FIREBIRD CONF 2025

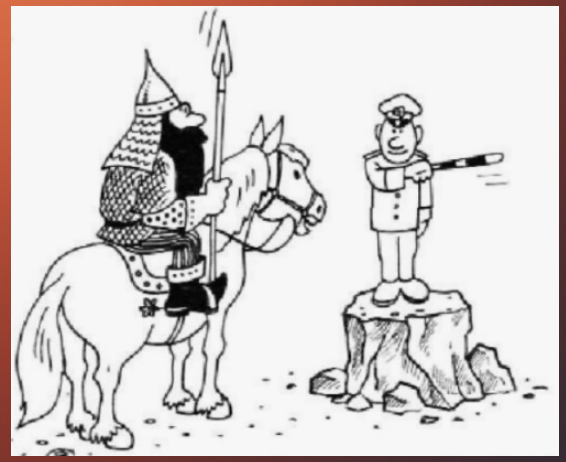
Александр
Шапошников

Миграция с Firebird на PostgreSQL Что может пойти не так?

Проблематика

Проект под Firebird работает несколько лет
Работает с проблемами
Заказчик верит, что проблемы
не в проекте или организации работы,
а в Firebird

- Firebird – плохая СУБД!
- Переходим на другую СУБД!
- Все используют PostgreSQL, она бесплатна – переходим на нее!



- Привлечение компаний-консультантов
- Заключение договора на техподдержку
  Переход на улучшенные версии Firebird
- Обучение и сертификация сотрудников
- Введение профессионального мониторинга СУБД

СУБД в этой ситуации значения не имеет. Вверху экрана - цитата из доклада о проблемах эксплуатации Postgres, с одним изменением - в оригинале вместо «Firebird» везде указано «PostgreSQL»

«Безвыходным мы называем положение, выход из которого нам не нравится...» Станислав Ежи Лец

Трудности миграции

Postgres – замечательная СУБД,
но она – ДРУГАЯ

Поговорим именно о тех отличиях,
которые могут оказаться
неприятными неожиданностями
при миграции

О тех вещах, которые мы считаем
при работе с Firebird
естественным и само собой
разумеющимся, и что может
оказаться далеко не так в
Postgres

Каждый, страдающий мигренью, хотел бы, чтобы она мигрировала в чью-то другую голову.

ТЕСТОВАЯ ТАБЛИЦА DAT

ПОЛЯ

```
CREATE TABLE DAT (ID BIGINT NOT NULL,
I1..I8 BIGINT,
N1..N8 DOUBLE PRECISION,
D1..D8 TIMESTAMP,
S1..S8 VARCHAR(100),
T1..T8 VARCHAR(1000));
PK$DAT PRIMARY KEY (ID);
```

ДАННЫЕ

В таблице – 10 млн записей

Первое поле группы заполнено всегда

Оставшиеся поля группы (2-8) с 50% вероятностью заполнено NULL

Данные таблицы идентичны в различных БД

ИНДЕКСЫ

```
X_I1 ON DAT (I1); X_I23 ON DAT (I2, I3); X_I456 ON DAT (I4, I5, I6);
X_N1 ON DAT (N1); X_N23 ON DAT (N2, N3); X_N456 ON DAT (N4, N5, N6);
X_D1 ON DAT (D1); X_D23 ON DAT (D2, D3); X_D456 ON DAT (D4, D5, D6);
X_S1 ON DAT (S1); X_S23 ON DAT (S2, S3); X_S456 ON DAT (S4, S5, S6);
X_F1 ON DAT (F1); X_F23 ON DAT (F2, F3); X_F456 ON DAT (F4, F5, F6);
X_T1 ON DAT (T1);
```

ОКРУЖЕНИЕ

Linux Mint 22.1(SSD, 32GbRAM)

Firebird 5.0.2, Postgres 17.4

Произведено базовое конфигурирование серверов БД.

Postgres не умеет останавливаться при Fetch-е без указания явной инструкции в тексте SQL

SELECT * FROM DAT (10 млн записей)

Firebird 5	Postgres 17
Результат: Записей 32 Prepare time = 16ms Execute time = 0ms	Total rows: 10 000 000 Query complete 00:01:15.039

Подобное поведение может оказаться крайне неприятным при работе с хранимыми процедурами и функциями

Нужно уметь вовремя остановиться...

Особенно хирургам и парикмахерам

ТЕСТОВАЯ ТАБЛИЦА DAT

ПОЛЯ

```
CREATE TABLE DAT (ID BIGINT NOT NULL,
I1..I8 BIGINT,
N1..N8 DOUBLE PRECISION,
D1..D8 TIMESTAMP,
S1..S8 VARCHAR(100),
T1..T8 VARCHAR(1000));
PK$DAT PRIMARY KEY (ID);
```

ДАННЫЕ

В таблице – 10 млн записей

Первое поле группы заполнено всегда

Оставшиеся поля группы (2-8) с 50% вероятностью заполнено NULL

Данные таблицы идентичны в различных БД

ИНДЕКСЫ

```
X_I1 ON DAT (I1); X_I23 ON DAT (I2, I3); X_I456 ON DAT (I4, I5, I6);
X_N1 ON DAT (N1); X_N23 ON DAT (N2, N3); X_N456 ON DAT (N4, N5, N6);
X_D1 ON DAT (D1); X_D23 ON DAT (D2, D3); X_D456 ON DAT (D4, D5, D6);
X_S1 ON DAT (S1); X_S23 ON DAT (S2, S3); X_S456 ON DAT (S4, S5, S6);
X_F1 ON DAT (F1); X_F23 ON DAT (F2, F3); X_F456 ON DAT (F4, F5, F6);
X_T1 ON DAT (T1);
```

ОКРУЖЕНИЕ

Linux Mint 22.1(SSD, 32GbRAM)

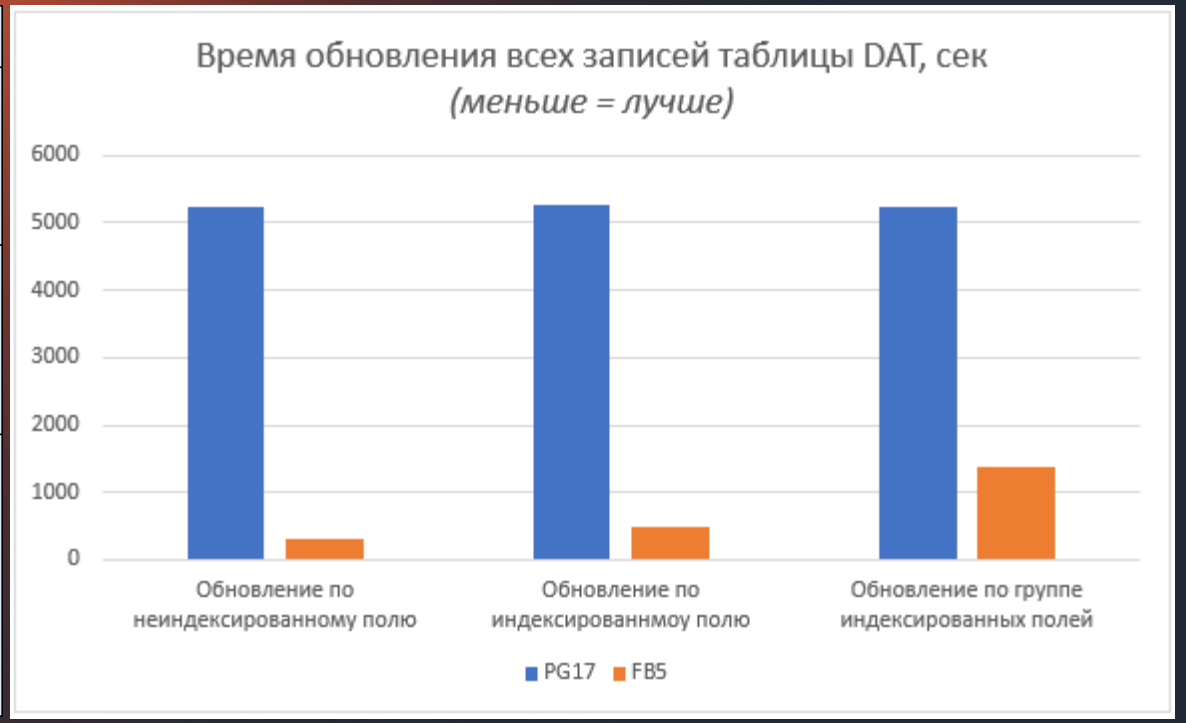
Firebird 5.0.2, Postgres 17.4

Произведено базовое конфигурирование серверов БД.

Особенность реализации MVCC в PostgeSQL и массовые обновления

MVCC в PostgreSQL: на каждое изменение записи создается новая строка. При отмене транзакции новая строка помечается на удаление, а при успешном завершении транзакции – старая

Обновление 10 млн записей	Firebird 5	Postgres 17
По неиндексированному полю ***update dat set n8=0***	5 минут 3 секунды	1 час 27 минут 23 секунды
По индексированному полю ***update dat set i4=-i4***	8 мин 16 секунд	1 час 27 минут 42 секунды
По группе индексированных полей ***update dat set i1=0, f2=true, n4=4, d6='1974-04-01', s3='FC Bayern', T1='1{...}Z'***	23 минуты 16 секунд	1 час 26 минут 29 секунд



+ раздувание (bloat) таблицы и индексов

Не обновление, а ОБНОВЛЕНЬИЩЕ!

(С) Нагиев

СХД и короткие высокочастотные запросы

Запросы со временем выполнения порядка 10 миллисекунд, выполняющиеся миллионы раз суммарно «просядут» по производительности

Причина: БД postgres – это множество файлов

Очень небольшой overhead (5-10мс) на обмен служебной информацией с СХД о файлах БД при выполнении каждого запроса становится существенным в процентном отношении

НЕ ЛЕЧИТСЯ!

Больше – не значит лучше

(с) Хромосомы

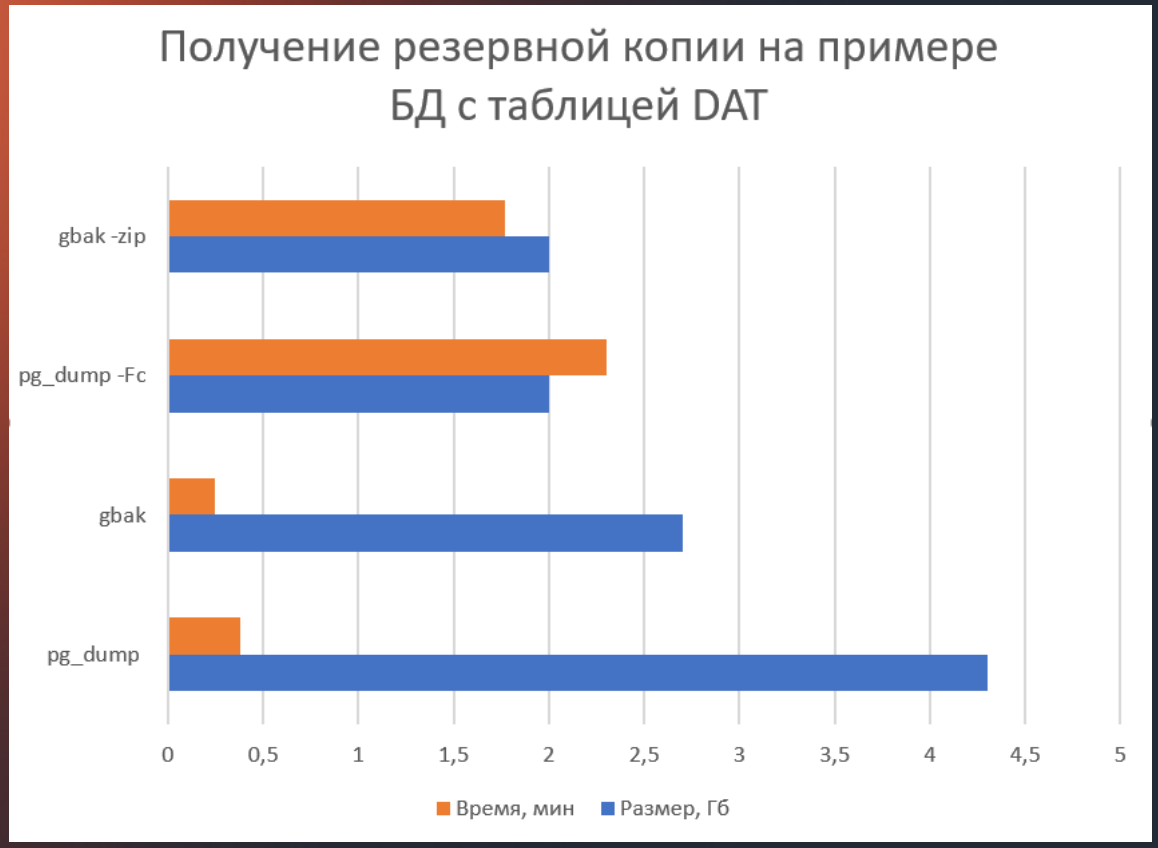
Особенности резервного копирования

В PostgreSQL с бэкапами все совсем не так, как в других БД – настолько, что это требует перелома психики и всех знаний, полученных при работе с другими СУБД.

Чтобы делать быстрые и надежные бэкапы в Postgres нужно понять главное: термин «база данных» в резервном копировании Postgres – это обман. Назовите ее чем угодно, но работайте только с кластером.

pg_dump/pg_restore – для миграции
pg_basebackup или pg_probackup (от Postgres Professional) – для горячего копирования

Nbackup от Firebird: счастье, о котором придется забыть

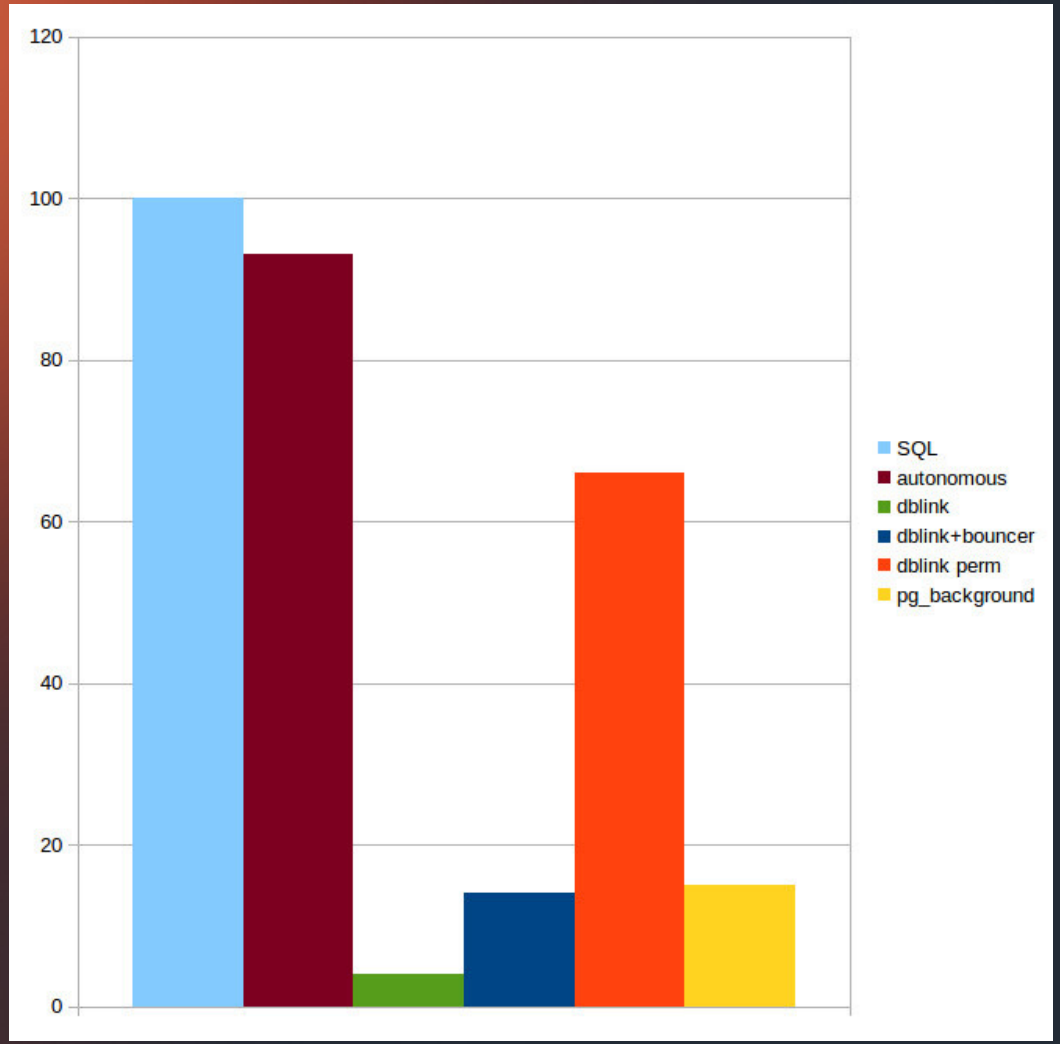


Ты спросишь меня, что важнее: Ты или бэкап?
Я отвечу тебе, что бэкап. И Ты уйдешь, а Я Раз!
И откатил назад

Отсутствие поддержки автономных транзакций в ванильной версии PG

В PostgreSQL автономных транзакций вообще нет. Их можно имитировать, запуская новое соединение при помощи dblink или pg_background, но это выливается в накладные расходы, сказывается на быстродействии и попросту неудобно

Только в Postgres Pro Enterprise (даже не Standard!) реализована полноценная поддержка автономных транзакций с выполнением автономной транзакции внутри того же серверного процесса.



Ограниченный функционал анонимного PSQL блока (DO)

Из документации postgres:

46.4. Анонимные блоки кода

“Анонимный блок кода не принимает аргументы, а любое значение, которое он мог бы вернуть, отбрасывается. В остальном он работает подобно коду функции“

Подобных ограничений нет в реализации анонимного блока EXECUTE BLOCK Firebird

Временные таблицы в PostgreSQL

ПРОБЛЕМЫ

Активное создание, модификация и удаление временных таблиц (включая работу с индексами) может привести к существенной деградации производительности

AVTOVACUUM не видит временные таблицы Соответсвующие операции очистки и анализа нужно выполнять вручную

Характерный паттерн работы с временными таблицами ведёт к переполнению кеша ОС мусором

pg_class (и прочие системные таблицы) разрастаются в процессе работы

Запрет записи во временные таблицы в ReadOnly-реплике
(при построении отчетности)

И РЕШЕНИЯ

Размещение временных таблиц на RAM-диске

Использование нежурналируемых (UNLOGGED) таблиц вместо временных

Использование Postgres Pro, где механизм временных таблиц существенно переработан

Ограничение использования временных таблиц в своих приложениях

- А умываться тогда как?
- А надо меньше пачкаться!

Ограничения Логической Репликации

Из документации postgres: Логическая репликация в настоящее время имеет ограничения и недостатки, описанные ниже...

Схема базы данных и команды DDL не реплицируются {...}

Данные последовательностей не реплицируются {...}

Большие объекты (*Примечание докладчика: не TOAST, а те, что в pg_large_object*) не реплицируются {...}

Репликация поддерживается только для таблиц {...}
Требуется корректная настройка подписки для репликации команды TRUNCATE и Секционированных таблиц

«Пишу с ошибками. Никак не нащупаю в грамматике математическую закономерность...»
Из фильма «Расписание на послезавтра»

32-битный счетчик транзакций

Остановки БД не произойдёт (благодаря механизму VACUUM FREEZE), но затраты на эту процедуру могут оказаться очень высокими

VACUUM FREEZE
спасает, но не более

64-разрядные счетчики транзакций - это принципиальная переделка ядра СУБД, к тому же и нужна только для сильно нагруженных систем, но для таких систем она не просто желательна.

Она необходима.

Совместимость многочисленных расширений при РАЗЛИЧНЫХ реализациях

Есть реализации (от Postgres Professional и прочих коммерческий версий Postgres)

Инертность сообщества

Отказ во включении в код из за размера и сложности

Счетчик транзакций в Firebird – 48-битный

Отзывчивость Firebird-Сообщества

Из фотоархива Дмитрия Кузьменко

Какой-то древнелохматый год

- Влад, нужен gbak с таймингом в логе... ОЧЕНЬ нужен!
- Ну, если так нужен... Сделаем

Последствия зависших транзакций VACUUM FULL и GFIX -sweep

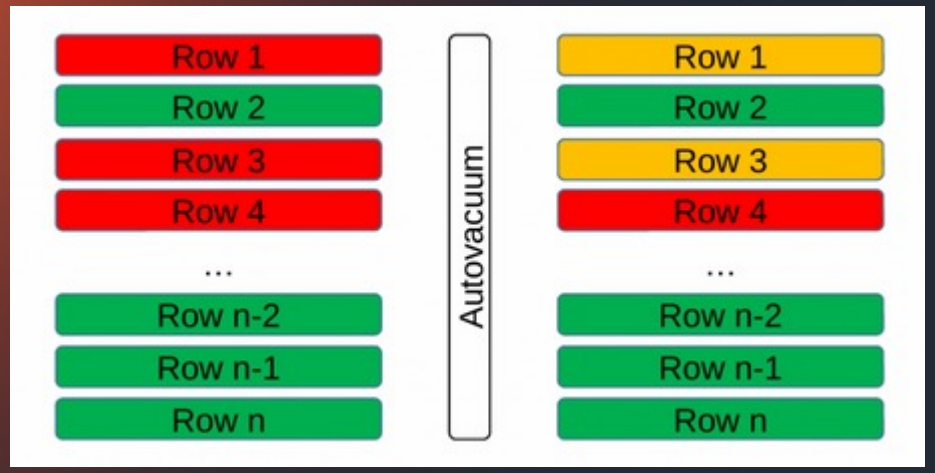
VACUUM FULL

Некоторые проблемы возникающие из-за зависших транзакций (раздувание таблиц и индексов) устраняются только через VACUUM FULL, а он

- выполняется гораздо дольше чем VACUUM
- запрашивает исключительную блокировку таблицы
- требует дискового пространства

(записывает новую копию таблицы и не освобождает старую до завершения операции)

Существует расширение pg_repack от Postgres Pro Standard для реорганизации таблиц без исключительной блокировки



GFIX -SWEEP

Устраняет все проблемы без блокировок базы данных и таблиц
Не отдаст место операционной системе, но прочистит таблицы и индексы

Postgres: Прочая специфика

- Одна транзакция на соединение
- «Процессная», а не потоковая архитектура
- Один или более процессов на соединение
- Более высокое потребление системных ресурсов
- Сложность настройки и администрирования
- Linux-ориентированность, отказ от поддержки Windows в некоторых коммерческих версиях

Postgres: Прочая специфика

- Различный PSQL и SQL
- Отсутствие имплементации пакетов
- Иная концепция реализации триггеров
- Перегружаемые функции
- Добавление поля только в конец таблицы
- Отсутствие проверки зависимостей при компиляции процедур и функций

Postgres: Прочая специфика

- Отсутствие Embedded-версии
- Невозможность управления планировщиком
- Изменение размера страницы БД только через перекомпиляцию
- Отсутствие удобного менеджера для работы с процедурами
- Отсутствие гарантий поддержки opensource расширений
- Специфика изменения размеров данных в БД

Дополнительные Материалы

- Неочевидные для начинающих тонкости Postgres
- PostgreSQL и временные таблицы
- Инструкция по бэкапу одной базы в Postgres – миф или реальность?
- Различия Postgres Pro Enterprise и PostgreSQL
- Тонкости эксплуатации PostgreSQL
- Как выжить разработке, когда прод переезжает на PostgreSQL
- Как PostgreSQL может сделать больно, когда не ожидаешь?
- Типовые ошибки в приложениях, которые ведут к bloat в postgresql
- 10 Things I Hate About PostgreSQL
- Странное поведение планировщика запросов PostgreSQL

Благодарности

Firebird Foundation

«Айбэйз»

Алексей Ковязин

За спонсирование создания данного контента

Алексей Бехтин

«РЕД СОФТ»

За многочисленные идеи и предоставленные материалы

Вот ДОМ, который построил Джим Старки
Серьезно он строил его, без запарки
Продуман фундамент, фасад и проводка
Ну что же тут скажешь – не дом, а находка

Вот Дмитрий Еманов, он – Главный Рабочий
Он ДОМ улучшает и делает больше...
У Джима приняв Знамя Огненной Птицы
Сейчас продолжает упорно трудиться

А как же реклама, а как же оценка?
Встречайте спеца – это Дима Кузьменко
Уже четверть века твердит ежедневно
Что нет ДОМА лучше... И это же верно...

А как же проектная документашка?
Порядок! Ошибки тут нет и промашки!
Оформлено лучше любого диплома –
Нам Симонов пишет про версии ДОМа

Роман Симаков занял должность прораба
Чтоб не было всяческих ям и ухабов
На долгом и трудном проектном пути
Чтоб легче, быстрее нам к счастью идти

И Леша Ковязин, в нем – наши надежды
Возглавил такой же проект зарубежный
Чтоб санкции не были ДОМу помехой
Вот только... Как жаль, что Ты, Леха, уехал...

Просторен и светел построенный ДОМ
Удобно и здорово жить вместе в нем