



## Проекты Логинова Дмитрия

[Главная](#)
[Статьи от автора](#)
[Работы автора](#)
[Гостевая книга](#)
[Об авторе](#)
[Регистрация](#)
[Добавить исходник](#)
[Что нового?](#)
[Полезные ссылки](#)


### Проблемы, которые могут возникать при использовании BDE и способы их решения

Версия 2.3 от 05.06.2008 г.

В настоящей статье в основном затрагиваются вопросы, касающиеся работы Borland Database Engine с таблицами Paradox, однако некоторые пункты относятся и к другим базам данных.

В настоящее время развитие BDE прекращено. Об успехах данного проекта говорит то, что и по сей день имеются программисты, использующие BDE при разработке своих проектов. Многие проекты, основанные во времена господства BDE, приходится поддерживать до сих пор.

После установки BDE нам предлагаются настройки, которые были актуальны в прошлом веке, но на сегодняшний день их явно недостаточно. Необходимо сразу увеличить численное значение ряда параметров настройки. Эти параметры можно менять как в реестре (они расположены в "HKEY\_LOCAL\_MACHINE\SOFTWARE\Borland\ Database Engine\Settings\SYSTEM\INIT"), так и с помощью программы BDEAdmin, которая устанавливается вместе в BDE и чаще всего ее можно найти в каталоге "C:\Program Files\Common Files\Borland Shared\BDE\". После запуска BDEAdmin зайдите на вкладку Configuration и выберите "System\INIT". Изменить в первую очередь следует следующие параметры:

- **SHAREDMEMSIZE** - размер совместно используемой области памяти (судя по названию). По умолчанию SHAREDMEMSIZE=2048, однако для нормальной работы требуется значение 8192. Можно увеличить его еще больше, однако при этом может потребоваться изменение параметра SHAREDMEMLOCATION (адрес памяти в HEX-формате, по которой следует расположить блок SHAREDMEMSIZE).
- **MEMSIZE** - размер оперативной памяти (в мегабайтах), который разрешено использовать BDE. По умолчанию MEMSIZE=16. Этого обычно хватает, но увеличение данного параметра должно привести к ускорению SELECT-запросов с большим объемом выборки. Рекомендуемое значение для данного параметра - 30% от объема ОЗУ, но не более 205.
- **MAXFILEHANDLES** - максимальное количество файлов, которые могут быть открыты BDE одновременно (в их число входят \*.db, \*.px, \*.val и др.). По умолчанию MAXFILEHANDLES=48. Рекомендуемое значение для данного параметра: 256.
- **LOCAL SHARE** - рекомендуется установить в True. Из справки назначение данного параметра можно понять следующим образом: LOCAL SHARE следует устанавливать в True, если с базой будет работать не только BDE, но еще и сторонние приложения. Однако это утверждение скрывает реальный механизм действия данной настройки, и чаще всего пользователи просто игнорируют данный параметр. А зря. Фактически данный параметр, установленный в True, отключает кэширование данных, выполняемое BDE, благодаря чему данные записываются в базу немедленно, после каждого вызова Post, что значительно уменьшает риск повреждения данных, и позволяет организовать более надежную многопользовательскую работу с базой в локальной сети. Естественно, модификация данных будет выполняться медленнее, однако и повреждения базы будут происходить гораздо реже. Кстати, не представляю, как вообще BDE может нормально работать в локальной сети с LOCAL SHARE=FALSE. Для корректной работы как минимум нужно, чтобы кэширование данных выполнялось на том же компьютере, где лежит база :) Дополнительную информацию читайте [здесь](#).

Изменение прочих параметров не так сильно влияет на работоспособность BDE, поэтому их можно и не трогать.

Ниже перечислены проблемы, которые возникали у меня в процессе работы с BDE и приведены способы их устранения.

1. Для корректной работы BDE требуется правильная установка. Для установки BDE (без SQL Links) можно воспользоваться архивом "C:\Program Files\Common Files\Borland Shared\BDE\bdeinst.cab". В нем лежит файл BdeInst.dll, который является инсталлятором BDE. Перед началом установки следует удалить предыдущую версию BDE (переименовать ветку реестра "HKEY\_LOCAL\_MACHINE\SOFTWARE\Borland\Database Engine" и каталог, в которой BDE был ранее установлен). Начать процесс инсталляции можно с помощью команды "regsvr32.exe BdeInst.dll". При этом, возможно, BDE ругнется на отсутствие места на жестком диске и предложит выбрать каталог установки. В результате установки в указанном каталоге появятся все необходимые для работы BDE файлы, включая BDEAdmin.exe и BDEADMIN.HLP. Сугубо "программерские" файлы (BDE32.HLP, LOCALSQL.HLP, \*.txt) установлены не будут! Описанный способ установки BDE хоть и является рабочим, но он не разрешен компанией Borland, поэтому является нелегальным! Легальная установка возможна только с помощью инсталляторов, имеющих разрешение (сертификат) от Borland на установку BDE (например, InstallShield Express). Эти инсталляторы корректно устанавливают BDE, настраивают алиасы и вносят необходимую информацию в реестр Windows.
2. Эпизодически появляется ошибка "Insufficient memory for this operation". Она появляется при слишком большом числе подключений. Например, при стандартной конфигурации BDE данная ошибка появится, если установить более 10 подключений к базам данных и для каждого подключения открыть хотя бы одну таблицу (для того, чтобы убедиться в этом, запустите 11 экземпляров программы SQL Explorer (она лежит в каталоге "\$(DELPHI)\dbexplor.exe") и попробуйте в каждом из них открыть какую-либо табличку Paradox - должна произойти указанная ошибка). Для исправления данного ограничения необходимо увеличить параметр SHAREDMEMSIZE. По умолчанию это значение равно 2048 (ограничение на 10 подключений). Если изменить данную переменную на 4096, то BDE станет поддерживать одновременно 21 подключение. Рекомендуется

изменить данный параметр на 8192.

3. Многопоточное приложение виснет при работе с таблицами базы данных. Проблема возникает из-за того, что во всех потоках по умолчанию используется объект TSession с именем "Default". Похоже, что в недрах BDE какие-то ресурсы защищаются от одновременного доступа с помощью критической секции, либо мьютекса. Перед выполнением SQL-запроса происходит захват объекта синхронизации, а после выполнения запроса захват данного объекта снимается. Если при выполнении запроса произойдет исключение, то снятие блокировки может не произойти (это бывает не всегда, а только в некоторых ситуациях, вероятно, где-то по ошибке пропущен оператор TRY..FINALLY), и выполнение любого запроса из параллельного потока приведет к его зависанию, т.к. он будет вечно ждать снятия блокировки. Для решения данной проблемы следует для каждого потока создавать свой объект TSession с уникальным именем SessionName, и назначать SessionName всем используемым в данном потоке DB-компонентам. SessionName лучше сделать глобально уникальным (это предотвратит зависание, если в качестве объекта синхронизации используется именованный мьютекс).
4. Данные иногда не сохраняются при выходе из программы. Многие с этой проблемой сталкивались, кто-то списывает подобные проблемы на "кривизну" рук пользователя, кому-то предстоит с ней столкнуться. Данное утверждение очень легко проверить: достаточно в компоненте TTable добавить/изменить/удалить несколько записей и снять приложение из диспетчера задач. Данные так и не будут записаны в базу данных, поскольку физически запись данных в DB-файл происходит только при вызове метода TDataSet.Close. До этого вся работа с данными выполняется в памяти, либо в кэше на жестком диске. Если несколько приложений параллельно работают с одной и той же таблицей, то все изменения сохраняются в тот же самый (причем общий для всех) кэш, и физически запись в DB-файл произойдет только после того, как все приложения закроют набор данных. Если в ходе работы с таблицей базы данных одно из таких приложений было снято из диспетчера задач (его могли снять в результате зависания, вызванного теми или иными обстоятельствами), то можно хоть месяц вносить изменения в таблицу с помощью остальных приложений (такие случаи на практике встречались), все равно они будут обречены - в DB-файл ничего сохранено не будет. Даже подтверждение транзакции методом TDataBase.Commit не приведет к сохранению данных! Для недопущения описанной ситуации не следует держать таблицы открытыми длительное время, т.е. нужно как можно реже использовать "живые" наборы данных (см. [Как работает BDE](#)). Для автоматической записи на диск изменений после каждого вызова метода Post следует выставить в TRUE указанный выше параметр LOCAL SHARE. Также с помощью DBI-функций DbUseIdleTime() и DbSaveChanges() можно заставить BDE в любой момент сохранить изменения в DB-файл (см. [Запись буфера BDE на диск](#)).
5. Таблицы Paradox иногда повреждаются. Это может произойти по разным причинам (ошибка одновременного доступа к файлу, ошибка при сохранении пакета кэшированных изменений, сбой во время сохранения кэша BDE в базу, ошибка записи на диск / чтения с диска, ошибка RAM, неправильная установка параметров BDE и др.). Описание некоторых ошибок, приводящих к повреждениям баз данных (Interbase/Firebird), смотрите [здесь](#). Для исправления поврежденных таблиц Paradox предназначена библиотека TUTIL32.DLL, заголовочный файл для которой вы можете взять [отсюда](#). Восстановить таблицу удастся не всегда, поэтому делайте регулярно бэкап баз данных (только не функцией CopyFile(); лучше - с помощью запроса "INSERT INTO Table1 SELECT \* FROM Table2", либо функцией dbiCopyTable()).
6. Портятся файлы индекса первичного ключа (PX-файлы). При открытии таблицы выдается ошибка "Index out of date". Это также может произойти по разным причинам (например, при неправильно установленном языковом драйвере). Для устранения проблемы нужно либо удалить PX-файл (но кто его будет пересобирать?), либо восстановить его средствами модуля PXHEADER.PAS, скачать который вы можете [отсюда](#).
7. Не удастся открыть таблицу из-за поврежденного VAL-файла. Для устранения проблемы следует удалить поврежденный VAL-файл (в этом файле хранятся ограничения на поля таблицы, которые использует разве что Database Desktop). Обычно и без них можно прекрасно обойтись.
8. Во время периодической проверки таблиц базы данных утилитой TUTIL32.DLL иногда при выполнении запросов выдается ошибка "File is locked". Не следует обращаться к таблицам, пока с ними работает указанная утилита.
9. В рамках одной транзакции невозможно изменить более 255 записей (это ограничение описано в статье [Ограничения BDE](#)). Обойти данное ограничение не удастся. В таких случаях нужно либо отказаться от длительных транзакций (но при этом сначала нужно очень тщательно проверить каждую запись на правильность перед сохранением в БД), либо использовать кэшированные изменения (но данный механизм работает ОЧЕНЬ медленно, причем скорость нелинейно падает, в зависимости от числа измененных записей, и при изменении нескольких сотен записей может потребоваться несколько минут для завершения обработки, что чаще всего совершенно недопустимо).
10. При попытке открытия таблицы происходит ошибка "Table TABLENAME locked by user USERNAME". Ошибка бывает, когда пользователь стартовал транзакцию, после чего, в рамках кэшированных изменений, внес в таблицу изменения, но потом программу сняли из диспетчера задач (например, в результате ее зависания по тем или иным причинам) не дождавшись завершения транзакции. Для устранения данной проблемы необходимо закрыть ВСЕ приложения, которые могут использовать BDE (в том числе и Delphi). После этого таблицу можно будет открыть без проблем, т.к. при очередном запуске BDE исправит (или удалит) ошибочный файл блокировки данной таблицы.
11. Иногда (очень редко) возникает ошибка "Initialization failed". Одна из возможных причин (толком не изучена) - не установлены каталоги NET DIR и PRIVATEDIR. Обычно после перезапуска всех приложений, использующих BDE, ошибка исчезает.
12. Если изменить настройки BDE (например, SHAREDMEMSIZE) при наличии запущенных приложений, работающих с BDE, то при запуске очередного приложения может возникнуть ошибка "An error occured while attempting to initialize the Borland Database Engine (error \$251E)". После перезапуска всех, использующих BDE, приложений ошибка исчезнет.
13. BDE может выполнять запросы, которые вместо таблицы ссылаются на SQL-файлы. Например, есть следующий файл:

```
[MyQuery.sql]
SELECT g.Name, g.Kolvo, g.RPrice FROM Goods g
```

Далее на основании этого запроса мы хотим получить список неповторяющихся имен товаров. Используется следующий SQL-запрос:

```
SELECT DISTINCT Name FROM "MyQuery.sql"
```

К сожалению, обработка данного запроса приводит к ошибке "Operation not applicable", хотя формально все правильно. Зато так все работает:

```
[MyQuery.sql]
SELECT g.Name, g.Kolvo, g.RPrice AS RPrice FROM Goods g
```

т.е. нужно в SQL-файле добавить псевдоним хотя бы для одного поля.  
SQL-файлы - это более эффективная альтернатива вложенным SELECT-запросам.

14. Никогда при работе с BDE не используйте вложенные SELECT-запросы. BDE работает с ними очень долго и неоптимально. При достаточном знании языка SQL практически всегда можно обойтись без вложенных запросов, если правильно применить оператор JOIN (подробное описание синтаксиса языка SQL, используемого BDE, вы можете найти в "C:\Program Files\Common Files\Borland Shared\BDE\LOCALSQL.HLP").

15. В SQL-запросе можно дать любому полю русско-язычное имя. Пример:  

```
SELECT g.Name T."Имя товара", g.Kolvo T."Количество", g.RPrice T."Стоимость товара" FROM Goods g
```

Вместо "T" можно использовать и другие символы или слова.

16. При создании таблицы можно использовать служебные слова языка SQL. Пример:

```
CREATE TABLE MyTable
(
  ID INTEGER,
  T."TYPE" CHAR,
  T."COUNT" INTEGER
)
```

17. В рабочей таблице Paradox можно в любой момент добавить, изменить, удалить поле. Для этих целей существует функция DbfDoRestructure(). Минимальную информацию по ней можно почерпнуть из справочного файла "C:\Program Files\Common Files\Borland Shared\BDE\BDE32.HLP". К сожалению, в справке не говорится, каким образом можно, к примеру, добавить новое поле. Этот пробел восполняют поисковые машины. Такой запрос в Google: "DbfDoRestructure add field" вернет все необходимое. Кстати, есть готовый компонент-обертка над DbfDoRestructure() - TRestructure ([www.torry.net](http://www.torry.net)), но за него (неспроста наверно) автор просит \$129 ;)

18. В компоненте TDataBase есть свойство DatabaseName. Не нужно записывать в него путь к базе данных! Это просто строка текста, по которой компоненты TTable и TQuery отыскивают компонент TDataBase в приложении и работают с базой данных. Текст может быть любой, например "MyDB". Базу данных следует указывать либо с помощью свойства TDataBase.Alias, либо следующим образом: DataBase1.Params.Values['PATH'] := 'C:\DBPath' (при этом потребуется указать DriverName='STANDARD'). Данный подход позволяет использовать несколько компонентов TDataBase для работы с одной базой данных (например, один TDataBase служит для изменения данных в рамках транзакции, а второй используется в генераторе отчетов, работающем в параллельном потоке и транзакция ему не нужна).

19. Рекомендуется, где это возможно, использовать компонент TQuery вместо TTable. Компонент TQuery в сочетании со знанием языка SQL дают на порядок больше возможностей. Почти все, что есть в TTable, можно сделать через TQuery. Для редактирования набора данных следует включить режим RequestLive. Для создания связи Master-Detail следует в дочернем TQuery составить запрос вида: "SELECT \* FROM ChildTable WHERE ID = :IDMainTable" и подключить компонент TDataSource. Для компонента TTable реализовать связь Master-Detail можно только для ключевых полей, состоящих не более чем из одного поля. В TQuery подобных ограничений не может быть в принципе. Создать таблицу через TQuery также проще, чем через TTable (см. п. 14). Добавить/изменить поле невозможно ни в TTable, ни в Query. Мне известно только несколько ситуаций, в которых использование TTable оправдано:

- пакетное копирование данных из одного набора данных в другой с помощью метода BatchMove;
- экспорт данных в DBF-файл. TTable при создании новой таблицы позволяет скопировать структуру полей из указанного набора данных (Table1.FieldDefs.Assign(Query1.FieldDefs)), т.е. нет необходимости составлять скрипт создания таблицы базы данных.

20. Для изменения языкового драйвера (например, для DBF-таблиц) следует открыть ветку реестра "HKEY\_LOCAL\_MACHINE\SOFTWARE\Borland\Database Engine\Settings\DRIVERS\FOXPROVINIT" и установить требуемое значение переменной LANGDRIVER (например, LANGDRIVER=db866ru0). Аналогичного результата можно добиться с помощью TSession.ModifyDriver(), однако функция сработает только если нет открытых наборов данных, а результат ее работы будет виден только после перезапуска всех приложений, работающих с BDE. Также обращение к этой функции может приводить к ошибке, если в реестре был удален раздел "TABLE CREATE" (иногда данный раздел удаляет BDE в результате каких-то сбоев).

21. Иногда (крайне редко, раз год) при работе с базой данных возникает ошибка "Insufficient disk space", несмотря на то, что на диске свободно еще 40.0 Гбайт. BDE (вернее, IDAPI32.DLL) использует тип DWORD при определении размера свободного места на жестком диске. Какой-бы размер Windows не вернул, с этого значения берутся только 4 младших байта, поэтому, если размер свободного места на диске кратен 4294967295, то BDE считает, что свободного места вообще нет и выдает указанную ошибку. По теме данной ошибки в Интернете очень много обсуждений, например [вот это](#). Я предлагаю для решения данной проблемы автоматически создавать временный файл и периодически корректировать его размер так, чтобы объем свободного места на диске не был кратен 4294967295.

22. При попытке добавления данных в таблицу Paradox стала возникать ошибка "Table is Full". Срабатывает ограничение на размер таблицы - 256 Мб. [Здесь](#) указывается, как изменить настройки BDE, чтобы обойти данное ограничение. Рекомендую обратить внимание на 2 пункт, связанный с dbiDoRestucture().
23. При динамическом создании TQuery в дополнительном потоке не удастся подключиться к базе данных через компонент TDataBase, т.к. выдается ошибка "Unknown database. Alias XXX". Для успешного подключения требуется, чтобы у обоих компонентов свойства SessionName и DatabaseName были одинаковыми. В каждом дополнительном потоке должен быть создан свой компонент TSession с уникальным именем SessionName, которое следует присвоить остальным BD-компонентам, работающим в данном потоке.
24. Не удастся подключить двух и более клиентов к DCOM-серверу с модулем данных TRemoteDataModule, т.к. выдается ошибка "Name not unique in this context". Ошибка возникает из-за того, что для каждого клиента создается собственный экземпляр TRemoteDataModule, соответственно в сервере будут присутствовать несколько компонентов TDataBase с одним и тем же именем DatabaseName. Проблема можно решить одним из способов, в зависимости от модели сервера (tmApartment или tmFree). При использовании tmApartment все подключения работают в контексте одного потока (он не является основным), поэтому достаточно создать один общий TDataBase и расположить его, к примеру, на главной форме (либо же оставить TDataBase на TRemoteDataModule и выставить свойство TDataBase.HandleShared в True - в этом случае все компоненты TDataBase с одинаковым DatabaseName фактически будут использовать одно подключение). При использовании tmFree каждое подключение работает в отдельном потоке, поэтому для каждого из подключений необходимо создавать индивидуальный TDataBase и TSession. Т.к. при этом одновременных подключений к базе данных может быть много, следует изменить параметр SHAREDMEMSIZE в соответствии с данными выше рекомендациями.
25. У компонента TDataBase есть свойство KeepConnection со значением по умолчанию = True. При этом соединение с базой данных устанавливается при открытии любого связанного набора данных и остается на все время работы с базой данных. Если KeepConnection=False, то соединение разорвется автоматически, как только будут закрыты все связанные наборы данных. Если в системе установлено одновременно слишком много подключений к базам данных, то можно нарваться на описанные выше ограничения BDE. Один из способов минимизации числа подключений - контроль над параметром KeepConnection. Оставляйте KeepConnection=True только там, где это действительно нужно (я рекомендую выставлять KeepConnection=False исключительно по собственному опыту). Также не держите подолгу открытыми "живые" наборы данных.
26. TQuery ведет себя по-разному, в зависимости от того, включен ли CachedUpdates или нет. Тестирование выполнялось на запросе, подобном следующему: SELECT \* FROM TABLE WHERE ОСТАТОК > 0. Если CachedUpdates=False, то при внесении изменений запись исчезнет из набора данных, если ей присвоить ОСТАТОК=0. При CachedUpdates=True исчезновения записи не происходит.
27. При работе с таблицами Paradox можно использовать механизм транзакций. В рамках одной транзакции имеется ограничение - 255 изменений на одну таблицу. Для старта транзакции предназначен метод TDataBase.StartTransaction, а для завершения - TDataBase.Commit (подтверждение) либо TDataBase.Rollback (откат). Подтверждение транзакции для Paradox сводится всего-лишь к снятию блокировок измененных записей и очистке лога транзакции. Откат транзакции - это более сложная операция, которая потом обработкой лога транзакции должна вернуть базу данных в исходное состояние. В реализации функции отката транзакции для таблиц Paradox есть ошибки. Описание некоторых из них приведено [здесь](#). Вот еще одна ошибка функции Rollback, с которой пришлось столкнуться: имеется TQuery с CachedUpdates=True, в операторе SELECT которого есть условие WHERE по полю "Остаток". Текст запроса примерно такой: SELECT \* FROM TABLE WHERE ОСТАТОК > 0. После обработки запроса (при активной транзакции) в наборе данных оказалось 3 записи, значения поля "Остаток" которых равны соответственно {1, 5, 1}. Далее изменяем все три записи, уменьшая значения в поле "Остаток" на 1 (получим соответственно {0, 4, 0}), после чего вызываем TQuery.ApplyUpdates, TQuery.Close и откатываем транзакцию. В результате в наших трех записях окажутся значения {1, 4, 1} (а по-правильному должны быть {1, 5, 1}), т.е. вторую запись Rollback проигнорировал. Таким образом, если изменения записей в рамках транзакции ведут к делению данных с помощью условия WHERE на 2 части, то откат транзакции НЕ восстанавливает записи, которые УДОВЛЕТВОРЯЮТ условию WHERE.
28. Как было сказано, в рамках одной транзакции допускается вносить в таблицу не более 255 изменений. Видимо, имеется в виду, что нельзя изменить/добавить/удалить более 255 записей, а одной измененной записи соответствует одно изменение. На практике одной измененной записи может соответствовать N изменений (т.е. N записей в логе транзакции), где N - кол-во столбцов изменяемой таблицы. Достаточно создать N наборов данных, в каждом из них установить курсор на строку I, затем изменить значения полей записи (в каждом наборе данных нужно изменить только одно поле записи). Если в таблице имеется 10 полей (N = 10), то описанным способом получится изменить только 25 записей (всего 250 изменений). При попытке изменить 26-ю запись BDE выдаст ошибку: "Operation not applicable. Too many open tables." Это сообщение вводит лишь в заблуждение, т.к. по правильному сообщение об ошибке должно быть следующим: "Too many record locks on table".
29. При работе BDE с таблицами Paradox операция изменения существующих записей (UPDATE) выполняется относительно долго, тогда как SELECT сравним по скорости с любой современной СУБД. На скорость UPDATE влияют разные факторы: длина имени поля, порядок расположения в таблице, тип поля, наличие индекса на данное поле и т.д. Если с двумя практически одинаковыми полями скорость UPDATE будет отличаться по порядку, то с этим вряд ли можно что-нибудь сделать, логика BDE нам неизвестна. Реальный случай: активная транзакция, в таблице Goods есть отнотипные поля Ostatok (порядковый номер 13) и Kolvo (порядковый номер 14). Для изменения значений этих полей у 255 записей требуется 22 секунды (это очень много), причем 150 мс занимает модификация поля Kolvo, а 21 секунду - модификация поля Ostatok. Скорость при изменении поля Ostatok падает в 140 раз.