

Приложение 1. FIBPlus 6.4.2 свойства, события, методы компонентов.

Оглавление

ТрFIBDatabase.....	7
Свойства.....	7
AliasName	7
BlobSwapSupport	7
CacheShemaOptions.....	7
Connected.....	7
ConnectParams.....	7
DBName.....	8
DBParams.....	8
DBParams.Add('user_name=XUSER');.....	8
DBParams.Add('password=XUSER');.....	8
DBParams.Add('lc_ctype=WIN1251');.....	8
DBParams.Add('sql_role_name=XROLE');.....	8
Дополнительные параметры подключения можно посмотреть в документации к серверу (APIGuide.pdf, DevGuide.pdf).....	8
DefaultTransaction.....	8
DefaultUpdateTransaction.....	9
DesignDBOptions.....	9
LibraryName.....	9
SaveAliasParamsAfterConnect.....	9
SQLDialect.....	9
SQLLogger.....	9
SynchronizeTime.....	9
TimeOut.....	10
UpperOldNames.....	10
UseLoginPrompt.....	10
UseRepositories.....	10
WaitForRestoreConnect.....	10
Описание событий.....	10
AfterConnect.....	10
AfterDisconnect.....	10
AfterEndTransaction.....	11
AfterLoadBlobFromSwap.....	11
AfterRestoreConnect.....	11
AfterSaveBlobToSwap.....	11
AfterStartTransaction.....	11
BeforeConnect.....	11
BeforeDisconnect.....	11
BeforeEndTransaction.....	11
BeforeLoadBlobFromSwap.....	12
BeforeSaveBlobToSwap.....	12
BeforeStartTransaction.....	12
OnAcceptCacheSchema.....	12
OnErrorRestoreConnect.....	12
OnLostConnect.....	12
Public свойства.....	13
Методы.....	16

TpFIBTransaction.....	20
Свойства.....	20
DefaultDatabase.....	20
Timeout.....	20
TimeoutAction.....	20
TPBMode.....	20
TRParams.....	20
UserKindTransaction.....	20
События.....	20
AfterEnd.....	20
AfterSQLExecute.....	20
AfterStart.....	21
BeforeEnd.....	21
BeforeSQLExecute.....	21
BeforeStart.....	21
OnTimeout.....	21
Методы.....	21
TpFIBQuery.....	23
Свойства.....	23
Conditions.....	23
Database.....	23
GoToFirstRecordOnExecute.....	23
Options.....	23
ParamsCheck.....	23
SQL.....	23
Transaction.....	23
Свойства.....	23
AfterExecute.....	23
BeforeExecute.....	23
OnBatchError.....	24
OnBatching.....	24
OnExecuteError.....	24
OnSQLChanging.....	24
TransactionEnded.....	24
TransactionEnding.....	24
Методы.....	24
Public свойства.....	26
TpFIBStoredProc.....	28
TpFIBDataset.....	29
Свойства.....	29
Active.....	29
Filter.....	29
FilterOptions.....	29
AllowedUpdateKinds.....	29
AutoCalcFields.....	29
AutoCommit.....	29
AutoUpdateOptions.....	29
CacheModelOptions.....	30
Conditions.....	31
Container.....	31
Database.....	31

DataSet_ID.....	31
Description.....	31
DefaultFormats.....	31
DetailCondition.....	32
FieldOriginalRule.....	32
Options.....	32
PrepareOptions.....	33
ReceiveEvents.....	34
RefreshTransactionKind.....	34
SQLs.....	34
SQLScreenCursor.....	34
Transaction.....	34
UniDirectional.....	34
UpdateRecordTypes.....	34
UpdateTransaction.....	35
События.....	35
AfterEndTransaction.....	35
AfterEndUpdateTransaction.....	35
AfterFetchRecord.....	35
AfterStartTransaction.....	35
AfterStartUpdateTransaction.....	35
BeforeEndTransaction.....	35
BeforeEndUpdateTransaction.....	35
BeforeFetchRecord.....	35
BeforeStartTransaction.....	35
BeforeStartUpdateTransaction.....	35
DatabaseDisconnected.....	35
DatabaseDisconnecting.....	36
DatabaseFree.....	36
OnApplyDefaultValue.....	36
OnAskRecordCount.....	36
OnCompareFieldValues.....	36
OnFieldChange.....	36
OnFillClientBlob.....	36
OnLockError.....	36
OnUserEvent.....	36
TransactionEnded.....	37
TransactionEnding.....	37
TransactionFree.....	37
Методы.....	37
ТрFIBUpdateObject.....	45
Свойства.....	45
Conditions.....	45
DataSet.....	45
ExecuteOrder.....	45
KindUpdate.....	45
OrderInList.....	45
ТDataSetContainer.....	46
Свойства.....	46
Active.....	46
MasterContainer.....	46

События.....	46
OnApplyDefaultValue.....	46
OnCompareFieldValues.....	46
OnDataSetError.....	46
OnDataSetEvent.....	46
OnUserEvent.....	47
Методы.....	47
TSIBfibEventAlerter.....	48
Свойства.....	48
AutoRegister.....	48
Database.....	48
Events.....	48
События.....	48
TpFIBErrorHandler.....	49
Свойства.....	49
Options.....	49
TpFIBClientDataSet.....	50
TpFIBDatasetProvider.....	50
TFIBSQLMonitor.....	51
Свойства.....	51
TraceFlags.....	51
События.....	51
OnSQL.....	51
TFIBSQLLogger.....	52
Свойства.....	52
Database.....	52
ActiveStatistics.....	52
ActiveLogging.....	52
ApplicationID.....	52
LogFileName.....	52
StatisticsParams.....	52
LogFlags.....	52
ForceSaveLog.....	53
Методы.....	53
TpFIBCustomService.....	54
Свойства.....	54
Handle: TISC_SVC_HANDLE.....	54
ServiceParamBySPB.....	54
Active.....	54
ServerName.....	54
Protocol.....	54
Params.....	54
LoginPrompt.....	54
LibraryName.....	54
SQLLogger.....	54
События.....	54
OnAttach.....	54
OnLogin.....	55
Методы.....	55
TpFIBServerProperties.....	55
Свойства.....	55

Option.....	55
DatabaseInfo.....	55
LicenseInfo.....	55
LicenseMaskInfo.....	55
VersionInfo.....	56
ConfigParams.....	56
Методы.....	56
ТрFIBSecurityService.....	56
Свойства.....	56
SecurityAction.....	56
UserName.....	56
Password.....	56
SQLRole.....	57
FirstName, MiddleName, LastName.....	57
UserID.....	57
GroupID.....	57
UserInfo.....	57
UserInfoCount.....	57
Методы.....	57
ТрFIBBackupService.....	58
Свойства.....	58
BackupFile.....	58
DatabaseName.....	58
Option.....	58

TrFIBDatabase

Компонент, инкапсулирующий соединение с базой данных сервера InterBase/Firebird

Свойства

AliasName

Группа опций, которая позволяет задать псевдоним базы данных. Псевдоним хранится в реестре Windows и содержит информацию, необходимую для подключения к БД.

BlobSwapSupport

Группа опций, которая позволяет задать режим кэширования BLOB-полей на клиенте.

```
TBlobSwapSupport = class(TPersistent)
  property Active: Boolean default False;
  property SwapDir: string;
  property MinBlobSizeToSwap: Integer default 0;
end;
```

Подробное описание использования этой опции приведено в разделе [Кэширование блоб-полей на клиенте](#)

CacheSchemaOptions

Группа опций, которая позволяет задать режим кэширования информации, получаемой из базы данных и от сервера InterBase, которая появляется при работе приложения: подготовленные запросы, служебная информация FIBPlus о значениях по умолчанию для полей, и многое другое.

```
TCacheSchemaOptions = class(TPersistent)
  property LocalCacheFile: string;
  property AutoSaveToFile: Boolean .. default False;
  property AutoLoadFromFile: Boolean .. default False;
  property ValidateAfterLoad: Boolean .. default True;
end;
```

FIBPlus позволяет использовать эту информацию многократно, а также не терять ее между сеансами.

Свойство **LocalCacheFile** позволяет задать имя файла, в котором будет сохраняться эта информация. **AutoSaveToFile** отвечает за автоматическую запись кеша в файл при закрытии приложения. **AutoLoadFromFile** отвечает за загрузку кеша из файла. И, наконец, **ValidateAfterLoad** указывает, стоит ли проверять сохраненный кеш после загрузки.

Connected

Группа опций, которая позволяет управлять состоянием соединения с базой данных.

ConnectParams

Группа опций, которая позволяет задать основные параметры подключения.

```
TConnectParams=class(TPersistent)
    property UserName: string;
    property RoleName: string;
    property Password: string;
    property CharSet : string;
end;
```

DBName

Группа опций, которая позволяет задать строку для подключения. Строка подключения может включать имя сервера, протокол и базу данных. Так, например, для локального подключения пишется только имя файла:

```
D:\IB7Book\Program\Base\Bpexempl.ib
```

- Для соединения с удаленным сервером по протоколу TCP/IP нужно дописать имя сервера:

```
netserver:D:\IB7Book\Program\Base\Bpexempl.ib
```

- Для соединения с удаленным сервером по протоколу TCP/IP и порту 3051 нужно дописать порт

```
netserver/3051:D:\IB7Book\Program\Base\Bpexempl.ib
```

- Для протокола NetBEUI

```
\\netserver\D:\IB7Book\Program\Base\Bpexempl.
```

- Для IPX/SPX (NetWare servers)

```
netserver@vol1:\IB7Book\Program\Base\Bpexempl.
```

- для UNIX или Linux сервера

```
netserver:/user/IB7Book/Program/Base/Bpexempl.
```

DBParams

Группа опций, которая позволяет задать параметры подключения. Частично пересекается с опциями [ConnectParams](#). Все возможные параметры можно добавлять напрямую сразу в буфер, например:

```
DBParams.Add('user_name=XUSER');
```

```
DBParams.Add('password=XUSER');
```

```
DBParams.Add('lc_ctype=WIN1251');
```

```
DBParams.Add('sql_role_name=XROLE');
```

Дополнительные параметры подключения можно посмотреть в документации к серверу (APIGuide.pdf, DevGuide.pdf).

DefaultTransaction

Группа опций, которая позволяет задать транзакцию по умолчанию. При присоединении

компонентов TpFIBDataSet, TpFIBQuery, TpFIBStoredProc к компоненту TpFIBDatabase свойство Transaction этого компонента будет автоматически скопировано в свойство Transaction нового компонента.

DefaultUpdateTransaction

Группа опций, которая позволяет задать транзакцию по умолчанию для обновления в компонентах TpFIBDataSet. Т.е. при заполнении свойства Database компонента TpFIBDataSet будет автоматически установлено свойство UpdateTransaction.

DesignDBOptions

Группа опций, которая позволяет задать поведение компонента TpFIBDatabase в режиме проектирования.

```
TDesignDBOption = (ddoIsDefaultDatabase, ddoStoreConnected, ddoNotSavePassword);  
TDesignDBOptions = set of TDesignDBOption;
```

ddoIsDefaultDatabase определяет, будет ли данный компонент соединением по умолчанию для добавляемых компонентов TpFIBDataSet, TpFIBQuery, TpFIBStoredProc. Если опция включена, то при добавлении новых компонентов в проект им будет автоматически присваиваться свойство Database и, как следствие, Transaction и UpdateTransaction.

ddoStoreConnected определяет, будет ли сохраняться состояние подключения при компиляции приложения.

ddoNotSavePassword определяет, будет ли сохраняться пароль для подключения в свойствах компонента.

LibraryName

Группа опций, которая позволяет задать имя клиентской библиотеки InterBase/Firebird. Данная опция может быть изменена только в зарегистрированной версии FIBPlus.

SaveAliasParamsAfterConnect

Если включена эта опция, то при удачном соединении с базой данных информация будет записана в реестр Windows как псевдоним (псевдоним задается в свойстве [AliasName](#))

SQLDialect

Эта опция позволяет задать диалект базы данных

SQLLogger

Эта опция позволяет задать компонент для логгирования обращений к БД. Подробнее смотрите в описании компонента TSQLLogger.

SynchronizeTime

Если включена эта опция, то при успешном подключении к БД время рабочей станции будет синхронизировано с временем на сервере.

TimeOut

Эта опция позволяет задать интервал времени в миллисекундах, после которого становится ясно, что произошла ошибка соединения с БД.

UpperOldNames

Если установлена эта опция, все имена в SQL-запросах будут переводиться в верхний регистр. Это позволяет избежать ошибок с именами объектов при написании запросов.

UseLoginPrompt

Если установлена эта опция, то, несмотря на пароль, прописанный в информации для подключения, будет вызваться диалог подключения.

UseRepositories

Эта опция позволяет задать режим использования репозитория:

```
TFIBUseRepository = (urFieldsInfo,urDataSetInfo,urErrorMessagesInfo);  
TFIBUseRepositories = set of TFIBUseRepository;
```

urFieldsInfo использовать репозиторий информации о полях

urDataSetInfo использовать репозиторий информации о датасетах

urErrorMessagesInfo использовать репозиторий информации об ошибках

Подробно работа с репозиториями описывается в разделе «Работа с репозиториями». Также Вы можете посмотреть примеры из комплекта FIBPlusExamples:

FIBPlusExamples\src\DataSetRepository\

FIBPlusExamples\src>ErrorMessagesRepository\

FIBPlusExamples\src\FieldsRepository\

WaitForRestoreConnect

Эта опция позволяет задать интервал времени в миллисекундах, по истечении которого будет производиться очередная попытка восстановить соединение. Подробнее об обработке потери соединения Вы можете прочитать в разделе [Обработка потери соединения](#)

Описание событий

AfterConnect

```
procedure (Sender: Tobject);
```

Событие возникает после соединения с базой данных посредством использования метода Open или установки в истину свойства Connected.

AfterDisconnect

```
procedure (Sender: Tobject);
```

Событие генерируется после отсоединения от БД методом Close или установкой в False свойства Connected.

AfterEndTransaction

procedure (EndingTR: TFIBTransaction; Action: TTransactionAction; Force: Boolean);

Событие возникает после завершения транзакции, при этом передаются завершившаяся транзакция и действие, которым она завершилась.

TTransactionAction = (TARollback, TARollbackRetaining, TACommit, TACommitRetaining)

AfterLoadBlobFromSwap

procedure (const TableName, FieldName: String; RecordKeyValues: array of Variant; const FileName: String);

Событие возникает после загрузки BLOB-поля с диска.

AfterRestoreConnect

procedure (Database: TFIBDatabase);

Событие возникает после того, как соединение было успешно восстановлено после разрыва. Подробное описание работы смотрите в разделе "Обработка потери соединения" и в демонстрационном примере:

FIBPlusExamples\src\ConnectionLost\

AfterSaveBlobToSwap

procedure (const TableName, FieldName: String; RecordKeyValues: array of Variant; const FileName: String);

Событие возникает после сохранения BLOB-поля на диск

AfterStartTransaction

procedure (Sender: TObject);

Событие возникает после старта транзакции.

BeforeConnect

procedure (Database: TFIBDatabase; LoginParams: TStrings; var DoConnect: Boolean);

Событие возникает непосредственно перед попыткой соединения с БД.

BeforeDisconnect

procedure (Sender: TObject);

Событие возникает перед отсоединением от БД.

BeforeEndTransaction

procedure (EndingTR: TFIBTransaction; Action: TTransactionAction; Force: Boolean);

Событие возникает перед завершением транзакции.

BeforeLoadBlobFromSwap

procedure (EndingTR: TFIBTransaction; Action: TTransactionAction; Force: Boolean);

Событие возникает перед загрузкой BLOB-поля из кеша. Подробнее смотрите раздел [Кеширование блоб-полей на клиенте](#)

BeforeSaveBlobToSwap

procedure (const TableName, FieldName: String; RecordKeyValues: array of Variant; Stream: TStream; var FileName: String; var CanSave: Boolean);

Событие возникает перед сохранением BLOB-поля на диск.

BeforeStartTransaction

procedure (Sender: TObject);

Событие возникает перед стартом транзакции.

OnAcceptCacheSchema

procedure (const ObjName: String; var Accept: Boolean);

Событие возникает в момент принятия решения о том, стоит ли загружать информацию из кеша для объекта ObjName. Здесь вы можете разрешить или запретить загрузку кеша объекта. Подробнее читайте в разделе [Кэширование метаданных](#).

OnErrorRestoreConnect

procedure (Database: TFIBDatabase; E: EFIBError; var Actions: TonLostConnectActions);

Событие возникает при ошибке очередной попытки восстановления подключения к БД. При этом вы можете проанализировать полученную ошибку и задать действие, которое следует выполнить в этой ситуации. Подробнее читайте в разделе [Обработка потери соединения](#).

TonLostConnectActions = (laTerminateApp, laCloseConnect, laIgnore, laWaitRestore);

Действие может принимать одно из следующих значений

laTerminateApp	закрыть приложение;
laCloseConnect	закрыть соединение;
laIgnore	игнорировать;
laWaitRestore	продолжить ожидание восстановления соединения.

OnLostConnect

procedure (Database: TFIBDatabase; E: EFIBError; var Actions: TonLostConnectActions);

Событие возникает в момент потери соединения. Подробное описание работы события смотрите в разделе "Обработка потери соединения" и в демонстрационном примере FIBPlusExamples\src\ConnectionLost\OnTimeout

procedure (Sender: TObject);

Событие возникает в момент превышения времени таймаута соединения или транзакции.

Public свойства

DBParamByDPB[const Idx: Integer]: string;

Свойство позволяет получить строковое значение параметра подключения по индексу

FIBBaseCount: Integer;

Свойство позволяет задать количество объектов сервера БД, TrFIBTransaction, TrFIBDataSet, TrFIBQuery.

FIBBases[Index: Integer]: TFIBBase;

Свойство позволяет получить объект БД по индексу

Handle: TISC_DB_HANDLE;

Дескриптор подключения к базе данных

HandleIsShared: Boolean;

Свойство возвращает True, если текущее подключение «разделяемое». Т.е., если соединение в TrFIBDatabase получилось путем присвоения свойству Handle значения из другого, уже существующего Handle. Актуально при использовании внутри dll.

TransactionCount: Integer;

Свойство позволяет получить количество транзакций

FirstActiveTransaction: TFIBTransaction;

Указатель на первую активную транзакцию

ActiveTransactionCount: Integer;

Свойство позволяет получить количество активных транзакций

Transactions[Index: Integer]: TFIBTransaction;

Свойство позволяет получить транзакцию по ее индексу

AttachmentID: Long;

Уникальный идентификатор подключения к БД.

Allocation: Long;

Информация от вызова IB API функции isc_database_info с параметром isc_info_allocation. Подробности смотрите в документации к серверу (APIGuide.pdf)

BaseLevel: Long;

Информация от вызова IB API функции isc_database_info с параметром isc_info_base_level. Подробности смотрите в документации к серверу (APIGuide.pdf)

DBFileName: string;

Свойство позволяет получить имя файла БД

DBSiteName: string;

Свойство позволяет получить имя хоста (сервера), на котором работает БД.

IsRemoteConnect: boolean;

Свойство возвращает True, если соединение удаленное

DBImplementationNo: Long;

Информация от вызова IB API функции isc_database_info с параметром isc_info_implementation. Подробности смотрите в документации к серверу (APIGuide.pdf)

DBImplementationClass: Long;

Информация от вызова IB API функции isc_database_info с параметром isc_info_implementation. Подробности смотрите в документации к серверу (APIGuide.pdf)

NoReserve: Long;

Информация от вызова IB API функции isc_database_info с параметром isc_info_no_reserve. Подробности смотрите в документации к серверу (APIGuide.pdf)

ODSMajorVersion: Long;

Свойство позволяет получить младшую цифру версии ODS

ODSMajorVersion: Long;

Свойство позволяет получить старшую цифру версии ODS

PageSize: Long;

Свойство позволяет получить размер страницы БД

Version: string;

Свойство позволяет получить строковое сообщение о версии сервера.

FBVersion: string;

Свойство позволяет получить строковое сообщение о версии сервера. Актуально только для клонов Firebird. Если строка пустая, то версия сервера InterBase, иначе - Firebird.

ServerMajorVersion: integer;

Свойство позволяет получить старшую цифру версии сервера

ServerMinorVersion: integer;

Свойство позволяет получить младшую цифру версии сервера

ServerBuild: integer;

Свойство позволяет получить номер сборки сервера

ServerRelease: integer;

Свойство позволяет получить номер релиза сервера

CurrentMemory: Long;

Свойство возвращает True, если текущая память занята сервером.

ForcedWrites: Long;

Свойство позволяет получить значение ForcedWrites для подключенной БД

MaxMemory: Long;

Свойство позволяет получить максимальный размер памяти, который занимал сервер для работы с данной БД.

SweepInterval: Long;

Свойство позволяет получить количество транзакций, после которого стартует автоматическая сборка мусора.

UserNames: TstringList;

Свойство позволяет получить список подключенных пользователей, для Firebird 1.5 - только для SuperServer.

TrFIBDatabase предоставляет доступ к функциям IB API. Подробное описание этих параметров можно найти в документации сервера (APIGuide.pdf, OpGuide.pdf)

NumBuffers: Long;

Fetches: Long;

Marks: Long;

Reads: Long;

Writes: Long;

BackoutCount: TstringList;

DeleteCount: TstringList;

ExpungeCount: TstringList;

InsertCount: TstringList;

PurgeCount: TstringList;

ReadIdxCount: TstringList;

ReadSeqCount: TstringList;

UpdateCount: TstringList;

IndexedReadCount[const TableName:string]: integer;

NonIndexedReadCount[const TableName:string]: integer;

InsertsCount[const TableName:string]: integer;

UpdatesCount[const TableName:string]: integer;

DeletesCount[const TableName:string]: integer;

AllModifications: integer;

LogFile: Long;

CurLogFileName: string;

CurLogPartitionOffset: Long;

Префикс WAL относится к серверам под NewWare.

NumWALBuffers: Long;

WALBufferSize: Long;

WALCheckpointLength: Long;

WALCurCheckpointInterval: Long;

WALPrvCheckpointFilename: string;

WALPrvCheckpointPartOffset: Long;

WALGroupCommitWaitUSecs: Long;

WALNumIO: Long;

WALAvergeIOSize: Long;

WALNumCommits: Long;

WALAvergeGroupCommitSize: Long;

DBSQLDialect: Word;

Это свойство позволяет получить диалект БД

ReadOnly: Long;

Это свойство возвращает True, если БД только для чтения

DatabaseName: string;

Это свойство позволяет получить имя БД

DifferenceTime: double;

Это свойство позволяет получить дельту времени на серверной машине и на машине клиента.

ServerActiveTransactions: TstringList;

Это свойство позволяет получить список активных транзакций на сервере

OldestTransactionID: Long;

Это свойство позволяет получить ID старейшей транзакции. Подробности читайте в документации по серверу.

OldestActiveTransactionID: Long;

Это свойство позволяет получить ID старейшей заинтересованной транзакции. Подробности читайте в документации по серверу

ClientLibrary: TlibClientLibrary;

Это свойство позволяет получить ссылку на клиентскую библиотеку

SQLStatisticsMaker: ISQLStatMaker;

Это свойство позволяет получить ссылку на счетчик статистики

Методы

procedure RegisterBlobFilter (BlobSubType:integer; EncodeProc, DecodeProc: PIBBlobFilterProc);

Этот метод позволяет регистрировать Blob-Filter для подтипа BlobSubType

procedure RemoveBlobFilter (BlobSubType:integer);

Этот метод позволяет удалить Blob-Filter для подтипа BlobSubType

procedure CheckActive;

Этот метод позволяет проверить, подключена ли БД. В том случае, если БД не подключена, метод генерирует исключение.

procedure CheckInactive;

Этот метод противоположен предыдущему методу CheckActive. Он позволяет проверить, не подключена ли БД. В том случае, если БД подключена, метод генерирует исключение.

procedure CheckDatabaseName;

Этот метод проверяет, заполнено ли имя базы данных. Если имя БД не заполнено, он генерирует исключение.

procedure Close;

Этот метод позволяет закрыть соединение.

procedure CreateDatabase;

Этот метод позволяет создать БД. Базовые параметры подключения должны быть заполнены.

procedure DropDatabase;

Этот метод позволяет физически удалить БД.

function FindTransaction(TR: TFIBTransaction): Integer;

Этот метод возвращает индекс транзакции.

procedure ForceClose;

Этот метод позволяет принудительно закрыть БД.

function IndexOfDBConst(const st: string): Integer;

Этот метод возвращает индекс константы БД

procedure Open; **virtual**;

Этот метод позволяет открыть БД

function TestConnected: Boolean;

Этот метод позволяет тестировать подключение к БД. Он возвращает True в случае удачной попытки подключения.

function GetServerTime:TDateTime;

Этот метод возвращает серверное время.

function ClientVersion: string;

Этот метод возвращает версию клиентской библиотеки (WI-V6.3.2.4731 Firebird 1.5) .

function ClientMajorVersion:integer;

Этот метод возвращает старшую цифру версии клиенткой библиотеки.

function ClientMinorVersion: Integer;

Этот метод возвращает младшую цифру версии клиенткой библиотеки.

function IsFirebirdConnect: Boolean;

Этот метод возвращает значение True, если есть соединение с сервером Firebird.

function NeedUnicodeFieldsTranslation: Boolean;

Это внутренний флаг. Для клиентской библиотеки метод возвращает информацию о том, нужна ли UTF перекодировка для полей UNICODE_FSS. Данная информация нужна не всегда, это зависит от CHAR_SET подключения.

function IsUnicodeConnect :boolean;

Этот метод возвращает True, если это соединение CHARSET UNICODE_FSS.

function GetContextVariable (ContextSpace:TFBContextSpace;**const** VarName:string ;
aTransaction: TFIBTransaction=nil): Variant;
TFBContextSpace = (csSystem, csSession, csTransaction);

Этот метод возвращает значение контекстной переменной. Только для Firebird версии 2.0.

procedure SetContextVariable(ContextSpace:TFBContextSpace;**const**
VarName,VarValue:string ;aTransaction:TFIBTransaction = nil);

Этот метод позволяет установить значение контекстной переменной. Только для Firebird версии 2.0.

procedure StartTransaction;

Этот метод позволяет стартовать транзакцию.

procedure Commit;

Этот метод позволяет завершить транзакцию по Commit.

procedure Rollback;

Этот метод позволяет завершить транзакцию по Rollback

procedure CommitRetaining;

Этот метод позволяет завершить транзакцию по CommitRetaining

procedure RollbackRetaining;

Этот метод позволяет завершить транзакцию по RollbackRetaining.

function Gen_Id(**const** GeneratorName: string; Step: Int64; aTransaction:
TFIBTransaction = nil): Int64;

Этот метод позволяет получить значение генератора с именем, шагом и в определенной транзакции. Если транзакция не задана, то она будет создана автоматически.

function Execute(**const** SQL: string): boolean;

Этот метод позволяет выполнить sql-запрос и возвращает True в случае удачного выполнения запроса.

procedure CreateGUIDDomain;

Этот метод позволяет создать домен FIBGUID char(16) character set octets в БД.

function QueryValue(**const** aSQL: string; FieldNo: integer; aTransaction:
TFIBTransaction=nil): Variant; **overload**;

Этот метод позволяет получить значение поля с номером FieldNo запроса с текстом aSQL. Если транзакция не задана, то она будет создана автоматически.

function QueryValue(**const** aSQL: string;FieldNo:integer; ParamValues:array of

```
variant;aTransaction:TFIBTransaction=nil ):Variant; overload;
```

Эта функция аналогична функции, описанной выше, но дает возможность передавать параметры.

```
function QueryValues(const aSQL: string;aTransaction:TFIBTransaction=nil):Variant; overload;
```

Эта функция аналогична функции, описанной выше, но позволяет получить запись целиком в вариантный массив.

```
function QueryValues(const aSQL: string; ParamValues:array of variant;  
aTransaction: TFIBTransaction=nil ): Variant; overload;
```

Эта функция аналогична функции, описанной выше, но позволяет получить запись целиком в вариантный массив и, кроме того, передавать параметры.

```
function QueryValueAsStr(const aSQL: string;FieldNo:integer):string; overload;
```

Эта функция возвращает строковое представление поля.

```
function QueryValueAsStr(const aSQL: string;FieldNo:integer; ParamValues:array  
of variant):string; overload;
```

Эта группа перегруженных методов QueryValueAsStr позволяет получить результатом строковое представление выполнения SQL-запроса первой строки, которую он выполняет.

Внимание: Семь методов, описанные выше, должны возвращать только одну запись!

```
function EasyFormatsStr: Boolean;
```

Этот метод позволяет выяснить, нужно ли обрамлять кавычками имена таблиц и полей или нужно приводить их к верхнему регистру.

TrFIBTransaction

Это очень важный компонент, инкапсулирующий транзакцию, без которого невозможно обойтись.

Свойства

DefaultDatabase

Это свойство возвращает базу данных для транзакции

Timeout

Это свойство возвращает время в миллисекундах, после которого будет выполнен TimeoutAction

TimeoutAction

Это свойство возвращает действие, которое выполнится при наступлении события Timeout. Оно описано следующим образом:

```
TTransactionAction = (TARollback, TARollbackRetaining, TACcommit, TACcommitRetaining);
```

TPBMode

Используя это свойство, можно настроить параметры транзакции. Его возможные параметры таковы:

```
TPBMode = (tpbDefault, tpbReadCommitted, tpbRepeatableRead)
```

TRParams

В этом свойстве можно задать свои собственные параметры

UserKindTransaction

Используя это свойство, можно выбрать пользовательский вариант настроек. Наиболее удобно работать в редакторе «Edit Transaction Parameters», который можно вызвать из контекстного меню компонента.

События

AfterEnd

```
procedure (EndingTR: TFIBTransaction; Action: TTransactionAction; Force: Boolean);
```

Это свойство генерируется при завершении транзакции

AfterSQLExecute

```
procedure (Query: TFIBQuery; SQLType: TFIBSQLTypes);
```

Это свойство генерируется после выполнения SQL.

AfterStart

procedure (Sender: TObject);

Это свойство генерируется после старта транзакции.

BeforeEnd

procedure (EndingTR: TFIBTransaction; Action: TTransactionAction; Force: Boolean);

Это свойство генерируется после завершения транзакции.

BeforeSQLExecute

procedure (Query: TFIBQuery; SQLType: TFIBSQLTypes);

Это свойство генерируется перед выполнением запроса.

BeforeStart

procedure (Sender: TObject);

Это свойство генерируется перед стартом транзакции

OnTimeout

procedure (Sender: TObject);

Это свойство генерируется при таймауте.

Методы

function MainDatabase: TFIBDatabase;

Этот метод возвращает основную базу данных.

function FindDatabase(db: TFIBDatabase): Integer;

Этот метод возвращает индекс базы данных.

procedure CheckInTransaction;

Этот метод генерирует исключение, если транзакция активна.

procedure CheckNotInTransaction;

Этот метод генерирует исключение, если транзакция не активна.

procedure StartTransaction; **virtual**;

Этот метод стартует транзакцию.

procedure Commit; **virtual**;

Этот метод подтверждает транзакцию по Commit.

procedure CommitRetaining; **virtual**;

Этот метод подтверждает транзакцию по commitRetaining.

procedure Rollback; **virtual**;

Этот метод откатывает транзакцию по Rollback.

procedure RollbackRetaining; **virtual**;

Этот метод откатывает транзакцию по RollbackRetaining.

procedure ExecSQLImmediate(**const** SQLText:string);

Этот метод выполняет запрос с использованием `isc_dsql_execute_immediate` без подготовки, и, как следствие, без рутинных операций с Handle запросов. Подробнее смотрите в IB API.

procedure SetSavePoint(**const** SavePointName:string);

Этот метод создает точку восстановления (Savepoint).

procedure RollBackToSavePoint(**const** SavePointName:string);

Этот метод отменяет все изменения, сделанные до точки восстановления (Savepoint).

procedure ReleaseSavePoint(**const** SavePointName: string);

Этот метод освобождает все точки восстановления (Savepoint).

procedure CloseAllQueryHandles;

Этот метод закрывает все Handle запросов, ассоциированных с этой транзакцией.

function IsReadCommittedTransaction:boolean;

Этот метод возвращает True, если транзакция имеет параметр ReadCommitted.

procedure StartTransaction; **override**;

Этот метод стартует транзакцию.

function FIBQueryCount: integer;

Этот метод возвращает количество запросов, связанных с транзакцией.

function FIBDataSetsCount:integer;

Этот метод возвращает количество датасетов, связанных с транзакцией.

TrFIBQuery

Свойства

Conditions

Смотрите подробное описание в разделе "Выполнение SQL-запросов. Условия" о TrFIBDataSet Руководства пользователя.

Database

Это свойство возвращает базу данных, для которой будет работать транзакция.

GoToFirstRecordOnExecute

Свойство влияет на выполнение селективных SQL. Если оно установлено в True, то первый fetch будет сделан сразу же после выполнения. Если оно установлено в False, то первый fetch будет сделан только после вызова Next. Опция имеет смысл только для селективных запросов.

Options

Это свойство аналогично свойствам TrFIBDataSet и описывается следующим образом:

```
TrFIBQueryOption = (qoStartTransaction, qoAutoCommit, qoTrimCharFields,  
qoNoForceIsNull, qoFreeHandleAfterExecute);  
TrFIBQueryOptions=set of TrFIBQueryOption;
```

ParamsCheck

Установите это свойство в False, чтобы выполнять DDL запросы через компонент.

SQL

Это свойство возвращает текст выполняемого запроса

Transaction

Это свойство возвращает транзакцию, в контексте которой будет выполнен запрос.

Свойства

AfterExecute

```
procedure (Sender: Tobject); TnotifyEvent;
```

Это свойство генерируется после выполнения SQL-запроса (вызова TrFIBQuery.ExecQuery, ExecProc, ExecWP).

BeforeExecute

```
procedure (Sender: Tobject);
```

Это свойство генерируется перед выполнением.

OnBatchError

```
procedure (E: EFIBError; var BatchErrorAction: TbatchErrorAction);
```

Это свойство генерируется при ошибке выполнения пакетной обработки.

OnBatching

```
procedure (BatchOperation: TBatchOperation;  
  RecNumber: Integer; var BatchAction: TbatchAction);
```

Это свойство генерируется при очередном выполнении пакетной обработки.

OnExecuteError

```
procedure (pFIBQuery: TpFIBQuery; E: EFIBError; var Action: TdataAction);
```

Это свойство генерируется при ошибке выполнения запроса.

OnSQLChanging

```
procedure (Sender: Tobject);
```

Это свойство генерируется при изменении текста SQL.

TransactionEnded

```
procedure (Sender: Tobject);
```

Это свойство генерируется после завершения транзакции.

TransactionEnding

```
procedure (Sender: Tobject);
```

Это свойство генерируется при завершении транзакции.

Методы

```
function TableAliasForField(FieldIndex:integer):string; overload;  
function TableAliasForField(const aFieldName:string):string; overload;
```

Эти методы возвращают псевдоним таблицы для поля по индексу поля либо по его имени.

Подробнее о методах для пакетной обработки читайте в разделе "Выполнение SQL-Запросов. Пакетная обработка" Руководства пользователя.

```
function BatchInput(InputObject: TFIBBatchInputStream) :boolean;  
function BatchOutput(OutputObject: TFIBBatchOutputStream):boolean;  
procedure BatchInputRawFile(const FileName:string);  
procedure BatchOutputRawFile(const FileName:string;Version:integer=1);  
procedure BatchToQuery(ToQuery:TFIBQuery;Mappings:TStrings);
```

```
procedure CheckValidStatement;
```

Этот метод проверяет SQL на валидность.

```
procedure Close;
```

Этот метод закрывает запрос.

function Current: TFIBXSQlda;

Этот метод возвращает буфер текущей записи.

procedure ExecQuery; **virtual**;

Этот метод выполняет запрос.

procedure FreeHandle;

Этот метод закрывает запрос с флагом DSQL_drop. При этом Handle запроса на сервере уничтожается и повторно его использовать уже нельзя.

function Next: TFIBXSQlda;

Этот метод производит fetch следующей записи.

procedure Prepare;

Этот метод подготавливает запрос

Данные функции аналогичны одноименным функциям TpfibDataSet:

function FieldByName(const FieldName: string): TFIBXSQlvar;

function FindField(const FieldName: string): TFIBXSQlvar;

function FN(const FieldName: string): TFIBXSQlvar;

function FieldByOrigin(const TableName, FieldName: string): TFIBXSQlvar; **overload**;

function SQLFieldName(const aFieldName: string): string;

procedure PrepareArrayFields;

procedure PrepareArraySqlVar(SqlVar: TFIBXSQlvar; const RelName, SQLName: string;
IsField: boolean);

procedure SetParamValues(const ParamValues: array of Variant); **overload**;

procedure SetParamValues(const ParamNames: string; ParamValues: array of
Variant); **overload**;

procedure ExecWP(const ParamValues: array of Variant); **overload**;

procedure ExecWP(const ParamNames: string; ParamValues: array of Variant);
overload;

procedure ExecWPS(const ParamSources: array of ISQLObject);

Методы позволяют выполнить запрос и одновременно передать параметры этого запроса на сервер.

function IsProc :boolean;

Этот метод установлен в True, если текст SQL - это выполнение процедуры (execute procedure xxx).

function ParamByName(const ParamName: string): TFIBXSQlvar;

Этот метод возвращает параметр по его имени.

function FindParam (const aParamName: string): TFIBXSQlvar;

Этот метод ищет параметр по имени (в том числе и в макросах)

procedure RestoreMacroDefaultValues;

Этот метод устанавливает значения макросов в значения по умолчанию.

function FieldCount: integer;

Этот метод возвращает количество полей запроса.

```
function ParamCount:integer;
```

Этот метод возвращает количество параметров.

```
function ParamName(ParamIndex:integer):string;
```

Этот метод возвращает параметр по индексу.

```
function FieldName(FieldIndex:integer):string;
```

Этот метод возвращает поле по индексу.

```
function FieldsCount:integer;
```

Этот метод возвращает количество полей.

```
function FieldExist(const FieldName:string; var FieldIndex:integer):boolean;
```

Этот метод проверяет существование поля с именем FieldName.

```
function ParamExist(const ParamName:string; var ParamIndex:integer):boolean;
```

Этот метод проверяет существование параметра с именем ParamName.

```
function FieldValue(const FieldName:string;Old:boolean):variant; overload;
```

```
function FieldValue(const FieldIndex:integer;Old:boolean):variant; overload;
```

```
function ParamValue(const ParamName:string):variant; overload;
```

```
function ParamValue(const ParamIndex:integer):variant; overload;
```

Этот метод возвращает значения полей и параметров

```
function ReadySQLText(ForChangeExecSQL:boolean=True): string;
```

Этот метод возвращает реальный текст SQL, который уходит на сервер при использовании параметров и макросов.

```
procedure AssignProperties(Source: TFIBQuery);
```

Этот метод копирует все свойства компонента-параметра

Public свойства

```
property Bof: Boolean read FBOF;
```

Это свойство установлено в True, если достигнуто начало набора данных.

```
property DBHandle: PISC_DB_HANDLE read GetDBHandle;
```

Это свойство возвращает Handle запроса.

```
property Eof: Boolean read GetEOF;
```

Это свойство установлено в True, если достигнут конец выборки

```
property FldByName(const FieldName: string): TFIBXSQLVAR read FieldByName;
```

Это свойство возвращает значение поля по имени.

```
property Fields(const Idx: Integer): TFIBXSQLVAR read GetFields;
```

Это свойство возвращает значение поля по индексу.

```
property FieldIndex(const FieldName: string): Integer read GetFieldIndex;
```

Это свойство возвращает индекс поля по имени.

```
property Open: Boolean read Fopen;
```

Это свойство показывает, активен ли запрос.

property Params: TFIBXSQLDA **read** GetSQLParams;

Это свойство возвращает массив параметров запроса.

property Plan: **string** **read** GetPlan;

Это свойство возвращает план выполнения запроса.

property Prepared: Boolean **read** Fprepared;

Это свойство установлено в True, если запрос подготовлен на сервере.

property RecordCount: Integer **read** GetRecordCount;

Это свойство возвращает количество отфетченных записей.

property RowsAffected: Integer **read** GetRowsAffected;

Это свойство возвращает количество записей, измененных запросом.

property AllRowsAffected: TAllRowsAffected **read** GetAllRowsAffected;

Это свойство возвращает количество операций, произведенных запросом.

```
TAllRowsAffected =
record
    Updates: integer;
    Deletes: integer;
    Selects: integer;
    Inserts: integer;
end;
```

property SQLType: TFIBSQLTypes **read** FSQLType;

TFIBSQLTypes = (SQLUnknown, SQLSelect, SQLInsert, SQLUpdate, SQLDelete, SQLDDL, SQLGetSegment, SQLPutSegment, SQLExecProcedure, SQLStartTransaction, SQLCommit, SQLRollback, SQLSelectForUpdate, SQLSetGenerator, SQLSavePointOperation)

property TRHandle: PISC_TR_HANDLE **read** GetTRHandle;

Это свойство возвращает Handle – внутренний дескриптор соединения на сервере, который идентифицирует каждую операцию с БД, сообщая, от какого соединения приходят запросы. Подробности смотрите в APIGuide.pdf

property ProcExecuted: boolean **read** FProcExecuted **write** FprocExecuted;

Это свойство установлено в True, если процедура в запросе выполнена через execute procedure.

property OnSQLFetch: TOnSQLFetch **read** FOnSQLFetch **write** FonSQLFetch;

TOnSQLFetch = **procedure** (RecordNumber:integer; **var** StopFetching:boolean) **of** **object**;

Это свойство генерируется при fetch очередной записи.

property CursorName : **string** **read** FCursorName **write** FcursorName;

Это свойство возвращает имя курсора

property OrderClause: **string** **read** GetOrderString **write** SetOrderString;

Это свойство возвращает секцию order by

```
property GroupByClause: string read GetGroupByString write SetGroupByString;
```

Это свойство возвращает секцию group by

```
property FieldsClause: string read GetFieldsClause write SetFieldsClause;
```

Это свойство возвращает секцию полей

```
property PlanClause: string read GetPlanClause write SetPlanClause;
```

Это свойство позволяет задать план выполнения запроса.

```
property SQLKind: TSQLKind read GetSQLKind;
```

TSQLKind = (skUnknown, skSelect, skUpdate, skInsert, skDelete, skExecuteProc, skDDL, skExecuteBlock) ;

TpFIBStoredProc

Это наследник TpFIBQuery. Отличается свойством StoredProcName, которое позволяет задать имя хранимой процедуры.

TrFIBDataset

Является потомком TDataSet и поддерживает все его методы и свойства. Здесь будут перечислены только специфические свойства TrFIBDataSet.

Свойства

Active

Смотрите справку по TDataSet.

Filter

Смотрите описание свойства в справке по Delphi/C++Builder, оно ничем не отличается от свойств в TDataSet.

FilterOptions

Смотрите описание свойства с справке по Delphi/C++Builder, оно ничем не отличается от свойств в TDataSet.

AllowedUpdateKinds

Объявленные как

```
TUpdateKind = (ukModify, ukInsert, ukDelete);  
TUpdateKinds = set of TUpdateKind;
```

Это свойство позволяет задать режим обновления датасета. ukModify определяет, будут ли производиться модификации датасета, ukInsert вставки и ukDelete удаления. Если опция выключена, но в момент операции библиотека будет вызывать тихое исключение Abort, то операция не будет выполняться.

AutoCalcFields

Значение опции аналогично значению опции для TDataSet.

AutoCommit

Если опция установлена в True, то, в зависимости от настроек [UpdateTransaction](#), после каждой модифицирующей операции Insert/Update/Delete будет вызываться принудительный Commit/CommitRetaining.

AutoUpdateOptions

Очень важная группа опций. Если понимать, как работает данная группа опций, можно избежать значительного количества проблем.

```
TAutoUpdateOptions= class (TPersistent)  
  property AutoParamsToFields: Boolean .. default False;  
  property AutoReWriteSqls: Boolean .. default False;  
  property CanChangeSQLs: Boolean .. default False;  
  property GeneratorName: string;  
  property GeneratorStep: Integer .. default 1;
```

```

property KeyFields: string;
property ParamsToFieldsLinks: TStrings;
property SeparateBlobUpdate: Boolean .. default False;
property UpdateOnlyModifiedFields: Boolean .. default False;
property UpdateTableName: string;
property WhenGetGenID: TWhenGetGenID .. default wgNever;
end;

```

```
TWhenGetGenID= (wgNever, wgOnNewRecord, wgBeforePost);
```

Если опция `AutoRewriteSQLs` установлена в `True`, то, при наличии пустых `SQLText` для `InsertSQL`, `UpdateSQL`, `DeleteSQL` и `RefreshSQL`, будет автоматически производиться их генерация на основе `KeyFields`, `UpdateTableName`.

Если опция `CanChangeSQLs` установлена в `True`, то разрешена перезапись непустых SQL.

Опции `GeneratorName` и `GeneratorStep` задают, соответственно, имя и шаг генератора

Опция `KeyFields` содержит список ключевых полей

Опция `SeparateBlobUpdate` управляет записью BLOB полей в базу данных. Если эта опция установлена в `True`, то сначала будет производиться запись строки без BLOB поля, а затем, в случае успеха, будет записываться само BLOB поле.

Если опция `UpdateOnlyModifiedFields` установлена в `True` и если также установлены `CanChangeSQLs`, то для каждой операции обновления будет формироваться новый SQL-запрос, в котором будут представлены только реально измененные поля.

`UpdateTableName` должна содержать имя обновляемой таблицы

`WhenGetGenId` позволяет задать режим использования генератора: никогда, на новую запись, непосредственно перед операцией `Post`.

Т.е., при использовании настроек `AutoUpdateOptions` FIBPlus позволяет избавиться от генерации SQL в режиме `design time` и переложить эту задачу на время исполнения программы. Для этого нужно лишь приписать имя обновляемой таблицы и ключевое поле.

Данные код взят из примера `AutoUpdateOptions`:

```

pFIBDataSet1.SelectSQL.Text := 'SELECT * FROM EMPLOYEE';
pFIBDataSet1.AutoUpdateOptions.AutoReWriteSqls := True;
pFIBDataSet1.AutoUpdateOptions.CanChangeSQLs := True;
pFIBDataSet1.AutoUpdateOptions.UpdateOnlyModifiedFields := True;
pFIBDataSet1.AutoUpdateOptions.UpdateTableName := 'EMPLOYEE';
pFIBDataSet1.AutoUpdateOptions.KeyFields := 'EMP_NO';
pFIBDataSet1.AutoUpdateOptions.GeneratorName := 'EMP_NO_GEN';
pFIBDataSet1.AutoUpdateOptions.WhenGetGenID := wgBeforePost;
pFIBDataSet1.Open;

```

CachedUpdates

Это свойство соответствует аналогичному свойству компонента `TDataSet`

CacheModelOptions

Это «революционное» новшество FIBPlus. Используя эту опцию, Вы можете выбрать модель хранения кеша датасета. Подробное описание работы в режиме ограниченного кеша описано в разделе ...

```

TCacheModelOptions = class (TPersistent)
property CacheModelKind: TCacheModelKind default cmkStandard;

```

```

    property BufferChunks: Integer default vBufferCacheSize;
    property PlanForDescSQLs: string;
end;
TCacheModelKind= (cmkStandard, cmkLimitedBufferSize);

```

CacheModelKind может принимать значение cmkStandard cmkLimitedBuffersSize. При установлении этой опции в cmkLimitedBuffersSize датасет будет загружать в память приложения столько записей, сколько установлено в свойстве BuffersChunks. В дополнении к этому можно указать план для выполнения обратных запросов (быстрое извлечение последних записей выборки).

Conditions

Подробнее о дополнительных условиях читайте в Руководстве пользователя.

Container

Подробнее об использовании контейнеров смотрите описание компонента TpFIBDatasetContainer.

Database

Это свойство используется для подключения датасета к базе данных.

DataSet_ID

Свойство позволяет задать код датасета, хранящегося в репозитории. Если значение свойства отлично от нуля, то при открытии датасета будет произведена выгрузка настроек из репозитория датасетов и заполнено свойство Description. Подробнее о использовании репозитория FIBPlus читайте в разделе «Репозитории FIBPlus» а также смотрите демонстрационные примеры XXXRepository

Description

Это свойство описание датасета, которое используется в репозитории датасетов.

DefaultFormats

Эта группа опций позволяет задать форматы отображения стандартных полей для этого датасета. Описание выглядит следующим образом:

```

TFormatFields = class (TPersistent)
property DateTimeDisplayFormat: string;
property NumericDisplayFormat: string;
property NumericEditFormat: string;
property DisplayFormatDate: string;
property DisplayFormatTime: string;
end;

```

Т.е., вы можете задать формат отображения для полей TdateTimeField/TdataField/TtimeField, отображения и редактирования TnumericField.

DetailCondition

Это группа опций облегчает работу в режиме master-detail. Вот ее описание:

```
TDetailCondition=(dcForceOpen,dcIgnoreMasterClose,dcForceMasterRefresh,dcWaitEndMasterScroll);
```

```
TDetailConditions= set of TDetailCondition;
```

dcForceOpen	если эта опция включена, то детальные датасет будет открываться автоматически при открытии мастера
dcIgnoreMasterClose	опция означает, что детальный датасет не будет закрываться в случае закрытия мастера
dcForceMasterRefresh	при обновлении детального датасета будет производиться обновление мастер-датасета – будет вызываться его RefreshSQL
dcWaitEndMasterScroll	опция означает, что при прокрутке мастера выжидается некоторое время и только потом происходит переоткрытие детали. Опция позволяет избежать лишней работы при прокрутке мастер-датасета

FieldOriginalRule

Это свойство позволяет руководить заполнением свойства TField.Origin и может принимать несколько значений:

```
TfieldOriginRule =(forNoRule, forTableAndFieldName, forClientFieldName, forTableAliasAndFieldName);
```

Значениями, соответственно, являются: не заполнять, для таблицы и имени поля, просто имя поля на клиенте, для алиаса таблицы и имени поля. Свойство заполняется только в режиме выполнения программы.

Options

Эта группа опций - одна из основных и необходимых для понимания тонкостей работы с TpFIBDataSet. Описание группы опций:

```
TpFIBDsOption = (poTrimCharFields, poRefreshAfterPost, poRefreshDeletedRecord, poStartTransaction, poAutoFormatFields, poProtectedEdit, poKeepSorting, poPersistentSorting, poVisibleRecno, poNoForceIsNull, poFetchAll, poFreeHandlesAfterClose, poCacheCalcFields);
```

```
TpFIBDsOptions= set of TpFIBDsOption;
```

poStartTransaction	стартовать транзакцию при открытии датасета, если она не активна
poTrimCharFields	усекать концевые пробелы для полей типа CHAR/VARCHAR
poRefreshAfterPost	выполнять RefreshSQL, после фиксации изменений в БД, после метода Post;
poRefreshDeletedRecord	удалять из кеша запись после выполнения RefreshSQL, если тот не вернул записи
poAutoFormatFields	использовать автоматическое форматирование для датасета
poProtectedEdit	использовать защищенное редактирование (подробнее описано в разделе Защищенное редактирование)
poKeepSorting	помещать добавленные либо измененные записи в правильную позицию буфера, которая отвечает локальной сортировке датасета. Подробнее работа с локальной сортировкой будет рассмотрена в разделе Локальная сортировка
poPersistentSorting	сохранять сортировку и восстанавливать при следующем открытии датасета.

poVisibleRecno	при включенной опции добавляется поле RecNo
poNoForcelsNull	если опция выключена, то для параметров в случае если они NULL условие where вида FIELD1 = :FIELD1 будет заменяться на where FIELD IS NULL. В некоторых случаях такое поведение может быть нежелательно. Например, если нужно вернуть информацию о типе параметра, следует включить эту опцию.
poFetchAll	при включенной опции будет производиться полный fetch данных, что может потребоваться, например, для справочников
poFreeHandlesAfterClose	эта опция отвечает за то, чтобы автоматически сразу же после закрытия запроса (модифицирующий запрос, либо селективный после fetch всех записей) вызывалось автоматическое освобождения ресурсов, связанных с запросом FreeHandle
poCacheCalcFields	при включенной опции будут кэшироваться вычисляемые поля.

PrepareOptions

Это также ключевые опции для тонкой настройки работы датасета

```
TpPrepareOption = (pfSetRequiredFields, pfSetReadOnlyFields,
pfImportDefaultValues, psUseBooleanField, psUseGuidField, psSQLINT64ToBCD,
psApplyRepository, psGetOrderInfo, psAskRecordCount, psCanEditComputedFields,
psSetEmptyStrToNull, psSupportUnicodeBlobs, psUseLargeIntField);
TpPrepareOptions=set of TpPrepareOption;
```

pfSetRequiredFields	если установлена эта опция, то при открытии датасета будут установлены поля, которые обязательны для заполнения;
pfSetReadOnlyFields	если установлена эта опция, то при открытии датасета будут заполняться поля только для чтения;
pfImportDefaultValues	если установлена эта опция, то будут извлекаться значения по умолчанию (default), и при вставке поля датасета будут заполняться этими значениями.
psUseBooleanField	если установлена эта опция, то будет использоваться эмуляция Boolean-полей. Подробнее смотрите в разделе Эмуляция Boolean-полей
psUseGuidField	если установлена эта опция, то будут использоваться поля типа GUID. Подробнее смотрите в разделе Работа с GUID-полями
psSQLINT64ToBCD	если установлена эта опция, то поля типа BIGINT будут переведены в поля типа BCD. Эта опция позволяет не терять точность при работе с BIGINT полями.
PsApplyRepository	эта опция позволяет вычитать информацию из репозитория полей при открытии датасета. Подробнее смотрите в разделе «Репозитории FIBPlus» Руководства пользователя.
psGetOrderInfo	эта опция позволяет получить порядок сортировки при открытии выборки. Используйте в сочетании с poKeepSorting.
psAskRecordCount	если установлена эта опция, то будет генерироваться дополнительный запрос select count(*) from ... и в свойстве RecordCount будет правильное количество записей. Используйте эту опцию с осторожностью, так как: 1) делается дополнительный запрос на сервер, который может начать сборку мусора; 2) не для всех запросов возможен корректный результат
PsCanEditComputedFields	если установлена эта опция, то разрешено редатирование

	вычисляемых полей.
psSetEmptyStrToNull	если установлена эта опция, то пустые строки будут изменены на NULL
psSupportUnicodeBlobs	установите эту опцию в True, если вы используете поля типа Unicode BLOB
psUseLargeIntField	установите эту опцию в True, если вы используете поля типа LargeInt.

Также для PrepareOptions и Options есть редакторы, облегчающие комплексную настройку датасета. Помимо этого, в инструменте FIBPlusTools эти опции можно настроить для компонентов TpFIBDataSet, вновь добавляемых в проект.

ReceiveEvents

Это свойство возвращает список пользовательских событий TpFIBDatasetContainer, на которые будет реагировать датасет.

RefreshTransactionKind

Это свойство позволяет задать, в какой транзакции (читающей или обновляющей) будет вызваться RefreshSQL.

SQLs

Это свойство содержит тексты SQL-запросов.

SQLScreenCursor

Это свойство позволяет задать курсор для долгих операций датасета.

Transaction

Это свойство возвращает транзакцию, в которой производится чтение данных.

UniDirectional

Если это свойство установлено в True, ваш датасет станет однонаправленным и не будет кэшировать результаты выполнения. В каждый момент будет доступна лишь одна запись. Это может понадобиться, например, для больших отчетов.

UpdateRecordTypes

Это свойство позволяет задать режим отображения записей в режиме кэшированные обновлений:

```
TCachedUpdateStatus = (cusUnmodified, cusModified, cusInserted, cusDeleted,
    cusUninserted, cusDeletedApplied);
TFIBUpdateRecordTypes = set of TCachedUpdateStatus;
```

Поддерживаются следующие режимы отображения записей: отображение немодифицировавшихся; модифицировавшихся; вставленных; удаленных; удаленных, с уже примененным удалением.

UpdateTransaction

Это свойство задает транзакцию, в которой будут производиться модифицирующие запросы

События

Рассмотрим события, уникальные для TpfIBDataSet. События TDataSet смотрите в справке по Delphi. Многие события понятны без описания, мы же обратим внимание на некоторые из них.

AfterEndTransaction

```
procedure (EndingTR: TFIBTransaction; Action: TTransactionAction; Force: Boolean);
```

AfterEndUpdateTransaction

```
procedure TForm1.pFIBDataSet1AfterEndUpdateTransaction(  
    EndingTR: TFIBTransaction; Action: TTransactionAction; Force: Boolean);
```

AfterFetchRecord

```
procedure (FromQuery: TFIBQuery; RecordNumber: Integer; var StopFetching: Boolean);
```

AfterStartTransaction

```
procedure (Sender: TObject);
```

AfterStartUpdateTransaction

```
procedure TForm1.pFIBDataSet1AfterStartUpdateTransaction(Sender: TObject);
```

BeforeEndTransaction

```
procedure (EndingTR: TFIBTransaction; Action: TTransactionAction; Force: Boolean);
```

BeforeEndUpdateTransaction

```
procedure (EndingTR: TFIBTransaction; Action: TTransactionAction; Force: Boolean);
```

BeforeFetchRecord

```
procedure (EndingTR: TFIBTransaction; Action: TTransactionAction; Force: Boolean);
```

BeforeStartTransaction

```
procedure (Sender: TObject);
```

BeforeStartUpdateTransaction

```
procedure (Sender: TObject);
```

DatabaseDisconnected

```
procedure (Sender: TObject);
```

DatabaseDisconnecting

```
procedure (Sender: TObject);
```

DatabaseFree

```
procedure (Sender: TObject);
```

OnApplyDefaultValue

```
procedure (DataSet: TDataSet; Field: TField; var Applied: Boolean);
```

Это событие позволяет переопределить значения полей, выставленных по умолчанию, при вставке новой записи.

OnAskRecordCount

```
procedure (DataSet: TFIBDataSet; var SQLText: String);
```

Это событие возникает при выставленной опции psAskRecordCount и позволяет изменить SQL по умолчанию для получения количества записей

OnCompareFieldValues

```
function (Field: TField; const S1, S2: Variant): Integer;
```

Это событие позволяет задать свою функцию сравнения для локальной сортировки. Различные кодировки (разные наборы символов) требуют разных методов сортировки. Поэтому для кодировки None и Unicode_FSS подойдет стандартный метод, т.е., ничего переопределять не нужно. Для национальных кодировок нужно ставить ANSI кодировку AnsiCompareString (функция датасета).

OnFieldChange

```
procedure (Sender: Tfield);
```

Это событие возникает при изменении значения поля.

OnFillClientBlob

```
procedure TForm1.pFIBDataSet1FillClientBlob(DataSet: TFIBCustomDataSet;  
Field: TFIBBlobField; Stream: TFIBBlobStream);
```

Это событие генерируется при заполнении BLOB поля на клиенте (подробности читайте в статье [«Работа с BLOB-полями в клиентских приложениях InterBase и Firebird на основе компонентов FIBPlus»](#))

OnLockError

```
procedure (DataSet: TDataSet; LockError: TLockStatus; var ErrorMessage: String;  
var Action: TDataAction);
```

Это событие возникает при выставленной опции roProtectedEdit, если попытка блокирования не была успешной. Смотрите раздел [Защищенное редактирование](#) Руководства пользователя для подробностей.

OnUserEvent

```
procedure (Sender: TObject; Receiver: TDataSet; const EventName: String; var  
Info: String);
```

Это событие возникает при получении пользовательского события. Способы передачи этого

события смотрите в описании `TrFIBDatasetContainer` Руководства пользователя.

TransactionEnded

```
procedure (Sender: Tobject);
```

Это событие возникает после завершения транзакции.

TransactionEnding

```
procedure (Sender: Tobject);
```

Это событие возникает в момент завершения транзакции.

TransactionFree

```
procedure (Sender: Tobject);
```

Это событие возникает при высвобождении транзакции.

Методы

Данные методы используются для сравнения Ansi-строк:

```
function AnsiCompareString(Field:TField;const val1, val2: variant): Integer;  
function StdAnsiCompareString(Field:TField;const S1, S2: variant): Integer;
```

`StdAnsiCompareString` – задает следующий порядок:

```
a  
A  
б  
Б
```

`AnsiCompareString` – задает следующий порядок:

```
A  
Б  
a  
б
```

Метод соответствует разным сортировкам сервера при различных сочетаниях чарсетов (charset) и коллэитов (collate).

```
function FN(const FieldName: string): TField;  
function FBN(const FieldName: string): Tfield;
```

Данные функции аналогичны `TdataSet.FieldName`, но более короткие в написании.

```
procedure ApplyConditions(Reopen :boolean = False);
```

Этот метод применяет дополнительные условия. Параметр `Reopen` указывает, нужно ли переоткрывать `TrFIBDataSet`

```
procedure CancelConditions;
```

Этот метод отменяет все дополнительные условия.

```
procedure CloseOpen(const DoFetchAll:boolean);
```

Этот метод переоткрывает `TrFIBDataSet`. Параметр `DoFetchAll` указывает, делать ли полный fetch данных

```
procedure StartTransaction;
```

```
procedure BatchInput(InputObject: TFIBBatchInputStream; SQLKind: TpSQLKind
=skInsert);
procedure BatchOutput(OutputObject: TFIBBatchOutputStream);
```

Подробности смотрите в разделе [Пакетные изменения](#) Руководства пользователя.

```
function CachedUpdateStatus: TcachedUpdateStatus;
```

Этот метод позволяет получить статус записи при использовании кэшированных обновлений.

```
procedure CancelUpdates; virtual;
```

Этот метод отменяет все изменения, сделанные в режиме кэшированных обновлений

```
procedure FetchAll;
```

Этот метод производит полный fetch датасета

```
procedure RevertRecord;
```

При использовании режима CachedUpdates этот метод возвращает запись к начальному состоянию.

```
procedure Undelete;
```

Этот метод отменяет удаление записи.

```
procedure DisableScrollEvents;
```

```
procedure EnableScrollEvents;
```

```
procedure DisableCloseOpenEvents;
```

```
procedure EnableCloseOpenEvents;
```

```
procedure DisableCalcFields;
```

```
procedure EnableCalcFields;
```

```
function ArrayFieldValue(Field:TField):Variant;
```

```
procedure SetArrayValue(Field:TField;Value:Variant);
```

```
function GetElementFromValue( Field:TField; Indexes:array of integer):Variant;
```

```
procedure SetArrayElementValue(Field:TField;Value:Variant; Indexes:array of
integer );
```

Для получения подробностей смотрите раздел «Работа с полями-массивами» Руководства пользователя.

```
function GetRelationTableName(Field:TObject):string;
```

Этот метод возвращает имя отношения для поля.

```
function GetRelationFieldName(Field:TObject):string;
```

Этот метод возвращает имя поля.

```
procedure MoveRecord(OldRecno,NewRecno:integer); virtual;
```

Этот метод перемещает поле в кеше с позиции OldRecNo на позицию NewRecNo

```
procedure DoSortEx(Fields: array of integer; Ordering: array of Boolean);
overload;
```

```
procedure DoSortEx(Fields: TStrings; Ordering: array of Boolean); overload;
```

```
procedure DoSort(Fields: array of const; Ordering: array of Boolean); virtual;
```

Данные методы – это методы локальной сортировки.

```
function CreateCalcField(FieldClass:TFieldClass; const
aName,aFieldName:string;aSize:integer):TField;
```

```
function CreateLookUpField(FieldClass:TFieldClass; const
aName,aFieldName:string;aSize:integer; aLookupDataSet: TDataSet; const
aKeyFields, aLookupKeyFields, aLookupResultField: string ):TField;
```

Этот метод позволяет быстро создавать Calc-, Lookup-поля.

```
function GetFieldOrigin(Fld:TField):string;
```

Этот метод возвращает оригинальное имя поля Tfield.Origin

```
function FieldByOrigin(const aOrigin:string):TField; overload;
```

```
function FieldByOrigin(const TableName,FieldName:string):TField; overload;
```

Этот метод получает объект-поле по оригинальному имени

```
function FieldByRelName(const Fname:string):TField;
```

Этот метод возвращает первое поле для 'Select AAA as Name from Table1'
FieldByRelName('AAA').

```
function ReadySelectText:string;
```

Этот метод позволяет увидеть запрос, который в действительности отправляется на сервер.
Полезен при работе с дополнительными условиями.

```
function TableAliasForField(const aFieldName:string):string;
```

Этот метод возвращает псевдоним таблицы для поля.

```
function SQLFieldName(const aFieldName:string):string;
```

Этот метод возвращает реальное имя поля.

```
procedure RestoreMacroDefaultValues;
```

Этот метод устанавливает значения макросов в значения по умолчанию (default).

```
function IsComputedField(Fld:Variant):boolean;
```

Этот метод возвращает True, если поле вычисляемое.

```
function DomainForField(Fld:Variant):string;
```

Этот метод возвращает домен поля.

```
function SortInfoIsValid:boolean;
```

Этот метод проверяет информация о сортировке на валидность

```
function IsSortedField(Field:TField; var FieldSortOrder:TSortFieldInfo):boolean;
```

Этот метод получает информацию о порядке сортировке для поля. Тип TsortFieldInfo объявлен как:

```
TSortFieldInfo = record
  FieldName: string;           //имя поля
  InDataSetIndex: Integer;     // порядковый номер поля в датасете, т.е., индекс в
                               // коллекции Fields.
  InOrderIndex: Integer;       // порядковый номер поля в order. Т.е., например,
                               // для order by 2,3,1 второе поле датасета будет
                               // первым в ордере. InDataSetIndex=1, а
                               // InOrderIndex=0 (нумерация с нуля)
  Asc: Boolean;                //True, если порядок полей по возрастанию
  NullsFirst: Boolean;         //если установлено в Null значение перед
                               //остальными
end;
```

```
function SortFieldsCount:integer;
```

Этот метод возвращает количество полей сортировки

function SortFieldInfo (OrderIndex:integer): TSortFieldInfo;

Этот метод возвращает информацию о сортировке поля в позиции OrderIndex

function SortedFields:string;

Этот метод возвращает строку с полями сортировки, перечисленными через ','.

function CompareBookmarks (Bookmark1, Bookmark2: TBookmark): Integer; **override**;

Этот метод сравнивает две закладки.

function BlobModified (Field: TField): boolean;

Этот метод возвращает True, если BLOB поле Field было модифицировано.

function CreateBlobStream (Field: TField; Mode: TBlobStreamMode): TStream;
override;

Этот метод создает BLOB Stream для поля.

function GetRecordFieldInfo (Field: TField; **var** TableName,FieldName:string; **var** RecordKeyValues:TDynArray):boolean;

Этот метод получает информацию о поле.

function RecordFieldValue (Field:TField;RecNumber:integer):Variant; **overload**;

function RecordFieldValue (Field:TField;aBookmark:TBookmark):Variant; **overload**;

Эти методы возвращают значение поля и значение поля по закладке соответственно.

function Locate(**const** KeyFields: String; **const** KeyValues: Variant; Options: TLocateOptions): Boolean; **override**;

function LocatePrior(**const** KeyFields: String; **const** KeyValues: Variant; Options: TLocateOptions): Boolean;

function LocateNext(**const** KeyFields: String; **const** KeyValues: Variant; Options: TLocateOptions): Boolean;

Это стандартная функция датасета: поиск записи, в дополнение к нему поиск следующей записи и поиск предыдущей.

function ExtLocate(**const** KeyFields: String; **const** KeyValues: Variant; Options: TExtLocateOptions): Boolean;

function ExtLocateNext(**const** KeyFields: String; **const** KeyValues: Variant; Options: TExtLocateOptions): Boolean;

function ExtLocatePrior(**const** KeyFields: String; **const** KeyValues: Variant; Options: TExtLocateOptions): Boolean;

Это уникальные функции FIBPlus, которые работают аналогично предыдущим, но позволяют более гибкое управление.

TExtLocateOptions = (eloCaseInsensitive, eloPartialKey, eloWildCards, eloInSortedDS, eloNearest, eloInFetchedRecords)

eloCaseInsensitive игнорировать регистр при сравнении строк ;

eloPartialKey поиск по частичному совпадению;

eloWildCards поиск будет производиться по маске (подобно тому, как в операторе LIKE);

eloInSortedDS поиск производится в отсортированном датасете. Если датасет сортирован по этому полю, то поиск будет работать быстрее, чем обычно;

eloNearest (только в комбинации с eloInSortedDS). В результате операции,

если запись не будет найдена, указатель текущей записи будет спозиционирован на то место, где должен был быть результат;

`eloInFetchedRecords` поиск производится только в тех записях, которые уже `fetched` и находятся в буфере датасета.

procedure RefreshFilters;

Этот метод обновляет фильтр.

При обычной работе с фильтрами нужно делать так:

`DataSet.Filtered := False;`

`DataSet.Filter := <строка фильтра>;`

`DataSet.Filtered := True;`

Используя FIBPlus, можно написать проще:

`DataSet.Filter := <строка фильтра>;`

`DataSet.RefreshFilters;`

procedure CacheDelete;

Этот метод удаляет запись из кэша датасета, но реального удаления не происходит.

procedure CacheOpen;

Этот метод открывает датасет, но не делает `fetch` данных. Необходимо, чтобы было подключение к БД.

procedure RefreshClientFields (ForceCalc:boolean=True);

Этот метод пересчитывает `Calculated` поля без переоткрытия запроса.

function CreateCalcFieldAs (Field:TField):TField;

Этот метод создает вычисляемое поле с таким же типом, как у поля-параметра.

procedure CopyFieldsStructure (Source:TFIBCustomDataSet;RecreateFields:boolean);

Этот метод копирует структуру полей из датасета-источника.

procedure CopyFieldsProperties (Source, Destination:TFIBCustomDataSet);

Этот метод копирует свойства полей из одного датасета в другой.

procedure AssignProperties (Source:TFIBCustomDataSet);

Этот метод копирует все свойства датасета-параметра.

procedure OpenAsClone (DataSet:TFIBCustomDataSet);

Этот метод открывает датасет как копию датасета-параметра.

procedure Clone (DataSet:TFIBCustomDataSet; RecreateFields:boolean);

Этот метод клонирует данные датасета-параметра.

function CanCloneFromDataSet (DataSet:TFIBCustomDataSet):boolean;

Эта функция возвращает `True`, если может быть создана копия датасета-параметра

function PrimaryKeyFields (const TableName: string): string;

Этот метод возвращает имя ключевого поля для таблицы.

function FetchNext (FetchCount:Dword):integer;

Этот метод делает `fetch` следующих записей в количестве, указанном в параметре.

```
procedure ReopenLocate(const LocateFieldNames:string);
```

Этот метод переоткрывает TrFIBDataSet с позиционированием на том же месте, где был курсор перед закрытием. Параметр определяет, по каким полям будет сделан последующий Locate. Если полей несколько, их нужно писать через ','

```
function AllFieldValues: variant;
```

Этот метод возвращает вариантный массив – текущую строку датасета.

```
procedure FullRefresh;
```

Этот метод производит переоткрытие датасета. При этом отключаются методы, влияющие на отображение: визуальные компоненты данных и прокрутка.

```
function FieldsCount:integer;
```

Этот метод возвращает количество полей.

```
function FieldName(FieldIndex:integer):string;
```

Этот метод возвращает имя поля по индексу.

```
function FieldExist(const FieldName:string; var FieldIndex:integer):boolean;
```

Этот метод проверяет существование поля в TDataSet, и в случае успеха (если поле существует) его индекс возвращается в FieldIndex.

```
function ParamExist(const ParamName:string; var ParamIndex:integer):boolean;
```

Этот метод проверяет существование параметра в TDataSet, и в случае успеха (если параметр существует) его индекс возвращается в FieldIndex.

```
function FieldValue(const FieldName:string; Old:boolean):variant; overload;
```

```
function FieldValue(const FieldIndex:integer;Old:boolean):variant; overload;
```

Этот метод возвращает значение поля по имени или индексу.

```
function ParamValue(const ParamName:string):variant; overload;
```

```
function ParamValue(const ParamIndex:integer):variant; overload;
```

Этот метод возвращает значение параметра по индексу или имени

```
procedure SetParamValue(const ParamIndex:integer; aValue:Variant);
```

Этот метод устанавливает значение параметра

```
function RecordCountFromSrv: integer; dynamic;
```

Этот метод возвращает количество записей на сервере. Используется для получения реального количества записей на сервере, когда fetch выборки выполнен не до конца. Так, например, этот метод используется при опции psAskRecordCount.

```
function VisibleRecordCount: Integer;
```

Этот метод возвращает количество видимых записей, например, в сетке данных.

```
function CanEdit: Boolean; override;
```

Этот метод возвращает True, если датасет поддерживает операцию Edit;

```
function CanInsert: Boolean; override;
```

Этот метод возвращает True, если датасет поддерживает операцию Insert;

```
function CanDelete: Boolean; override;
```

Этот метод возвращает True, если датасет поддерживает операцию Delete.

```
function ExistActiveUO(KindUpdate: TUpdateKind): boolean;
function AddUpdateObject(Value: TpFIBUpdateObject): integer;
procedure RemoveUpdateObject(Value: TpFIBUpdateObject);
```

Эти методы проверяют существование дополнительного обработчика TpFIBUpdateObject, а также удаляют или добавляют дополнительный обработчик TpFIBUpdateObject.

```
function ParamByName(const ParamName: string): TFIBXSQLVAR;
```

Этот метод возвращает параметр по имени.

```
function FindParam(const ParamName: string): TFIBXSQLVAR;
```

Этот метод ищет параметр, в том числе, и на уровне макросов. Если макросы содержат параметры, то для их заполнения нужно использовать именно этот метод.

```
function RecordStatus(RecNumber: integer): TupdateStatus;
```

Этот метод возвращает статус для кэшированных обновлений.

```
procedure CloneRecord(SrcRecord: integer; IgnoreFields: array of const);
```

Этот метод копирует запись по индексу SrcRecord. Второй параметр указывает, какие поля игнорировать при клонировании (например, ключевые поля)

```
procedure CloneCurRecord(IgnoreFields: array of const);
```

Этот метод клонирует текущую запись.

```
procedure CommitUpdToCach;
```

```
procedure ApplyUpdToBase;
```

```
procedure ApplyUpdates;
```

Это методы для кэшированных обновлений в дополнение к стандартным ApplyUpdates, CancelUpdates. Стандартные операции датасета неадекватно работают при фильтрации, поэтому рекомендуем использовать эти дополнительные методы.

```
procedure SaveToStream(Stream: TStream; SeekBegin: boolean);
```

```
procedure LoadFromStream(Stream: TStream; SeekBegin: boolean);
```

```
procedure SaveToFile(const FileName: string);
```

```
procedure LoadFromFile(const FileName: string);
```

Эти методы позволяют сохранить, а затем загрузить кеш датасета в файл или в поток. Датасет должен быть подключен к БД.

```
function LockRecord(RaiseErr: boolean= True): TlockStatus;
```

Этот метод позволяет производить пессимистическое блокирование записи.

```
function FieldByFieldNo(FieldNo: Integer): Tfield;
```

Этот метод возвращает поле по числовому параметру FieldNo

```
function ParamNameCount(const aParamName: string): integer;
```

Этот метод возвращает количество уникальных имен параметров, если один и тот же параметр используется несколько раз.

```
function ParamCount: integer;
```

Этот метод возвращает количество параметров.

```
procedure ExecUpdateObjects (KindUpdate: TUpdateKind; Buff: Pointer;
aExecuteOrder: TFIBOrderExecUO);
```

Этот метод выполняет дополнительные обновляющие запросы TpFIBUpdateObject, ассоциированные с датасетом.

```
procedure DoUserEvent (Sender: TObject; const UDE: string; var Info: string);
dynamic;
```

Этот метод отправляет пользовательское событие.

```
procedure OpenWP(const ParamValues: array of Variant); overload;
procedure OpenWP(const ParamNames : string; const ParamValues: array of Variant);
overload;
procedure OpenWPS(const ParamSources: array of ISQLObject);
procedure ReOpenWP(const ParamValues: array of Variant); overload;
procedure ReOpenWP(const ParamNames : string; const ParamValues: array of
Variant); overload;
procedure ReOpenWPS(const ParamSources: array of ISQLObject);
```

Это группа перегруженных методов, которые позволяют выполнить открытие и переоткрытие датасета с одновременной передачей параметров.

```
procedure BatchRecordToQuery (ToQuery: TFIBQuery);
procedure BatchAllRecordsToQuery (ToQuery: TFIBQuery);
```

Подробности смотрите в разделе "Выполнение SQL-Запросов. Пакетная обработка"

Руководства пользователя.

```
procedure AutoGenerateSQLText (ForState: TDataSetState);
function GenerateSQLText (const TableName, KeyFieldNames: string; SK: TpSQLKind;
IncludeFields: TIncludeFieldsToSQL=ifsAllFields): string;
function GenerateSQLTextWA (const TableName: string; SK: TpSQLKind;
IncludeFields: TIncludeFieldsToSQL=ifsAllFields): string;
procedure GenerateUpdateBlobsSQL;
procedure GenerateSQLs;
function CanGenerateSQLs: boolean;
```

Это группа методов для автоматической генерации обновляющих SQL-запросов.

```
function KeyField: Tfield;
```

Этот метод возвращает объект ключевого поля

```
function SqlTextGenID: string;
```

Этот метод возвращает текст для получения значения генератора

```
procedure IncGenerator; virtual;
```

Этот метод увеличивает значение генератора.

```
function AllKeyFields(const TableName: string): string;
```

Этот метод возвращает имена ключевых полей.

Следующие методы используются для манипуляции с кешем датасета

```
procedure CacheModify( aFields: array of integer; Values: array of Variant;
KindModify: byte );
procedure CacheEdit(aFields: array of integer; Values: array of Variant);
procedure CacheAppend(aFields: array of integer; Values: array of Variant);
overload;
procedure CacheAppend(Value: Variant; DoRefresh: boolean = False); overload;
```

```
procedure CacheInsert(aFields: array of integer; Values: array of Variant);  
overload;  
procedure CacheInsert(Value: Variant; DoRefresh: boolean = False); overload;  
procedure CacheRefresh(FromDataSet: TDataSet; Kind: TCachRefreshKind ; FieldMap:  
Tstrings);  
procedure CacheRefreshByArrMap( FromDataSet: TDataSet; Kind: TCachRefreshKind;  
const SourceFields, DestFields: array of string );
```

TrFIBUpdateObject

Это объект-наследник TrFIBQuery, который позволяет выполнить дополнительные действия для TrFIBDataSet при вставке, модификации или удалении записей. Про наследуемые свойства и методы TrFIBQuery читайте в соответствующем разделе «Выполнение по TrFIBQuery» Руководства пользователя.

Свойства

Conditions

Смотрите соответствующее свойство TrFIBDataSet или TrFIBQuery в разделе "Выполнение SQL-Запросов. Условия" Руководства пользователя.

DataSet

Это свойство возвращает датасет, для которого будет работать дополнительное действие TrFIBUpdateObject.

ExecuteOrder

Это свойство возвращает порядок выполнения действия, которое может быть выполнено до основного действия или после.

KindUpdate

Это свойство возвращает тип объекта обновления: вставка, модификация, удаление.

OrderInList

Это свойство возвращает порядок в списке объектов с одинаковым KindUpdate

TDataSetContainer

Этот объект позволяет задать одинаковое поведение для родственных объектов `TpFIBDataSet`.

Компонент `TDataSetContainer` позволяет централизованно обрабатывать события от разных компонентов `TpFIBDataSet`, а также посылать им сообщения, при получении которых они также могут производить какие-то дополнительные действия. Кроме того, он может использоваться, чтобы задать одинаковую функцию сравнения полей для локальной сортировки

Свойства

Active

Установите это свойство в `True`, если компонент активен.

MasterContainer

Если свойство установлено в `True`, компонент может быть подчиненным другому контейнеру

События

События повторяют избранные события `TpFIBDataSet`, поэтому ищите описания в соответствующем разделе о `TpFIBDataSet` Руководства Пользователя.

OnApplyDefaultValue

```
procedure (DataSet: TDataSet; Field: TField; var Applied: Boolean);
```

Это событие генерируется при применении значения по умолчанию для поля

OnCompareFieldValues

```
function (Field: TField; const S1, S2: Variant; var Compared: Boolean): Integer;
```

Это событие позволяет задать свою функцию сортировки для функции `DoSort/DoSortEx`.

Контейнер имеет смысл для всех присоединенных датасетов, и, следовательно, функция сортировки будет применена для всех датасетов.

OnDataSetError

```
procedure (DataSet: TDataSet; Event: TKindDataSetError; E: EDatabaseError; var
Action: TdataAction);
```

```
TKindDataSetError = (deOnError, deOnPostError, deonDeleteError);
```

Это событие генерируется при ошибке и позволяет выполнить стандартное действие редактирования (`deOnError`), подтверждения (`deOnPostError`) и удаления (`deonDeleteError`).

OnDataSetEvent

```
procedure (DataSet: TDataSet; Event: TkindDataSetEvent);
```

```
TkindDataSetEvent = (deBeforeOpen, deAfterOpen, deBeforeClose, deAfterClose,
```

deBeforeInsert, deAfterInsert, deBeforeEdit, deAfterEdit, deBeforePost, deAfterPost, deBeforeCancel, deAfterCancel, deBeforeDelete, deAfterDelete, deBeforeScroll, deAfterScroll, deOnNewRecord, deOnCalcFields, deBeforeRefresh, deAfterRefresh)

Это событие генерируется при получении события датасета

OnUserEvent

```
procedure (Sender: TObject; Receiver: TDataSet; const EventName: String; var Info: String);
```

Это событие генерируется при получении пользовательского события.

Методы

```
procedure AddDataSet (Value:TDataSet);
```

Этот метод позволяет добавить датасет в контейнер.

```
procedure RemoveDataSet (Value:TDataSet);
```

Этот метод позволяет удалить датасет из контейнера.

```
function DataSetCount:integer;
```

Этот метод возвращает количество датасетов в контейнере

```
function DataSet (Index:integer):TDataSet;
```

Этот метод возвращает датасет по индексу

TSIBfibEventAlerter

Этот компонент позволяет получать события базы данных.

Свойства

AutoRegister

Если свойство установлено в True, то при создании формы события будут зарегистрированы автоматически, иначе нужно вызвать метод Register самостоятельно.

Database

Это свойство задает объект TrFIBDataBase, события которого он будет принимать.

Events

Это свойство возвращает StringList с событиями, которые будут обрабатываться компонентом TSIBfibEventAlerter.

События

Существует только одно событие OnEventAlert, объявленное как:

procedure (Sender: TObject; EventName: **String**; EventCount: Integer);

EventName возвращает имя произошедшего события, EventCount – количество произошедших событий.

Помните, что события отсылаются только при подтверждении транзакции. Это может накладывать ограничения на способ использования данных событий.

ТрFIBErrorHandler

Этот объект позволяет централизованно обрабатывать все ошибки библиотеки, что, на наш взгляд, довольно удобно.

Свойства

Options

```
TOptionErrorHandler = (oeException, oeForeignKey, oeLostConnect, oeCheck,  
    oeUniqueViolation);
```

```
TOptionsErrorHandler = set of TOptionErrorHandler;
```

Это свойство отвечает за типы ошибок, которые будут попадать в обработчик
OnFIBErrorEvent

```
TOnFIBErrorEvent = procedure (Sender: TObject; ErrorValue: EFIBError;  
    KindIBError: TKindIBError; var DoRaise: boolean) of object;
```

```
TKindIBError = (keNoError, keException, keForeignKey, keLostConnect,  
    keSecurity, keCheck, keUniqueViolation, keOther);
```

TpFIBClientDataSet

Это прямой потомок TClientDataSet, который введен для корректной работы с BIGINT-полями. Используется совместно с TpFIBDataSetProvider. Из новых методов есть только OpenWP, специфичный для TpFIBDataSet.

Подробности читайте во встроенной справке по Delphi.

TpFIBDataSetProvider

Этот объект осуществляет связь между датасетом и клиентским датасетом. Это прямой потомок TDataSetProvider, поэтому подробности читайте в справке по Delphi. Объект введен для корректной работы с BIGINT- полями.

TFIBSQLMonitor

Этот объект позволяет осуществить мониторинг всех действий с БД, которые производит приложение, использующее FIBPlus.

Свойства

TraceFlags

Это свойство задает типы событий, которые будет отслеживать компонент. Свойство объявляется следующим образом:

```
TFIBTraceFlag = (tfQPrepare, tfQExecute, tfQFetch, tfConnect, tfTransact,
tfService, tfMisc);
TFIBTraceFlags = set of TFIBTraceFlag;
```

tfQPrepare	определяет, будут ли отслеживаться операции Prepare;
tfQExecute	отслеживать выполнение;
tfQFetch	отслеживать фетч;
tfConnect	события соединения;
tfTransact	события транзакций старта, завершения транзакций;
tfService	работа с сервисами;
tfMisc	служебные запросы библиотеки.

События

OnSQL

Это событие будет вызываться каждый раз при выполнении операций, выставленных в TraceFlags. Свойство объявлено следующим образом:

```
TSQLEvent = procedure (EventText: String; EventTime : TDateTime) of object;
```

TFIBSQLLogger

Этот компонент предназначен для ведения статистики работы с БД, а также логгирования SQL-запросов. Смотрите демонстрационный пример SQLLogger:
FIBPlusExamples\src\SQLLogger\.

Свойства

Database

Это свойство возвращает базу данных, запросы которой отслеживает компонент.

ActiveStatistics

Включите эту опцию, чтобы отслеживать статистику БД

ActiveLogging

Включите эту опцию, чтобы вести лог БД.

ApplicationID

Это свойство возвращает строку, которая будет определять ваше приложение. Например, это свойство пригодится для идентификации приложений при наличии нескольких компонент для записи статистики в БД.

LogFileName

Это свойство возвращает имя файла, в который будет писаться лог БД.

StatisticsParams

Это свойство возвращает параметры, которые будут сохранены в статистике. Вот возможные значения параметров:

```
TFIBStatisticsParam = (fspExecuteCount, fspPrepareCount, fspSumTimeExecute,  
fspAvgTimeExecute, fspMaxTimeExecute, fspLastTimeExecute);
```

<code>fspExecuteCount</code>	количество выполнений определенного запроса;
<code>fspPrepareCount</code>	количество операций подготовки запроса;
<code>fspSumTimeExecute</code>	суммарное время выполнения запроса;
<code>fspAvgTimeExecute</code>	среднее время выполнения запроса;
<code>fspMaxTimeExecute</code>	максимальное время выполнения запроса;
<code>fspLastTimeExecute</code>	последнее время выполнения запроса.

LogFlags

Это свойство отвечает за то, какие типы операций будут записываться в лог. Повторяют события TFIBSQLMonitor:

```
TLogFlag = (lfQPrepare, lfQExecute, lfQFetch, lfConnect, lfTransact,  
lfService, lfMisc);
```

ForceSaveLog

Если включена эта опция, то события будут записываться по мере поступления.

Методы

```
procedure Clear;
```

```
procedure SaveStatisticsToFile(const FileName:string);
```

Этот метод позволяет сохранить статистику в файл БД

```
procedure SortStatisticsForPrint(const VarName:string;Ascending:boolean);
```

Этот метод позволяет сортировать статистику БД

```
function ExistStatisticsTable:boolean;
```

Этот метод проверяет, создана ли в базе таблица статистики БД

```
procedure CreateStatisticsTable;
```

Этот метод позволяет создать таблицу для хранения статистики в БД

```
procedure SaveStatisticsToDB(ForMaxExecTime:integer=0);
```

Этот метод позволяет сохранить статистику в БД

```
procedure SaveLog;
```

Этот метод позволяет сохранить лог статистики БД

TpFIBCustomService

Свойства

Handle: TISC_SVC_HANDLE

Это свойство возвращает хендл сервиса

ServiceParamBySPB

Это свойство позволяет получить параметр сервиса по имени

Active

Это логическое свойство управляет состоянием соединения с сервисом

ServerName

Это строковое свойство имя сервера, обязательное для заполнения.

Protocol

Это свойство возвращает протокол, по которому будет осуществляться взаимодействие с сервисом. Тип Tprotocol описан следующим образом:

```
TProtocol = (TCP, SPX, NamedPipe, Local)
```

Params

Это свойство возвращает параметры сервиса

LoginPrompt

Это свойство позволяет определить, следует ли выводить диалог авторизации.

LibraryName

Это свойство возвращает имя клиентской библиотеки

SQLLogger

Это свойство логирует вызовы сервиса.

События

OnAttach

Это событие TNotifyEvent, которое возникает при присоединении к сервису

OnLogin

Это событие возникает при присоединении к сервису

```
TLoginEvent = procedure (Database: TpFIBCustomService; LoginParams: TStrings) of
object;
```

Методы

Существуют два основных метода присоединения и отсоединения от сервиса.

```
procedure Attach;
procedure Detach;
```

TpFIBServerProperties

Это событие позволяет получать информацию о сервере и обслуживаемых базах данных.

Свойства

Option

TPropertyOption задает опции для получаемой информации о БД. Объявляется следующим образом:

```
TPropertyOption = (Database, License, LicenseMask, ConfigParameters, Version)
```

Если опция установлена в True, то после вызова метода Fetch будет заполнена соответствующая запись.

DatabaseInfo

Это свойство содержит информацию о базах данных и заполняется после вызова методов Fetch или FetchDatabaseInfo.

```
TDatabaseInfo = record
  NoOfAttachments: Integer;    //количество соединений с сервером
  NoOfDatabases: Integer;      //количество соединенных баз данных
  DbName: Variant;             //имена соединенных баз данных
end;
```

LicenseInfo

Это свойство содержит информацию о лицензии сервера

```
TLicenseInfo = record
  Key: Variant;                //ключ
  Id: Variant;                  //код
  Desc: Variant;                //описание
  LicensedUsers: Integer;       //пользователи
end;
```

LicenseMaskInfo

```
TLicenseMaskInfo = record
  LicenseMask: Integer;         //маска
  CapabilityMask: Integer;      //маска
end;
```

VersionInfo

```
TVersionInfo = record
  ServerVersion: String;           //версия сервера
  ServerImplementation: string;   //внутренняя информация о сборке
  ServiceVersion: Integer;        //версия севрисов
end;
```

ConfigParams

```
TConfigParams = record
  ConfigFileData: TConfigFileData;
  BaseLocation: string;
  LockFileLocation: string;
  MessageFileLocation: string;
  SecurityDatabaseLocation: string;
end;
```

```
TConfigFileData = record
  ConfigFileValue:Variant;
  ConfigFileKey:Variant;
end;
```

Методы

При помощи доступных методов можно получить всю информацию либо по всем опциям сразу, либо по каждой опции отдельно:

```
procedure Fetch;
procedure FetchDatabaseInfo;
procedure FetchLicenseInfo;
procedure FetchLicenseMaskInfo;
procedure FetchConfigParams;
procedure FetchVersionInfo;
```

TrFIBSecurityService

Этот метод используется для управления пользователями сервера.

Свойства

SecurityAction

```
TSecurityAction = (ActionAddUser, ActionDeleteUser, ActionModifyUser,
ActionDisplayUser)
```

Это свойство задает такие возможные действия как: получение информации о пользователях, а также добавление, модификация и удаление пользователей.

UserName

Это свойство возвращает имя пользователей на сервере

Password

Это свойство возвращает пароль на сервере

SQLRole

Это свойство возвращает роль пользователя на сервере

FirstName, MiddleName, LastName

Это свойство возвращает необязательную общую информацию о пользователе на сервере

UserID

Это свойство возвращает код пользователя на сервере

GroupID

Это свойство возвращает код группы. В настоящее время свойство сервером не используется.

UserInfo

Это индексированное свойство, в котором содержится информация о пользователе в позиции Index. Информация о пользователе представлена объектом TUserInfo:

```
TUserInfo = class (TObject)
public
  UserName: string;
  FirstName: string;
  MiddleName: string;
  LastName: string;
  GroupID: Integer;
  UserID: Integer;
end;
```

UserInfoCount

Это свойство возвращает количество пользователей на сервере, заполняется при выполнении метода DisplayUsers

Методы

```
procedure DisplayUsers;
```

Этот метод возвращает информацию о пользователях в свойства UserInfo

```
procedure DisplayUser (UserName: string);
```

Этот метод возвращает информацию о пользователе с именем, заполненным в свойстве UserName

```
procedure AddUser;
```

Этот метод добавляет пользователя с именем, заполненным в свойстве UserName. Свойства UserName и Password должны быть предварительно заполнены для добавляемого пользователя.

```
procedure DeleteUser;
```

Этот метод удаляет пользователя с именем, заполненным в свойстве UserName

```
procedure ModifyUser;
```

Этот метод модифицирует информацию о пользователе с именем, заполненным в свойстве UserName. Аналогично методам, описанным выше, должны быть заполнены основные

свойства UserName и Password.

TpFIBBackupService

Данный компонент позволяет выполнить резервирование базы данных. Он четвертый в иерархии наследования, это добавляет ему целую группу свойств и методов.

TpFIBControlService

procedure ServiceStart //запускает сервис
property IsServiceRunning //показывает, активен ли сервис

TpFIBControlAndQueryService

function GetNextLine : **string**; //получает следующую строку из выходного буфера
property Eof: **boolean** //если возвращает True, то достигнут конец буфера

TpFIBBackupRestoreService

property Verbose: **Boolean** //выводить ли лог работы сервиса
property OnTextNotify //событие возникает при получении очередной строки буфера

Свойства

BackupFile

Tstrings, в который нужно поместить целевые имена файлы бэкапа.

DatabaseName

Свойство возвращает имя базы данных, для которой будет создана резервная копия (backup).

Option

Задаёт опции процесса резервирования. Подробное описание опций можно получить в OpGuide.pdf документации по InterBase.

```
TBackupOption = (
  IgnoreChecksums,       //игнорировать контрольную сумму
  IgnoreLimbo,           //игнорировать лимбо-транзакции
  MetadataOnly,          //резервировать только метаданные
  NoGarbageCollection,   //не производить сборку мусора
  OldMetadataDesc,       //совместимость со старыми версиями
  NonTransportable,       //показывает, создавать ли backup, понятный другим
                            версиями сервера, либо только для использования тем
                            сервером, который его сделал (если есть два сервера,
                            например, версии 1.0 и 1.5, то, если сервером 1.0
                            создать backup с опцией NonTransportable, то для этого
                            backup нельзя будет сделать restore на версии 1.5)
  ConvertExtTables);     //содержимое внешних таблиц будет включено в backup, при
                            restore внешние таблицы будут создаваться в основной
```

базе данных

```
TBackupOptions = set of TBackupOption;
```

Работа с сервисом производится следующим образом:

```
//Delphi
BackupService1.Active := True;
BackupService1.Verbose := True;
BackupService1.ServiceStart;
while not BackupService1.Eof do
    Memol.Lines.Add(BackupService1.GetNextLine);
BackupService1.Active := False;

//C++
BackupService1->Active = true;
BackupService1->Verbose = true;
BackupService1->ServiceStart();
while (!BackupService1->Eof)
    Memol->Lines->Add(BackupService1->GetNextLine());
BackupService1->Active = false;
```