|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | **База данных Access. Особенности работы в многопользовательском режиме** |  | |
| После создания приложения Access оно покидает среду разработки и направляется для использования заказчиками. При этом могут возникнуть проблемы, связанные с работой этого приложения в многополь­зовательской среде. Наиболее серьезной из этих проблем является разрешение конфликтов, возникающих при одновременном обращении нескольких пользователей к записям базы данных Access |
| **Конфликты доступа**  Важно помнить, что работа с базой данных в многопользовательской среде может вызывать пробле­мы, связанные с блокировкой данных и конфликтами доступа к ним. Подавление сообщений об ошиб­ках, недостаточное внимание к подобным вопросам либо надежда на благоприятное стечение обстоятельств не решает проблему. Несмотря на кажущуюся сложность работы в многопользовательской среде, понять механизм действия блокировки данных и способ обслуживания механизмом Jet нескольких пользователей достаточно просто. Пренебрежение подобными вопросами, как правило, приводит к возникновению бо­лее сложных проблем, затрагивающих пользователей, клиентов и влияющих на репутацию разработчика. Если для приложения не предусмотрено эффективное решение вопросов работы в многопользовательской среде, при работе с ним неизбежно будут возникать следующие проблемы:  • Новые записи не сохраняются. После ввода информации пользователь обнаруживает, что в базе данных она не появилась. Если подобная ошибка не повторяется, это говорит не об отсутствии проблемы, а о ненадежности приложения.  • Изменения существующих записей не сохраняются. Пользователь может даже не заметить, что вне­сенные им изменения не сохранились. Однако в дальнейшем может оказаться, что либо таблица ито­говых данных отсутствует, либо в инвентарном списке слишком много элементов, или может случиться так, что заказ важного клиента будет отправлен по неверному адресу. Подобные серьез­ные проблемы вполне могут оказаться не выявленными вовремя. Как правило, от них страдают посторонние люди.  • Пользователи получают невразумительные сообщения о невозможности обеспечить доступ к данным. Хотя последствия подобного рода неприятностей не столь драматичны, как в предыдущих случаях, для пользователя будет сильным разочарованием необходимость работы с приложением, которое не может справиться даже с такой несложной проблемой.  Проблемы работы приложений Access в многопользовательской среде связаны не только с установкой и снятием блокировок записей. Поскольку такое приложение существует в виде единого файла и, по край­ней мере, часть его форм может быть непосредственно связана с данными, любой обзор проблем при­менения в многопользовательском режиме должен охватывать вопросы работы на уровне файла, аспекты конфигурирования, а также технологий разработки интерфейсов и свойств запросов и форм. Типичное приложение Access требует определенного сочетания различных приемов работы в распределенном режи­ме, поскольку в разных частях приложения подобные проблемы решаются различными способами.  **Конфигурация**  Для обслуживания нескольких пользователей приложение Access необходимо на файловом уровне кон­фигурировать по-разному. Каждый способ имеет свои преимущества и недостатки, некоторые из них пе­речисляются ниже.  • Сетевое размещение. В данной конфигурации единый MDB-файл располагается на сетевом серве­ре, и пользователи получают доступ к базе данных при обращении к серверу. Данные и выполня­емые модули могут содержаться в едином MDB-файле либо размещаться на файловом сервере в виде нескольких отдельных файлов. Преимуществом данной конфигурации является простота поддержки, поскольку при необходимости в обновлении нуждается лишь выполняемый файл. Однако, поскольку все формы, отчеты, модули, запросы, ЕХЕ-файлы Access, а также все библиотеки DLL и т.п. должны передаваться по сети на рабочую станцию, сетевой трафик неоправданно возрастает, а производительность значительно снижается. Вероятно, в подобных конфигурациях следует исполь­зовать связанные формы. Далее рассматриваются проблемы связывания форм с данными и возни­кающие при этом конфликты доступа.  • Разделенная база данных с размещенными в сети данными. Такая конфигурация по традиции на­зывается конфигурацией удаленной базы данных (отметим, что значение слова "удаленная" в чрез­вычайно динамичную эпоху Internet постепенно меняется и вскоре может устареть), поскольку данные отделены от выполняемого модуля или программного кода, хотя механизм баз данных и остается локальным. В отличие от конфигурации клиент-сервер, механизм баз данных Access на пользовательском ПК получает, обрабатывает, блокирует и снимает блокировку с данных, находя­щихся в MDB-файле на сетевом сервере. Работа в такой конфигурации зависит от механизмов баз данных одновременно работающих пользователей, а также от возможностей файлового сервера, касающихся поддержания сетевого графика. До настоящего времени при размещении приложений баз данных Access предпочтение отдают именно этому методу. Его преимуществом является высо­кая производительность и управляемость при корректном использовании. Поскольку при размеще­нии данных в сети по каналам связи передаются только они, сетевой трафик значительно снижается. Основной недостаток данной конфигурации заключается в том, что на каждом клиентском ПК необходимо устанавливать Access и выполняемый MDE- (скомпилированный вариант базы данных MDB) либо MDB-файл, что осложняет поддержку приложения. Тем не менее, существуют спосо­бы решения подобной проблемы.  • Репликация. При использовании схемы репликации пользователи совместно обрабатывают данные, хотя данные на самом деле не являются общими, как это имеет место в схемах сетевого распре­деления или в разделенных базах данных. В схеме репликации каждый пользователь или небольшая группа пользователей имеет собственную копию данных, которые посредством механизма репликации Jet синхронизируются с другой базой или базами данных. Одно из преимуществ такой схемы, когда каждому пользователю предоставляется копия данных, состоит в полном исключении проблемы блокировки, но вместо них возникают проблемы репликации, степень сложности которых являет­ся практически такой же. Другим значительным преимуществом репликации является возможность асинхронного доступа к данным для отключенных от сети пользователей. Вместе с тем существует еще один недостаток такой схемы: при совместном использовании источника данных даже небольшой группой пользователей все же существует возможность возникновения как конфликтов доступа, так и проблем репликации.  • Конфигурация клиент-сервер. В Access 2000 появилась новая возможность создания клиент-сервер­ных приложений на базе проекта Microsoft Access. В такой конфигурации удаленными являются как данные, так и механизм баз данных. Если данными управляет SQL Server, Oracle или какой-либо иной сервер баз данных, расположенный на центральном компьютере, он также решает вопросы блокировки и проблемы работы в многопользовательской среде. Это не означает, что разработчик избавлен от необходимости решения всех связанных с ними задач, просто ему приходится иметь дело с иными наборами свойств, возможностей и правил. Основными преимуществами такой конфигу­рации являются высокая производительность, стабильность, возможность обслуживания большого количества пользователей и выполнения множества задач. Наибольший недостаток данной конфи­гурации состоит в высокой стоимости и значительной сложности.  В данной главе рассматриваются вопросы, которые являются общими для сетевых конфигураций: схе­мы разделенной базы данных и реализации архитектуры клиент-сервер. О репликации рассказывается в главе 22.  При использовании Access существует ряд установок на уровне базы данных (для форм, запросов, наборов записей) и на уровне выполняемого кода, которые требуют взаимного координирования в це­лях обеспечения надежного обслуживания приложением нескольких пользователей. В различных частях приложения представлены почти все обсуждаемые в главе темы. Как для разных приложений, так и в пределах одной программы возможно применение разных технологий, а также их комбинаций. Основой организации работы в многопользовательской среде является планирование, прогнозирование и тестирование.  **Access и способы блокировки в Jet**  Механизм Jet имеет схему блокировки, которая позволяет эффективно обслуживать несколько пользо­вателей. При использовании Jet с Access, а не с VB или каким-либо иным инструментом разработки необходимо учитывать, что некоторые действия выполняются по умолчанию. Данный раздел посвящен изучению этих вопросов.  **Основные сведения о блокировке**  Перед использованием приложения многопользовательской базы данных его следует разместить таким образом, чтобы пользователи имели к нему доступ, а также настроить для совместного использования. Существует несколько способов достижения этой цели.  В диалоговом окне Options (Параметры), отображающемся при выполнении команд меню Tools Options (Сервис / Параметры), во вкладке Advanced (Другие) имеется параметр Default open mode (Режим откры­тия, определенный по умолчанию). Здесь можно определить режим открытия базы данных, т.е. должна ли она открываться для монопольного доступа (только для одного пользователя на весь сеанс работы) или для общего доступа.  Если выбран режим Exclusive (монопольный доступ), базу данных имеет право открывать только один пользователь. В этом случае Access изменяет заголовок LDB-файла, тем самым блокируя его (подробнее об этом см. в разделе "LDB-файл") и запрещая доступ к данным для всех других пользователей. Очевид­но, для многопользовательского приложения такая настройка использоваться не должна. Однако такие процедуры, как сжатие и восстановление, следует выполнять над базой данных, открытой для монополь­ного доступа.  Режим Shared (Общий доступ) позволяет открывать базу данных нескольким пользователям одновре­менно. При этом Access в момент открытия базы данных заносит информацию о подключившихсякнейпользователях в LDB-файл и задействует механизм блокировки и освобождения страниц и строк.  Эти и другие параметры можно задавать в командной строке во время запуска приложения Access. Некоторые из них перечислены в табл. 1.  **Таблица 1 Параметры командной строки при запуске Access**   |  |  | | --- | --- | | Параметр | Описание | | /Excl | База данных открывается для монопольного доступа. Данный параметр можно использовать даже тогда, когда для базы данных установлен режим Shared. | | /Ro | Только для чтения. Хотя база данных доступна для нескольких пользователей, в нее невозможно вносить изменения, при этом блокировка не используется. | | Отсутствует | Режим, определенный по умолчанию. Отсутствие параметра /Excl или /Ro позволяет открывать базу данных в режиме общего доступа либо в соответствии с определенным для нее режимом. |   **СОВЕТ**  **При необходимости совместного использования важно предотвратить открытие базы данных в монопольном режиме. Этого можно добиться, отключив параметр OpenExclusive при определении настроек для рабочих групп и задании параметров защиты данных в приложении. Более подробно данная тема рассматривается в другой статье.**  Задавая параметры базы данных, разработчик может выбирать режим блокировки записи по умолча­нию: блокировку на уровне строки либо на уровне страницы.  **Сравнение блокировки на уровне страницы с блокировкой на уровне строки**  В прошлом Access были присущи недостатки, связанные с появлением конфликтов доступа при исполь­зовании несовершенного способа хранения и блокировки записей. Поскольку Access поддерживает пере­менную длину записей, простая реализация блокировки на уровне строки была затруднена. Обеспечивая преимущества такой структуры записей, Access был вынужден хранить записи в статической страничной структуре объемом 2 Кб (при использовании механизма баз данных Jet 4.0 для приложения Access 2000 объем страницы данных составляет 4 Кб). При умышленной либо случайной блокировке записи блокиро­валась вся страница, что приводит к недоступности всех ее записей. Несмотря на эффективность такого метода, его применение приводит к возникновению различных проблем, связанных с конфликтами до­ступа, а также сокращает число одновременно работающих пользователей приложения Access. Таким образом, при использовании Access возможности разработчика были ограниченны.  В Access 2000 механизм баз данных Jet 4.0 позволяет разработчикам выбирать метод блокировки по умолчанию: на уровне строки либо на уровне страницы. Теперь пользователь может блокировать только редактируемую запись, а не все записи на странице. Поскольку отдельная запись может блокироваться лишь на короткое время (например, при выполнении операторов SQL Delete, Update или Insert), вероятность конфликта двух пользователей во время ее редактирования ниже, чем при одновременной блокировке нескольких записей в схеме страничной блокировки. Ранее вероятность конфликта умножалась на число записей на странице, определение которого было затруднено. Количество записей на странице данных зависело от размера записей и от времени их ввода, поэтому предвидеть вероятность конфликта было затруднительно.  Режим блокировки на уровне строки определен по умолчанию, но это не означает, что он во всех случаях является оптимальным. Если первостепенной задачей ставится производительность приложения, а конфликты возникают достаточно редко либо поддаются контролю, такой способ блокировки может привести к снижению производительности. Рассмотрим пример компьютерной системы банка, осуществ­ляющего международные торговые операции, в которой ввод записей производится гораздо чаще, чем их редактирование. Поскольку для подобной системы чрезвычайно важна высокая производительность, а ее снижение допускается лишь в случае конфликта, остальные операции базы данных должны выполняться с максимально возможной скоростью. В подобном случае может использоваться страничная блокировка.  С другой стороны, если к базе данных должен обеспечиваться доступ многих пользователей, а одно­временное редактирование каждым пользователем более одной записи недопустимо, возможно примене­ние блокировки на уровне строки. Это тем более верно при активном редактировании базы данных. Возвращаясь к примеру банковской системы, следует отметить, что записи базы данных с информацией о депозитах и изъятиях со счетов клиентов должны быть легкодоступны. При редактировании записи ее следует блокировать, в противном случае существует риск, что внесенные пользователем изменения бу­дут перезаписаны конкурирующим пользователем. Более того, редактирование одной записи не должно препятствовать редактированию соседней записи другим пользователем. Блокировка на уровне строки может применяться в ситуациях, когда запись должна оставаться открытой какое-то время, в течение которого ее не могут редактировать другие пользователи. Примером может служить просмотр информации о кли­енте для подведения баланса либо оценки его кредитоспособности. До завершения просмотра и принятия решения изменять запись нежелательно. Если запись остается открытой в течение нескольких минут, желательно избегать блокировки нескольких других записей на время ее редактирования. Как правило, следует избегать даже одной записи в течение достаточно долгого периода времени, если только это не является абсолютно необходимым.  Таким образом, разработчики имеют возможность управлять доступом многих пользователей на уров­нях страницы либо строки данных, что придает приложениям достаточную гибкость.  **LDB-файл**  Файл блокировки - это специальный временный файл, создаваемый при открытии базы данных Access. Он содержит информацию о применяемых в базе данных блокировках, а также о ее пользователях. При закрытии базы данных файл удаляется. Его имя совпадает с именем соответствующей базы данных, но он имеет расширение LDB. Этот файл всегда располагается в том же каталоге, что и база данных.  Сравнение оптимистической, пессимистической блокировок и блокировки на уровне строки  Разработчик может справедливо предполагать, что в многопользовательском приложении рано или поздно возникнет конфликт доступа при обращении к одной и той же записи. Единственное разумное решение такой проблемы заключается в выборе соответствующих параметров блокировки. Существует два варианта блокировки: оптимистическая и пессимистическая.  **Оптимистическая блокировка**  Оптимистическая блокировка используется в Access по умолчанию, она проста в реализации, и обычно предпочтение отдают именно ей. При оптимистической блокировке записи предполагается, что конфликты доступа маловероятны и что запись блокируется лишь в момент ее фактического обновления. Это обес­печивает высокую степень доступности данных, поскольку право долговременного либо исключительно­го доступа к ним никому не предоставляется. В соответствии с вышесказанным при открытии записи для редактирования остальные пользователи также могут открывать ее для редактирования, причем преиму­щество сохранения внесенных изменений имеет первый пользователь. Хотя оптимистическая блокировка проста в реализации и обычно не порождает проблем доступа пользователей к своим данным, однако при ее использовании одним из наиболее важных вопросов работы с базами данных в многопользовательской среде является вопрос о том, чьи изменения следует сохранять. Когда пользователь **А** открывает запись для редактирования и накладывает на нее оптимистическую блокировку, ничто не мешает пользователю **Б** открыть эту же запись для внесения изменений. Если **Б** сохранит изменения раньше, чем это сделает **А,** пользователь **А** получит следующее сообщение:  **"The Microsoft Jet database engine stopped the process because you and another user are attempting to change the same data at the same time."**    **("Механизм баз данных MicrosoftJet остановил процесс, поскольку вы и другой пользователь одновременно предприняли попытку доступа к тем же данным".)**  В подобной ситуации предшествующие версии Access отображали маловразумительное диалоговое окно, предлагающее пользователю выбрать один из вариантов: сохранить свои изменения, не сохранять их либо скопировать информацию в буфер обмена. Такое диалоговое окно конфликта доступа при сохранении из­менений просто не предоставляло достаточно информации для того, чтобы сделать разумный выбор. Новое диалоговое окно конфликта доступа при сохранении изменений не предоставляет возможности выбора и является однозначным. Далее рассматриваются некоторые способы наилучшей обработки ошибок при возникновении подобной ситуации.  **Пессимистическая блокировка**  Пессимистическая блокировка является противоположностью оптимистической. При пессимистической блокировке записи или страницы она становится недоступной для других пользователей с момента на­чала редактирования записи до момента ее сохранения. Такой способ блокировки используется многими другими базами данных, поэтому он знаком большинству разработчиков, а его результаты не должны вызывать вопросов у пользователей. Хотя пессимистическая блокировка исключает присущие оптимисти­ческой блокировке конфликты доступа при записи изменений, она также не лишена недостатков.  При использовании пессимистической блокировки вероятность конфликтов при обращении к данным может быть уменьшена. Когда используется блокировка на уровне страницы, появляется дополнительная проблема, связанная с блокировкой всех записей на странице в течение определенного периода времени. Если обычно процесс редактирования оказывается достаточно длительным и существует много конкури­рующих пользователей, пессимистическую блокировку следует применять с осторожностью. В некоторых приложениях, например баз данных для хранения информации о продажах и товарах, вероятно, предпоч­тение следует отдавать пессимистической блокировке, поскольку основные операции связаны с обработ­кой существующих записей. В то же время для систем отслеживания изменений данных во времени пессимистическая блокировка негативно отражается на производительности. Большинство касающихся пес­симистической блокировки предупреждений и оговорок относятся к способу страничной блокировки в Access. Теперь, когда в Access имеется возможность выполнять блокировку на уровне строки, пессимис­тическая блокировка должна получить более широкое распространение и применение.  **Блокировка на уровне строки**  Основным преимуществом блокировки на уровне строки является расширение доступа к базе данных для многих пользователей. При блокировке единственной редактируемой записи многим пользователям пре­доставляется доступ к большему объему данных без возникновения конфликтов блокировки или доступа к записям. Использование блокировки на уровне строки также позволяет разработчикам расширить гра­ницы использования пессимистической блокировки. Таким образом, пользователям предоставляются бо­лее знакомые и очевидные условия работы, в ходе которой они выполняют несложные операции открытия записи, ее редактирования и сохранения изменений. В предшествующих версиях Access пессимистическая блокировка не могла получить широкого распространения, поскольку страничный способ блокировки ог­раничивал количество одновременно работающих пользователей, которые должны были мириться с воз­можностью блокировки внесенных ими изменений другими пользователями. При этом разработчикам приходилось создавать схемы реализации привычных для пользователей условий работы (расширяющиеся записи, временные таблицы и т.п.). Блокировка на уровне строки является главным достижением в Jet 4.0. Она должна найти csoe применение в наиболее популярных и надежных приложениях.  **Свойство RecordLocks и связанные интерфейсные элементы**  При открытии в Access связанной формы или набора записей имеется возможность наложения бло­кировки на соответствующий набор записей. Конечно, эти параметры можно использовать только при ра­боте с Jet, тогда как при использовании конфигурации приложения Access клиент-сервер предполагается установка режима No Locks (отсутствует).  Существует три режима блокировки:  • No Locks (отсутствует) - эквивалентен оптимистической блокировке,  • Edited Records (изменяемой записи) - эквивалентен пессимистической блокировке,  • All Records (всех записей) - блокировка всех записей набора. В многопользовательских приложениях этот режим следует использовать с осторожностью.  **СОВЕТ**  **Связывание пользовательского интерфейса с данными является простым способом предоставления пользователям доступа к данным, но при этом теряется контроль над данными. В связанной с пессимистически блокированной за­писью форме пользователь увидит в панели выбора записи символ "перечеркнутое О", но информация о том, кто заблокировал запись, отсутствует. Если панель выбора записи не отображается, пользователь услышит звуковой сигнал.**  **Методы блокировки в Jet**  Блокировка - обычное и необходимое явление в базе данных. Чтобы убедиться в корректности типа и установить продолжительность блокировки, необходимо при ее появлении иметь возможность получать о ней информацию. Данный раздел будет полезен при анализе особенностей блокировки в приложении, который проводится для проверки соответствия возможностей приложения цели, с которой оно создава­лось.  **Определение состояния блокировки**  Как отмечалось ранее, фактическая блокировка записи или страницы записей происходит в различное время, а сама блокировка может быть оптимистической либо пессимистической. Кроме того, разные части приложения (или разные приложения) могут использовать различные виды блокировки тех же записей в то же самое время. Поэтому в разные моменты времени будут возникать различные ошибки. Возникаю­щая ошибка зависит от состояния блокировки.  В ADO существует свойство набора записей **LockType,** содержащее информацию о применяемом к записям типе блокировки. Это свойство доступно для чтения и записи до момента открытия набора за­писей, если набор записей уже открыт, оно доступно только для чтения. Значения свойства **LockType**для Microsoit.Jet.OLEDB.4.0 приводятся в табл. 2. При использовании других поставщиков могут применяться иные константы. Для определения поддерживаемых поставщиком параметров следует использовать метод **. Supports** с параметрами **adUpdate** либо **adUpdateBatch.**  **Таблица 2 Константы для свойства LockType в Jet 4.0 при использовании провайдера Microsoft.Jet.OLEDB.4.0**   |  |  | | --- | --- | | Константа | Описание | | **adLockReadOnly** | Значение, принятое по умолчанию. Набор записей не может изменяться, открывается только для чтения, и к данным не применяются никакие блокировки. | | **adLockPessimistic** | Пессимистическая блокировка при редактировании. | | **adLockOptimistic** | Оптимистическая блокировка при вызове обработчика события **Update.** | | **adLockBatchOptimistic** | Оптимистическая блокировка для режима группового обновления. |   **ПРИМЕЧАНИЕ**  Если свойство **CursorLocation**имеет значение **adUseClient ,**значение**adLockPessimistic** не поддерживается, однако при этом ошибка возникать не будет. Jet подставляет в свойство **LockType**другое подходящее значение. Так про­исходит потому, что при использовании значения**adUseClient** сервер не отслеживает состояние текущей записи, и поэтому пессимистическая блокировка невозможна.  **ПРИМЕЧАНИЕ**  ADOR является подмножеством объектной модели ADO и содержит только объекты **RecordSet и Field.**Он может создаваться специально либо передаваться от сервера клиенту. Объект ADOR поддерживает единственное значение **свойства LockType - adLockBatchOptimistic.**  При разработке, тестировании и поддержке приложения важно иметь информацию о состоянии бло­кировки записи. Необходимо проверить соответствие каждого процесса обработки данных требованиям, предъявляемым к приложению. Подобная процедура затруднений не вызывает. Следует остановить выпол­нение программы и проверить значение свойства набора записей **LockType** (рис. 1).  **РИСУНОК 1**Свойство LockType отображает •остояние блокировки набора wnuceu .  Для индикации режима редактирования набора записей предназначено другое свойство. До вхождения в режим редактирования свойство**EditMode**содержит значение**adEditNone.** Во время редактирования записи оно содержит значение**adEditInProgress.** После успешного обновления записи свойство**EditMode** вновь принимает значение**adEditNone.** Остальные значения свойства**EditMode** описываются в табл. 3.  **Габлица 3 Значения свойства EditMode набора записей ADO**   |  |  | | --- | --- | | Константа | Описание | | **adEditNone** | Редактирование не выполняется. | | **adEditInProgress** | Данные в текущей записи изменились, но сохранение еще не выполнялось. | | **adEditAdd** | Данное значение свойство **EditMode** принимает после вызова метода **AddNew**. Оно показывает, что буфер копирования содержит еще не сохраненную новую запись. | | **adEditDelete** | Текущая запись была удалена. |   Значение свойства **EditMode** отражает состояние буфера, используемого для создания и редактирова­ния записей. Оно используется, когда при выходе из режима редактирования выбран соответствующий метод**(Update** или**CancelUpdate).**  **Тестирование блокировок**  Применяемые к записям блокировки можно протестировать, просмотрев значения свойств**LockType и EditMode,** но обычно гораздо более важно выяснить тип блокировки, налагаемой другим пользователем на требуемые данные. Единственный способ выполнения поставленной задачи фактически состоит**в не-**обходимости вызова ошибки конфликта доступа.  При возникновении ошибки провайдер OLEDB Jet выдает определенную информацию о типе блоки­ровки, применяемой другим пользователем. В случае конфликта следует проверить свойство подключения:  **Connection.Errors( index ). SQLState**  для точного выяснения вида возникшей ошибки. В табл. 4 приводятся некоторые коды ошибок конф-ликта доступа, возвращаемые при обращении к свойству**.SQLState.**  **Таблица 4 Коды ошибок блокировки, возвращаемые поставщиком Jet 4.0 OLEDB**   |  |  | | --- | --- | | **Код** | Сообщение | | **3006** | Database <name> is exclusively locked. (База данных <имя> используется в монопольном режиме.) | | **3008** | The table <name> is already opened by another user, or it's already open through the user interface and can't be manipulated programmatically. (База данных <имя> уже открыта другим пользователем, либо она открыта посредством пользовательского интерфейса, и ее программная обработка запрещена.) |   **Таблица 4 Коды ошибок блокировки, возвращаемые провайдером Jet 4.0 OLEDB (продолжение)**   |  |  | | --- | --- | | Код | Сообщение | | 3009 | You tried to lock table <tablename> while opening it, but the table can't be locked because it's currently in use. Wait a moment, and then try the operation again. (Предпринята попытка блокировки таблицы <имятаблицы>, но эта операция невозможна, поскольку в данный момент таблица используется. Подождите немного, а затем повторите попытку.) | | 3027 | Can't update; database or object is read-only. (Обновление невозможно. База данных или объект доступен только для чтения.) | | 3046 | Couldn't save; currently locked by another user. (Сохранение невозможно. Объект блокирован другим пользователем.) | | 3158 | Couldn't save record; currently locked by another user. (Сохранение записи невозможно. Она блокирована другим пользователем.) | | 3164 | The field can't be updated because another user or process has locked the corresponding record or table. (Обновление поля невозможно, поскольку соответствующая запись или таблица блокирована другим пользователем либо процессом.) | | 3186 | Couldn't save; currently locked by user <name> on machine <name>. (Сохранение невозможно. Объект блокирован пользователем <имя> на компьютере <имя>.) | | 3187 | Couldn't read; currently locked by user <name> on machine <name>. (Чтение невозможно. Объект блокирован пользователем <имя> на компьютере <имя>.) | | 3188 | Couldn't update; currently locked by another session on this machine. (Обновление невозможно. Объект блокирован в течение другого сеанса работы на этом компьютере.) | | 3189 | Table <name> is exclusively locked by user <name> on machine <name>. (Таблица <имя> блокирована в монопольном режиме пользователем <имя> на компьютере <имя>.) | | 3197 | The Microsoft Jet database engine stopped the process because you and another user are attempting to change the same data at the same time. (Механизм баз данных Microsoft Jet остановил процесс вследствие одновременной попытки двух пользователей изменить одни и те же данные.) | | 3202 | Couldn't save; currently locked by another user. (Сохранение невозможно. Объект блокирован другим пользователем.) | | 3211 | The database engine couldn't lock table <name> because it's already in use by another person or process. (Механизм баз данных не выполнил блокировку таблицы <имя>, поскольку она уже используется другим лицом либо процессом.) | | 3212 | Couldn't update; currently locked. (Обновление невозможно. Объект блокирован.) | | 3218 | Couldn't update; currently locked by user <name> on machine <name>. (Обновление невозможно. Объект блокирован пользователем <имя> на компьютере <имя>.) | | 3260 | Table <name> is exclusively locked by user <name> on machine <name>. (Таблица <имя> блокирована в монопольном режиме пользователем <имя> на компьютере <имя>.) | | 3261 | Couldn't lock table <name>; currently in use by user <name> on machine <name>. (Блокировка таблицы <имя> невозможна. Таблица используется пользователем <имя> на компьютере <имя>.) |    Массив ошибок также содержит и другую потенциально полезную информацию об ошибке блокиров­ки: данные о блокировке, используемой другим пользователем. Свойства **NativeError и Number** сообща­ют о блокировке, препятствующей выполнению требуемой операции. Сочетания этих свойств и их значения приведены в табл. 5.  **Таблица 5 Сочетания свойств NativeError и Number объекта Connection. Errors для иденти­фикации типа блокировки**   |  |  |  |  | | --- | --- | --- | --- | | Пользователь | Свойство | Значение | Тип блокировки | | Данный пользователь | Connection.Errors(0).NativeError | -533791822 | Пессимистическая | | Данный пользователь | Connection.Errors(O).Number | -105514571 | Оптимистическая | | Другой пользователь | Connection.Errors(O).NativeEr | -2147467259 | Пессимистическая | | Другой пользователь | Connection.Errors(O).Number | -2147217887 | Оптимистическая |    На данном этапе смысл этих значений не совсем очевиден, разве что их можно использовать при выборе момента для еще одной попытки обновления данных. Однако реальную ситуацию, когда они могут понадобиться, предвидеть трудно. Пока будем считать их применение делом будущего.  **Использование блокировки страниц**  Как уже говорилось, в течение длительного периода в Access не существовало возможности непосред­ственной блокировки отдельных записей, предоставлялась лишь блокировка целых страниц. Чтобы исполь­зовать преимущества более высокой производительности при задействовании страничной блокировки, необходимо отключить установленный по умолчанию параметр блокировки на уровне строк. Для этого следует выполнить команды меню Tools / Options [ Advanced и отключить флажок Open databases with row-level locking (Блокировка записей при открытии базы данных).  **Обработка ошибок блокировки при работе в многопользовательской среде**  Любая многопользовательская система должна предвидеть ошибки блокировки. Различные системы обрабатывают возникающие в определенных ситуациях ошибки по-разному. Кроме того, в случаях возник­новения ошибок блокировки различные системы предоставляют разработчикам и пользователям неодина­ковую информацию. В данном разделе рассматриваются некоторые настройки блокировки и связанные с нею ошибки, с которыми чаще всего приходится сталкиваться при разработке приложений в Access 2000. Здесь также поясняются некоторые технологии предотвращения и обработки этих ошибок.  Настройки блокировки Access  Лучший способ обработки возникающих при работе в многопользовательской среде ошибок состоит в их предотвращении. В Access имеется несколько свойств, которые можно использовать для снижения ча­стоты возникновения конфликтов доступа. Соответствующие параметры можно отыскать во вкладке Advanced диалогового окна Options. Однако сами по себе они не осуществляют обработку подобных ошибок.  • Number of Update Retries (Число повторов обновления) - управляет количеством попыток, кото­рые Access предпринимает при сохранении или обновлении заблокированной записи. Допустимые значения находятся в интервале 0-10.  • ODBC Refresh Interval (Период обновления ODBC (с)) - период обновления в секундах при ис­пользовании базы данных ODBC. Допустимые значения находятся в интервале 1-32766.  • Refresh Interval (Период обновления (с)) - период обновления записей в секундах в режиме про­смотра Datasheet (Таблица) или Form (Форма). Допустимые значения находятся в интервале 1-32766.  • Update Retry Interval (Период повтора обновления (мс)) - промежуток времени в миллисекундах, по истечении которого Access предпринимает следующую попытку сохранения измененной записи, которая ранее была блокирована. Допустимые значения находятся в интервале 1-1000.  Конфликт записи  Ошибка Write Conflict (Ошибка конфликта при записи) (см. табл. 4, ошибка 3197) является одной из наиболее неприятных ошибок, возникающих при работе приложения Access в многопользовательской среде. Она возникает в случаях, когда пользователь А открывает запись с оптимистической блокировкой и во время ее редактирования к ней обращается пользователь Б, изменяя и сохраняя ее. Когда пользо­ватель А завершает работу над записью и предпринимает попытку ее сохранения, он получает сообще­ние об ошибке. В предшествующих версиях Access в подобных ситуациях отображалось маловразумительное диалоговое окно, в котором предлагалось перезаписать изменения другого пользователя (при этом не со­общалось, какие именно), отказаться от только что внесенных изменений (что никогда не пользовалось популярностью) либо скопировать данные в буфер обмена (и что делать дальше?).  В настоящее время способ внутренней обработки ошибок подвергся изменениям. В Access 2000 конф­ликт записи приводит к игнорированию внесенных пользователем А изменений. Хотя подобная мера кажется излишне суровой, она наилучшим образом соответствует ситуации, когда большинство многополь­зовательских приложений Access поспешно создаются людьми, которые не всегда достаточно хорошо разбираются в вопросах многопользовательского применения. По крайней мере, такая обработка конфликта записи является решительной и окончательной, а пользователям не придется искать ответ на вопрос, над которым они никогда не задумывались. Если приложение должно обрабатывать конфликт записи иным способом, необходимо создать пользовательскую процедуру обработки ошибки.  **Блокированная запись**  Когда в ходе обычного использования приложения пользователь А пытается изменить запись, редак­тируемую пользователем Б, первый из них получит сообщение об ошибке 3260 (Запись блокирована - см. табл. 4). Как правило, подпрограмма обработки ошибок предпринимает заданное число попыток со­хранения записи пользователя А перед тем, как предложить ему подтвердить необходимость дальнейших попыток либо отказаться от изменения записи. Если примененная пользователем Б блокировка является пессимистической, она снимается сразу после обновления записи в базе данных. Как правило, этот пе­риод времени очень короток.  **Транзакции**  Транзакция объединяет в себе отдельные либо элементарные операции и выполняетихкак единый процесс**.**Весь набор команд транзакции завершается успешно (изменения сохраняются) либо весь откло­няется (происходит откат). В случае неудачи транзакции база данных возвращается в исходное состояние (выполняется операция отката), в котором она была до начала выполнения транзакции. Это гарантиру­ет, например, что при внесении в товарный чек данных о товаре они одновременно удаляются из ин­вентарной описи. Когда один счет увеличивается, тогда другой уменьшается, а при записи изменений соответствующая информация вносится в контрольную таблицу. В весьма динамичной среде многопользо­вательского приложения выполняющий подобные обновления и дополнения пользователь в ходе выпол­нения отдельных операций подобного процесса, скорее всего, столкнется с ошибками блокировки записей, оставляя незавершенными балансовые счета, завышенные сведения о количестве товаров в описи либо внесенные, но несохраненные изменения. Короче говоря, транзакции помогают поддерживать целостность данных в условиях частых блокировок нескольких записей. В многопользовательском приложении транзакции следует использовать во всех возможных случаях.  Однакб транзакции имеют не одни лишь преимущества. Чтобы обеспечить выполнение всех изменений транзакция собирает информацию о блокировках. Транзакции устанавливают все требуемые приложени­ем блокировки и не снимают их до завершения всего процесса, при этом ошибки возникать не должны. Поскольку существует вероятность установки большого количества блокировок и время их действия ока­зывается более длительным, чем если бы они устанавливались только частью всего процесса, возможность одновременного доступа многих пользователей при обращении приложения к данным в действительнос­ти снижается. Однако в любом случае база данных, допускающая использование большим количеством пользователей, но не обеспечивающая целостность данных, ценится невысоко. Поэтому компромисс между целостностью данных и возможностью доступа со стороны многих пользователей является оправданным.  Транзакции являются методами объекта  **ADOConnection.**    |  |  | | --- | --- | | Метод | Описание | | **BeginTrans** | С вызова данного метода начинается выполнение транзакции. | | **CommitTrans** | Завершает последовательность процессов и в случае отсутствия ошибок сохраняет внесенные изменения в базе данных., | | **RollbackTrans** | В случае появления ошибки отменяет процесс и возвращает базу данных в состояние, в котором она была до выполнения команды **BeginTrans**. |     В листинге 1 приводится пример использования транзакции.  Листинг 1 Использование транзакции в VBA:  Function TestTrans() As Boolean  Dim conn As ADODB.Connection  Dim rst As ADODB.Recordset  On error resume Err\_TestTrans  Set conn = New ADODB.Connection  Conn.BeginTrans  'выполнениепроцессов, подобныхоператорам SQL, либометодов .Edit,  '.Update,  .AddNew Methods  ' В случае отсутствия ошибок изменения сохраняются.  Conn.CoimnitTrans  Exit Function  Err\_TestTrans:  'В случае возникновения ошибок выполняется откат транзакции.  Conn.RollbackTrans  ………………………..  EndFunction  Транзакции могут быть вложенными, когда сохранение результата одной транзакции зависит от кор­ректности выполнения другой. В случаях вложения транзакций необходимо проверять корректность их вы­полнения, начиная с низшего (внутреннего) и заканчивая наивысшим (внешним) уровнем.  Блокировка Oracle/SQL Server  При работе с Oracle, SQL Server, Informix или любым другим серверным механизмом баз данных Access более не осуществляет управление блокировкой. Однако основная концепция остается неизменной - Access управляет доступом к записям в базе данных, обеспечивая многим пользователям одновременный доступ к ней.  Серверы данных прекрасно справляются сэтой задачей. Поскольку данные размещаютсявместе**с** ме­ханизмом баз данных, блокировка записи и управление одновременным использованием может осуществ­ляться быстро, четко и надежно. Каждое из этих приложений хранит информацию о блокировке в таблице, постоянно находящейся в памяти, и может применять наиболее эффективный тип блокировки, выпол­нять операцию и снимать блокировку за считанные секунды.  При использовании Microsoft SQL Server могут применяться следующие типы блокировок:  • Shared Lock (Общая блокировка). Подобная блокировка используется в операциях обработки данных, доступных только для чтения. Общие блокировки позволяют другим пользователям читать запись или страницу, являющуюся объектом общей блокировки. На запись или страницу может одновременно налагаться несколько общих блокировок. Такие блокировки снимаются по окончании использова­ния**данных.**  • Exclusive Lock (Монопольная блокировка). Такая блокировка используется при выполнении по от­ношению к данным операторов SQL**UPDATE,  DELETE** или **INSERT.** При этом на монопольно бло­кированные данные не могут налагаться никакие другие блокировки до тех пор, пока SQL Server не снимет монопольную блокировку.  • Live Lock (Временная блокировка). Подобная блокировка является запросом на монопольную бло­кировку, возникающим после четырех последовательных неудачных попыток применения монополь­ной блокировки данных. Такая блокировка возникает в случаях наличия слишком большого количества перекрывающихся общих блокировок. В подобной ситуации SQL Server перестает при­менять общие блокировки. Временные блокировки предотвращают монополизацию таблицы или страницы общими блокировками (при операциях считывания) и запрещают операции, связанные с записью**(UPDATE, DELETE, INSERT).** Они также предотвращают ситуацию, называемую "на­сыщением блокировки".  Существуют и другие используемые SQL Server стратегии обработки проблем одновременного досту­па. К ним относятся динамическая блокировка на уровне строки (SQL Server 7.0), предотвращение, выявление и исправление взаимной блокировки, управление оптимистической блокировкой, а также наращивание масштабируемых блокировок.  Динамическая блокировка данных на уровне строки соответствующим диспетчером SQL Server дина­мически настраивает конфигурацию сервера блокировок, исходя из размера и характеристик использова­ния базы данных. Это в значительной степени уменьшает необходимость в ручной настройке и поддержке сервера блокировки.  При предотвращении, выявлении и исправлении взаимной блокировки SQL Server выявляет момент возникновения конфликта двух транзакций. В подобной ситуации выясняется, что одна транзакция моно­польно блокировала данные, требуемые другой транзакцией, а та, в свою очередь, монопольно блоки­ровала данные, требуемые первой. Обе транзакции не могут снять свои блокировки. Без вмешательства механизма баз данных они остаются в состоянии взаимной блокировки, для снятия которой SQL Server осуществляет откат одной транзакции, завершает другую и снова запускает первую.  **SQL Server активно предотвращает взаимные блокировки, значительно уменьшая количество блоки­ровок в таблицах.**  Для обеспечения работы многих пользователей SQL Server использует подход оптимистического управ­ления одновременным доступом. При оптимистическом подходе для просмотра данных пользователи мо­гут применять курсор со стороны сервера, не вызывая их блокировки. Вместо этого SQL Server выявляет изменения возвращаемых строк и затем действует соответствующим образом (обычно отклоняя изменение, если только оно не сопровождается явной инструкцией блокировки). Такой подход увеличивает объем доступных данных без интенсивного управления блокировкой и вызываемой этим перегрузки.  **Резюме**  В данной главе исследовались вопросы работы в многопользовательской среде, начиная с конфигури­рования файла базы данных и заканчивая выбором наборов записей и работой со страницами и записями данных. Подробно рассматривались различные стратегии блокировки, сравнивались методы оптимистической и пессимистической блокировки, описывались преимущества новых возможностей механизма баз данных Jet при блокировке на уровне строки. Приводились аргументы выбора применяемых приложением типов блокировки, а также давались советы по использованию сообщений об ошибках, поступающих от постав­щика OLEDB. В целом эта глава призвана помочь в выработке эффективной стратегии и тактики при создании нового приложения, она может также оказаться полезной и при отладке существующей базы данных.  Приведенная информация носит обзорный характер. Основную же ценность представляет опыт прак­тической разработки приложений для работы в многопользовательской среде. При этом следует уделять особое внимание разрешению конфликтов доступа, которые неминуемо возникают в реальной ситуации. Однако в Access имеются специальные средства для решения подобных проблем. |