Современные базы данных – это сложные многофункциональные программные системы, работающие в открытой распределенной среде. Они уже сегодня доступны для использования в деловой сфере и выступают не просто в качестве технических и научных решений, но как завершенные продукты, предоставляющие разработчикам мощные средства управления данными и богатый инструментарий для создания прикладных программ и систем.

Администрирование базами данных предусматривает выполнение функций, направленных на обеспечение надежного и эффективного функционирования системы баз данных, адекватности содержания базы данных информационным потребностям пользователей, отображения в базе данных актуального состояния предметной области.

Необходимость персонала, обеспечивающего администрирование данными в системе БД в процессе функционирования, является следствием централизованного характера управления данными в таких системах, постоянно требующего поиска компромисса между противоречивыми требованиями к системе в социальной пользовательской среде. Хотя такая необходимость и признавалась на ранних стадиях развития технологии баз данных, четкое понимание и структуризация функций персонала, занятого администрированием, сложилось только вместе с признанием многоуровневой архитектуры СУБД.

**Управление данными в базах данных**

Непосредственное управление данными во внешней памяти. Эта функция включает обеспечение необходимых структур внешней памяти как для хранения непосредственных данных, входящих в БД, так и для служебных целей, например, для убыстрения доступа к данным в некоторых случаях (обычно для этого используются индексы). В некоторых реализациях СУБД активно используются возможности существующих файловых систем, в других работа производится вплоть до уровня устройств внешней памяти. Но подчеркнем, что в развитых СУБД пользователи в любом случае не обязаны знать, использует ли СУБД файловую систему, а если использует, то как организованы в ней файлы. В частности, СУБД поддерживает собственную систему именования объектов БД (это очень важно, поскольку имена объектов базы данных соответствуют именам объектов предметной области).

Существует множество различных способов организации внешней памяти баз данных. Как и все решения, принимаемые при организации баз данных, конкретные методы организации внешней памяти необходимо выбирать в тесной связи со всеми остальными решениями.

Управление буферами оперативной памяти. СУБД обычно работают с БД значительного размера; по крайней мере этот размер обычно существенно превышает доступный объем оперативной памяти. Понятно, если при обращении к любому элементу данных будет производиться обмен с внешней памятью, то вся система будет работать со скоростью устройства внешней памяти. Единственным же способом реального увеличения этой скорости является буферизация данных в оперативной памяти. И даже если операционная система производит общесистемную буферизацию (как в случае ОС UNIX), этого недостаточно для целей СУБД, которая располагает гораздо большей информацией о полезности буферизации той или иной части БД. Поэтому в развитых СУБД поддерживается собственный набор буферов оперативной памяти с собственной дисциплиной замены буферов. При управлении буферами основной памяти приходится разрабатывать и применять согласованные алгоритмы буферизации, журнализации и синхронизации. Заметим, что существует отдельное направление СУБД, которые ориентированы на постоянное присутствие в оперативной памяти всей БД. Это направление основывается на предположении, что в предвидимом будущем объем оперативной памяти компьютеров сможет быть настолько велик, что позволит не беспокоиться о буферизации. Пока эти работы находятся в стадии исследований.

Управление транзакциями. Транзакция – это последовательность операций над БД, рассматриваемых СУБД как единое целое. Либо транзакция успешно выполняется, и СУБД фиксирует (COMMIT) изменения БД, произведенные ею, во внешней памяти, либо ни одно из этих изменений никак не отражается в состоянии БД. Понятие транзакции необходимо для поддержания логической целостности БД.

Таким образом, поддержание механизма транзакций – обязательное условие даже однопользовательских СУБД (если, конечно, такая система заслуживает названия СУБД). Но понятие транзакции гораздо существеннее во многопользовательских СУБД. То свойство, что каждая транзакция начинается при целостном состоянии БД и оставляет это состояние целостным после своего завершения, делает очень удобным использование понятия транзакции как единицы активности пользователя по отношению к БД. При соответствующем управлении параллельно выполняющимися транзакциями со стороны СУБД каждый пользователь может в принципе ощущать себя единственным пользователем СУБД (на самом деле, это несколько идеализированное представление, поскольку пользователи многопользовательских СУБД порой могут ощутить присутствие своих коллег).

С управлением транзакциями в многопользовательской СУБД связаны важные понятия сериализации транзакций и сериального плана выполнения смеси транзакций. Под сериализаций параллельно выполняющихся транзакций понимается такой порядок планирования их работы, при котором суммарный эффект смеси транзакций эквивлентен эффекту их некоторого последовательного выполнения. Сериальный план выполнения смеси транзакций – это такой способ их совместного выполнения, который приводит к сериализации транзакций. Понятно, что если удается добиться действительно сериального выполнения смеси транзакций, то для каждого пользователя, по инициативе которого образована транзакция, присутствие других транзакций будет незаметно (если не считать некоторого замедления работы для каждого пользователя по сравнению с однопользовательским режимом).

Существует несколько базовых алгоритмов сериализации транзакций. В централизованных СУБД наиболее распространены алгоритмы, основанные на синхронизационных захватах объектов БД. При использовании любого алгоритма сериализации возможны ситуации конфликтов между двумя или более транзакциями по доступу к объектам БД. В этом случае для поддержания сериализации необходимо выполнить откат (ликвидировать все изменения, произведенные в БД) одной или более транзакций. Это один из случаев, когда пользователь многопользовательской СУБД может реально (и достаточно неприятно) ощутить присутствие в системе транзакций других пользователей.

**Управление безопасностью в СУБД**

Системы управления базами данных стали основным инструментом, обеспечивающим хранение больших массивов информации. Современные информационные приложения опираются, как уже говорилось, в первую очередь, на многопользовательские СУБД. В этой связи пристальное внимание в настоящее время уделяется проблемам обеспечения информационной безопасности, которая определяет степень безопасности организации, учреждения в целом.

Под информационной безопасностью понимают защищенность информации от случайных и преднамеренных воздействий естественного или искусственного характера, чреватых нанесением ущерба владельцам или пользователям информации.

Защите подлежат не только данные; в базе данных нарушения в защите могут повлиять на другие части системы, что повлечет за собой разрушение и базы данных. Поэтому защита баз данных охватывает и оборудование, и программное обеспечение, и персонал, и, собственно, данные.

Таким образом, защита баз данных предусматривает предотвращение любых преднамеренных и непреднамеренных угроз с использованием компьютерных и некомпьютерных средств контроля. Следует также защищать: современные информационные системы; глобальную связанность (выход в Internet); разнородность (различные платформы); технологию «клиент-сервер».

Рассмотрим основные программно-технические меры, применение которых позволит решить некоторые из вышеперечисленных проблем: аутентификация и идентичность; управление доступом; поддержка целостности; протоколирование и аудит; защита коммуникаций между клиентом и сервером; отражение угроз, специфичных для СУБД.

Аутентификация и идентичность. Проверка подлинности пользователя приложений базы данных чаще всего осуществляется либо через соответствующие механизмы операционной системы, либо через определенный SQL-оператор: пользователь идентифицируется своим именем, а средством аутентификации служит пароль. Авторизация –предоставление прав (привилегий), позволяющих владельцу иметь законный доступ к объектам базы данных. Аутентификация – механизм определения того, является ли пользователь тем, за кого он себя выдает. Эта процедура позволяет организовать контролируемый доступ к информационной системе (пользователь – идентификатор – пароль). Пароль – наиболее распространенный метод аутентификации, но он не дает абсолютной гарантии, что пользователь является именно тем, за кого себя выдает. При использовании такого подхода создаются значительные сложности для повторных проверок и исключает подобные проверки перед каждой транзакцией. Средства аутентификации на основе личных карточек или эквивалентного механизма дали бы приложению большую свободу в реализации контроля за подлинностью пользователей.

Управление доступом. После получения права доступа к СУБД пользователь автоматически получает привилегии, связанные с его идентификатором. Это может относиться к процедурам доступа к объектам базы данных, к операциям над данными. Для основных объектов базы данных могут быть построены таблицы, в которых указывается набор действий, доступных каждому пользователю системы.

Каждому возможному действию над данными таблицы ставится в соответствие двоичное значения, общий результат возможных операция получается путем суммирования набранных пользователем значений.

Привилегии в СУБД могут быть разделены на две категории: привилегии безопасности и привилегии доступа. Привилегии безопасности позволяют выполнять административные действия, привилегии доступа определяют права доступа субъектов к определенным объектам.

До выполнения процедуры присваивания привилегий их необходимо создать.

Привилегии можно подразделить в соответствии с видами объектов, к которым они относятся: таблицы и представления, процедуры, базы данных сервер базы данных. Применительно к таблицам могут быть определены следующие права доступа: право на выборку, удаление, обновление, добавление, право на использование внешних ключей, ссылающихся на данную таблицу. По умолчанию пользователь не имеет никаких прав доступа ни таблицам, ни к представлениям. По отношению к процедурам можно предоставить право на их выполнение, однако при этом не указываются привилегии на право доступа к объектам, обрабатываемым процедурами. Это позволяет выделять неконтролируемый доступ для выполнения строго определенных операций над данными. По отношению к базе данных выделяемые права на самом деле являются запретительными: ограничение на число операций ввода/вывода строк, число строк, возвращаемым одним запросом.

Оператор позволяет реализовать следующие виды ограничений доступа:

операционные ограничения (за счет прав доступа операторов выборки, вставки, удаления и обновления, применяемые ко всем или отдельным столбцам таблицы);

ограничения по значениям (за счет механизма представлений);

ограничения на ресурсы (за счет привилегий к базам данных).

СУБД предоставляет специфическое средство управление доступом – представления, которые позволяют делать видимыми только отдельные столбцы базовых таблиц. В результате выполнения запроса к представлению, а не к таблице может быть возвращена таблица из нуля строк, а не код ответа, свидетельствующий о нарушении прав доступа. Это важно, поскольку этот ответ лишает возможности поиска ответа другим путем, например, через анализ кодов, ответов, возвращаемых после обработки SQL-запросов.

Управление доступом базируется на реализации следующего минимального набора действий: произвольное управление доступом; обеспечение безопасности повторного использования объектов; использование меток безопасности; принудительное управление доступом.

Произвольное управление доступом – метод ограничения доступа к объектам, основанный на учете личности субъекта или групп, в которую субъект входит. Группа – это именованная совокупность пользователей; объединение субъектов в группы облегчает процесс администрирования данных, и строится на основе формальной структуры организации. Эта технология обеспечивает владельцу объекта (представления, сервера базы данных, процедуры, таблицы) передачу по своему усмотрению привилегий другому лицу. Этим лицом в данной ситуации может выступать субъект-пользователь, группа пользователей и такой возможный носитель привилегий как роль.

**Резервное копирование и восстановление БД**

Поскольку данные, хранимые компьютерными средствами подвержены потерям и повреждениям, вызываемым разными событиями, важно обеспечить средства восстановления данных. Приведение базы данных точно в то состояние, которое существовало перед отказом не всегда возможно, но процедуры восстановления базы данных могут привести ее в состояние, существовавшее незадолго до отказа. Восстановление базы данных применяется при повреждениях, не позволяющих пользователю открыть базу данных или работать с ней. Одной из причин повреждения базы данных может быть воздействие компьютерных вирусов или наличие дефектов (физических или логических) на диске. Повреждение базы данных может проявляться при попытке пользователя открыть, сжать, зашифровать или дешифровать БД. При наличии повреждений базы данных, созданной в СУБД Access, для ее восстановления необходимо: § закрыть базу данных. При работе в сети необходимо убедиться, что другие пользователи закрыли базу данных; § создать резервную копию базы данных; § выполнить команду Сервис / Служебные данные / Восстановить; § указать имя папки и файла восстанавливаемой базы данных. Следующим способом обслуживания базы данных является резервное копирование. Основным назначением резервного копирования базы данных является предотвращение потери информации и реализуется путем одноразового или периодического копирования и архивирования наиболее ценной информации. Резервное копирование заключается в создании резервной копии базы данных и размещении на вспомогательных носителях информации: жестких дисках, дискетах, накопителях на оптических дисках, магнитных лентах. Резервная копия может быть точной копией исходной БД или сжатой (архивной) копией. Сжатие может осуществляться аппаратно или программно. Аппаратное сжатие предпочтительно с точки зрения временных затрат на эту операцию, но увеличивает стоимость аппаратной части. Резервное копирование может осуществляться во время работы с БД (режим online) или в другое время. Копия может создаваться по инициативе оператора, либо автоматически в заданное время путем запуска соответствующей утилиты. При организации резервного копирования администратор решает такие вопросы как: § какие устройства выбрать для резервного копирования; § когда и с какой частотой выполнять резервное копирование. Важно периодически проверять корректность выполненного резервирования информации путем пробного восстановления. Репликация (replication) – создание специальных копий (реплик) базы данных, с которыми пользователи могут работать одновременно на разных рабочих станциях.

**Журнал транзакций. Восстановление через откат-накат**

Журнализация. Одно из основных требований к СУБД – надежное хранение данных во внешней памяти. Под надежностью хранения понимается то, что СУБД должна быть в состоянии восстановить последнее согласованное состояние БД после любого аппаратного или программного сбоя. Обычно рассматриваются два возможных вида аппаратных сбоев: так называемые мягкие сбои, которые можно трактовать как внезапную остановку работы компьютера (например, аварийное выключение питания), и жесткие сбои, характеризуемые потерей информации на носителях внешней памяти. Примерами программных сбоев могут быть аварийное завершение работы СУБД (из-за ошибки в программе или некоторого аппаратного сбоя) или аварийное завершение пользовательской программы, в результате чего некоторая транзакция остается незавершенной. Первую ситуацию можно рассматривать как особый вид мягкого аппаратного сбоя; при возникновении последней требуется ликвидировать последствия только одной транзакции.

Но в любом случае для восстановления БД нужно располагать некоторой дополнительной информацией. Другими словами, поддержание надежного хранения данных в БД требует избыточности хранения данных, причем та их часть, которая используется для восстановления, должна храниться особо надежно. Наиболее распространенный метод поддержания такой избыточной информации – ведение журнала изменений БД.

Журнал – это особая часть БД, недоступная пользователям СУБД и поддерживаемая особо тщательно (иногда поддерживаются две копии журнала, располагаемые на разных физических дисках), в которую поступают записи обо всех изменениях основной части БД. В разных СУБД изменения БД журнализуются на разных уровнях: иногда запись в журнале соответствует некоторой логической операции изменения БД (например, операции удаления строки из таблицы реляционной БД), а порой запись соответствует минимальной внутренней операции модификации страницы внешней памяти. В некоторых системах одновременно используются оба подхода.

Во всех случаях придерживаются стратегии «упреждающей» записи в журнал (так называемого протокола Write Ahead Log – WAL). Грубо говоря, эта стратегия заключается в том, что запись об изменении любого объекта БД должна попасть во внешнюю память журнала раньше, чем измененный объект попадет во внешнюю память основной части БД. Известно, если в СУБД корректно соблюдается протокол WAL, то с помощью журнала можно решить все проблемы восстановления БД после любого сбоя.

Самая простая ситуация восстановления – индивидуальный откат транзакции. Строго говоря, для этого не требуется общесистемный журнал изменений БД. Достаточно для каждой транзакции поддерживать локальный журнал операций модификации БД, выполненных в этой транзакции, и производить откат транзакции выполнением обратных операций, следуя от конца локального журнала. В некоторых СУБД так и делают, но в большинстве систем локальные журналы не поддерживают, а индивидуальный откат транзакции выполняют по общесистемному журналу, для чего все записи от одной транзакции связывают обратным списком (от конца к началу).

При мягком сбое во внешней памяти основной части БД могут находиться объекты, модифицированные транзакциями, не закончившимися к моменту сбоя, и могут отсутствовать объекты, модифицированные транзакциями, которые к моменту сбоя успешно завершились (по причине использования буферов оперативной памяти, содержимое которых при мягком сбое пропадает). При соблюдении протокола WAL во внешней памяти журнала должны гарантированно находиться записи, относящиеся к операциям модификации обоих видов объектов. Целью процесса восстановления после мягкого сбоя является состояние внешней памяти основной части БД, которое возникло бы при фиксации во внешней памяти изменений всех завершившихся транзакций и которое не содержало бы никаких следов незаконченных транзакций. Чтобы этого добиться, сначала производят откат незавершенных транзакций (undo), а потом повторно воспроизводят (redo) те операции завершенных транзакций, результаты которых не отображены во внешней памяти. Этот процесс содержит много тонкостей, связанных с общей организацией управления буферами и журналом. Более подробно мы рассмотрим это в соответствующей лекции.

Для восстановления БД после жесткого сбоя используют журнал и архивную копию БД. Грубо говоря, архивная копия – это полная копия БД к моменту начала заполнения журнала (имеется много вариантов более гибкой трактовки смысла архивной копии). Конечно, для нормального восстановления БД после жесткого сбоя необходимо, чтобы журнал не пропал. Как уже отмечалось, к сохранности журнала во внешней памяти в СУБД предъявляются особо повышенные требования. Тогда восстановление БД состоит в том, что исходя из архивной копии по журналу воспроизводится работа всех транзакций, которые закончились к моменту сбоя. В принципе можно даже воспроизвести работу незавершенных транзакций и продолжить их работу после конца восстановления. Однако в реальных системах это обычно не делается, поскольку процесс восстановления после жесткого сбоя является достаточно длительным.

Источики:

<https://smekni.com/a/115319/administrirovanie-bazy-dannykh/>

<https://zinref.ru/000_uchebniki/02800_logika/011_lekcii_raznie_33/2093.htm#:~:text=%D0%90%D0%B4%D0%BC%D0%B8%D0%BD%D0%B8%D1%81%D1%82%D1%80%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%20%D0%B1%D0%B0%D0%B7%D0%B0%D0%BC%D0%B8%20%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85%20%D0%BF%D1%80%D0%B5%D0%B4%D1%83%D1%81%D0%BC%D0%B0%D1%82%D1%80%D0%B8%D0%B2%D0%B0%D0%B5%D1%82%20%D0%B2%D1%8B%D0%BF%D0%BE%D0%BB%D0%BD%D0%B5%D0%BD%D0%B8%D0%B5,%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85%20%D0%B0%D0%BA%D1%82%D1%83%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B3%D0%BE%20%D1%81%D0%BE%D1%81%D1%82%D0%BE%D1%8F%D0%BD%D0%B8%D1%8F%20%D0%BF%D1%80%D0%B5%D0%B4%D0%BC%D0%B5%D1%82%D0%BD%D0%BE%D0%B9%20%D0%BE%D0%B1%D0%BB%D0%B0%D1%81%D1%82%D0%B8>.

<https://www.opennet.ru/docs/RUS/db_admin/>

http://www.bseu.by/it/tohod/lekcii9.htm