



Миграция приложения с MS SQL Server на PostgreSQL

Структура доклада



1. Стоимость владения - когда софт дороже железа
2. Инструменты для работы с PostgreSQL
3. Инструменты мониторинга
4. Миграция схемы данных и подводные камни
5. Разница в синтаксисе
6. Миграция схемы и данных средствами efCore
7. Результаты нагрузочного тестирования
8. Синхронизация и проверка целостности данных

Продукты входящие в MS SQL Server

1. SQL Server
2. SQL Server Integration Services (SSIS)
3. SQL Server Analysis Services (SSAS)
4. SQL Server Reporting Services (SSRS)



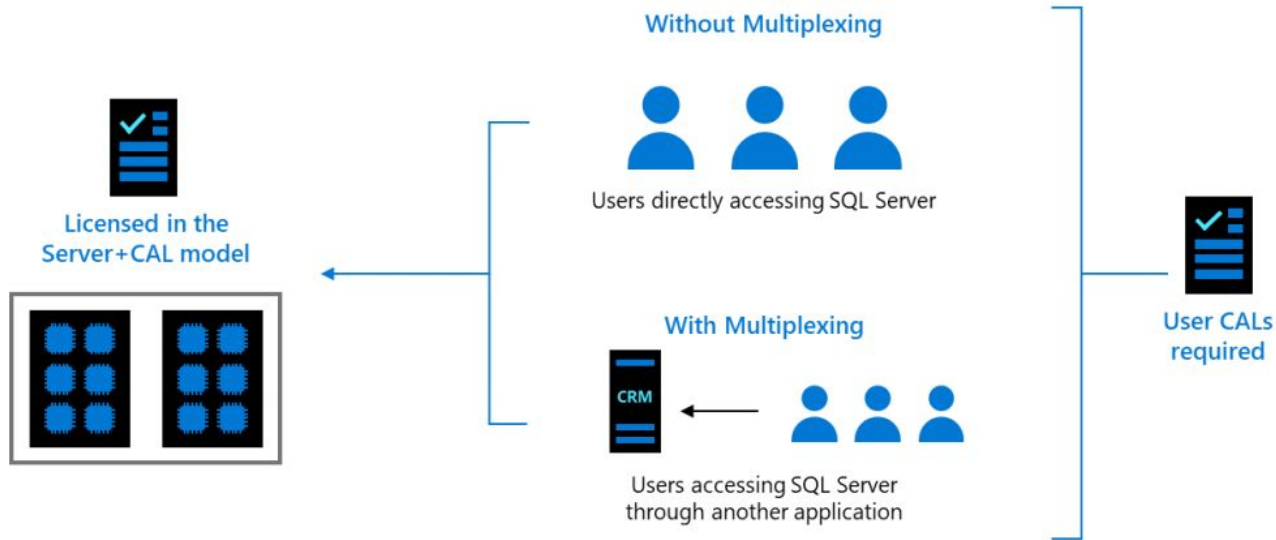
<https://docs.microsoft.com/ru-ru/sql/reporting-services/create-deploy-and-manage-mobile-and-paginated-reports?view=sql-server-ver15>

Различные издания MS SQL Server

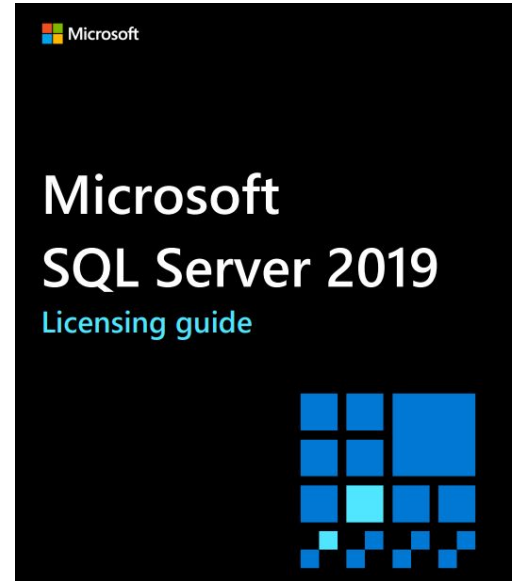
SQL Server 2019 Edition	Database Engine (DBE) capacity limits		
	Max compute capacity	Max memory utilization - DBE	Max DB Size
Enterprise Per Core	OS max	12 TB	524 PB
Standard Per Core	Lesser of 4 sockets or 24 cores	128 GB	524 PB
Standard Server + CAL	24 core limit	128 GB	524 PB
Express	Lesser of 1 socket or 4 cores	1 GB	10 GB
Developer	OS max	OS max	OS max

Выпуски	Цена для Open No Level (долл. США)	Модель лицензирования	Доступность канала
Enterprise	\$13,748 ^[1]	Пакет на 2 ядра	Корпоративное лицензирование, хостинг
Standard — на ядро	\$3,586 ^[1]	Пакет на 2 ядра	Корпоративное лицензирование, хостинг
Standard — сервер	\$899 ^[1]	Сервер ^[2]	Корпоративное лицензирование, хостинг
Standard — CAL	\$209	CAL	Корпоративное лицензирование, хостинг
Developer	Бесплатно	На пользователя	Бесплатное скачивание
Web	Цены уточняйте у партнера по размещению	Неприменимо	Только хостинг
Express	Бесплатно	Неприменимо	Бесплатное скачивание

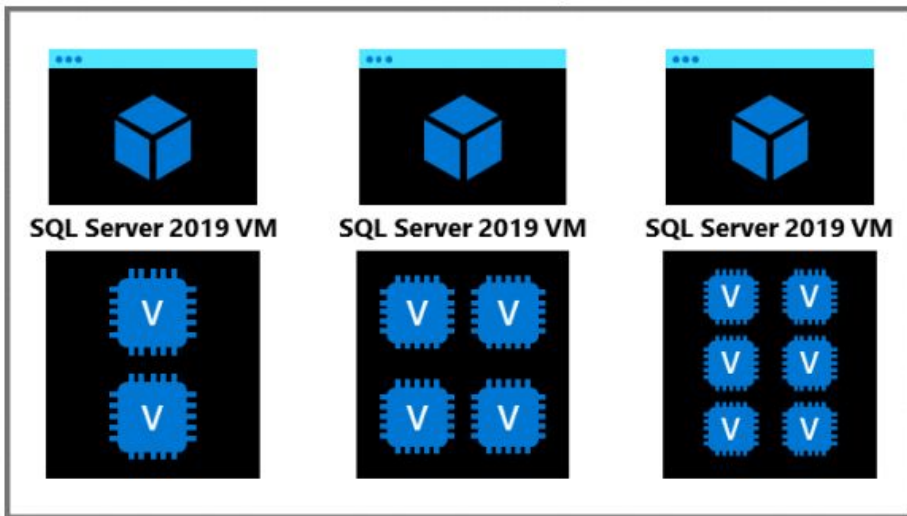
Модель лицензирования Server + CAL



This figure illustrates the licenses used in the Server+CAL licensing model via multiplexing.



Модель лицензирования Core-based



Virtual cores	2	4	6	
Licenses	4	4	6	14 Total core licenses Purchase seven 2-pack SKUs of core licenses

Когда софт в разы дороже железа



Лицензии

- MS SQL core based licence (16 core)

Цена: 28 688 usd

- MS Windows Server Standard (16 core)

Цена: 15 522 usd

- <https://www.microsoft.com/ru-ru/windows-server/pricing>
- <https://www.microsoft.com/ru-ru/sql-server/sql-server-2019-pricing>

Сервер HPE ProLiant DL360 Gen10

- Intel Xeon (5218) 16 core 2.3 GHz
- RAM 32Gb (max 1.5Tb)
- RAID P408i-a

Цена: 357 100 rub ~ 4 700 usd

- https://www.citilink.ru/catalog/computers_and_notebooks/servers_and_net equipments/servers/1191586/properties/

Сравнение стоимости в облачном провайдере AWS

Instance type: m5.2xlarge

- 8 vCPUs
- 32 Gb memory

Windows Server and SQL Server on Amazon EC2 estimate

Amazon Elastic Block Storage (EBS) pricing (monthly)	95.20 USD
Amazon EC2 Reserved instances (monthly)	0.00 USD
Amazon EC2 On-Demand instances (monthly)	1,305.24 USD
Total monthly cost:	1,400.44 USD

Cancel

Add to my estimate

Amazon EC2 estimate

Amazon Elastic Block Storage (EBS) pricing (monthly)	3.57 USD
Amazon EC2 Instance Savings Plans instances (monthly)	0.00 USD
Amazon EC2 On-Demand instances (monthly)	335.80 USD
Total monthly cost:	339.37 USD

Cancel


Add to my estimate

Реестр отечественного ПО и 20% НДС



4.1. СУБД

Запрещено	Разрешено
<ol style="list-style-type: none">1. EnterpriseDB2. IBM DB23. InterSystems Caché4. Microsoft Access5. Microsoft SQL Server6. Oracle Database7. Oracle MySQL (Standard Edition, Enterprise Edition, Cluster Carrier Grade Edition)8. Oracle NoSQL Database9. Redis Enterprise10. SAP HANA11. SAP Adaptive Server Enterprise (ASE) / Sybase Adaptive Server Enterprise (ASE)12. SAP SQL Anywhere / Sybase SQL Anywhere13. Splunk	<ol style="list-style-type: none">1. Любые СУБД из Единого реестра2. СУБД с открытой лицензией, в частности:<ol style="list-style-type: none">а) CouchDBб) Elasticsearchв) Firebird (рекомендуется перейти на российский аналог Ред База Данных)г) Hiveд) MariaDBе) MongoDBж) Oracle MySQL (Community Edition)з) PostgreSQL (рекомендуется перейти на российский аналог PostgresPro)и) Redis (open-source edition)3. СУБД стран, не налагающих санкции, например:<ol style="list-style-type: none">а. Tmaxsoft Tiberio



Инструменты для работы с PostgreSQL

DBeaver Community - структура таблицы

Database Navigator

Enter a part of table name here

▼ MonocardIdentity dev35 - tst-pgdev-01.1

▼ MonocardIdentity_develop35

▼ public

- AspNetRoleClaims 24Kb
- AspNetRoles 64Kb
- AspNetUserClaims 24Kb
- AspNetUserLogins 24Kb
- AspNetUserRoles 1.8Mb
- AspNetUserTokens 16Kb
- AspNetUsers 4.2Mb
- __EFMigrationsHistory
- __MigrationHistory 48Kb
- get_load_average_copy
- pg_buffercache
- pg_qualstats
- pg_qualstats_all
- pg_qualstats_by_query
- pg_qualstats_pretty
- pg_stat_statements

AspNetUsers

Table Name: AspNetUsers Object ID: 103000

Tablespace: Owner: monocard_owner

Has Oids Partitions

Extra Options:

Partition by:

Comment:

#	Name	Data type	Not Null	Length	Collation	Access Method	Predicate	Unique	Rel Size	Identit
1	Id	varchar	<input checked="" type="checkbox"/>	450	default					
2	Email	varchar	<input type="checkbox"/>	256	default					
3	NormalizedEmail	varchar	<input type="checkbox"/>	256	default					
4	EmailConfirmed	bool	<input checked="" type="checkbox"/>	1						
5	PasswordHash	text	<input type="checkbox"/>		default					
6	SecurityStamp	text	<input type="checkbox"/>		default					
7	PhoneNumber	text	<input type="checkbox"/>		default					
8	PhoneNumberConfirmed	bool	<input checked="" type="checkbox"/>	1						
9	TwoFactorEnabled	bool	<input checked="" type="checkbox"/>	1						
10	LockoutEndDateUtc	timestamptz	<input type="checkbox"/>							
11	LockoutEnd	timestamptz	<input type="checkbox"/>							
12	LockoutEnabled	bool	<input checked="" type="checkbox"/>	1						
13	AccessFailedCount	int4	<input checked="" type="checkbox"/>							
14	UserName	varchar	<input type="checkbox"/>	256	default					
15	NormalizedUserName	varchar	<input type="checkbox"/>	256	default					
16	ConcurrencyStamp	text	<input type="checkbox"/>		default					
	AspNetUsers_pkey					btree		<input checked="" type="checkbox"/>	520Kb	
	AspNetUsers_pkey					btree		<input type="checkbox"/>	432Kb	
	emailindex					btree		<input type="checkbox"/>	432Kb	
	normalizedusernameindex					btree	("NormalizedUserName" I...	<input checked="" type="checkbox"/>	432Kb	
	usernameindex					btree	("UserName" IS NOT NULL)	<input checked="" type="checkbox"/>	432Kb	

DBeaver Community - план выполнения

The screenshot displays the DBeaver Community interface. On the left, the 'Database Navigator' shows a tree view of the 'MonocardIdentity dev35' database, including tables like 'AspNetUsers' (4.2Mb) and 'AspNetRoles'. The main window shows a SQL script with the following query:

```
select *
from public."AspNetUsers" usr
inner join public."AspNetUserRoles" usrRol
  on usrRol."UserId" = usr."Id"
inner join public."AspNetRoles" rol
  on rol."Id" = usrRol."RoleId"
where "NormalizedEmail" = 'USER637501948903282134@MONOPOLY.SU';
```

Below the query, the 'Execution plan - 1' is shown. The plan consists of a Nested Loop join, which is further broken down into nested loops and index scans.

Node Type	Entity	Cost	Rows	Time	Condition
√ Nested Loop		0.69 - 16.77	1	0.033	
√ Nested Loop		0.56 - 16.61	1	0.026	
Index Scan	AspNetUsers	0.28 - 8.30	1	0.014	((("NormalizedEmail")):te
Index Only Scan	AspNetUserRoles	0.28 - 8.30	1	0.008	("UserId" = (usr."Id")):tex
Index Scan	AspNetRoles	0.13 - 0.15	1	0.005	((("Id")):text = (usrrol."Ro

On the right side, the 'Details' for the 'Index Scan' node are shown:

Name	Value
√ General	
Node Type	Index Scan
Entity	AspNetUsers
Cost	0.28 - 8.30
Rows	1
Time	0.014
Condition	((("NormalizedEmail")):text = 'USER637501948903282134...
√ Details	
Parent-Relationship	Outer
Parallel-Aware	false
Scan-Direction	Forward
Index-Name	emailindex
Relation-Name	AspNetUsers
Alias	usr
Startup-Cost	0.28
Total-Cost	8.30
Plan-Rows	1
Plan-Width	373
Actual-Startup-Time	0.013
Actual-Total-Time	0.014
Actual-Rows	1
Actual-Loops	1

DBeaver Community - план выполнения

The screenshot displays the DBeaver Community interface. On the left is the Database Navigator showing a tree view of the 'Monocardidentity dev35' database, including a 'public' schema with various tables like 'AspNetRoleClaims' and 'AspNetUsers'. The main window shows a SQL script with the following query:

```
select *
from public."AspNetUsers" usr
inner join public."AspNetUserRoles" usrRol
on usrRol."UserId" = usr."Id"
inner join public."AspNetRoles" rol
on rol."Id" = usrRol."RoleId"
--where "NormalizedEmail" = 'USER637501948903282134@MONOPOLY.SU';
```

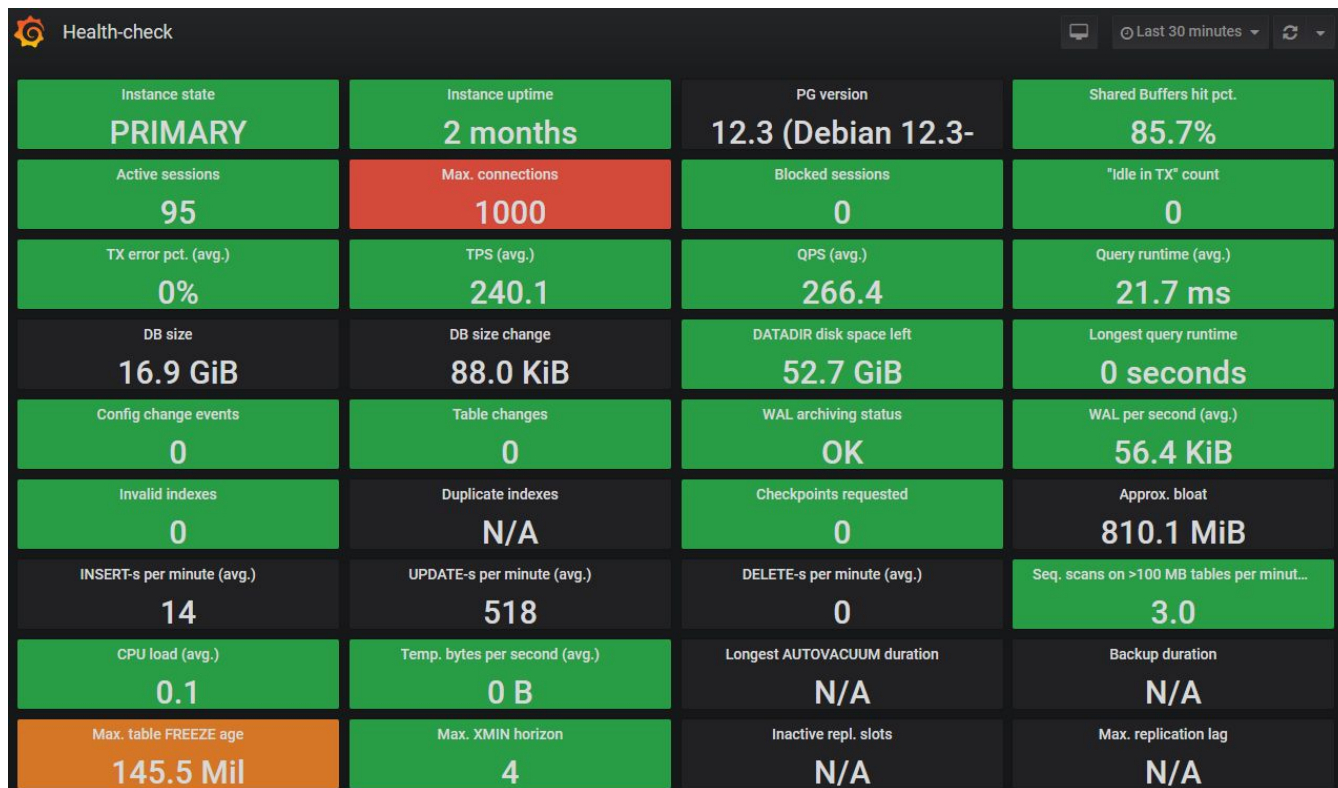
Below the query, the execution plan is shown. The plan consists of several nodes:

Node Type	Entity	Cost	Rows	Time	Condition
Hash Join		463.16 - 671.14	5825	17.483	((usrrol."RoleId")::text = (rol."Id")::text)
Hash Join		462.12 - 622.51	5825	15.822	((usrrol."UserId")::text = (usr."Id")::text)
Seq Scan	AspNetUserRoles	0.00 - 145.12	5825	0.695	
Hash		378.72 - 378.72	6675	12.270	
Seq Scan	AspNetUsers	0.00 - 378.72	6675	7.228	
Hash		1.02 - 1.02	2	0.010	
Seq Scan	AspNetRoles	0.00 - 1.02	2	0.006	

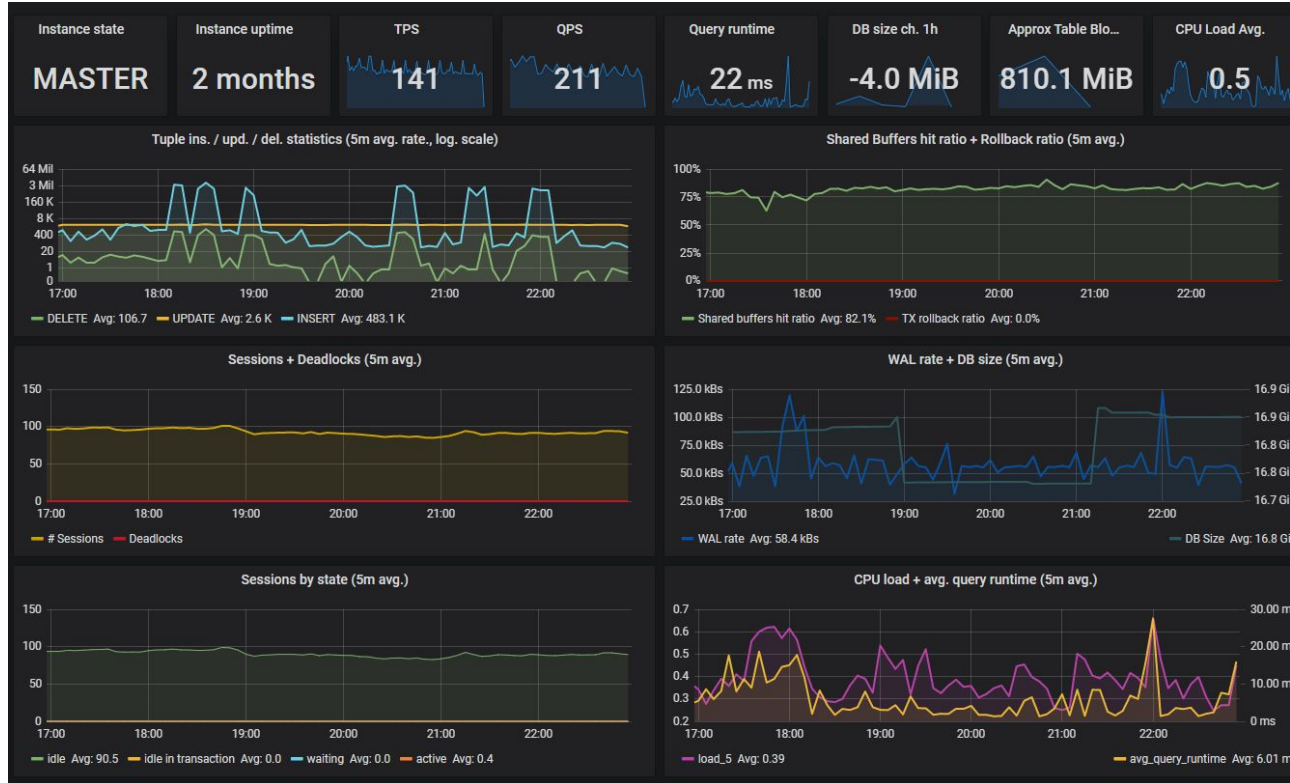
On the right side of the execution plan window, a 'Details' pane is open for the selected 'Hash Join' node, showing various statistics:

Name	Value
General	
Node Type	Hash Join
Entity	
Cost	462.12 - 622.51
Rows	5825
Time	15.822
Condition	((usrrol."UserId")::text = (usr."...
Details	
Parent-Relation	Outer
Parallel-Aware	false
Join-Type	Inner
Startup-Cost	462.12
Total-Cost	622.51
Plan-Rows	5812
Plan-Width	447
Actual-Startup-	12.299
Actual-Total-Tir	15.822
Actual-Rows	5825
Actual-Loops	1
Inner-Unique	true
Hash-Cond	((usrrol."UserId")::text = (usr."...

PG Watch - Health check dashboard




PG Watch - Db overview dashboard



PG Watch - Stat statements top

Total runtime	Query ID	Query
14.6 min	<u>7218968801041544637</u>	SELECT CASE WHEN \$1 IN (SELECT x4."ContractId" FROM "ContractTemplates" AS x4 WHERE
6.1 min	<u>-6226185490001164467</u>	SELECT CASE WHEN \$1 IN (SELECT x3."ContractId" FROM "FuelSupplySchemes" AS x3 WHERE
3.8 min	<u>5184857525036645206</u>	SELECT SUM(x."OwnCost") AS "OwnSum", SUM(x."OffsettingCost") AS "OffsettingSum" FROM "T (x.Transaction"."RefuelingDateLocal" > \$2)) AND ("x.Transaction"."Status" = \$4) GROUP BY x."C
25.0 s	<u>5063669567141266060</u>	SELECT o."Id", o."BaseFuelLimit", o."ContractId", o."CreatedAt", o."CreatedBy", o."DocumentNumbr o."OrganizationInn", o."OrganizationName", o."PersonId", o."RefuelNotAllowedReasons", o."RegN o."UpdatedAt" > \$1 ORDER BY o."UpdatedAt" LIMIT \$2

Calls	Query ID	Query
587.0 K	<u>7218968801041544637</u>	SELECT CASE WHEN \$1 IN (SELECT x4."ContractId" FROM "ContractTemplates" AS x4 WHERE x4
177.4 K	<u>4547141589903725484</u>	SELECT s."Id", s."LastSynchronizedAgreement", s."LastSynchronizedOperator", s."AgreementsAmo INNER JOIN "InfrastructureUnit_FuelStations" AS s ON h."FuelStationId" = s."Id" WHERE (((x."IsCl
88.0 K	<u>7676434299078060370</u>	SELECT o."Id", o."CreatedAt", o."CreatedBy", o."DeletedAt", o."LastDateUpdate", o."NetworkType", o. BY o."LastDateUpdate" DESC LIMIT \$2



Миграция схемы и подводные камни

Нюансы при миграции схемы в PostgreSQL

Чувствительность схемы данных к регистру (в двойных кавычках)

```
CREATE TABLE TableNameInLowerCase (  
    Id uuid NOT null  
)  
  
-- (ok)  
select Id  
from TableNameInLowerCase  
  
-- (ok)  
select id  
from tablenameinlowercase  
  
-- (ok)  
select ID  
from TABLENAMEINLOWERCASE
```

tablenameinlowercase

select Id from TableNameInLowerCase

id

```
CREATE TABLE "TableWithCaseSensitiveColumns" (  
    "Id" uuid NOT null,  
    "ID" uuid NOT null)  
  
-- (ok)  
select "Id"  
from "TableWithCaseSensitiveColumns"  
  
-- (wrong) incorrect table name  
select "Id"  
from "TABLEWITHCASESENSITIVECOLUMNS"  
  
-- (wrong) incorrect column name  
select "id"  
from "TableWithCaseSensitiveColumns"
```

Results

select "id" from "TableWithCaseSensitiveColumns"

SQL Error [42703]: ERROR: column "id" does not exist
Hint: Perhaps you meant to reference the column "TableWithCaseSensitiveColumns.Id".
Position: 42

! Используйте двойные кавычки только в сценариях, где регистр схемы важен !

Нюансы при миграции схемы в PostgreSQL

Уникальность имен идентификаторов индексов (indexes) и внешних ключей (foreign keys)

```
CREATE INDEX IX_ContractId_IsClosed
ON public."Payments"
USING btree ("ContractId", "IsClosed");

CREATE INDEX IX_ContractId_IsClosed
ON public."TransactionCosts"
USING btree ("ContractId", "IsClosed");
```

Results - 6

```
CREATE INDEX IX_ContractId_IsClosed ON public."TransactionCosts"
```



SQL Error [42P07]: ERROR: relation "ix_contractid_isclosed" already exists

```
CREATE INDEX IX_Payments_ContractId_IsClosed
ON public."Payments"
USING btree ("ContractId", "IsClosed");

CREATE INDEX IX_TransactionCosts_ContractId_IsClosed
ON public."TransactionCosts"
USING btree ("ContractId", "IsClosed");
```

Statistics

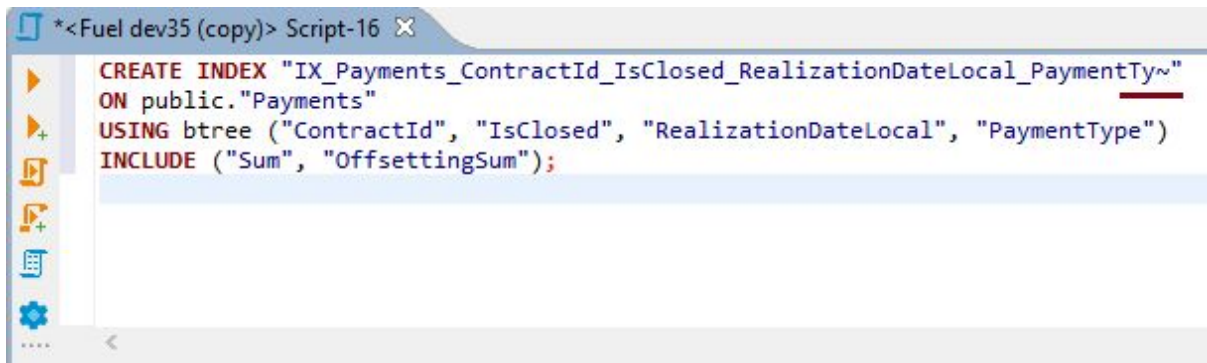
```
CREATE INDEX IX_TransactionCosts_ContractId_IsClosed ON public
```

Name	Value
Updated Rows	0

! Добавляйте имя таблицы в название индекса !

Нюансы при миграции схемы в PostgreSQL

Максимальная длина идентификатора 63 bytes = 63 chars
(table name, column name, index, foreign keys)



```
*<Fuel dev35 (copy)> Script-16 x
CREATE INDEX "IX_Payments_ContractId_IsClosed_RealizationDateLocal_PaymentTy~"
ON public."Payments"
USING btree ("ContractId", "IsClosed", "RealizationDateLocal", "PaymentType")
INCLUDE ("Sum", "OffsettingSum");
```

МАППИНГ ТИПОВ MS SQL В PostgreSQL

	SQL Server		PostgreSQL
1	BIGINT	64-bit integer	BIGINT
2	BINARY(<i>n</i>)	Fixed-length byte string	BYTEA
3	BIT	1, 0 or NULL	BOOLEAN
4	CHAR(<i>n</i>), CHARACTER(<i>n</i>)	Fixed-length character string, 1 = <i>n</i> ≤ 8000	CHAR(<i>n</i>), CHARACTER(<i>n</i>)
5	DATE	Date (year, month and day)	DATE
6	DATETIME	Date and time with fraction	TIMESTAMP(3)
7	DATETIME2(<i>p</i>)	Date and time with fraction	TIMESTAMP(<i>p</i>)
8	DATETIMEOFFSET(<i>p</i>)	Date and time with fraction and time zone	TIMESTAMP(<i>p</i>) WITH TIME ZONE
9	DECIMAL(<i>p,s</i>), DEC(<i>p,s</i>)	Fixed-point number	DECIMAL(<i>p,s</i>), DEC(<i>p,s</i>)
10	DOUBLE PRECISION	Double-precision floating-point number	DOUBLE PRECISION
11	FLOAT(<i>p</i>)	Floating-point number	DOUBLE PRECISION
12	IMAGE	Variable-length binary data, ≤ 2G	BYTEA
13	INT, INTEGER	32-bit integer	INT, INTEGER
14	MONEY	64-bit currency amount	MONEY
15	NCHAR(<i>n</i>)	Fixed-length Unicode UCS-2 string	CHAR(<i>n</i>)
16	NTEXT	Variable-length Unicode UCS-2 data, ≤ 2G ⚠	TEXT
17	NUMERIC(<i>p,s</i>)	Fixed-point number	NUMERIC(<i>p,s</i>)
18	NVARCHAR(<i>n</i>)	Variable-length Unicode UCS-2 string	VARCHAR(<i>n</i>)
19	NVARCHAR(max)	Variable-length Unicode UCS-2 data, ≤ 2G ⚠	TEXT
20	REAL	Single-precision floating-point number	REAL

- Полное мн-во типов данных
<https://www.postgresql.org/docs/9.5/datatype.html>
- Маппинг типов .NET
<https://www.npgsql.org/doc/types/basic.html>

<http://www.sqlines.com/sql-server-to-postgresql#data-types>

Нюансы с типами данных дата время

PostgreSQL type	Precision/Range	.NET Native Type	Precision/Range	Npgsql .NET Provider-Specific Type
timestamp without time zone	1 microsecond, 4713BC-294276AD	DateTime	100 nanoseconds, 1AD-9999AD	NpgsqlDateTime

.NET value	NpgsqlDbType	Action
DateTime	NpgsqlDbType.Timestamp (default)	Send as-is
DateTime(Kind=UTC,Unspecified)	NpgsqlDbType.TimestampTz	Send as-is
DateTime(Kind=Local)	NpgsqlDbType.TimestampTz	Convert to UTC locally before sending
DateTimeOffset	NpgsqlDbType.TimestampTz (default)	Convert to UTC locally before sending

1. различная точность и интервал для типа *datetime*
2. хитрая конверсия *datetime.kind* для типа *timestamp with time zone*
3. **не полная поддержка типа *datetimeoffset* (таймзона не хранится отдельно)**

Чувствительность к регистру поиска по строкам

Workaround 1: use collations for column or database (ef core 5.0 feature)

```
protected override void OnModelCreating(ModelBuilder modelBuilder)
{
    modelBuilder.HasCollation("my_collation", locale: "en-u-ks-primary", provider: "icu", deterministic: false);

    modelBuilder.Entity<Customer>().Property(c => c.Name)
        .UseCollation("my_collation");
}
```

i NOTE

It is not yet possible to use pattern matching operators such as LIKE on columns with a non-deterministic collation.

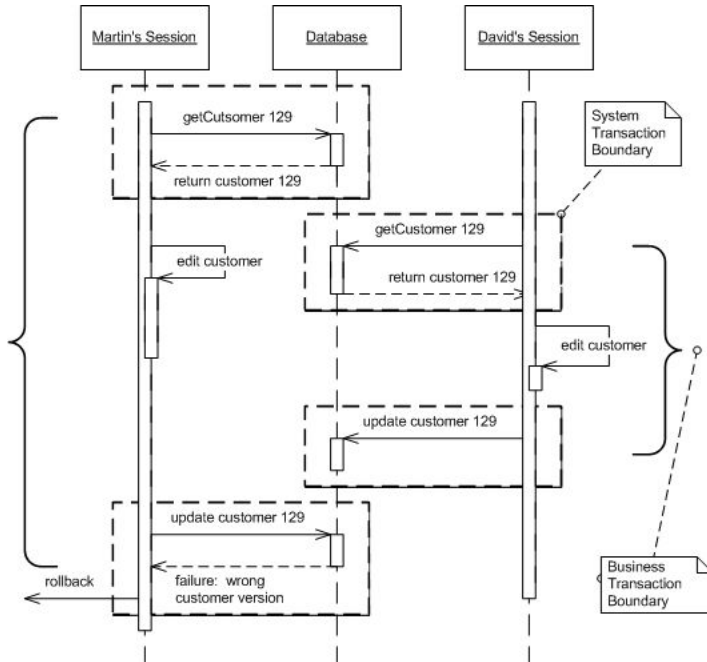
Чувствительность к регистру поиска по строкам

Workaround 2: add normalized column

```
1 CREATE TABLE [dbo].[AspNetRoles] (  
2 [Id] NVARCHAR (128) NOT NULL,  
3 [Name] NVARCHAR (256) NOT NULL,  
4 CONSTRAINT [PK_dbo.AspNetRoles] PRIMARY KEY CLUSTERED ([Id] ASC)  
5 );  
6 GO  
7  
8 CREATE UNIQUE NONCLUSTERED INDEX [RoleNameIndex]  
9 ON [dbo].[AspNetRoles]([Name] ASC);  
10 GO  
11  
12
```

```
1 CREATE TABLE [dbo].[AspNetRoles] (  
2 [Id] NVARCHAR (450) NOT NULL,  
3 [Name] NVARCHAR (256) NULL,  
4 [NormalizedName] NVARCHAR (256) NULL,  
5 [ConcurrencyStamp] NVARCHAR (MAX) NULL,  
6 CONSTRAINT [PK_AspNetRoles] PRIMARY KEY CLUSTERED ([Id] ASC) WITH (FILLFACTOR=100)  
7 );  
8 GO  
9  
10 CREATE UNIQUE NONCLUSTERED INDEX [RoleNameIndex]  
11 ON [dbo].[AspNetRoles]([Name] ASC) WHERE ([Name] IS NOT NULL) WITH (FILLFACTOR=100)  
12 GO  
13  
14 CREATE UNIQUE NONCLUSTERED INDEX [RoleNormalizedNameIndex]  
15 ON [dbo].[AspNetRoles]([NormalizedName] ASC) WHERE ([NormalizedName] IS NOT NULL) WITH (FILLFACTOR=100)  
16 GO  
17  
18
```


Нюансы с concurrency token - rowversion



Workaround with transaction id

<https://www.npgsql.org/efcore/modeling/concurrency.html>

```
class Blog
{
    ...
    public uint xmin { get; set; }
}
```

```
protected override void OnModelCreating(ModelBuilder modelBuilder)
=> modelBuilder.Entity<Blog>()
    .UseXminAsConcurrencyToken();
```

<https://martinfowler.com/eaaCatalog/optimisticOfflineLock.html>

Нюансы с concurrency control - Isolation Levels

There is no Isolation Level - READ UNCOMMITTED (reason MVCC)

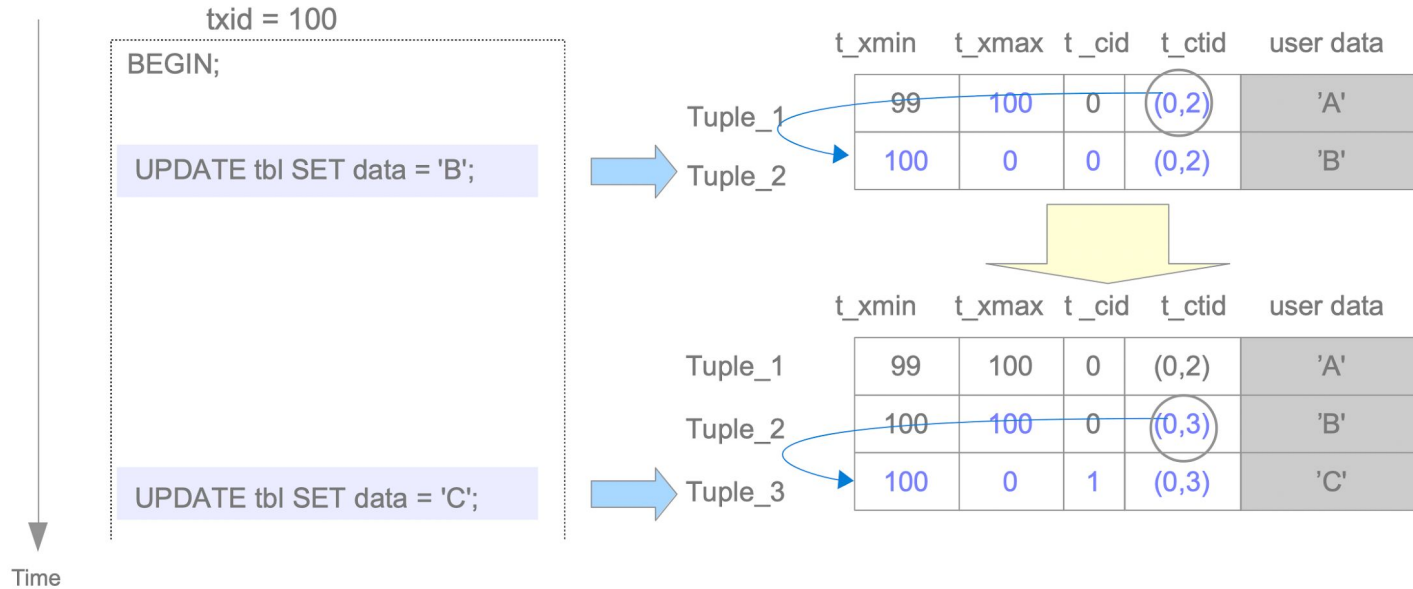
Transaction Isolation Level in PostgreSQL


PostgreSQL-implemented transaction isolation levels are described in the following table:

Isolation Level	Dirty Reads	Non-repeatable Read	Phantom Read	Serialization Anomaly
READ COMMITTED	Not possible	Possible	Possible	Possible
REPEATABLE READ* ¹	Not possible	Not possible	Not possible in PG; See Section 5.7.2. (Possible in ANSI SQL)	Possible
SERIALIZABLE	Not possible	Not possible	Not possible	Not possible

*1 : In version 9.0 and earlier, this level had been used as 'SERIALIZABLE' because it does not allow the three anomalies defined in the ANSI SQL-92 standard. However, with the implementation of SSI in version 9.1, this level has changed to 'REPEATABLE READ' and a true SERIALIZABLE level was introduced.

Нюансы с concurrency control - Update command





Разница в синтаксисе SQL запросов

PostgreSQL поддерживает **160** из **179** обязательных возможностей стандарта **SQL:2016 (ISO/IEC 9075)**.

Разница в синтаксисе

There is no TOP keywords, but fetch offset syntax works well

```
SQLQuery1.sql - tst-...nislav.flusov (309)*  X
SELECT top 10 *
FROM [dbo].[Payments]

SELECT *
FROM [dbo].[Payments]
order by CreatedAt
offset 128 rows
fetch next 10 rows only
```

```
select *
from public."Payments"
limit 10
offset 128;

-- use fetch syntax standard 2008
select *
from public."Payments"
order by "CreatedAt"
offset 128
fetch next 10 rows only;
```

<https://www.postgresql.org/docs/9.5/sql-select.html>

Разница в синтаксисе

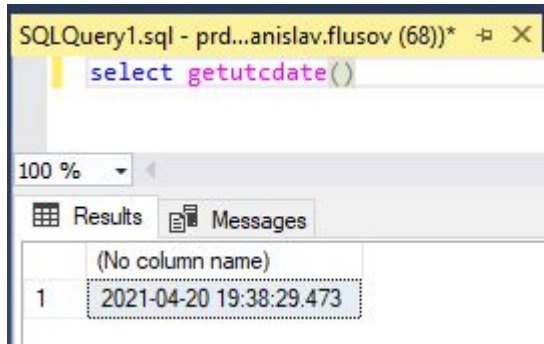
There are few functions for generating UUID

```
2 -- This involves the MAC address of the computer and a time stamp.
3 select uuid_generate_v1();
4
5 -- Generates a version 1 UUID,
6 -- but uses a random multicast MAC address instead of the real MAC address of the computer.
7 select uuid_generate_v1mc();
8
9 -- Generates a version 4 UUID, which is derived entirely from random numbers.
10 select uuid_generate_v4();
11
12 -- Create own function
13 create function newid() returns uuid
14 as
15 'select uuid_generate_v4()'
16 language sql
17
```

<https://www.postgresql.org/docs/current/uuid-osp.html>

Разница в синтаксисе - working with dates

There is no function **getutcdate()**, but we can create own function if its required



```
SQLQuery1.sql - prd...anislav.flusov (68)*
```

```
select getutcdate()
```

	(No column name)
1	2021-04-20 19:38:29.473

```
1  
2 -- use sql expression  
3 select now() at time zone 'UTC'  
4  
5 -- or create own function  
6 create function getutcdate() returns timestamp  
7 as  
8 'select now() at time zone ''UTC'''  
9 language sql  
10  
11 select getutcdate();
```

<https://www.postgresql.org/docs/current/sql-createfunction.html>

Разница в синтаксисе - working with dates

MS SQL Server:

```
1 | --Add 2 day to the current date  
2 | SELECT DATEADD(day, 2, GETDATE());
```

PostgreSQL:

```
1 | --Add 2 day to the current date  
2 | SELECT CURRENT_DATE + INTERVAL '2 day';
```

MS SQL:

```
1 | DATEPART( datepart , date )
```

PostgreSQL:

```
1 | date_part( text , timestamp )  
2 | date_part( text , interval )
```

<https://severalnines.com/database-blog/migrating-mssql-postgresql-what-you-should-know>

Разница в синтаксисе - working with strings



MS SQL Server:

```
1 | SELECT FirstName + LastName FROM employee;
```

PostgreSQL:

```
1 | SELECT FirstName || LastName FROM employee;
```

MS SQL Server:

```
1 | SELECT CHARINDEX('our', 'resource');
```

PostgreSQL:

```
1 | SELECT POSITION('our' in 'resource');
```

<https://severalnines.com/database-blog/migrating-mssql-postgresql-what-you-should-know>

Разница в синтаксисе - table hints are not supported

```
<table_hint> ::=  
{ NOEXPAND [ , INDEX ( <index_value> [ ,...n ] ) | INDEX = ( <index_value> ) ]  
| INDEX ( <index_value> [ ,...n ] ) | INDEX = ( <index_value> )  
| FORCESEEK [ ( <index_value> ( <index_column_name> [ ,... ] ) ) ]  
| FORCESCAN  
| HOLDLOCK  
| NOLOCK  
| NOWAIT  
| PAGLOCK  
| READCOMMITTED  
| READCOMMITTEDLOCK  
| READPAST  
| READUNCOMMITTED  
| REPEATABLEREAD  
| ROWLOCK  
| SERIALIZABLE  
| SNAPSHOT  
| SPATIAL_WINDOW_MAX_CELLS = <integer_value>  
| TABLOCK  
| TABLOCKX  
| UPDLOCK  
| XLOCK  
}
```

Reasons

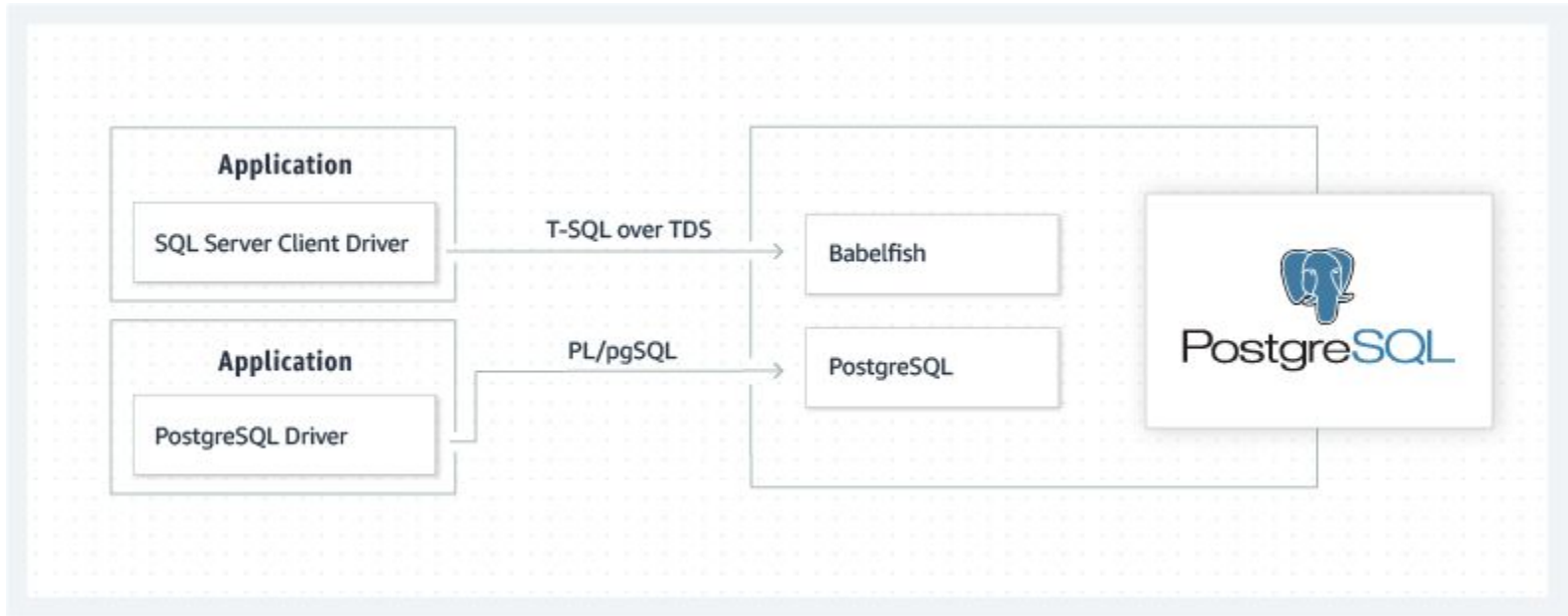
- MVCC - Multi-version Concurrency Control
- The planner is based on pure cost-based optimization



Революционный путь миграции для тех кто использует ADO.NET или Dapper

Coming soon 2021

Babelfish from AWS



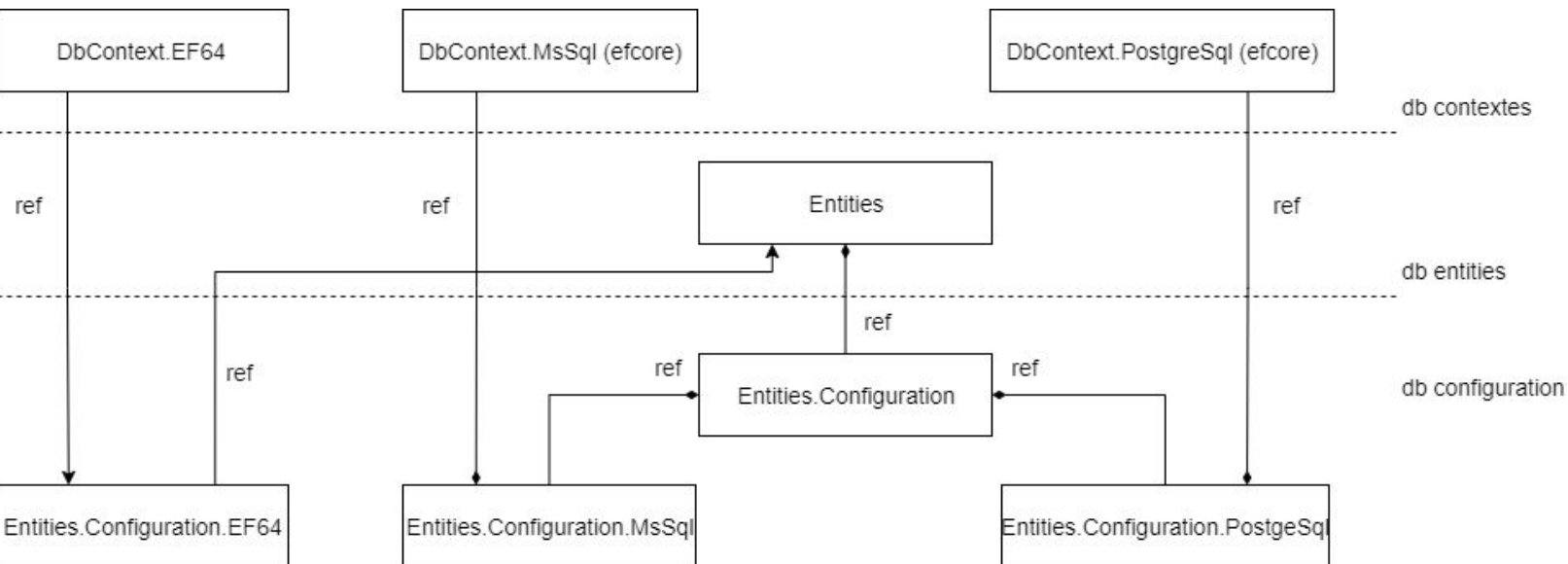


Миграция на PostgreSQL средствами ef core

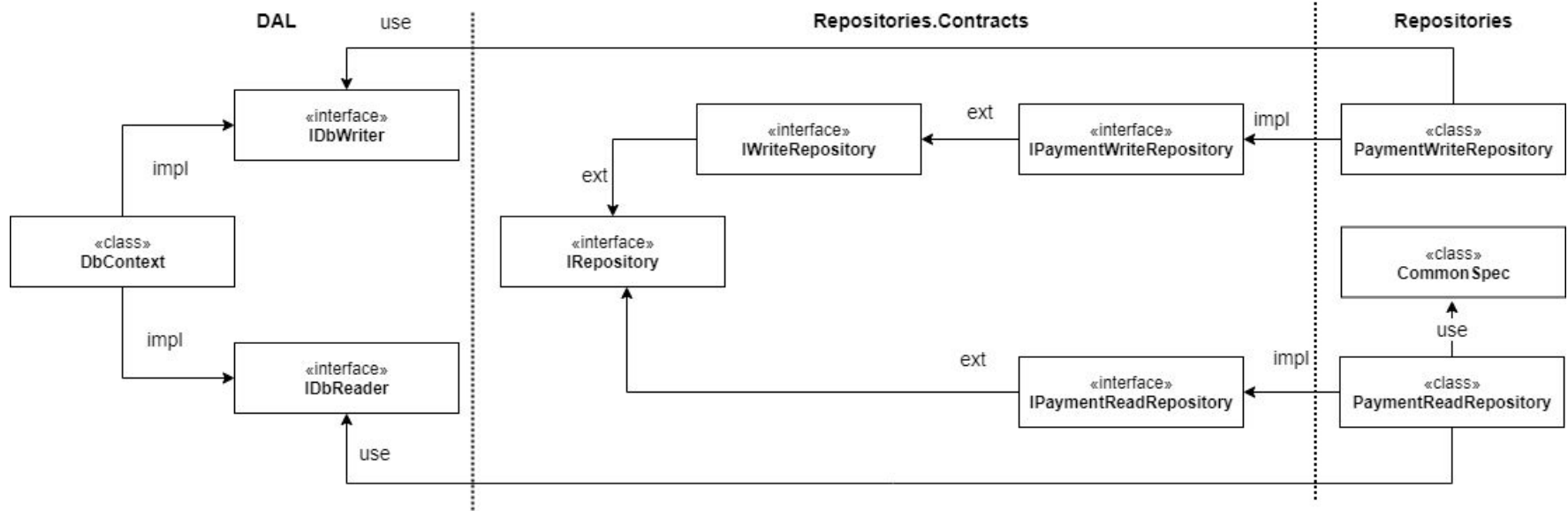
ef6.4 (ms sql) -> ef core 2.2 (ms sql) -> ef core 2.2 (pg sql)

- tables qty ~ 76
- qps (queries per second) ~ 248
- tps ~ 218

Разделение сущностей и конфигурация

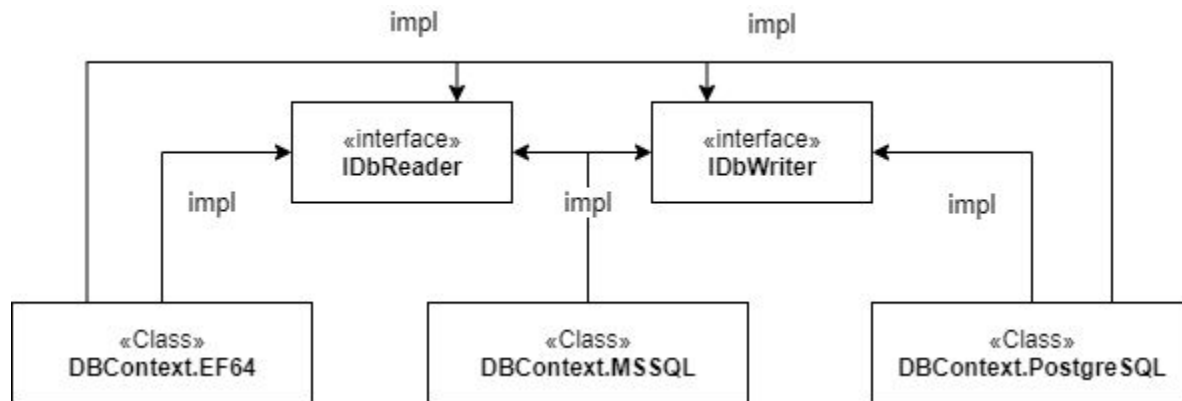


Введение слоя репозитория и спецификаций



Слой DAL

Абстракция от конкретной реализации DbContext, DbSet<T>, ToListAsync<T> и т.д.

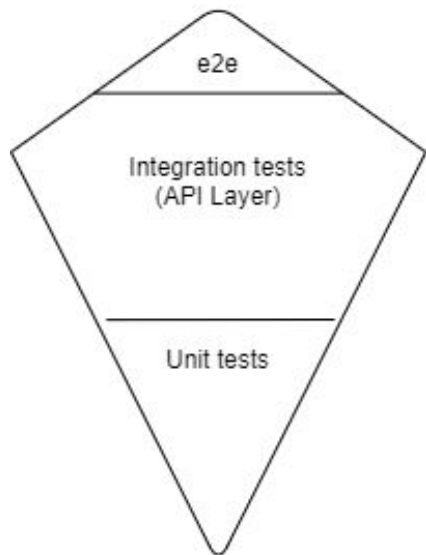


Композитные индексы и инклюды

```
{  
    return await dbReader.Read<Contract>()  
        .ByMemberId(clientId)  
        .OfType(ActivityType.FuelSale)  
        .NotDeleted()  
        .ActualAt(moment) // IQueryable<Contract>  
        .Asyncable() // IAsyncQuery<Contract>  
        .ToListAsync(cancellationToken); // Task<List<...>>  
}
```

```
builder.HasIndex(p:Contract => p.ExternalId).IsUnique();  
builder.HasIndex(p:Contract => new  
{  
    p.ContractId, p.ActivityType, p.IsClosed, p.ContractorIsVendor  
});  
builder.HasIndex(p:Contract => new  
{  
    p.ClientId, p.ActivityType, p.IsClosed, p.ContractorIsVendor  
});  
}
```

Интеграционные тесты и поиск багов



В своих тестах мы создаем реальные базы MS SQL или PostgreSQL с нуля

- проверяя совместимость миграций (последовательность)
- корректность работы приложений, используя разные контексты

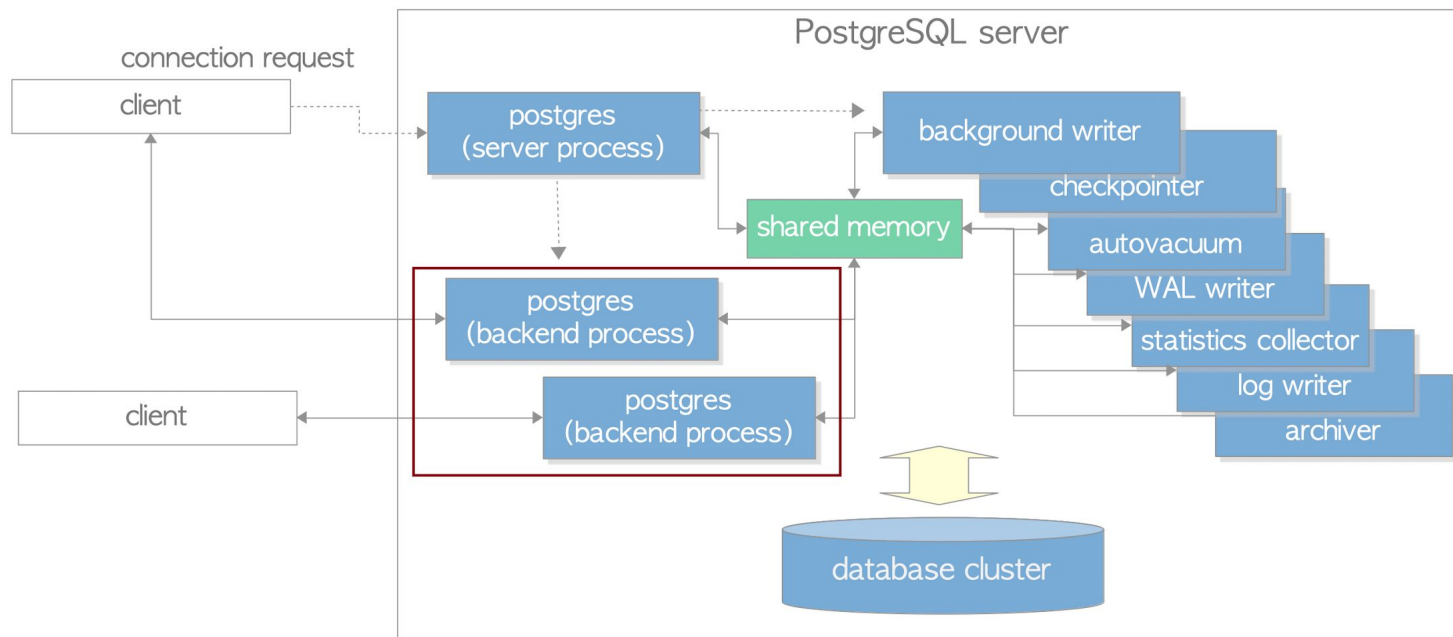
Почему реальные базы:

- Sqlite - не поддерживает множественные миграции (с изменением схемы)
- InMemoryDb - не поддерживает констрейнты: уникальные индексы, внешние ключи



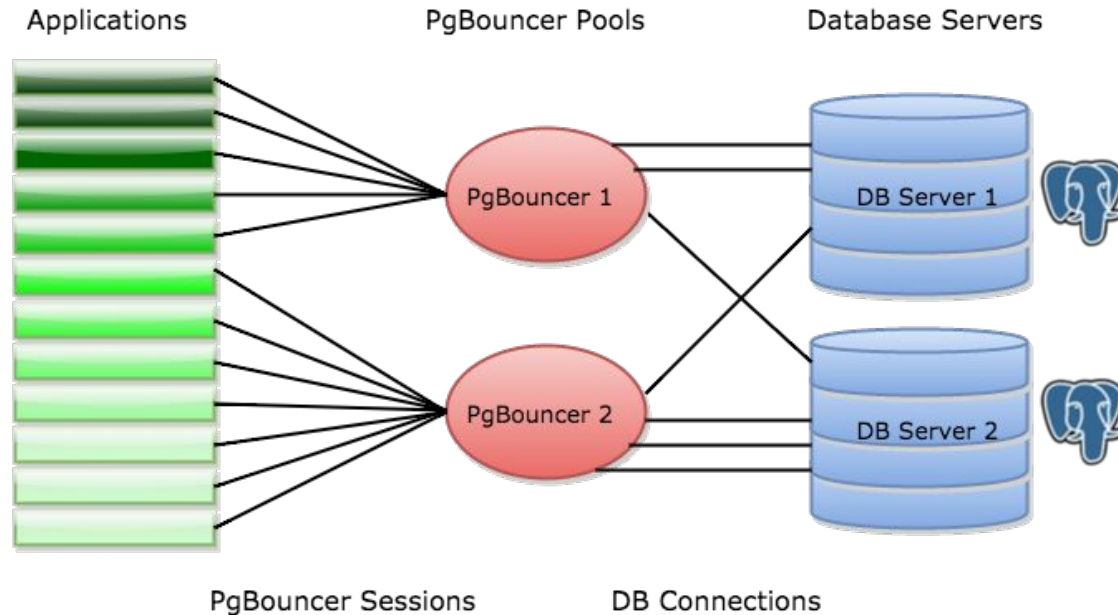
Нагрузочное тестирование и его результаты

Архитектура процессов и памяти

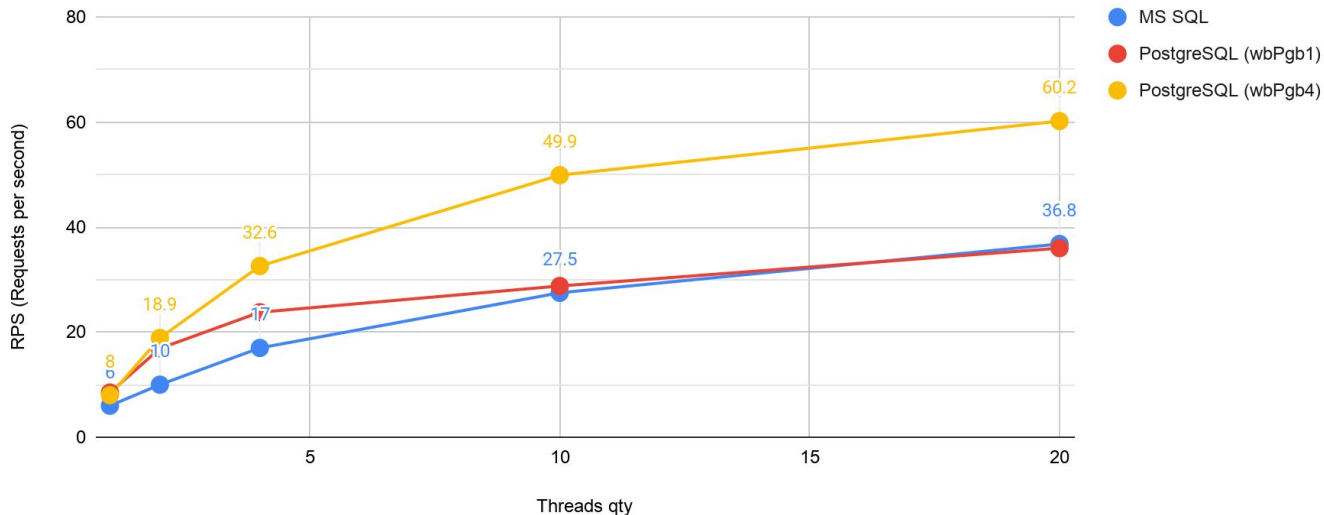


Нагрузочное тестирование и его результаты

PgBouncer - lightweight connection pooler




Нагрузочное тестирование и его результаты



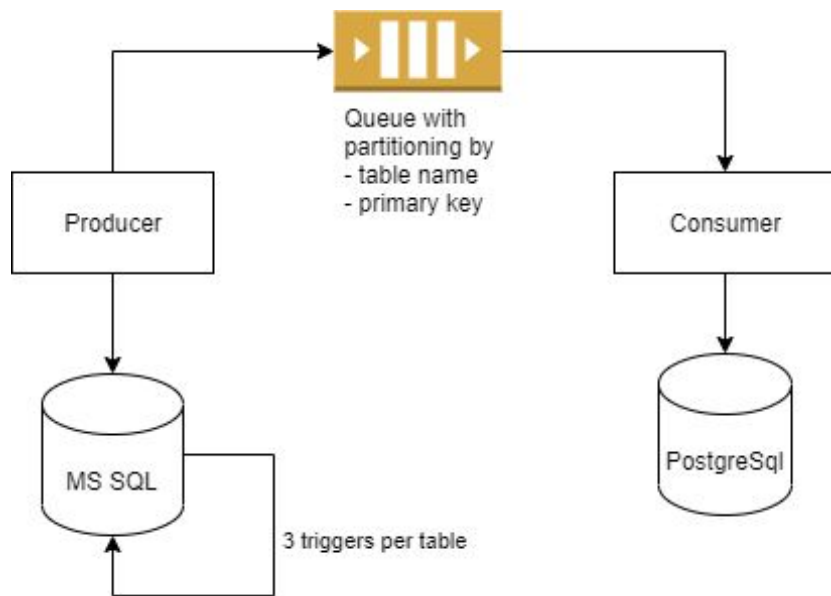
1. MS SQL - 16 vcpu, 44 Gb RAM
2. wbPgb1 - 4 vcpu, 8 Gb RAM, work mem = 32 Mb, shared buffer = 4096
3. wbPgb4 - 8 vcpu, 16 Gb RAM, work mem = 64 Mb, shared buffer = 8192

Db size ~ 20 Gb



Синхронизация данных и проверка целостности

Code first и синхронизация данных

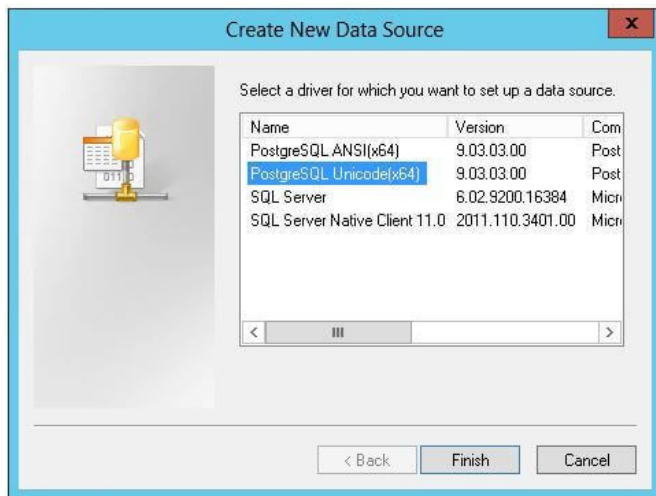


Проблемы с которыми мы столкнулись:

- нет решения, которое это сделает без простоя (~ 4 hour)
- маппинги типов, регистр колонок и других идентификаторов придется править вручную
- интеграционные тесты будут не репрезентативны по отношению к схеме и данным с прода

Проверка целостности данных после синхронизации

Add Linked Server to PostgreSQL



Простая агрегация всех значений

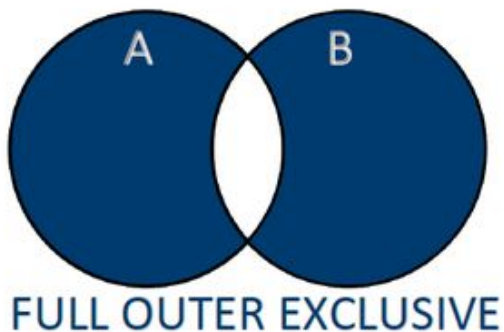
```
select
    sum([sum]) as sumPayments
    ,count(*) as qty
FROM [dbo].[Payments]

SELECT *
FROM OPENQUERY([prd-flpg], '
select
    sum("Sum")::money as sumPayments
    ,count(*) as qty
FROM "Payments"
') PG
```

<https://habr.com/ru/post/428443/>

Проверка целостности данных после синхронизации

Поиск расхождения в конкретных записях

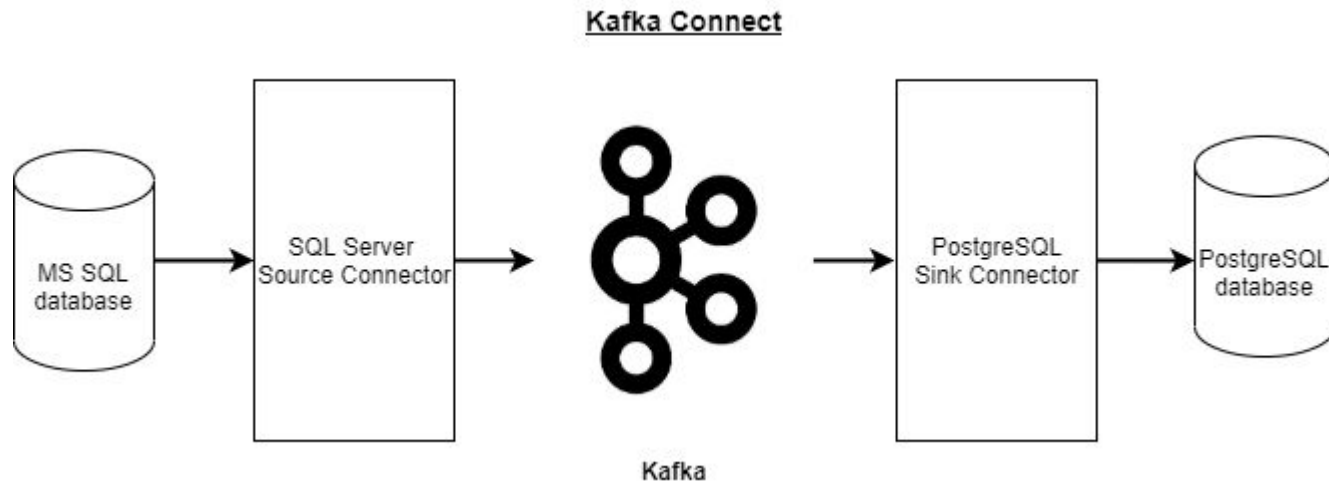


```
drop table if exists #Ids
select Id
into #Ids
from openquery([prd-flpg],
'SELECT "Id"
FROM "Payments"')
select count(*)
from Payments as ms
full join #Ids as pg
on pg.id = ms.id
where 1 != 1
or ms.Id is null
or pg.id is null
```

<https://habr.com/ru/post/428443/>

Синхронизация с использованием CDC + Kafka

Change Data Capture (CDC) is provided by SQL Standard (2016 SP1) or SQL Enterprise edition



<https://docs.confluent.io/debezium-connect-sqlserver-source/current/>

<https://docs.confluent.io/cloud/current/connectors/cc-postgresql-sink.html>

Благодарности



1. Спасибо командам **DevOps + DBA (Антону Смолькову, Алексею Казгунову)** за настроенный PostgreSQL, инструменты его мониторинга и бекапов
2. Спасибо моей команде за проделанную работу и в отдельности **Юрию Баранихину** за идеи абстракций *IDbReader* и *IDbWriter*
3. Спасибо команде нагрузочного тестирования (**Ивану Федупину**) за недельную перегрузку наших приложений и серверов



Спасибо за внимание

- telegram: <https://t.me/sflusov>
- vk: https://vk.com/stanislav_flusov