PostgresPro

Если ваш админ сказочный самурай

Или история о восстановлении очень нужных данных

Андрей Билле

Главный инженер департамента выпуска продуктов Postgres Professional


20:46

Что мы имеем?

PostgresPro

Одинокий кластер PostgreSQL
(сборка 1С от Postgres Professional)

01

Десятки баз 1С

02

≈ 10 активных баз

03

Решают восстановить
ещё 1 живую инфобазу
1С-ной утилитой
(многопоточно)

04

Что мы имеем?

PostgresPro

Одинокий кластер PostgreSQL
(сборка 1С от Postgres Professional)

01

Десятки баз 1С

02

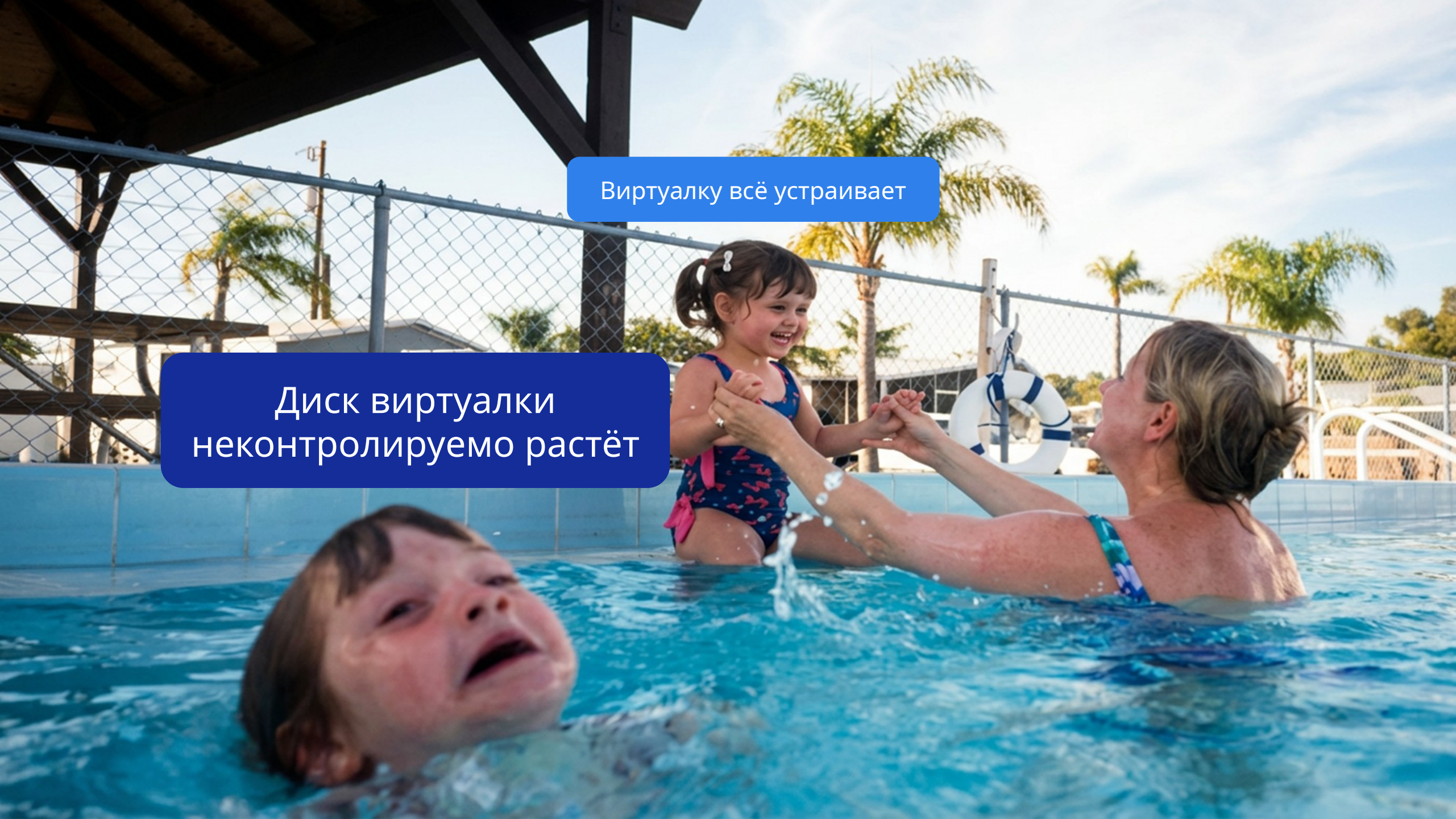
≈ 10 активных баз

03

Решают восстановить
ещё 1 живую инфобазу
1С-ной утилитой
(многопоточно)

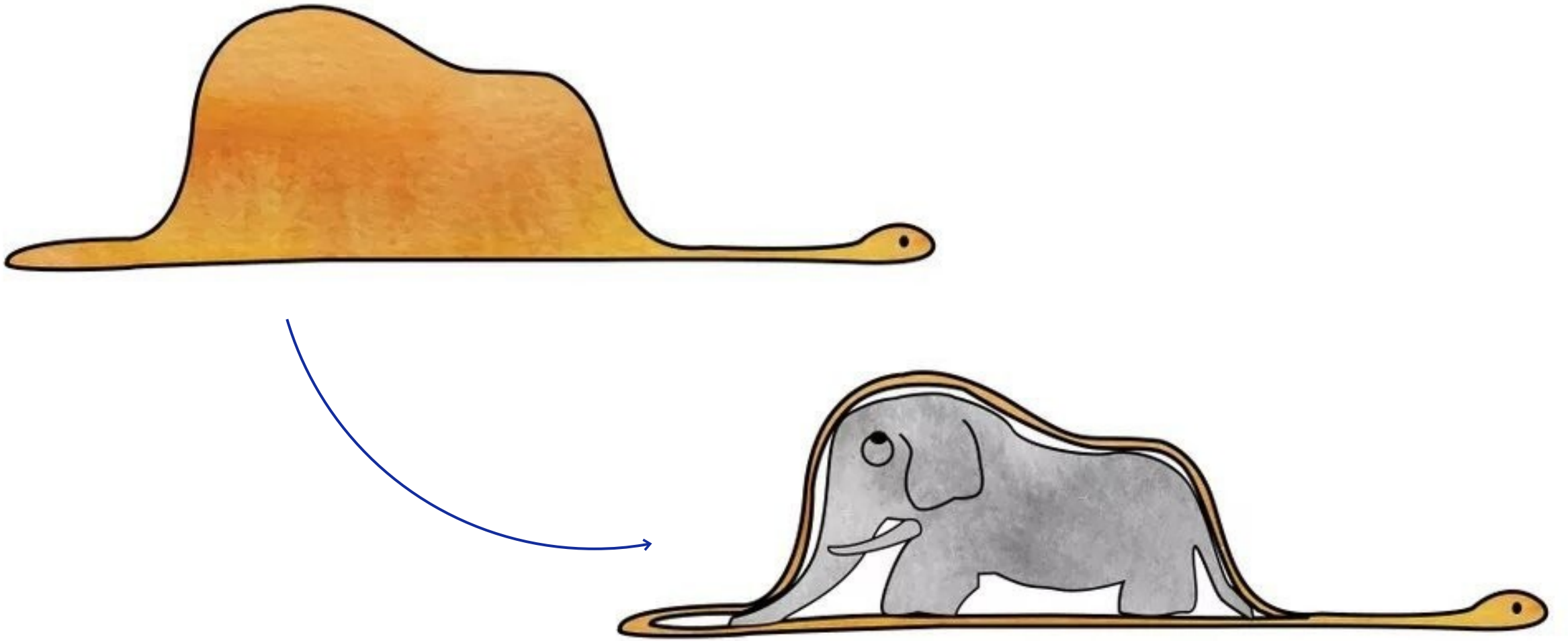
04


ALERT!

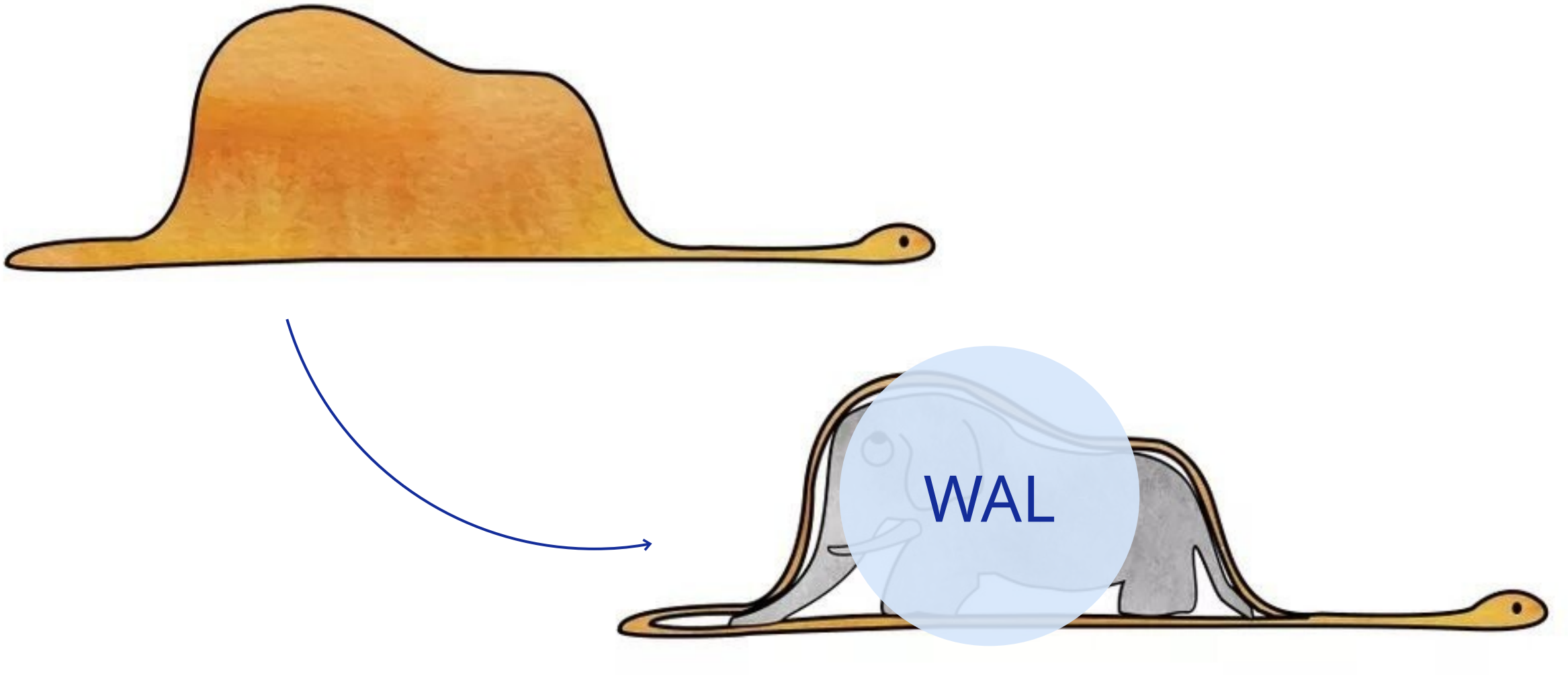

Виртуалку всё устраивает
Диск виртуалки неконтролируемо растёт


ALERT!
ALERT!
ALERT!
ALERT!
ALERT!
ALERT!
ALERT!
ALERT!
ALERT!
ALERT!
ALERT!

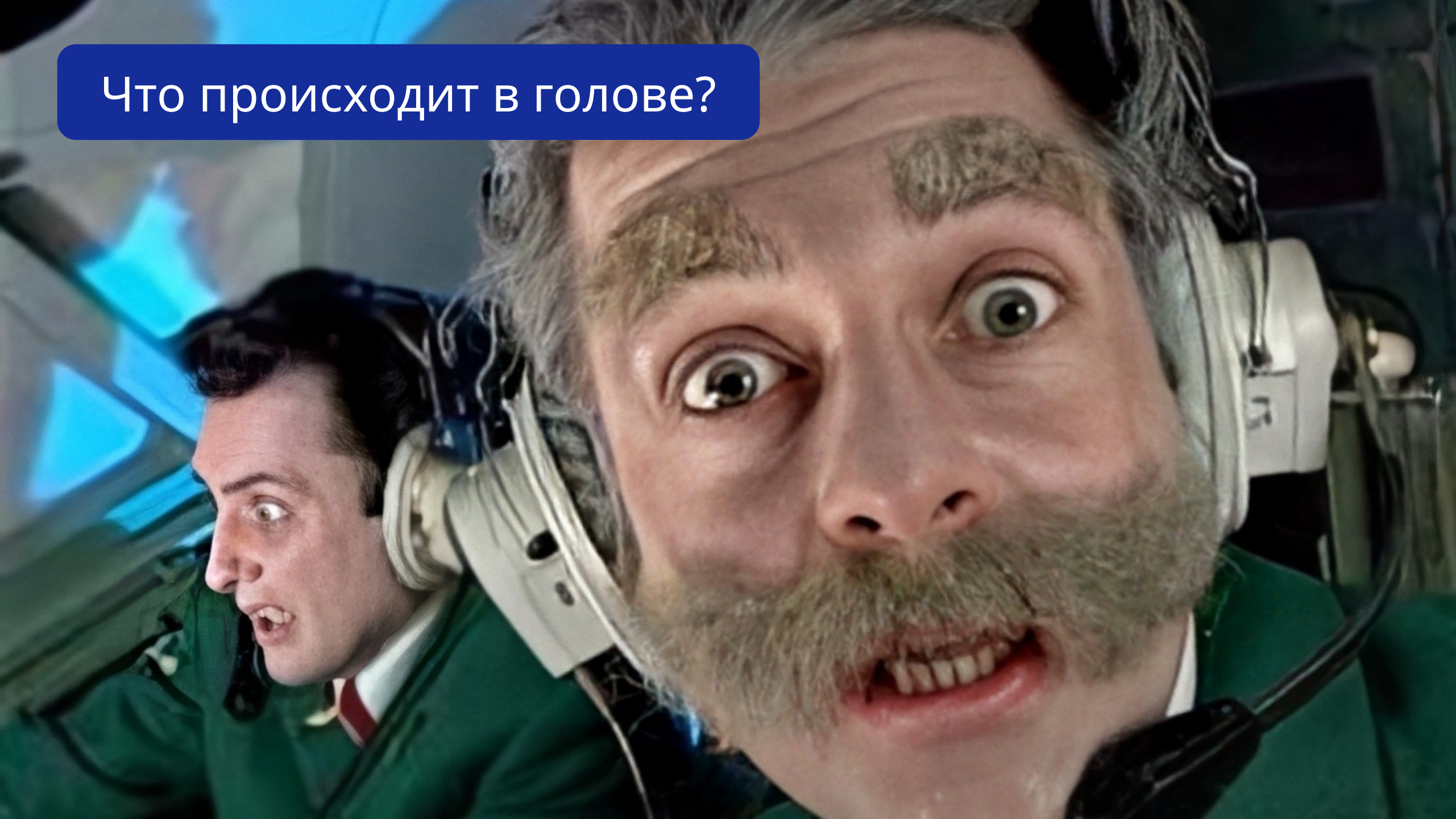
Кластер на виртуальной машине



Кластер на виртуальной машине




ALERT!
ALERT!
ALERT!
ALERT!
ALERT!
ALERT!
ALERT!
ALERT!
ALERT!
ALERT!


Что происходит в голове?

Кто виноват?


ALERT!
ALERT!
ALERT!
ALERT!
ALERT!
ALERT!
ALERT!
ALERT!

Что делают админы?

A: AAAAAAAAAAAAAAAA

B: Смотрят логи и ищут причину роста WAL

C: Удаляют WAL-записи

D: Аварийно останавливают сервер


sysxplore @sysxplore · 20h
Linux in a nutshell
root@terminal:~# love
-bash: love not found
root@terminal:~# happiness
-bash: happiness not found
root@terminal:~# peace
-bash: peace not found
root@terminal:~# kill
-bash: you need to specify whom to kill
59
555
3.5K
162K

Что делать?

pg_resetwal

Утилитой pg_resetwal можно отключить механизм восстановления

PostgresPro

База запустится, но останется гарантированно повреждённой

Никогда не используйте pg_resetwal, если не понимаете совершенно
точно, что он делает и какие будут последствия!

Утилита pg_resetwal позволит экземпляру постгреса запуститься, даже если половины базы уже нет, а вторая половина является бессмысленным набором байтов


Сервер стартовал!


А 1С нет...

Что же делать дальше?

Начали искать знакомых специалистов, стучать во все двери

Антон Дорошкевич

С 2004 спец в 1С, большой любитель Postgres

Ну теперь то уж точно всё заработает?

Что видит Антон?

- Пробует подключиться к базе не посредством 1С (pgAdmin, хоть и осуждаю)
- Запросы не выполняются, те что сложнее select 1;

Чуда не случилось!

Что делать дальше?

PostgresPro

Наши действия

PostgresPro

- Консоль не дали, (не)сломанный телефон
- Отладочные символы
- Пусть висит запрос, посмотрим что делает
- gdb / strace
- crash_info

А чем же занимается backend?

crash_info, не все kill одинаково вредны!
Тот, что 40 от postgrespro, бывает очень полезным

А чем же занимается backend?

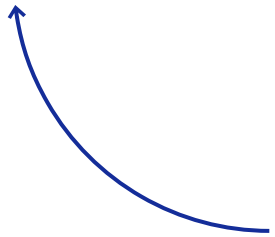
PostgresPro

```
# Backtrace
1   postgres + 0x694ea0            0x5c2c836e43fd  0x00005c2ca590b5a0
CrashInfoSignalHandler + 0x55d
  CrashInfoSignalHandler
/usr/src/postgrespro-1c-15-15.13-1.noble/build/../src/backend/crash_info/
crash_handler.c:535
2   libc.so.6 + 0x4532f            0x7e52c8245330  0x00005c2ca590b680
__sigaction + 0x50
is a signal frame
```

3 postgres + 0x1d31c0 0x5c2c83222257 0x00007fff5fa81f30 _bt_moveright + 0x97 _bt_moveright

```
/usr/src/postgrespro-1c-15-15.13-1.noble/build/../src/backend/access/nbtree/
nbtsearch.c:281
4   postgres + 0x1d6f10            0x5c2c83226062  0x00007fff5fa81f90
_bt_get_endpoint + 0x4f2
  _bt_search
/usr/src/postgrespro-1c-15-15.13-1.noble/build/../src/backend/access/nbtree/
nbtsearch.c:134
```

```
SET enable_indexscan TO off;
```



Пробуем вообще не использовать индексы, так как непонятно по какому мы там индексу бродим

Чем теперь занимается backend?

А если посмотреть внимательнее?

PostgresPro

```
# Planner queries:
00  SELECT DISTINCT att.attname as name, att.attnum as OID, pg_catalog.format_type(ty.oid,NULL) AS datatype,
att.attnotnull as not_null,
CASE WHEN att.atthasdef OR att.attidentity != '' OR ty.typdefault IS NOT NULL THEN True
ELSE False END as has_default_val, des.description, seq.seqtypid
FROM pg_catalog.pg_attribute att
   JOIN pg_catalog.pg_type ty ON ty.oid=atttypid
   JOIN pg_catalog.pg_namespace tn ON tn.oid=ty.typnamespace
   JOIN pg_catalog.pg_class cl ON cl.oid=att.attrelid
   JOIN pg_catalog.pg_namespace na ON na.oid=cl.relnamespace
   LEFT OUTER JOIN pg_catalog.pg_type et ON et.oid=ty.typelem
   LEFT OUTER JOIN pg_catalog.pg_attrdef def ON adrelid=att.attrelid AND adnum=att.attnum
   LEFT OUTER JOIN (pg_catalog.pg_depend JOIN pg_catalog.pg_class cs ON classid='pg_class'::regclass AND objid=cs.oid AND cs.relkind='S') ON
refobjid=att.attrelid AND refobjsubid=att.attnum
   LEFT OUTER JOIN pg_catalog.pg_namespace ns ON ns.oid=cs.relnamespace
   LEFT OUTER JOIN pg_catalog.pg_index pi ON pi.indrelid=att.attrelid AND indisprimary
   LEFT OUTER JOIN pg_catalog.pg_description des ON (des.objoid=att.attrelid AND des.objsubid=att.attnum AND des.classoid='pg_class'::regclass)
   LEFT OUTER JOIN pg_catalog.pg_sequence seq ON cs.oid=seq.seqrelid
WHERE
   att.attrelid = 3740287157::oid
   AND att.attnum > 0
   AND att.attisdropped IS FALSE
ORDER BY att.attnum
```

Системные индексы/каталог

PostgresPro

- Любой уважающий себя запрос, скорее всего, захочет узнать информацию о таблицах, которые в нем участвуют

Системные индексы/каталог

PostgresPro

- Любой уважающий себя запрос, скорее всего, захочет узнать информацию о таблицах, которые в нем участвуют
- Вся информация об этих таблицах хранится... в таблицах

Системные индексы/каталог

PostgresPro

- Любой уважающий себя запрос, скорее всего, захочет узнать информацию о таблицах, которые в нем участвуют
- Вся информация об этих таблицах хранится… в таблицах
- В случае ERP 1С таблиц много, полей много, каталог большой… чтобы это как-то быстро работало нам помогают индексы системных таблиц

Так чем же все-таки занимается backend?

Он ходит по кругу в каком-то
из системных индексов

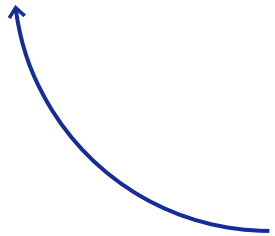
Дело было к вечеру, попросили сделать REINDEX SYSTEM — в некоторых случаях оно помогает

Прошла ночь… REINDEX не завершился

Отключение системных индексов

PostgresPro

ignore_system_indexes



Такую вот соломку для подобных случаев подстелили создатели Postgres

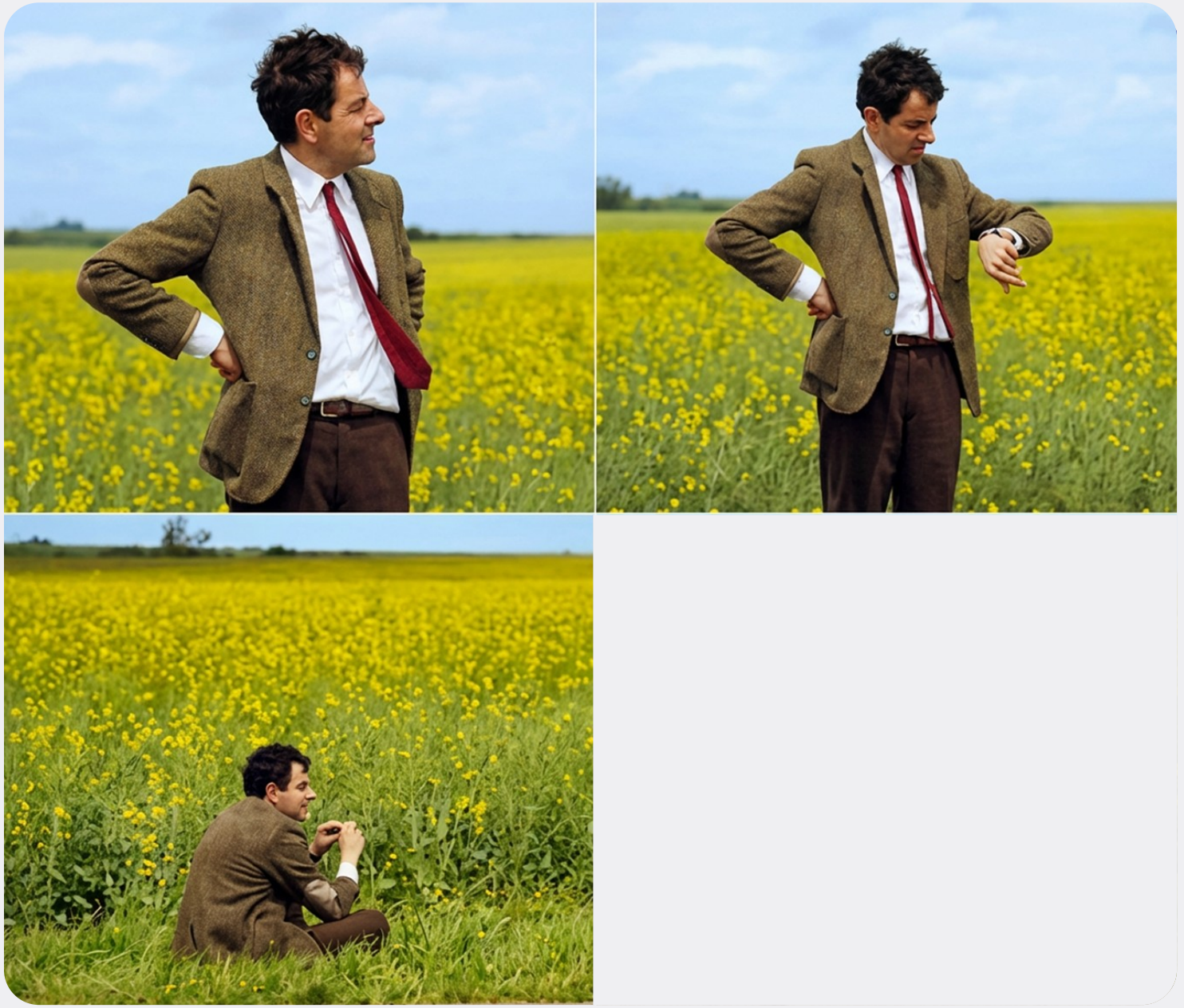
Отключаем системные индексы (не важно как), и пробуем спасать данные

В базе много-много таблиц. Баз у нас тоже немало
Побазово начинаем делать pg_dump...

pg_dump

pg_dump

pg_dump



pg_dump

pg_dump что-то делает, делает долго

В stat_activity виден кусок SELECT, посмотрим, что же это за SELECT?

Кто поможет?

crash_info!

И что же он делает?

PostgresPro

```
SELECT t.tableoid, t.oid,
..., pg_catalog.pg_get_indexdef(i.indexrelid)
AS indexdef, ...
FROM
unnest('{16385, ... ,136377}'::pg_catalog.oid[]) AS src(tbloid)
      JOIN pg_catalog.pg_index i ON (src.tbloid = i.indrelid) JOIN
pg_catalog.pg_class t ...,
indexname
```

Мы получаем определение индексов

А нужны ли они нам?

Без системных индексов это будет мучительно долго

Все данные о таблицах СУБД 1С хранит в таблице СУБД

По сути у них есть свой системный каталог

От нашего «пациента на аппаратах жизнеобеспечения» не требуется ни описание таблиц, ни индексов… Только данные

data-only — ключ к pg_dump

НО!

pg_dump что-то делает, делает долго...

Кажется, что этот слайд уже был...

Таблиц в базе много

Никто не ожидал что будет быстро.
Но на всякий случай просим еще раз по 40!

И что же он делает?

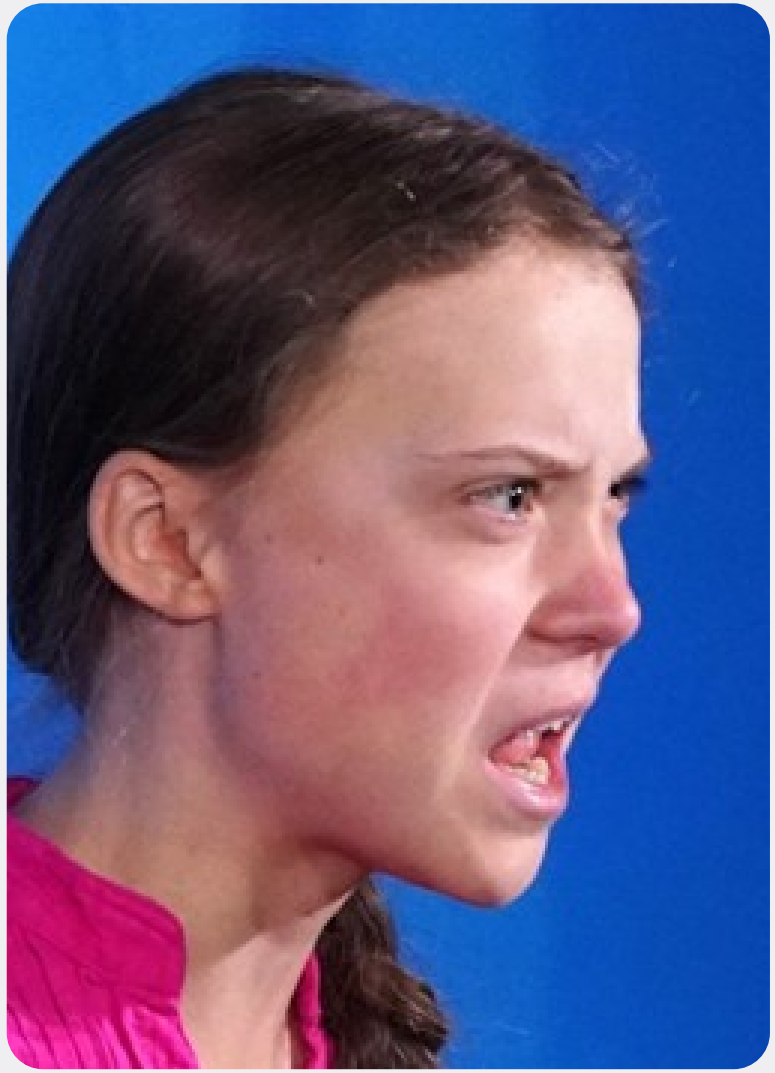
PostgresPro

```
SELECT t.tableoid, t.oid,
..., pg_catalog.pg_get_indexdef(i.indexrelid)
AS indexdef, ...
FROM
unnest('{16385, ... ,136377}'::pg_catalog.oid[]) AS src(tbloid)
      JOIN pg_catalog.pg_index i ON (src.tbloid = i.indrelid) JOIN
pg_catalog.pg_class t ...,
indexname
```


wat

Смотрим код

PostgresPro



```
pg_log_info("flagging inherited columns in subtables");
flagInhAttrs(fout, fout->dopt, tblinfo, numTables);

pg_log_info("reading partitioning data");
getPartitioningInfo(fout);

pg_log_info("reading indexes");
getIndexes(fout, tblinfo, numTables);

pg_log_info("flagging indexes in partitioned tables");
flagInhIndexes(fout, tblinfo, numTables);

pg_log_info("reading extended statistics");
getExtendedStatistics(fout);

pg_log_info("reading constraints");
getConstraints(fout, tblinfo, numTables);

pg_log_info("reading triggers");
getTriggers(fout, tblinfo, numTables);
```

Ну ладно, может, дождаться?

PostgresPro

НЕТ!

Стратегия была такова: если запустить дамп только данных, то он практически гарантировано свалится на какой-то побитой таблице

Ну ладно, может, дождаться?

PostgresPro

НЕТ!

Стратегия была такова: если запустить дамп только данных, то он практически гарантировано свалится на какой-то побитой таблице

...

Запускаем дамп еще раз, исключив эту таблицу

...

Ну ладно, может, дождаться?

PostgresPro

НЕТ!

Стратегия была такова: если запустить дамп только данных, то он практически гарантировано свалится на какой-то побитой таблице

...

Запускаем дамп еще раз, исключив эту таблицу

...

Умножаем на 3+ часа ожидания
(а, может, и больше, кто знает?)

НЕТ!

Бизнес простаивает уже сутки...

PostgresPro

Можно ли изменить поведение pg_dump?

```
-       pg_log_info("reading indexes");
-       getIndexes(fout, tblinfo, numTables);
+       if (!dataOnly)
+       {
+               pg_log_info("reading indexes");
+               getIndexes(fout, tblinfo, numTables);

-       pg_log_info("flagging indexes in partitioned tables");
-       flagInhIndexes(fout, tblinfo, numTables);
+               pg_log_info("flagging indexes in partitioned tables");
+               flagInhIndexes(fout, tblinfo, numTables);
+       }
```

Дело пошло, таблицы вычислены

По классике дихотомией ищем убитые строки

https://postgreshelp.com/postgresql-checksum/

Delete не работает

create table тоже не работает (скорее всего,
все начинает блудить в системных индексах, не изучали)

Спасает кластер “рядом” и fdw

Прошло около суток, базы ожили

На этом история могла закончиться, но кто мы такие, чтобы не поделиться ею с миром?

Поход в bugs

Возможно, стоило в hackers,
но вряд ли бы это что-то ускорило

С чем туда идти?

Искусственный пример

```
DO $$
DECLARE
 i INTEGER;
 j INTEGER;
BEGIN
 FOR i IN 1..15000 LOOP
 EXECUTE 'CREATE TABLE tab' i ' as SELECT 1 as f';
 FOR j IN 1..5 LOOP
 EXECUTE 'CREATE index idx_tab' i '_' j ' ON tab' i '(f)';
 END LOOP;
 END LOOP;
END;
$$;
—----------------------------------------------------
ignore_system_indexes = on
time pg_dump --data-only test > test.sql
```

https://pgpro.pro/dataonly

С чем туда идти?

real 62m44,582s
user 0m0,576s
sys 0m0,259s

С патчем

time pg_dump --data-only test > test.sql

real 7m45,533s
user 0m0,726s
sys 0m0,634s

https://pgpro.pro/dataonly

Как прошло обсуждение?

Да, возможно стоило идти не в bugs

Самый главный “цербер” настроен скептически

Потому что: “А мало ли что?”
Резать так ради “Unsupported bug” никто не собирается

Новая надежда

David Rowley предложил пойти другим путем — улучшить запрос

И даже что-то предложил

Но... это был фальшстарт. Потому что кэш.

Дэвид наносит ответный удар

PostgresPro

Патч уже посложнее, и «мы это берём», но на этом всё и заглохло, да и нас бы устроило едва ли

master
$ PGOPTIONS='-c ignore_system_indexes=1' time pg_dump --schema-only postgres >> /dev/null
2:23.75elapsed

$ PGOPTIONS='-c ignore_system_indexes=0' time pg_dump --schema-only postgres >> /dev/null
0:01.08 elapsed

patched:
$ PGOPTIONS='-c ignore_system_indexes=1' time pg_dump --schema-only postgres >> /dev/null
0:40.28elapsed

$ PGOPTIONS='-c ignore_system_indexes=0' time pg_dump --schema-only postgres >> /dev/null
0:00.78elapsed

i.e about 3.5x faster with ignore_system_indexes and 38% faster without.

Почему заглохло?

Got it. Definitely looks promising, but I'm too tired to review the whole change.

Regards, Tom Lane

Что в итоге?

PostgresPro

Самые важные данные в строю

Где-то клиенту повезло и побилось то, что и не нужно вовсе

Бизнес перепроверил все документы за сутки до падения

Последствия истории

PostgresPro

- Админы поняли как делать не надо,
  а как надо (кстати, когда делали как надо,
  развалился рейд… карма)
- Сколько-то часов переработок
- Купленный ент и поддержка
- Подаренный Антоном виски

Как жить?

- Не забываем делать бэкапы
- Заранее подключаем инструменты, которые в случае фаталити смогут нам облегчить расследование
- Архивы WAL

сколько уже про это все сказано...

PostgresPro

Отвечу на ваши вопросы!

Андрей Билле

Главный инженер департамента выпуска продуктов Postgres Professional