СИГМА
ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

7 ошибок, которые совершают разработчики при использовании СУБД PostgreSQL

Иван Чувашов

Ведущий разработчик-эксперт,
архитектор БД

1. VACUUM/AUTOVACUUM

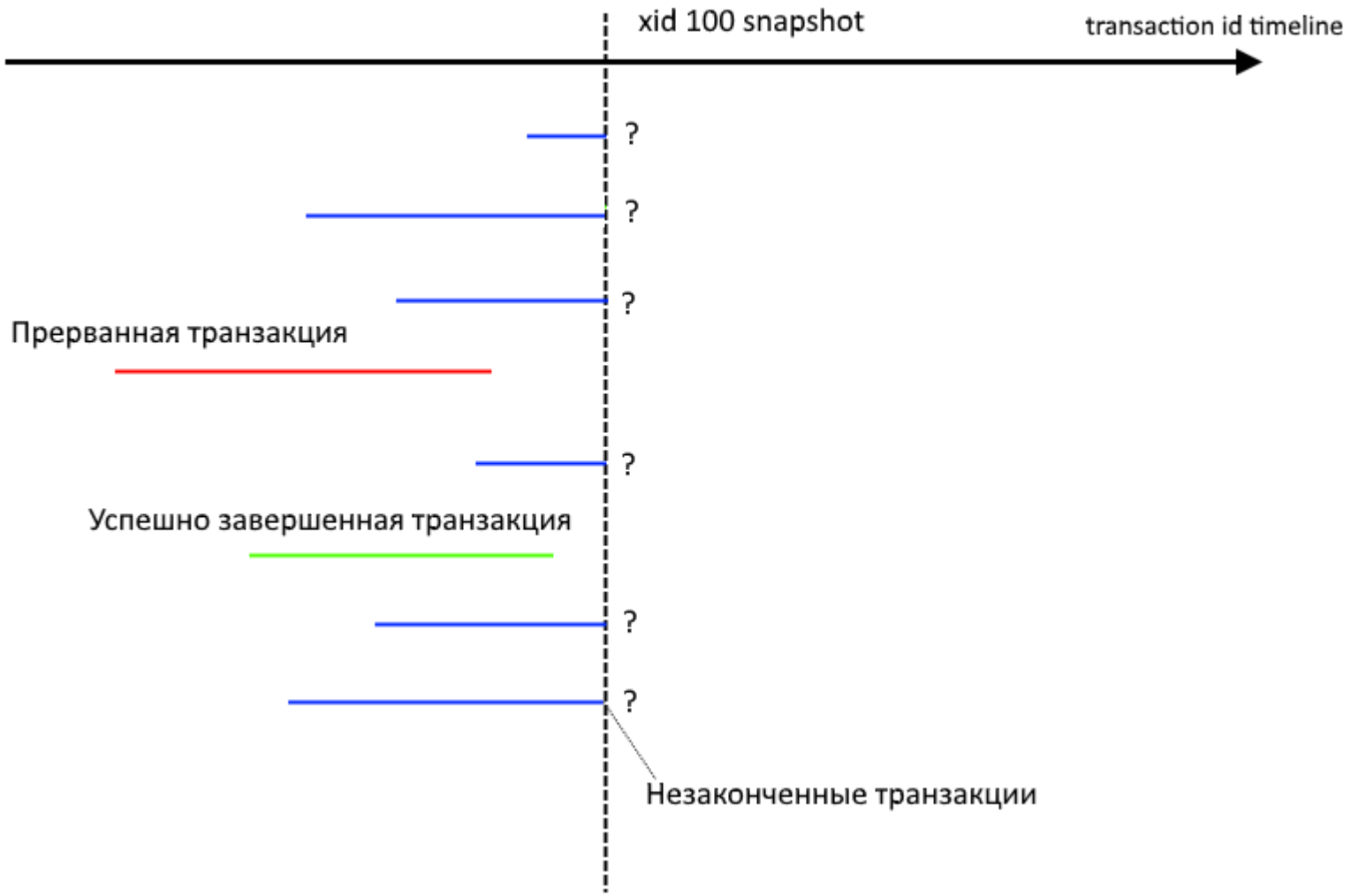
MCVV

MCVV (*MultiVersion Concurrency Control)* - механизм обеспечения **параллельного доступа к данным**, заключающийся в предоставлении каждому пользователю так называемого «снимка» БД, обладающего тем свойством, что вносимые пользователем изменения в БД невидимы другим пользователям до момента фиксации транзакции. Этот способ управления позволяет добиться того, что **пишущие транзакции не блокируют читающих, и читающие транзакции не блокируют пишущих**.

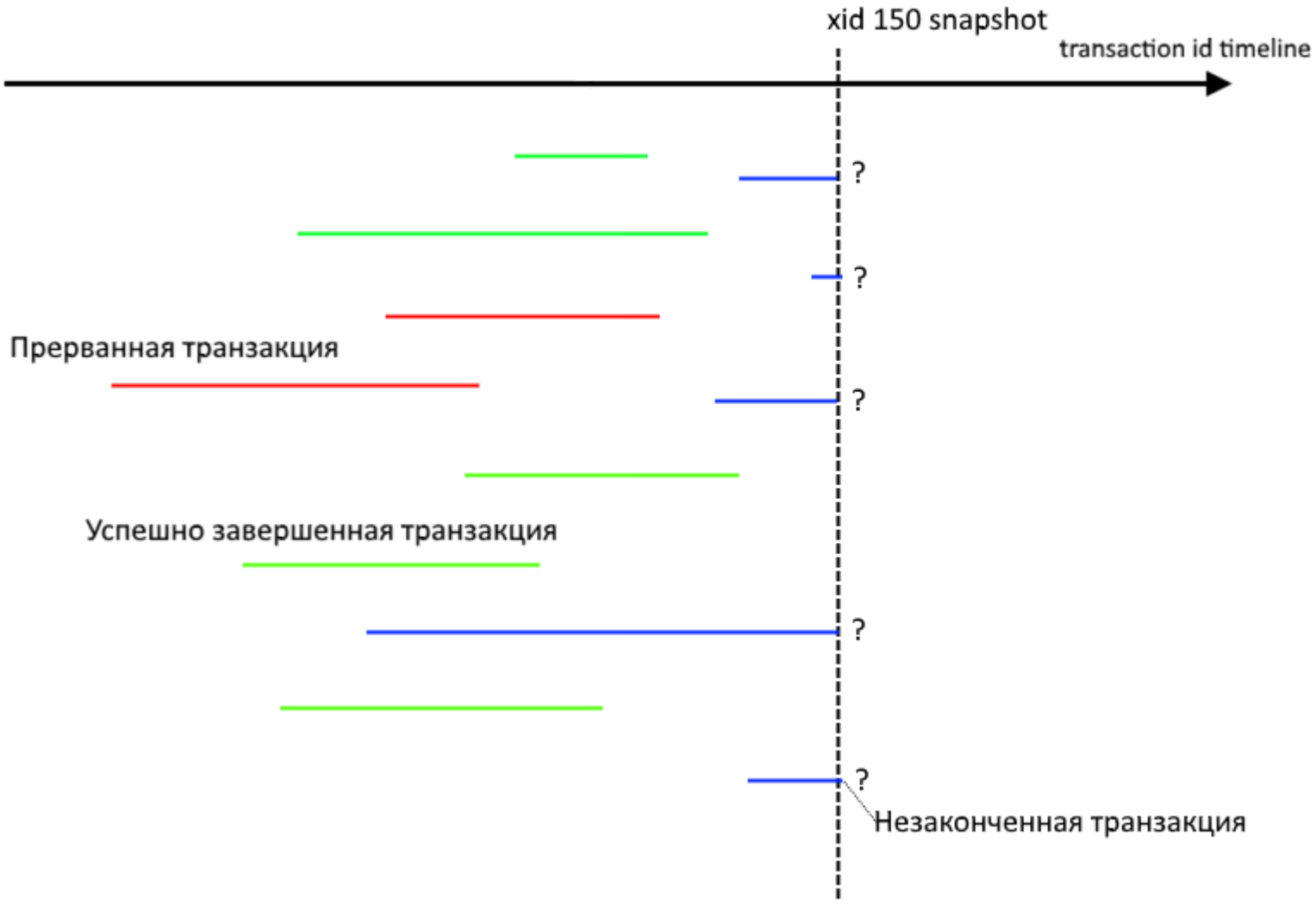
MVCC


xid 50 snapshot
transaction id timeline
?
?
?
?
Незаконченные транзакции

MVCC


xid 100 snapshot
transaction id timeline
?
?
?
Прерванная транзакция
?
Успешно завершенная транзакция
?
?
Незаконченные транзакции

MVCC


xid 150 snapshot
transaction id timeline
?
?
Прерванная транзакция
?
Успешно завершенная транзакция
?
?
Незаконченная транзакция

MVCC

xmin: 111
xmax:

INSERT строки транзакцией №111

xmin: 111
xmax: 222

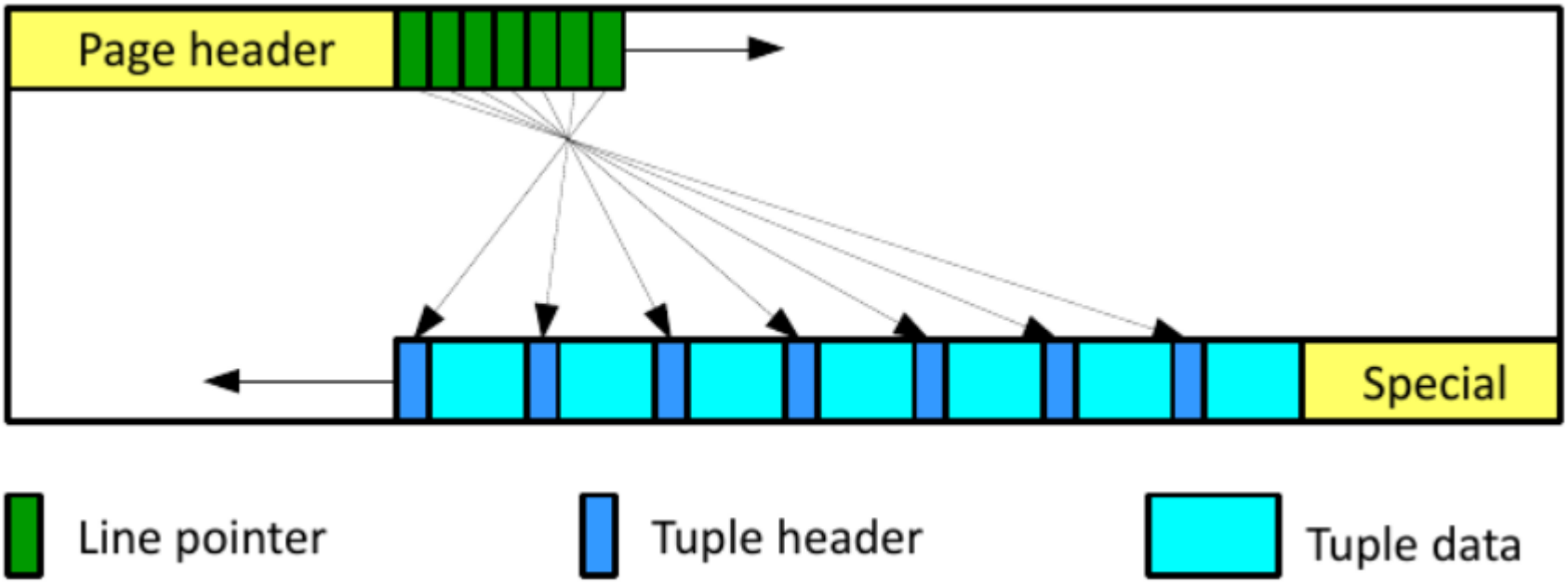
xmin: 222
xmax:

UPDATE строки транзакцией №222
UPDATE = DELETE + INSERT

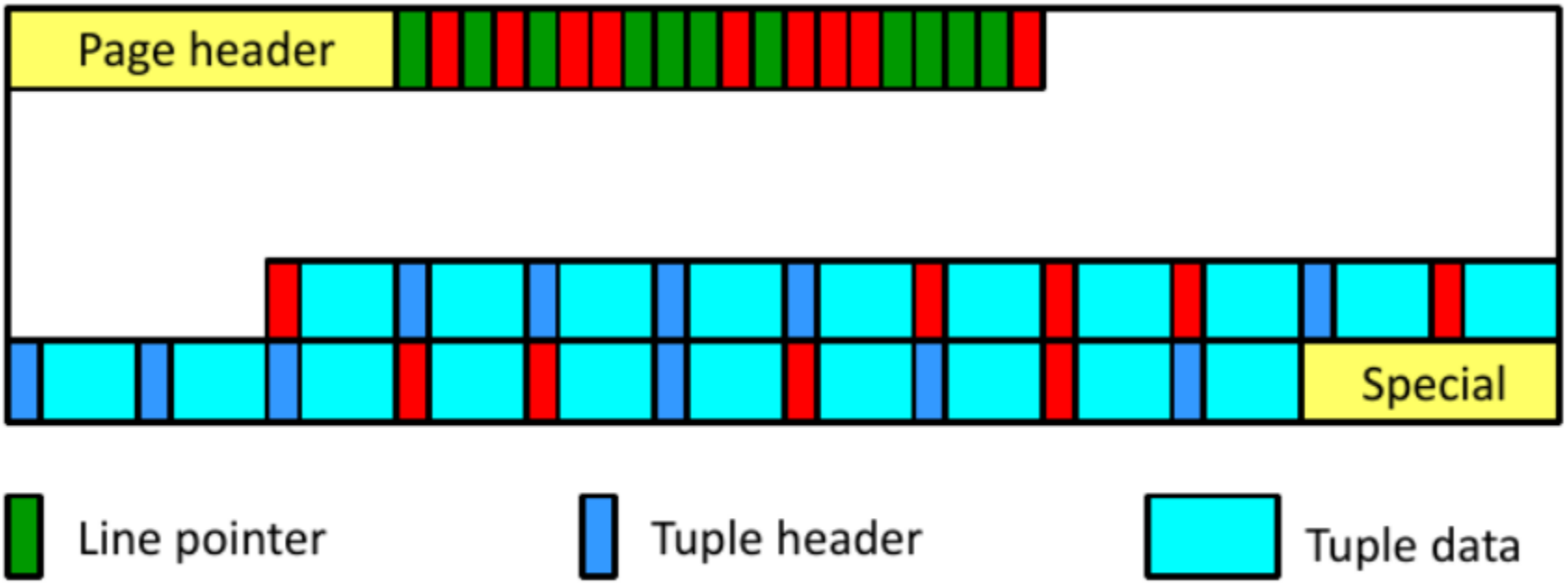
xmin: 222
xmax: 333

DELETE строки транзакцией №333

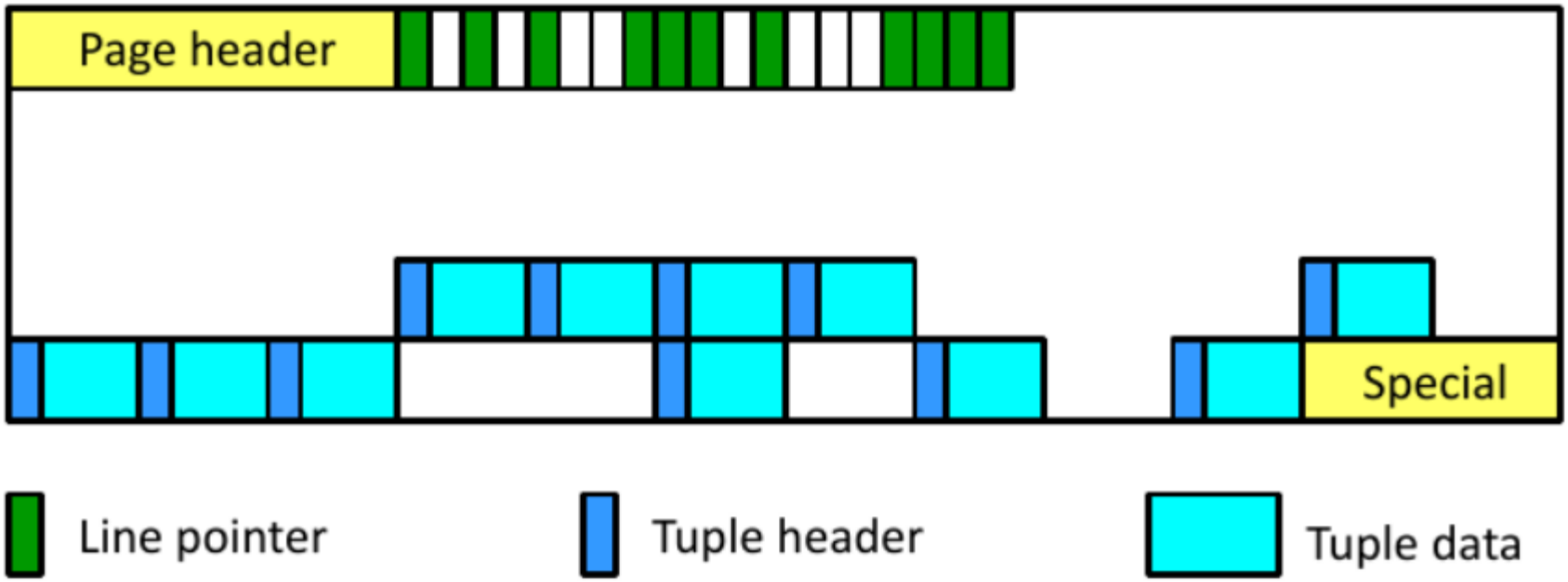
Как обстоят дела на уровне страницы



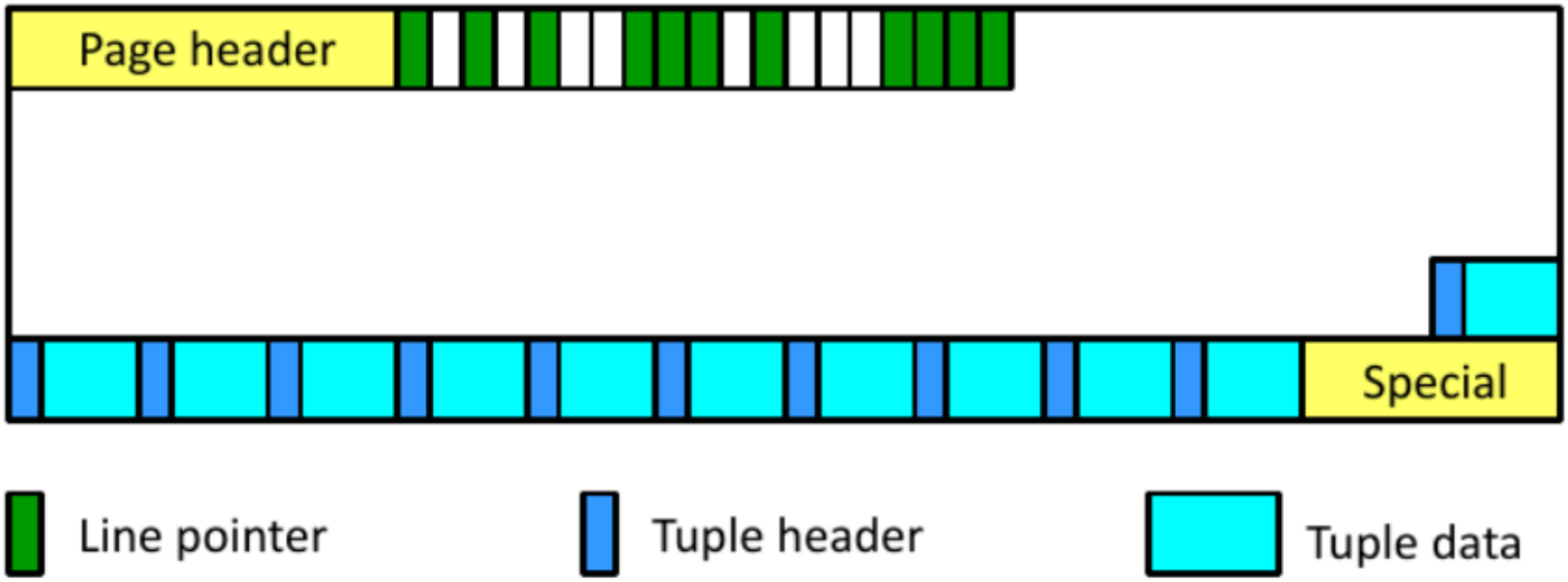
Как обстоят дела на уровне страницы



Как обстоят дела на уровне страницы



Как обстоят дела на уровне страницы



VACUUM/AUTOVACUUM

АЛЕКСЕЙ ЛЕСОВСКИЙ - Давайте отключим vacuum?!
https://pgconf.ru/2018/108354

2. LINUX + PostgreSQL

LINUX + PostgreSQL

Windows + PostgreSQL = головная боль

Файловая система

Файл **/etc/fstab**

barrier=0. Барьер нельзя отключать с плохими SSD и без защиты по питанию BBU.

Работа с дисками

Файл **/etc/sysctl.conf**

vm.dirty_background_bytes = 12058624

vm.dirty_bytes = 24117248

vm.swappiness = 1

Желательно:

1. Отдельный рейд для WAL
2. Отдельный рейд для хранения данных БД

Работа с памятью

vm.hugepages_treat_as_movable = 0

# nr_hugepages выставляем на 3-5% выше чем shared_buffers значение в страницах памяти
vm.nr_hugepages= 16384

# nr_overcommit_hugepages делаем 5-10% от nr_hugepages (значение так же в страницах памяти)
vm.nr_overcommit_hugepages = 820

# Минимальное количество килобайт памяти, которое должно держаться свободным
vm.min_free_kbytes = 524288

postgresql.conf

wal_buffers = 16Mb

checkpoint_timeout = 60min
checkpoint_completion_target = 0.9
checkpoint_warning = 10min

effective_io_concurrency – ставить больше!
random_page_cost = seq_page_cost=1 ! **Для SSD**

LINUX + PostgreSQL

Илья Космодемьянский - Как выжать максимум производительности из операционной системы и hardware
https://pgday.ru/files/papers/36/pgday.2015.kosmodemyansky.unix_and_hardware.pdf

3. Архитектура

Этапы зрелости проекта со стороны БД

1. Проект на этапе развития.

   Размер БД не большой

Программист: «У нас «база» тормозит!»
DBA: «Сейчас тут индекс сделаем»
DBA: «И тут индекс»
DBA: «А тут их нужно ДВА!»

Бесчисленно создаем индексы.

Этапы зрелости проекта со стороны БД

2. Проект сдали заказчику.
Размер БД быстро растет.

Программист: «У нас «база» тормозит!»
DBA: «?! Индексы уже не помогают»
DBA: «А давайте поменяем конфигурацию PostgreSQL»

Бездумно меняем конфигурацию сервера PostgreSQL

Этапы зрелости проекта со стороны БД

3. Проект год работает. Постоянные проблемы с производительность БД

Программист: «У нас «база» тормозит!»
DBA: «Индексы были. Postgresql.conf - правил»
DBA: «Нужно поменять архитектуру таблиц БД!»
DBA: «Это все PostgreSQL виноват! Вот если бы взяли Oracle/MSSQL …»

Запоздалые решения. Ухудшилось мнение о вас и о выбранных вами технологиях

Этапы зрелости проекта со стороны БД

4. Проект работает несколько лет. Заказчик «верит», что проблема не в проекте, а в СУБД. PostgreSQL – «плохая» СУБД

Программист: «У нас «база» тормозит!»
DBA: «Индексы были. Postgresql.conf - правил»
DBA: «Я тут читал, что PostgreSQL можно шардировать!!!»

Шардирование данных. Разделение БД на две: архивную и оперативную.

Как быть?

Все 4 пути нужно сразу закладывать в архитектуру:

1. Индексы.
2. Тюнинг настроек Postgresql.conf. Со сбором информации о поведении сервера БД.
3. Денормализация данных.
4. Разработанная архитектура **должна** позволить безболезненный переезд на две БД: оперативную и архивную.

Сбор статистики и мониторинг производительности

СИГМА
ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

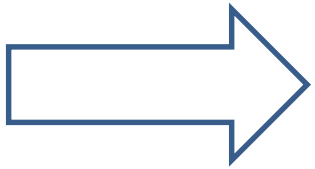
Пример

Таблица Table1 содержит 80 млн записей (19 Гб)
90 процентов запросов ищется по ключам полей param и param2.

19 индексов суммарно занимают место 28 Гб!!!

Решение

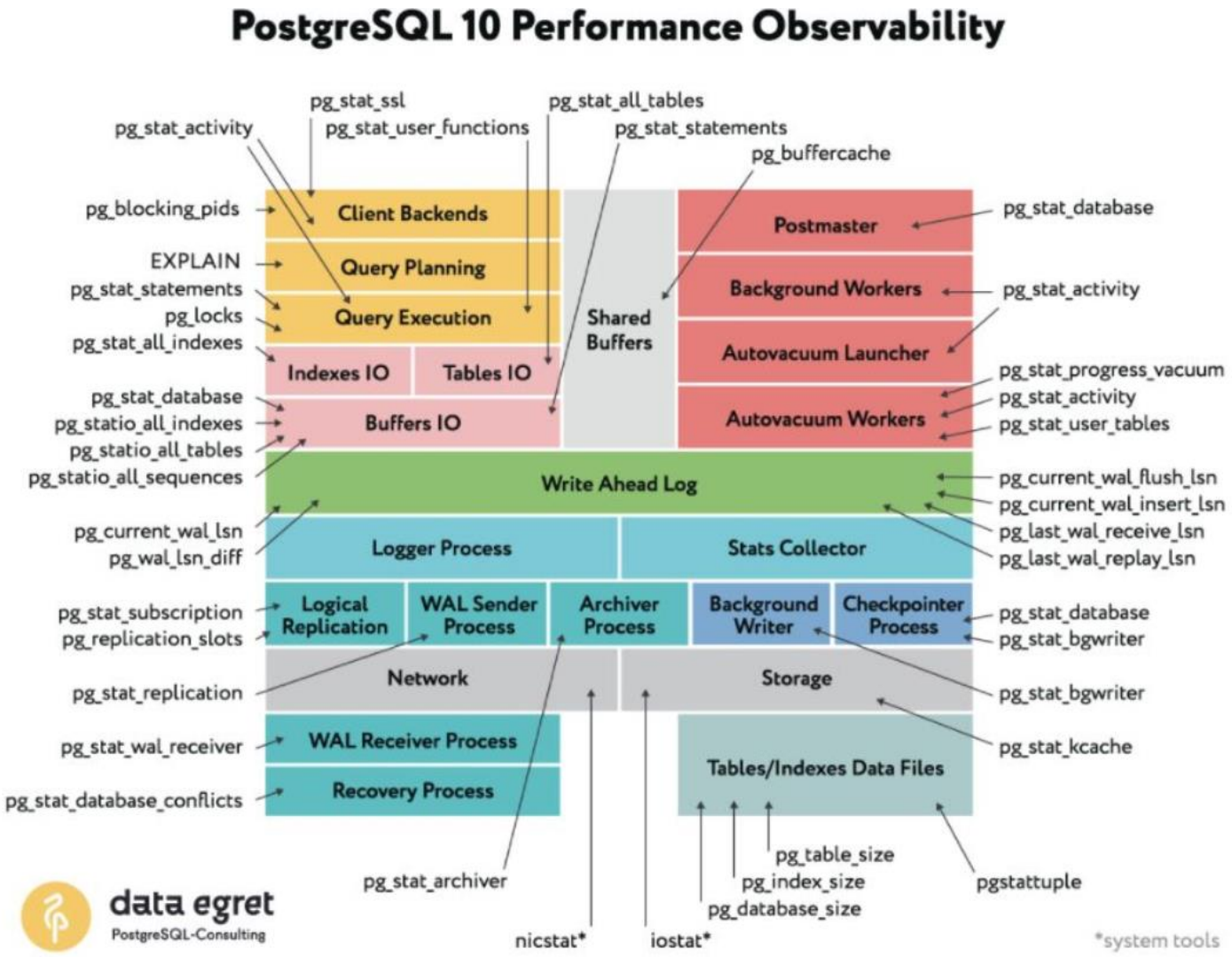
Field	Type
id	Int8
status	int8
param	jsonb
Param2	jsonb



Field0	Type
id	Int8
status	int8
Field1	string
field2	numeric
field3	int8
field4	string
param	jsonb
param2	jsonb

4. Статистика

Статистика


PostgreSQL 10 Performance Observability
pg_stat_ssl
pg_stat_activity
pg_stat_user_functions
pg_stat_all_tables
pg_stat_statements
pg_buffercache
pg_blocking_pids
Client Backends
Postmaster
pg_stat_database
EXPLAIN
Query Planning
pg_stat_statements
pg_locks
Query Execution
Shared Buffers
Background Workers
pg_stat_activity
pg_stat_all_indexes
Autovacuum Launcher
Indexes IO
Tables IO
pg_stat_progress_vacuum
pg_stat_database
pg_statio_all_indexes
Buffers IO
Autovacuum Workers
pg_stat_activity
pg_statio_all_tables
pg_stat_user_tables
pg_statio_all_sequences
Write Ahead Log
pg_current_wal_flush_lsn
pg_current_wal_insert_lsn
pg_current_wal_lsn
pg_wal_lsn_diff
Logger Process
Stats Collector
pg_last_wal_receive_lsn
pg_last_wal_replay_lsn
pg_stat_subscription
Logical Replication
WAL Sender Process
Archiver Process
Background Writer
Checkpointer Process
pg_stat_database
pg_replication_slots
pg_stat_bgwriter
Network
Storage
pg_stat_replication
pg_stat_bgwriter
pg_stat_wal_receiver
WAL Receiver Process
pg_stat_kcache
Tables/Indexes Data Files
pg_stat_database_conflicts
Recovery Process
pg_table_size
pg_stat_archiver
pg_index_size
pgstattuple
pg_database_size
nicstat*
iostat*
*system tools
data egret
PostgreSQL-Consulting

Статистика

Вы всегда должны наблюдать за вашим сервером БД

1. pg_stat_statements -
   Модуль pg_stat_statements предоставляет возможность отслеживать статистику выполнения сервером всех операторов SQL.
2. pg_log – журналы деятельности PostgreSQL.
3. Zabbix (диск, память, процессор)

Пример

pg_stat_statements

userid	dbid	queryid		calls	total_time	min_time	max_time	mean_time	stddev_time	rows	shared_blks_hit	shared_b
41231	16384	3997231863	SELECT "id", "userId", "tradePairId", "type", "amount", "price", "remain'	296985	56239.87	0.006	133.73	0.189369382	0.794850753	176827	10830119	598
41231	16384	1487737920	START TRANSACTION;	177186	80.71	0	0.045	0.000455499	0.000648551	0	0	0
41231	16384	895675780	SET SESSION TRANSACTION ISOLATION LEVEL SERIALIZABLE;	177186	1047.34	0.001	0.64	0.005910986	0.004892961	0	0	0
41231	16384	1364516930	COMMIT;	177181	75.52	0	0.073	0.000426248	0.00061551	0	0	0
10	16384	161687626	SET standard_conforming_strings=on;SET client_min_messages TO wa	149475	17410.22	0	3.206	0.116475819	0.181486819	0	548075	0
41231	16384	3514462075	SELECT "id", "userId", "tradePairId", "type", "amount", "price", "remain'	90843	20665.53	0.021	96.108	0.227486201	0.411419954	48	3090568	54
41231	16384	3513584419	SELECT "id", "userId", "tradePairId", "type", "amount", "price", "remain'	86155	22348.20	0.011	175.493	0.259395253	0.681636704	68	3184816	75
10	16384	1467684219	SELECT "id", "userId", "currencyId", "addressId", "address", "hash", "am	72049	7662.36	0.015	28.185	0.106349346	0.390287967	0	324252	6
10	16384	3704305974	SET standard_conforming_strings=on;SET client_min_messages TO wa	49825	9082.30	0.046	1.292	0.182283914	0.099521192	0	548075	0
41231	16384	161687626	SET standard_conforming_strings=on;SET client_min_messages TO wa	27654	2862.52	0	24.861	0.103511789	0.23427353	0	129052	0
10	16384	2311079100	UPDATE "currencies" SET "height"=?,"updatedAt"=? WHERE "id" = ?	27136	1133.18	0.013	14.192	0.04175936	0.113524799	27136	198890	2
10	16384	982876632	SELECT "id", "userId", "currencyId", "addressId", "address", "token", "st	16417	755.39	0.008	3.493	0.046012426	0.041266934	0	57069	1
41231	16384	1324327911	SELECT "id", "permalink", "srcCurrencyId", "dstCurrencyId", "fee", "lastP	13242	2132.46	0.034	38.758	0.161037381	0.38412519	172146	13241	1

Пример

pg_stat_statements

rolname	calls	total_time	mean_time	max_time	stddev_time	rows	query_text
vQRkNUQBeEKtXUPv	3366	14415199.6	4282.6	179364.6	14790.3	3366	select count(messages.id) from…
vQRkNUQBeEKtXUPv	14004	52894491.5	3777.1	117776.6	10765.9	405095	SELECT c0."id", c0."account_id…
vQRkNUQBeEKtXUPv	273	758793.9	2779.5	42796.4	6759.9	28379851	SELECT c0."id", c0."contact_id…
vQRkNUQBeEKtXUPv	899	2020241.2	2247.2	66362.4	7524.9	72791986	SELECT c0."id", c0."contact_gr…
vQRkNUQBeEKtXUPv	225	169522.6	753.4	1923.8	286.5	28097037	SELECT c0."id", c0."contact_id…
vQRkNUQBeEKtXUPv	21552	15569887.2	722.4	87465.8	1502.6	3518875	SELECT m0."id", m0."conversa…
vQRkNUQBeEKtXUPv	3365	1675108.5	497.8	42562.7	2509.0	3365	select count(distinct contacts.id…
vQRkNUQBeEKtXUPv	362	157875.2	436.1	15178.5	1687.2	1219499	SELECT c0."id" FROM "contact…
vQRkNUQBeEKtXUPv	1021	278391.0	272.7	12259.6	836.1	2846433	SELECT c0."id" FROM "contact…
vQRkNUQBeEKtXUPv	4033	962941.6	238.8	8889.4	341.8	1361015	UPDATE "conversations" AS c0…
vQRkNUQBeEKtXUPv	6244	1455266.7	233.1	24909.8	1061.3	7968037	SELECT c0."id" FROM "contact…
vQRkNUQBeEKtXUPv	46342	9350450.5	201.8	25518.8	619.1	60414070	SELECT c0."id", c0."conversati…
vQRkNUQBeEKtXUPv	22601	4334173.5	191.8	28550.8	548.5	22601	SELECT g0."id", g0."account_i…
vQRkNUQBeEKtXUPv	23605	4378991.6	185.5	10095.0	252.4	23605	SELECT c0."id", c0."account_id…
vQRkNUQBeEKtXUPv	53553	6954612.9	129.9	8430.6	318.6	33613916	SELECT p0."id", p0."phone", p…

СИГМА
ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

Статистика

ПАВЕЛ ТРУХАНОВ - Мониторинг Postgres по USE и RED (https://pgconf.ru/2019/242287)

АЛЕКСЕЙ ЛЕСОВСКИЙ - Поиск и устранение проблем в Postgres с помощью pgCenter (https://pgconf.ru/2019/242775)

5. Запросы

Запросы

Тормозят

1. Архитектуры БД.
2. Не правильных настройках СУБД PostgreSQL.
3. Длинные транзакции.

Архитектура БД

1. Денормализация БД. Часть полей в таблице могут повторяться или содержать справочные значения.
2. Секционирование. Это беда в PostgeSQL. Большие ограничения.
3. Частичные индексы. Огромное количество вариантов.
4. Разделение данных. Как вариант: оперативные и архивные.

postgresql.conf

1. Повторюсь
   effective_io_concurrency – ставить больше!
   random_page_cost = seq_page_cost =1! **Для SSD.**

2. Следить за статистикой.

3.
   max_parallel_workers_per_gather = разумное число.
   work_mem – подбирать согласно вашим запросам.
   maintenance_work_mem – в зависимости от обслуживания БД.

4. настраивать «агрессивный» автовакуум.

Запросы

Алексей Ермаков, Илья Космодемьянский -Оптимизация запросов в PostgreSQL - live at PG Day. (https://pgday.ru/files/papers/34/pgday2015.query-optimization-live.pdf)

6. Индексы

Индексы

1. Полезность индекса.
2. Повторяемость индекса. Название индекса разные, а набор полей одинаковый.
3. Можно ли сделать индекс уникальным?
4. Можно ли сделать индекс частичным?

Полезность индекса

```
SELECT
   t.tablename,
   indexname,
   c.reltuples AS num_rows,
   pg_size_pretty(pg_relation_size(quote_ident(t.tablename)::text)) AS table_size,
   pg_size_pretty(pg_relation_size(quote_ident(indexrelname)::text)) AS index_size,
   CASE WHEN indisunique THEN 'Y'
     ELSE 'N'
   END AS UNIQUE,
   idx_scan AS number_of_scans,   -- Количество запущенных сканирований по этому индексу
   idx_tup_read AS tuples_read,    -- Количество элементов индекса, возвращённых при сканированиях по этому
                                   индексу
   idx_tup_fetch AS tuples_fetched -- Количество живых строк таблицы, отобранных при простых сканированиях по
                                   этому индексу
FROM pg_tables t
LEFT OUTER JOIN pg_class c ON t.tablename=c.relname
LEFT OUTER JOIN
   ( SELECT c.relname AS ctablename, ipg.relname AS indexname, x.indnatts AS number_of_columns, idx_scan, idx_tup_read,
idx_tup_fetch, indexrelname, indisunique FROM pg_index x
        JOIN pg_class c ON c.oid = x.indrelid
        JOIN pg_class ipg ON ipg.oid = x.indexrelid
        JOIN pg_stat_all_indexes psai ON x.indexrelid = psai.indexrelid AND psai.schemaname = 'public' )
   AS foo
   ON t.tablename = foo.ctablename
WHERE t.schemaname='public'
ORDER by 7,8;
```

Индексы

Олег Бартунов, Александр Коротков, Федор Сигаев - Вся правда об индексах в PostgreSQL (http://www.sai.msu.su/~megera/postgres/talks/HighLoad-2013.pdf)

https://habr.com/ru/company/postgrespro/blog/326096/ - 6 частей

7. Расширения

Расширения

1. pgbouncer – пул соединений PostgreSQL.
2. pgpool-II – репликация и балансировщик нагрузки БД.
3. pg_pathman — это расширение PostgreSQL, реализующее оптимизированное решение для секционирования больших и распределённых баз данных.
4. pg_probackup — управление резервным копированием и восстановлением кластеров баз данных PostgreSQL.
5. pg_repack — утилита и расширение PostgreSQL для реорганизации таблиц.
6. Модуль pg_variables содержит функции для работы с переменными различных типов.
7. pgbench — программа для запуска теста производительности PostgreSQL.

Спасибо за внимание!