операции SAVE. При выполнении этой операции запоминаются состояние сканов данной транзакции, открытых к моменту выполнения SAVE, и координаты последней записи об изменениях в базе данных в журнале, произведенной от имени данной транзакции. Ответным параметром операции SAVE (а прямых параметров, кроме идентификатора транзакции, она не требует) является идентификатор точки сохранения.

ПРОХОЖДЕНИЕ ТРАНЗАКЦИЙ (2/2)

ОПЕРАЦИЯ RESTORE

Идентификатор точки сохранения в дальнейшем может быть использован как аргумент операции RESTORE, при выполнении которой производится восстановление базы данных по журналу (с использованием записей о ее изменениях от данной транзакции) до того состояния, в котором находилась база данных к моменту установки указанной точки сохранения. Кроме того, по локальной информации в оперативной памяти, привязанной к транзакции, восстанавливается состояние ее сканов. Откат к началу транзакции инициируется также вызовом операции RESTORE, но с указанием некоторого предопределенного идентификатора точки сохранения.

При выполнении своих транзакций пользователи System R изолированы один от другого, т.е. не ощущают того, что система функционирует в многопользовательском режиме. Это достигается за счет наличия в RSS механизма неявной синхронизации. До конца транзакции никакие изменения базы данных, произведенные в пределах этой транзакции, не могут быть использованы в других транзакциях (попытка использования таких данных приводит к временным синхронизационным блокировкам этих транзакций). При выполнении операции END TRANSACTION происходит "фиксация" изменений, произведенных в данной транзакции, т.е. они становятся видимыми в других транзакциях. Реально это означает снятие синхронизационных блокировок с объектов базы данных, изменявшихся в транзакции. Из этого следует, что после выполнения END TRANSACTION невозможны индивидуальные откаты данной транзакции. RSS просто делает недействительным идентификатор данной транзакции, и после выполнения операции окончания транзакции отвергает все операции с таким идентификатором.

ЯВНАЯ СИНХРОНИЗАЦИЯ

ОПЕРАЦИЯ ЯВНОЙ СИНХРОНИЗАЦИИ LOCK

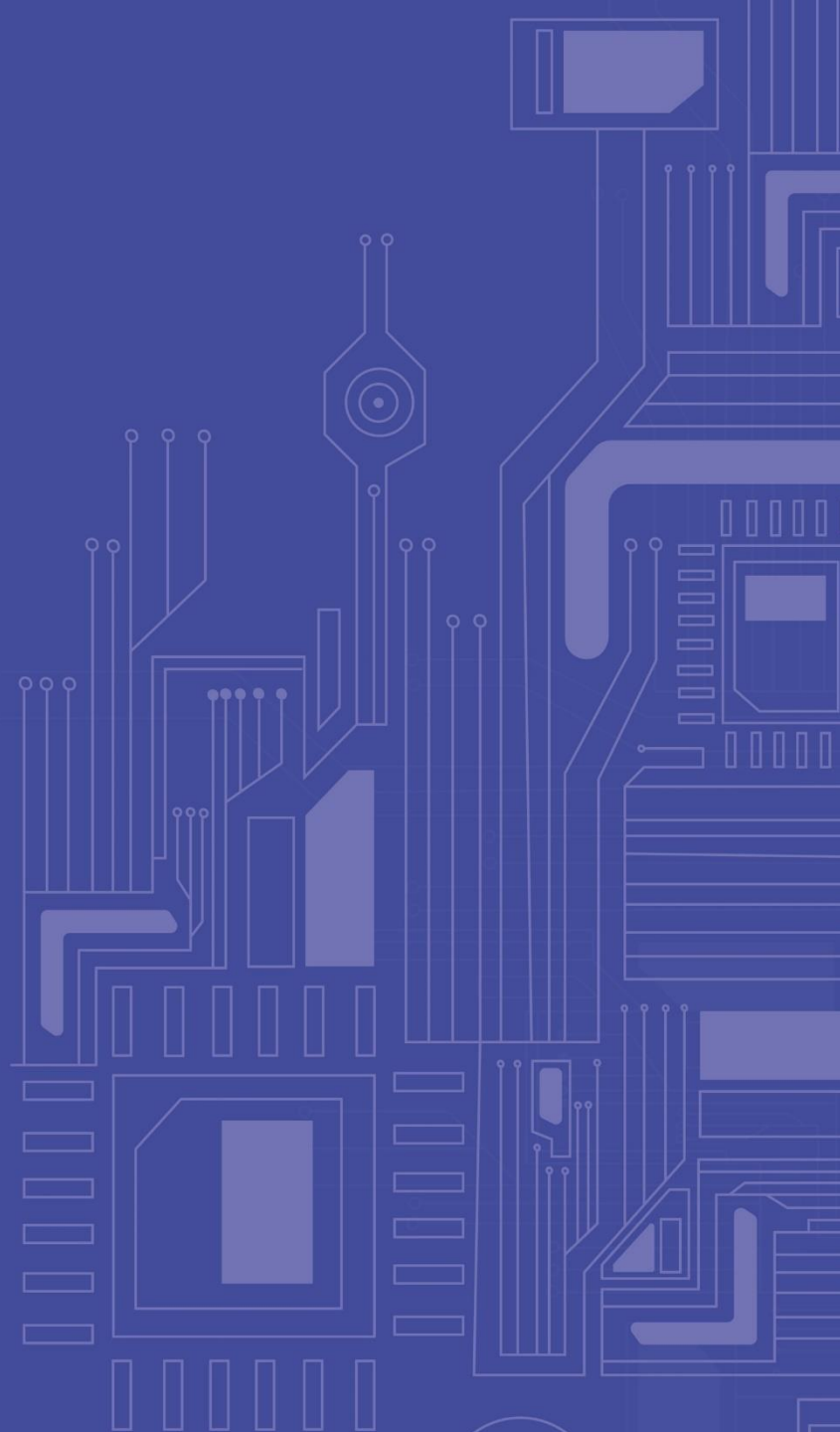
Эта операция позволяет установить явную синхронизационную блокировку указанной таблицы (параметром операции является идентификатор таблицы). Выполнение операции LOCK гарантирует, что никакая другая транзакция до конца данной не сможет изменить эту таблицу (вставить в нее новый кортеж, удалить или модифицировать существующий), если установлена блокировка в режиме чтения, или даже прочитать любой кортеж этой таблицы, если установлена монопольная блокировка.

На самом деле, логически эта операция избыточна, т.е. если бы ее не было, можно было бы реализовать SQL с использованием оставшейся части операций. Операция LOCK введена в интерфейс RSS для возможности оптимизации выполнения запросов.

ПРИМЕР ЯВНОЙ ВЫГОДЫ ОТ LOCK

Интерфейс RSS является покортежным. Следовательно, и информация для синхронизации носит достаточно узкий характер. В то же время, на уровне SQL имеется более полная информация. Например, если обрабатывается предложение SQL DELETE FROM table_name, то известно, что будут удалены все кортежи указанной таблицы. Понятно, что как бы не реализовывался механизм синхронизации в RSS, в данном случае выгоднее сообщить сразу, что изменения касаются всей таблицы.

ОБЩИЕ ПРИНЦИПЫ ОРГАНИЗАЦИИ ДАННЫХ ВО ВНЕШНЕЙ ПАМЯТИ В SQL-ОРИЕНТИРОВАННЫХ СУБД



ОСОБЕННОСТИ SQL-ОРИЕНТИРОВАННЫХ СУБД

- **Наличие двух уровней системы**: уровня непосредственного управления данными во внешней памяти (а также обычно управления буферами оперативной памяти, управления транзакциями и журнализацией изменений БД) и языкового уровня (уровня, реализующего язык SQL). При такой организации подсистема нижнего уровня должна поддерживать во внешней памяти набор базовых структур, конкретная интерпретация которых входит в число функций подсистемы верхнего уровня.
- **Поддержка таблиц-каталогов**. Информация, связанная с именованием объектов базы данных и их конкретными свойствами (например, структура ключа индекса), поддерживается подсистемой языкового уровня. С точки зрения структур внешней памяти таблица-каталог ничем не отличается от обычной таблицы базы данных.
- **Регулярность структур данных**. Поскольку основным объектом модели данных SQL является плоская таблица, основной набор объектов внешней памяти может иметь очень простую регулярную структуру.
- Необходимость обеспечения возможности эффективного выполнения операторов языкового уровня как над одной таблицей (простые селекция и проекция), так и над несколькими таблицами (наиболее распространено и трудоемко соединение нескольких таблиц). Для этого во внешней памяти должны поддерживаться дополнительные «управляющие» структуры – **индексы**.
- Наконец, для выполнения требования надежного хранения баз данных необходимо поддерживать избыточность хранения данных, что обычно реализуется в виде **журнала изменений базы данных**.

РАЗНОВИДНОСТИ ОБЪЕКТОВ ВО ВНЕШНЕЙ ПАМЯТИ

1. СТРОКИ ТАБЛИЦ – основная часть базы данных, большей частью непосредственно видимая пользователям;
2. УПРАВЛЯЮЩИЕ СТРУКТУРЫ – ИНДЕКСЫ, создаваемые по инициативе пользователя (администратора) или верхнего уровня системы из соображений повышения эффективности выполнения запросов и обычно автоматически поддерживаемые нижним уровнем системы;
3. ЖУРНАЛЬНАЯ ИНФОРМАЦИЯ, поддерживаемая для удовлетворения потребности в надежном хранении данных;
4. СЛУЖЕБНАЯ ИНФОРМАЦИЯ, поддерживаемая для удовлетворения внутренних потребностей нижнего уровня системы (например, информация о свободной памяти).

ХРАНЕНИЕ ТАБЛИЦ

ПО КОРТЕЖАМ (СТРОКАМ)

Наиболее распространенным является покортежное хранение таблиц (единицей физического хранения является кортеж).

Это обеспечивает быстрый доступ к целому кортежу.

Во внешней памяти дублируются общие значения разных кортежей одной таблицы и, могут потребоваться лишние обмены с внешней памятью, если нужна часть кортежа.

ПО СТОЛБЦАМ

Альтернативным (менее распространенным) подходом является хранение таблицы по столбцам, т.е. единицей хранения является столбец таблицы с исключенными дубликатами.

При такой организации суммарно в среднем тратится меньше внешней памяти, поскольку дубликаты значений не хранятся; за один обмен с внешней памятью в общем случае считывается больше полезной информации. Дополнительным преимуществом является возможность использования значений столбца таблицы для оптимизации выполнения операций соединения.

При этом требуются существенные дополнительные действия для сборки целого кортежа (или его части).

ХРАНЕНИЕ КОРТЕЖЕЙ (1/2)

Построчное хранение таблиц гораздо более распространено. В основном принципы хранения соответствуют System R.

- Каждый кортеж обладает уникальным идентификатором (tid), не изменяемым во все время существования кортежа и позволяющим выбрать кортеж в основную память не более чем за два обращения к внешней памяти.
- Обычно каждый кортеж хранится целиком в одной странице. Из этого следует, что максимальная длина кортежа любой таблицы ограничена размерами страницы. Возникает вопрос: как быть с «длинными» данными, которые в принципе не помещаются в одной странице? Применяется несколько методов. Наиболее простым решением является хранение таких данных в отдельных (вне базы данных) файлах с заменой «длинного» данного в кортеже на имя соответствующего файла. В некоторых системах такие данные хранились внутри базы данных в отдельном наборе страниц внешней памяти, связанном физическими ссылками. Оба эти решения сильно ограничивают возможность работы с длинными данными (как, например, удалить несколько байт из середины 2-мегабайтной строки?). В настоящее время все чаще используется метод, предложенный много лет тому назад в проекте Exodus , когда «длинные» данные организуются в виде B-деревьев последовательностей байт.
- Как правило, в одной странице данных хранятся кортежи только одной таблицы. Существуют, однако, варианты с возможностью хранения в одной странице кортежей нескольких таблиц. Это вызывает некоторые дополнительные расходы по части служебной информации (при каждом кортеже нужно хранить информацию о соответствующей таблице), но зато иногда позволяет резко сократить число обменов с внешней памятью при выполнении соединений.

ХРАНЕНИЕ КОРТЕЖЕЙ (2/2)

- Изменение схемы хранимой таблицы с добавлением нового поля не вызывает потребности в физической реорганизации таблицы. Достаточно лишь изменить информацию в описателе таблицы и расширять кортежи только при занесении информации в новое поле.
- Поскольку таблицы могут содержать неопределенные значения, необходима соответствующая поддержка на уровне хранения. Обычно это достигается путем хранения соответствующей шкалы при каждом кортеже, который в принципе может содержать неопределенные значения.
- Проблема распределения памяти в страницах данных связана с проблемами синхронизации и журнализации и не всегда тривиальна. Например, если в ходе выполнения транзакции некоторая страница данных опустошается, то ее нельзя перевести в статус свободных страниц до конца транзакции, поскольку при откате транзакции удаленные при прямом выполнении транзакции и восстановленные при ее откате кортежи должны получить те же самые идентификаторы.
- Распространенным способом повышения эффективности СУБД является кластеризация таблицы по значениям одного или нескольких столбцов. Полезной для оптимизации соединений является совместная кластеризация нескольких таблиц.
- С целью использования возможностей распараллеливания обменов с внешней памятью иногда применяют схему декластеризованного хранения таблиц: кортежи с общим значением столбца декластеризации размещают на разных дисковых устройствах, обмены с которыми можно выполнять параллельно.

ИНДЕКСЫ

Как бы не были организованы индексы в конкретной СУБД, их основное назначение состоит в обеспечении эффективного прямого доступа к кортежу таблицы по ключу. Одна организация индекса отличается от другой, главным образом, в способе поиска ключа с заданным значением.

- Обычно индекс определяется для одной таблицы, и ключом является значение ее поля (возможно, составного). Если ключом индекса является возможный ключ таблицы, то индекс должен обладать свойством уникальности, т.е. не содержать дубликатов ключа. На практике ситуация выглядит обычно противоположно: при объявлении первичного ключа таблицы автоматически заводится **уникальный индекс**, а единственным способом объявления возможного ключа, отличного от первичного, является явное создание уникального индекса. Это связано с тем, что для проверки сохранения свойства уникальности возможного ключа, так или иначе, требуется индексная поддержка.
- Поскольку при выполнении многих операций уровня SQL требуется сортировка кортежей таблиц в соответствии со значениями некоторых полей, полезным свойством индекса является обеспечение **последовательного просмотра кортежей таблицы в заданном диапазоне значений ключа** в порядке возрастания или убывания значений ключа.
- Одним из способов оптимизации выполнения эквисоединения таблиц является организация так называемых **мультииндексов** для нескольких таблиц, обладающих общими атрибутами. Любой из этих атрибутов (или их набор) может выступать в качестве ключа мультииндекса. Значению ключа сопоставляется набор кортежей всех связанных мультииндексом таблиц, значения выделенных атрибутов которых совпадают со значением ключа.


ВСЁ!