FIREBIRD CONF 2025

Логинов Дмитрий Сергеевич

Применение СУБД Firebird в программных продуктах ООО "АВТОМАТИКА плюс"

План доклада

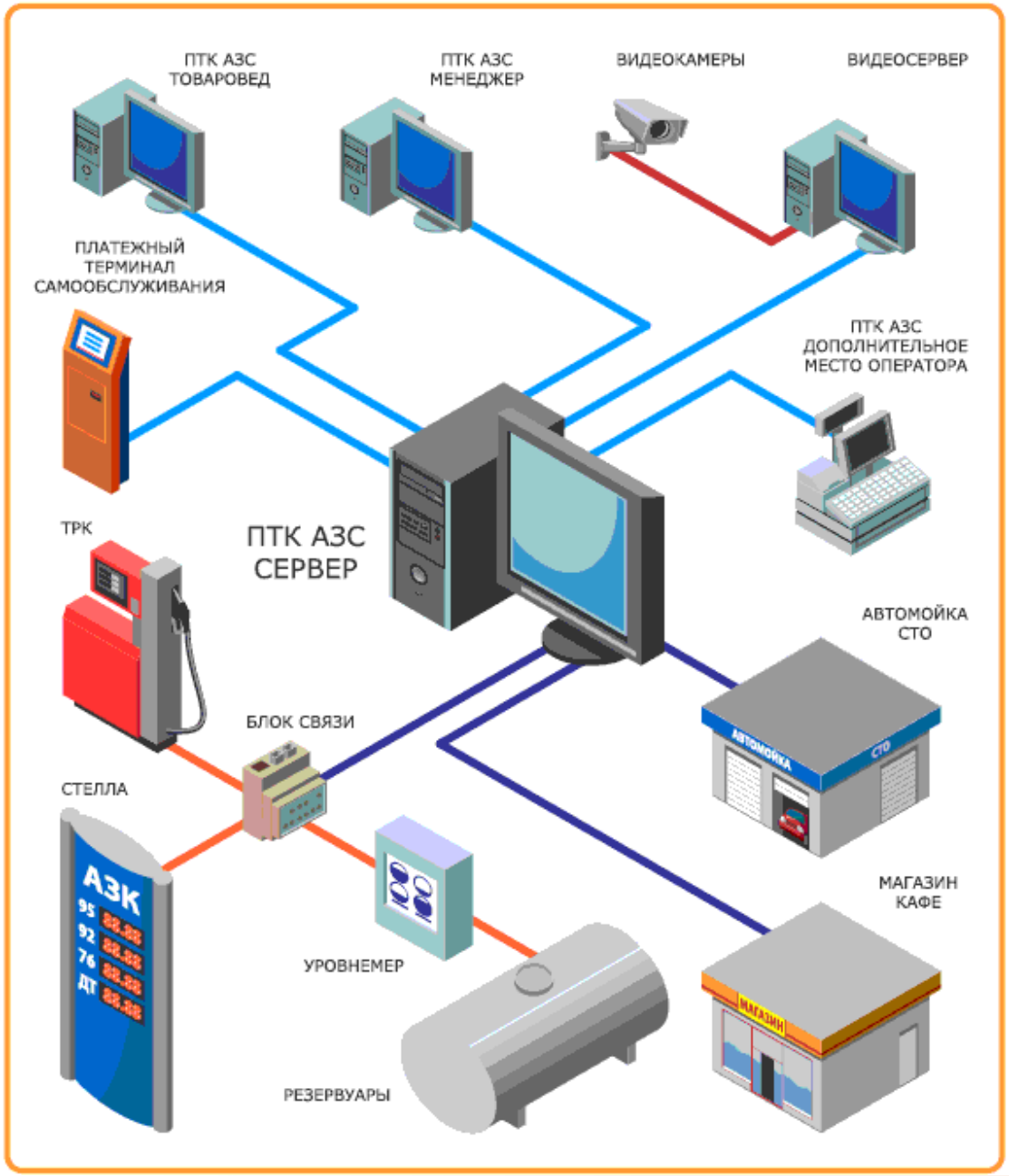
1. Знакомство с ООО «АВТОМАТИКА плюс»;
2. Опыт использования СУБД Firebird;
    1. Обзор возникающих проблем производительности и их решение;
    2. Причины перехода с Firebird 2.5 на Firebird 3.0 и результат;
3. Обзор работ докладчика (статьи по программированию, библиотеки), в том числе:
    1. Обзор библиотеки ibxFBUtils (Delphi, Lazarus), позволяющей упростить разработку приложений баз данных Firebird ;
    2. Обзор статьи по портированию VCL-приложения (IDE Delphi) на Linux (IDE Lazarus);

О компании ООО «АВТОМАТИКА плюс»

- Являемся IT-компанией
- Работаем на Российском рынке 28 лет
- Разрабатываем решения:
  - Программируемые контроллеры (ПКЭМ-3, УПУ-166, PPLC);
  - Системы с числовым программных управлением;
  - Системы управления АЗС, АГЗС и базами хранения нефтепродуктов;
  - Программное обеспечение для контроллеров и систем управления АЗС;
- Оказываем услуги:
  - Техническая поддержка;
  - Гарантийное обслуживание;
  - Обучение пользователей;
  - Доработка ПО под индивидуальные требования заказчиков;
- Сайт: https://www.automatikaplus.ru

ПТК АЗС: Автоматизация АЗС, АГЗС, НБ

- Назначение:
  - Программно-технический комплекс автозаправочной станции «ПТК АЗС» предназначен для автоматизации АЗС, АГЗС и НБ;
- Аппаратные компоненты:
  - Блок связи;
  - Щит управления оборудованием;
  - Персональный компьютер;
- Программные компоненты:
  - ПТК АЗС. Сервер – основное р.м.
  - ПТК АЗС. Клиент / Менеджер / Магазин;
  - ПТК АЗС. Офис;



ПТК АЗС: Автоматизация АЗС, АГЗС, НБ

- Основные функции:
  - управление топливораздаточными колонками;
  - мониторинг состояния ГСМ в резервуарах;
  - взаимодействие с торговым оборудованием;
  - продажа ГСМ и сопутствующих товаров по различным видам оплаты;
  - учёт остатков ГСМ и сопутствующих товаров;
  - формирование отчётов по операциям с ГСМ и сопутствующими товарами;
  - взаимодействие (интеграция) с различными электронными системами;

ПТК АЗС: Автоматизация АЗС, АГЗС, НБ

- Система «ПТК АЗС» разрабатывается с 1998г (на Delphi) и представлена во всех регионах России;
- Используют более 300 клиентов (сетей АЗС);
- Эксплуатируется более чем на 1000 АЗС и АЗК;
- Автоматизировано более 2000 рабочих мест кассиров, товароведов и другого персонала;
- Ежедневная выдача ГСМ более 5 млн. литров;
- Стоимость системы «ПТК АЗС», техподдержка, гарантия:
  - Стоимость базовой комплектации – около 100 тыс. руб.;
  - Разовая оплата, подписка не предусмотрена;
  - Обновления в рамках версии предоставляются бесплатно;
  - Гарантия – 1 год;
  - Уровень конкуренции – высокий, препятствует повышению цен;
  - Доходность АЗС низкая, также препятствует повышению цен.

Firebird – боец невидимого фронта

- Для хранения информации по операциям с ГСМ и сопутствующими товарами используется реляционная база данных.
- В 1998г был выбран движок BDE (таблицы Paradox). Доверия не оправдал! Недостатки:
  - Низкая надёжность и производительность;
  - Большое количество багов (никто не исправляет);
  - Высокие трудозатраты.
- В качестве СУБД был выбран Firebird. Причины:
  - Знаком разработчикам Delphi;
  - Бесплатный;
  - Поддержка SQL-стандартов;
  - Доступ «клиент-сервер»;
  - Нет аппаратных ограничений;
  - Минимум администрирования.

ПТК АЗС: путь к Firebird 3.0

- С 2009 по 2012 используется Firebird 2.1 (SuperServer);
  - Значительно увеличилась производительность и надежность хранения данных;
  - Уменьшились трудозатраты на восстановление системы после сбоев;
- С 2012 по 2022 используется Firebird 2.5 (SuperServer);
  - Улучшилась производительность офисной части «ПТК АЗС» при параллельных обращениях к базам данных от разных АЗС;
- Проблема Firebird 2.5 (SuperServer):
  - Из-за постепенного увеличения количества сервисов в офисной части больших сетей АЗС Firebird 2.5 (SuperServer) перестал справляться с нагрузкой. Решение: перейти на Firebird 3.0;
- С 2022 по н.в. используется Firebird 3.0 (SuperServer);
  - Многократно улучшилась производительность офисной части «ПТК АЗС». Теперь производительность напрямую зависит от аппаратной начинки ЭВМ;
- Поддержка Firebird 5.0 – в следующей версии!

КМАЗС: Автоматизация мобильных и ведомственных АЗС

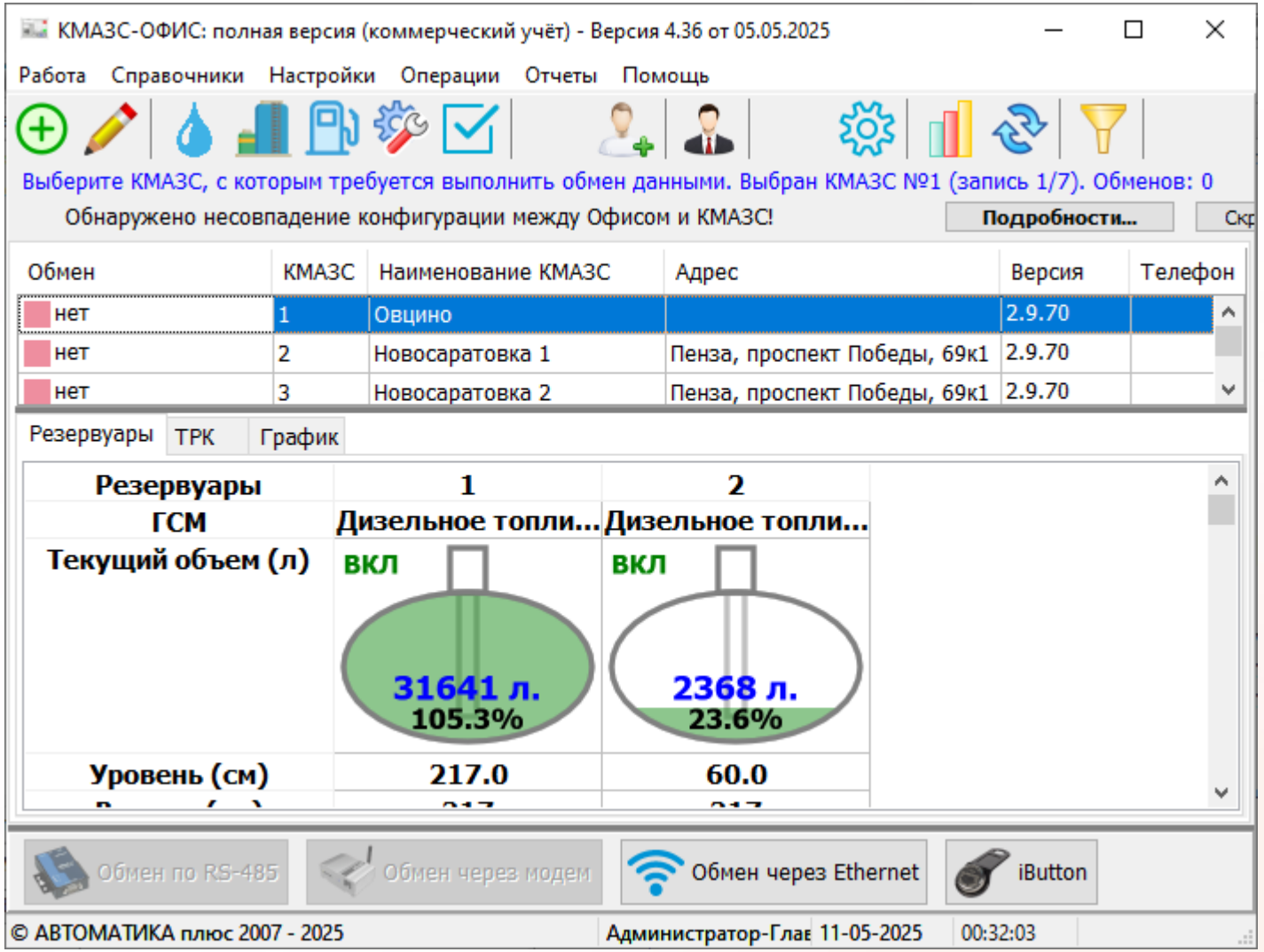
- Позволяет организовать выдачу ГСМ:
  - На территории предприятий;
  - На контейнерных АЗС;
  - С автомобильных топливозаправщиков;
  - С бензовозов;
  - С резервуаров (в полевых условиях).

КМАЗС: Автоматизация мобильных и ведомственных АЗС

- Возможности КМАЗС:
  - Авторизация водителей / техники – по бесконтактной карте либо пин-коду;
  - Управление электромеханическими либо интерфейсными ТРК;
  - Мониторинг состояния ГСМ в резервуарах;
  - Обслуживание водителей / техники выполняется в автономном режиме;
  - Хранение конфигурации, справочников пользователей и транзакций в энергонезависимой памяти;
  - Передача транзакций в офисное ПО по заданному каналу связи.

КМАЗС-ОФИС – офисная часть системы КМАЗС

- Назначение:
  - Ведение справочников;
  - Настройка параметров;
  - Обмен с контроллерами;
  - Учёт ГСМ;
  - Ведение счетов контрагентов;
  - Оформление приходов и откачек;
  - Формирование отчётов;
  - Импорт / экспорт данных;



КМАЗС-ОФИС – офисная часть системы КМАЗС

- Особенности КМАЗС-ОФИС:
  - Программа разрабатывается в Delphi;
  - Реализована поддержка Linux (с использованием Lazarus);
  - Используется Firebird 2.5 либо Firebird 3.0;
  - Протестирована работа с RedDatabase 3.0 и HQBird 2024 (в рамках FB3.0);
  - Используются компоненты IB Express (доступны в Delphi и Lazarus);
  - Программа устанавливается на инфраструктуре заказчика;
  - Архитектура: клиент-серверная (с одной базой данный одновременно могут работать более 50 экземпляров программы);
  - Программа может работать на любом отдалении от базы данных, однако время ping может влиять на производительность программы;
  - Обеспечивается одновременный обмен со всеми КМАЗС по заданным каналам связи (у некоторых клиентов более 100 контроллеров);

КМАЗС: информация

- Система «КМАЗС» разрабатывается с 2007г и представлена во всех регионах России;
- Используют более 500 организаций (в том числе очень крупные: Мосгортранс, Иркутская нефтяная компания, Мираторг);
- Эксплуатируется более чем на 3000 пунктах выдачи ГСМ;
- Автоматизировано более 2000 рабочих мест операторов;
- Ежедневная выдача ГСМ более 2 млн. литров;
- Стоимость контроллера КМАЗС – около 140 т.р.
- Стоимость системы ПО «КМАЗС-ОФИС», техподдержка, гарантия:
  - Стоимость основного рабочего места ПО «КМАЗС-ОФИС» – около 30 тыс. руб.;
  - Стоимость дополнительного рабочего места – около 15 тыс. руб;
  - Оплата разовая, подписка не предусмотрена;
  - Обновления в рамках версии предоставляются бесплатно;
  - Техподдержка бесплатная (в рамках актуальной версии);
  - Гарантия – 1 год;
  - Уровень конкуренции – очень высокий

КМАЗС-ОФИС и Firebird – проблемы

- 1. Проблемы взаимодействия с удалённо расположенной базой данных:
  - 1.1 Подвисания интерфейса пользователя (при большом времени Ping);
  - 1.2 Медленное подключение к БД (если часто создавать подключения к БД);
  - 1.3 Медленное выполнение автоматических операций (при большом времени Ping);
  - 1.4 Сбои в работе программы при перезагрузке службы Firebird на сервере;
  - 1.5 Зависание SQL-запросов на 2 часа (при сбоях в работе сети);
  - 1.6 Создание повышенной нагрузки на сеть:
    - при частых обращениях к БД;
    - при запросе большого объёма данных;
- 2. Проблемы из-за неаккуратного управления транзакциями:
  - 2.1 Постепенное ухудшение производительности Firebird, простейшие запросы выполняются несколько секунд;
- 3. Проблемы производительности при одновременном запуске обмена данными с большим количеством контроллеров КМАЗС:
  - 3.1 Увеличивается время ожидания ответов от Firebird (ответ на простейший SELECT приходится ожидать несколько секунд).

КМАЗС-ОФИС и Firebird – решение проблем

- 1.1 Подвисания интерфейса пользователя (при большом времени Ping > 20 мс)
  - Пример сценария для подвисания интерфейса:
    - Пользователь нажимает кнопку «Показать информацию о водителе»;
    - Программа выполняет серию запросов к БД – **интерфейс «зависает» (например, на 10 сек)**;
    - Окно «Информация о водителе» появляется (с большой задержкой);

  - При подвисании интерфейса программы:
    - Надпись «Не отвечает»;
    - Нет реакции на действия пользователя;
    - Проблемы с порядком расположения окон (могут уехать на задний план);
    - Риск необдуманных действий со стороны пользователя;

    - **Негатив со стороны пользователя** 

  - Решение проблемы:
    1) Создаётся модальное окно, блокирующее GUI (изначально прозрачное);
    2) Создаётся дополнительные поток;
    3) Обращения к БД выполняются в дополнительном потоке;
    4) Постепенно модальное окно становится непрозрачным, отображается общее время операции;
    5) Дополнительный поток завершается, модальное окно закрывается;

    6) Подвисания не произошло, пользователь счастлив 

КМАЗС-ОФИС и Firebird – решение проблем

- 1.2 Медленное подключение к БД (ping > 20 мс)
  - Обычное подключение к БД (без пула) занимает более 50 мс;
  - **Проблема**:
    - лишние задержки/подвисания при частых подключениях/отключениях;
  - **Решение**:
    - Использовать пул подключений, встроенный в библиотеку **ibxFBUtils**.
    - Получение подключение из пула обычно выполняется моментально (< 1 мс)

  - Типовая работа с БД в дополнительном потоке:

```
fdb := fb.Pool.GetConnection(@TranRead, nil);
try
  // Выполняем обращение к базе данных
finally
  fb.Pool.ReturnConnection(fdb);
end;
```

  - Особенность пула: транзакция read-only остаётся в активном состоянии после возврата подключения в пул.

КМАЗС-ОФИС и Firebird – решение проблем

- 1.3 Медленное выполнение автоматических операций (при большом времени Ping > 10 мс)
  - **Проблемы**:
    - лишние задержки/подвисания при SQL-запросах (SELECT / INSERT / UPDATE);
    - выполнение запросов в цикле приводит к неожиданным (плачевным) результатам;
  - **Решение**: повторное использование созданных объектов TIBDataSet (этап “Prepare” выполняется только при первом обращении).

    Библиотека ibxFBUtils для каждой транзакции хранит пул объектов TIBDataSet с быстрым поиском объекта по тексту SQL-запроса. Типичный код:

```
sql := 'SELECT NAME FROM USERS WHERE ID=:UserId';
ds := fb.GetAndOpenDataSet(nil, TranRead, sql, ['UserId'], [1]);
try
  // Выполняем обработку записей...
finally
  // ds.Free; - не удаляем объект!
  ds.Close; // Закрываем датасет для возможности повторного использования
end;
```

  - Аналогичный подход для операций INSERT и UPDATE:

```
fb.InsertRecordDB(nil, TranWrite, 'USERS', ['ID', 1, 'NAME', 'Иванов']);
fb.UpdateRecordDB(nil, TranWrite, 'USERS', ['ID’, 2], ['NAME', 'Петров']);
```

КМАЗС-ОФИС и Firebird – решение проблем

- 1.4 Сбои в работе программы при перезагрузке службы Firebird на сервере
  - **Проблемы**:
    1. Окна с ошибками при обращениях к БД;
    2. Пропадает информация из db-компонентов;
    3. Программа – в неработоспособном состоянии;
  - **Диагностика отсутствия подключения**:
    Периодический (раз в 30 сек) вызов метода TIBDatabase.TestConnected;
  - **Автоматическое восстановление подключения**:
    1. Открытие модального окна с запуском дополнительного потока;
    2. Попытка подключиться к базе данных в цикле;
    3. Визуализация ошибки подключения в модальном окне;
    4. Завершение дополнительного потока и закрытие модального окна;
    5. Переподключение к БД для основного объекта TIBDatabase;
  - **Решение проблемы исчезновения информации из db-компонентов**:
    Используются наборы данных «в памяти» (TRxMemoryData, TJvMemoryData);

КМАЗС-ОФИС и Firebird – решение проблем

- 1.5 Зависание SQL-запросов на 2 часа (при сбоях в работе сети)
  - **Примеры ситуаций**:
    1. Выдернули сетевой кабель из роутера;
    2. Аппаратный сбой в роутере;
    3. Сбой у провайдера;
    4. Ошибка в стороннем ПО (перехватчик TCP-пакетов);
  - **Следствие 1**: Windows не знает о сбое и не меняет состояние сокета;
  - **Следствие 2**: Программа ждет ответ от Firebird 2 часа (возникнет ошибка);
  - **Решение**:
    - Проверка значения параметра «KeepAliveTime» (DWORD) при запуске программы, выдача сообщения оператору;

    «HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters»,

КМАЗС-ОФИС и Firebird – решение проблем

- 1.6 Создание повышенной нагрузки на сеть при частых обращениях к БД (при автоматических операциях)

  - **Пример ситуации**:
    - Каждая программа 10 раз в секунду вызывает функцию fb.GetCurrentDateTime (несколько потоков);
    - Запущено 100 экземпляров программы, генерируется 1000 запросов в секунду (≈1МБ/с);

  - **Проблемы**:
    1. Достигается пропускная способность сети;
    2. Падает производительность на всех рабочих местах;
    3. Провайдер может отключить интернет;

  - **Диагностика проблемы**:
    используется ПО WireShark + экспорт в CSV + анализ в MS Excel

  - **Решения проблемы**:
    - Минимизация частоты обращений к Firebird;
    - Уменьшение размера запрашиваемого набора данных (количество столбцов и записей);
    - Для fb.GetCurrentDateTime – вызываем 1 раз в минуту и запоминаем разницу (FBTimeDelta) с локальным временем;

```
FBTimeDelta := Now - fb.GetCurrentDateTime();

function GetServerTimeFast: TDateTime;
begin
  Result := Now - FBTimeDelta;
end;
```

КМАЗС-ОФИС и Firebird – решение проблем

- 1.7 Создание повышенной нагрузки на сеть при запросе большого объёма данных
  - **Решение**: кэширование справочников учетных записей водителей и техники в оперативной памяти – кардинально снижает нагрузку на сеть и на Firebird;
- 2.1 Неаккуратное управление транзакциями приводит к постепенному ухудшению производительности Firebird
  - **Решение 1**: транзакции для записи – только «короткие»;
  - **Решение 2**: не используется редактирование данных в TDBGrid, изменение данных осуществляется в отдельных окнах;
  - **Решение 3**: периодическая проверка наличия длинных транзакций и уведомление пользователя при их обнаружении;

    Определяется разница между MON$NEXT_TRANSACTION и MON$OLDEST_ACTIVE
- 3. Проблемы производительности при запуске обмена данными с большим количеством контроллеров КМАЗС
  - **Решение**: ограничивается количество потоков, которые обращаются к БД одновременно. Учитывается количество реальных ядер процессора. Используется функция: GetLogicalProcessorInformation

Статьи о программировании

Статья «Многопоточное программирование в Delphi для начинающих»

Цель: обучить многопоточному программированию разработчиков с нулевыми знаниями в этой области;

Ссылка: https://github.com/loginov-dmitry/multithread

Статья «Особенности многопоточного сетевого программирования в Delphi» (2022г)

Цель: показать особенности сетевого программирования с учётом отсутствия в языке Delphi операторов, упрощающих асинхронное программирование;

Ссылка:

https://github.com/loginov-dmitry/multithread/blob/master/multithread_net_programming.md

Статьи о программировании

Статья «Упрощаем разработку приложений баз данных Firebird» (2012 год)

https://github.com/loginov-dmitry/ibxfbutils/blob/master/ibxfbutils-doc.md

Аудитория: разработчики на Delphi/Lazarus, использующие СУБД Firebird

Цель: дать подробную информацию о библиотеке ibxFBUtils

В статье подробно раскрыты темы:

1) Ограничения стандартных компонентов IBX;
2) Создание базы данных с нуля и коррекция структуры БД;
3) Использование пула подключений к базе данных;
4) Хранение конфигурационных параметров в базе данных;
5) Программный бэкап базы данных;
6) Удобный вызов конструкции EXECUTE BLOCK;
7) Простой вызов функций для модификации записей в таблице;
8) Базовые функции библиотеки.

Статьи о программировании

- **Статья «Упрощаем разработку приложений баз данных Firebird» (продолжение)**

Примеры подключения к базе данных:

Пример подключения без пула:

```
var
  fdb: TIBDatabase;
begin
  fdb := fb.CreateConnection('localhost', 3050, 'test_base', User, Password, CharSet);
  try
    // Действия с базой данных
  finally
    fb.FreeConnection(fdb); // Удаление подключения к БД
  end;
end;
```

Пример подключения с пулом:

```
var
  fdb: TIBDatabase; TranR: TIBTransaction;
begin
  fdb := fb.Pool.GetConnection(@TranR); // Получаем объект-подключение из пула
  try
    // Действия с базой данных
  finally
    fb.Pool.ReturnConnection(fdb); // Возвращаем объект-подключение в пул
  end;
end;
```

Статьи о программировании

- **Статья «Упрощаем разработку приложений баз данных Firebird» (продолжение)**

Примеры вставки, изменения, удаления:

Пример вставки записи:

```
fb.InsertRecordDB(fdb, tranW, 'TESTTABLE’,
    ['ID', 1, 'NAME', 'FIREBIRD', 'SUMMA', 100500]);
```

Пример изменения записи:

```
fb.UpdateRecordDB(fdb, tranW, 'TESTTABLE', ['ID', 1],
    ['SUMMA', 1000000, 'RECTYPE', 5]);
```

Пример удаления записи:

```
fb.DeleteRecordDB(fdb, ftran, 'TESTTABLE', ['ID', 1]);
```

Пример вставки либо изменения записи:

```
fb.UpdateOrInsertRecord(fdb, ftran,
    'TESTTABLE',                          // Имя таблицы
    ['ID', 'NAME', 'RECDATE', 'SUMMA'], // Имена полей таблицы
    [1, 'Наименование', Now,   100500], // Значения полей
    ['ID']);                              // Ключевое поле (для поиска совпадения)
```

Статьи о программировании

- **Статья «Конвертация VCL-приложения в Lazarus»**

https://github.com/loginov-dmitry/convert-to-lazarus-notes

Аудитория: разработчики на Delphi/Lazarus

Цель: задокументировать опыт портирования VCL-приложения в Lazarus

Причины разработки статьи:

1) Необходимость поддержки Российских дистрибутивов ОС Linux с целью импортозамещения;
2) Огромная сложность портирования VCL-приложения в Lazarus для поддержки Linux;
3) Недостаточное количество информации в Интернете по данной теме;

В статье подробно раскрыто большое количество тем, в том числе:

1) Нюансы установки и настройки IDE Lazarus;
2) Обзор компонентов (в том числе компоненты доступа к СУБД Firebird);
3) Подготовка проекта к портированию;
4) Конвертация форм dfm -> lfm;
5) Подключение стороннего менеджера памяти;
6) Особенности разработки служб;
7) Поддержка тёмной темы;
8) и многие другие темы.

Статьи о программировании

- **Статья «Конвертация VCL-приложения в Lazarus» (продолжение)**

Особенности статьи:

1) Разработана собственная утилита для конвертации pas и dfm-файлов. Её исходные коды доступны на github: https://github.com/loginov-dmitry/delphi-to-lazarus-converter
2) Разработан модуль MutexObjectUnit, в котором реализована кроссплатформенная обёртка TMutexObject (решает проблему отсутствия именованных мьютексов в линуксе): https://github.com/loginov-dmitry/multithread/blob/master/CommonUtils/MutexObjectUnit.pas
3) Разработана улучшенная версия менеджера привязок компонентов на форме;
4) Портированные исходники остаются совместимы с Delphi (под Windows);

О компонентах IBX от компании MWASoftware:

1) Обеспечена высокая степень совместимости с IBX в составе Delphi;
2) Отличия от Delphi имеются, библиотека ibxFBUtils их учитывает;
3) Нельзя использовать зарезервированные слова («from», «to») в качестве имён параметров.

Библиотеки с открытым исходным кодом

Все перечисленные библиотеки – кроссплатформенные (Delphi + Lazarus)

- ibxFBUtils - https://github.com/loginov-dmitry/ibxfbutils
- LDSLogger - https://github.com/loginov-dmitry/loginovprojects/tree/master/ldslogger
- MutexObjectUnit - https://github.com/loginov-dmitry/multithread/blob/master/CommonUtils/MutexObjectUnit.pas
- TimeIntervals - https://github.com/loginov-dmitry/multithread/blob/master/CommonUtils/TimeIntervals.pas
- LDSWaitFrm - https://github.com/loginov-dmitry/multithread/tree/master/ExWaitWindow
- SmartHolder - https://github.com/loginov-dmitry/loginovprojects/tree/master/smartholder

  TSmartHolder – аналог умных указателей (smart pointers), имеет преимущества:

  - Решает задачу автоматического удаления объектов без использования оператора try..finally;

  - За счёт избавления от try..finally улучшается качество кода.

Библиотеки с открытым исходным кодом

Примеры использования библиотеки SmartHolder

https://github.com/loginov-dmitry/loginovprojects/tree/master/smartholder

```
procedure MyProc;
var
  AList: TStringList;
  Ini: TIniFile;
  sql: string;
  ds: TIBDataSet;
  h: TSmartHolder;
begin
  AList := h.CreateStringList(); // 1-й способ

  Ini := h.RegObj(TIniFile.Create(IniFileName)) as TIniFile; // 2-й способ

  sql := 'SELECT NAME FROM USERS WHERE ID=:UserId';
  ds := fb.GetAndOpenDataSet(nil, TranRead, sql, ['UserId'], [1]);
  h.RegObj(ds); // 3-й способ

  AList.Add(ds.FieldByName('NAME').AsString);

  // Ini.Free; - не требуется
  // AList.Free; - не требуется
  // ds.Free; - не требуется
end;
```

Доклад окончен

Спасибо за внимание!

??? Вопросы ???

https://github.com/loginov-dmitry

Телеграм: @DmitrLoginov