PostgreSQL-Consulting

data egret

Архитектура СУБД PostgreSQL и ORACLE в сравнении

PGConf.Russia 2018

Виктор Егоров
vyegorov@dataegret.com

Предисловие

- архитектура *целиком* не рассматривается
- затронуты главные компоненты обех СУБД
- с акцентом на отличиях
- подробности можно найти в литературе

Литература

- официальная документация
- “PostgreSQL 9.6 High Performance”, Gregory Smith
- “Mastering PostgreSQL 9.6”, Hans-Jürgen Schönig
- “Oracle Performance Firefighting”, Craig Shallahamer
- “Forecasting Oracle Performance”, Craig Shallahamer
- “Expert Oracle Database Architecture”, Thomas Kyte, Darl Kuhn

О чём будем говорить?

1. общие черты
2. создание баз
3. хранение данных на носителях
4. структура блока и записи
5. процессы
6. память, общая и сессионная
7. redo и undo
8. бэкапы, репликация

- ACID совместимость
- версионный движок
  “писатели” не блокируют “читателей”
- транзакционный лог
- бэкапы и репликация
- SQL совместимость

- настройка *ORACLE_SID* (посредством */etc/oratab* и *oraenv*)
- подготовка *init.ora* файла в *$ORACLE_HOME/dbs*
- подключение и запуск экземпляра (*NOMOUNT*)
- *CREATE DATABASE* с указанием табличных областей, *redo* и прочих параметров
- создание системных представлений
- настройка *listener.ora* и *tnsnames.ora*
- управление табличными областями, параметрами и базами (в случае multitenant) по мере необходимости

- запуск *initdb* утилиты с указанием *PGDATA*
- правка *postgresql.conf* и, при необходимости, *pg_hba.conf*
- управление табличными областями, параметрами и базами по мере необходимости

- поддержка табличных областей — обе СУБД *создаются при инициализации новой базы*
- файлы с данными — только в ORACLE
- PostgreSQL работает с FS
- ORACLE может работать с raw-устройствами и использовать IO минуя кэш ядра

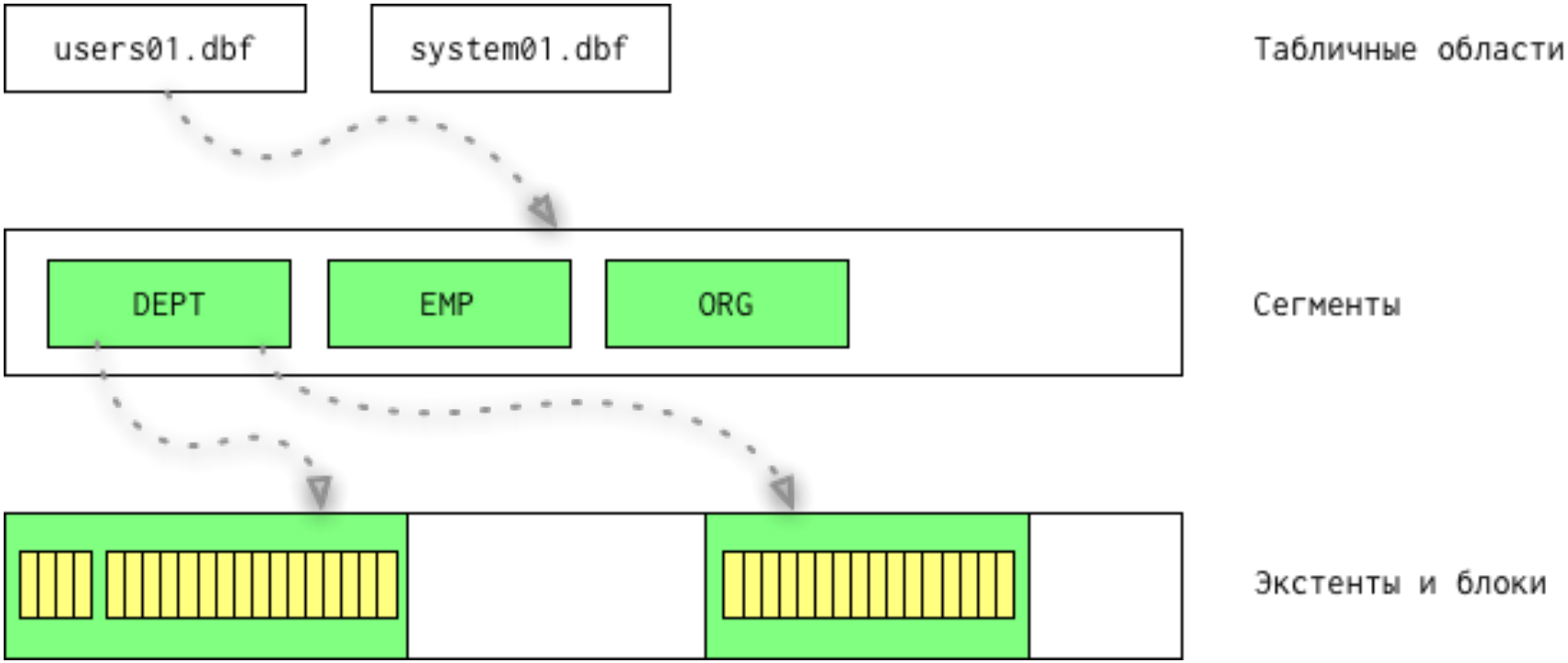
ORACLE

```
$ ls -l
-rw-r-----. 1 oracle oinstall sysaux01.dbf
-rw-r-----. 1 oracle oinstall system01.dbf
-rw-r-----. 1 oracle oinstall temp01.dbf
-rw-r-----. 1 oracle oinstall undotbs01.dbf
-rw-r-----. 1 oracle oinstall users01.dbf
-rw-r-----. 1 oracle oinstall usertbs01.dbf
```

PostgreSQL

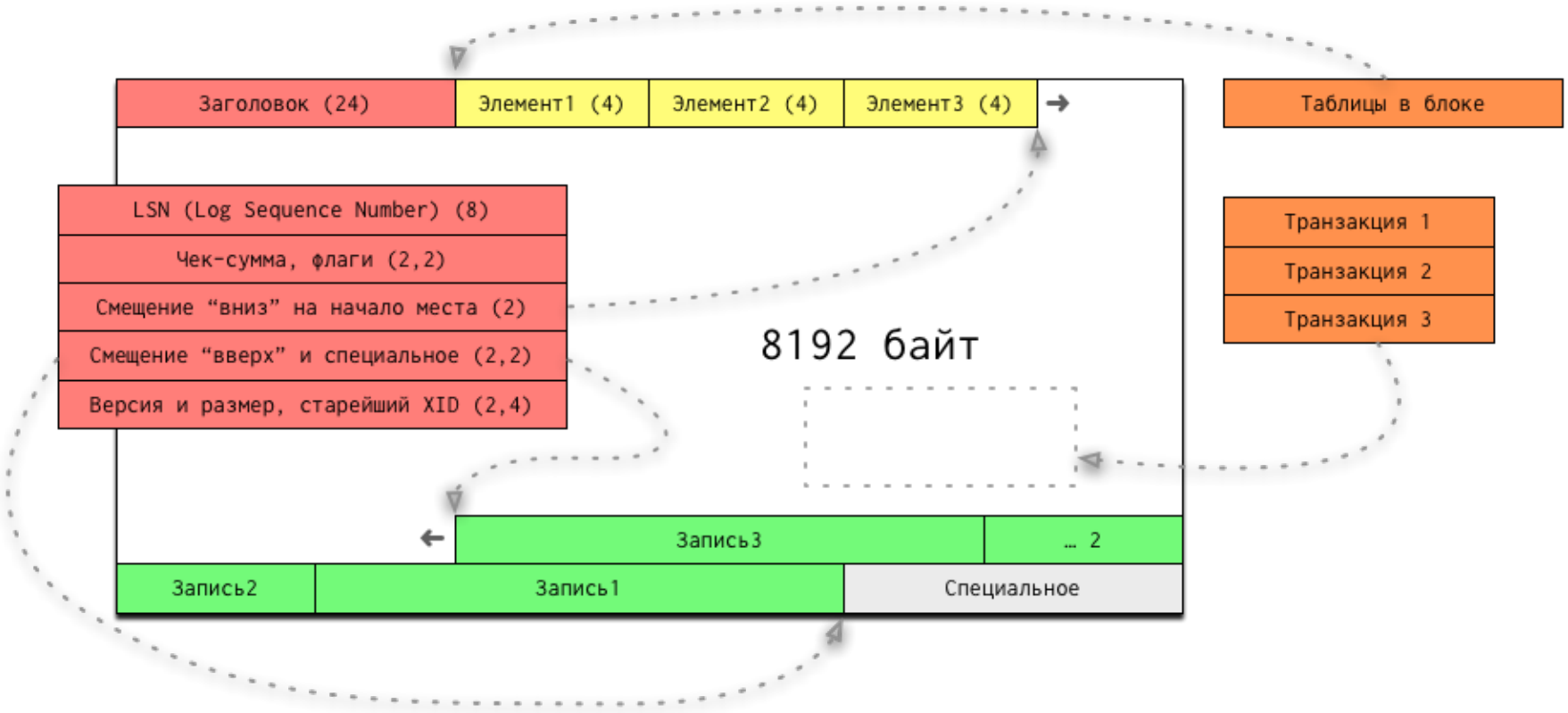
```
$ tree
|-- 1
|-- 13274
|-- 13275
    |-- 112
    |-- 113
    |-- 1247
    |-- 1247_fsm
    |-- 1247_vm
    |-- 1249
    | ...
    |-- pg_filenode.map
    |-- PG_VERSION

3 directories, 804 files
```


- Сегменты — обе СУБД
  ORACLE размещает в файлах данных, Postgres в индивидуальных файлах
- Экстенты — обе СУБД
  Postgres с 9.6 умеет выделять место экстентами
- Блоки — обе СУБД
- в ORACLE размеры блока / экстента / сегмента настраиваемы, в Postrges — вкомпилированные константы


users01.dbf
system01.dbf
Табличные области
DEPT
EMP
ORG
Сегменты
Экстенты и блоки

- блок имеет заголовок со служебной информацией и таблицу указателей на записи
- накладные расходы в ORACLE составляют “on average data block overhead total 84 to 107 bytes”
  включая таблицу указателей на записи
- накладные расходы в Postgres составляют 24 байта заголовка + 4*N
  где N — число записей в блоке
- наполнение блока ведётся с конца
- в ORACLE в блоке содержится список Транзакций, ждущих изменений данных в этом блоке
- в ORACLE блок может содержать записи от разных таблиц, в отличии от Postgres’a

4. Структура блока в PostgreSQL (*и отличия ORACLE*)



- заголовок записи в Postgres составляет 23 байта + необязательная NULL-маска (битовая)
- в ORACLE заголовок записи гораздо меньше, 3 байта в оптимальном случае
- в ORACLE отсутствует NULL-маска

- Postgres обеспечивает версионность на уровне записи
  что напрямую влияет на размер заголовка записи
- в Postgres запись не может выходить за пределы блока,
  большие значения хранятся в специальных TOAST-таблицах
- в ORACLE длинные записи формируют цепочки из нескольких блоков

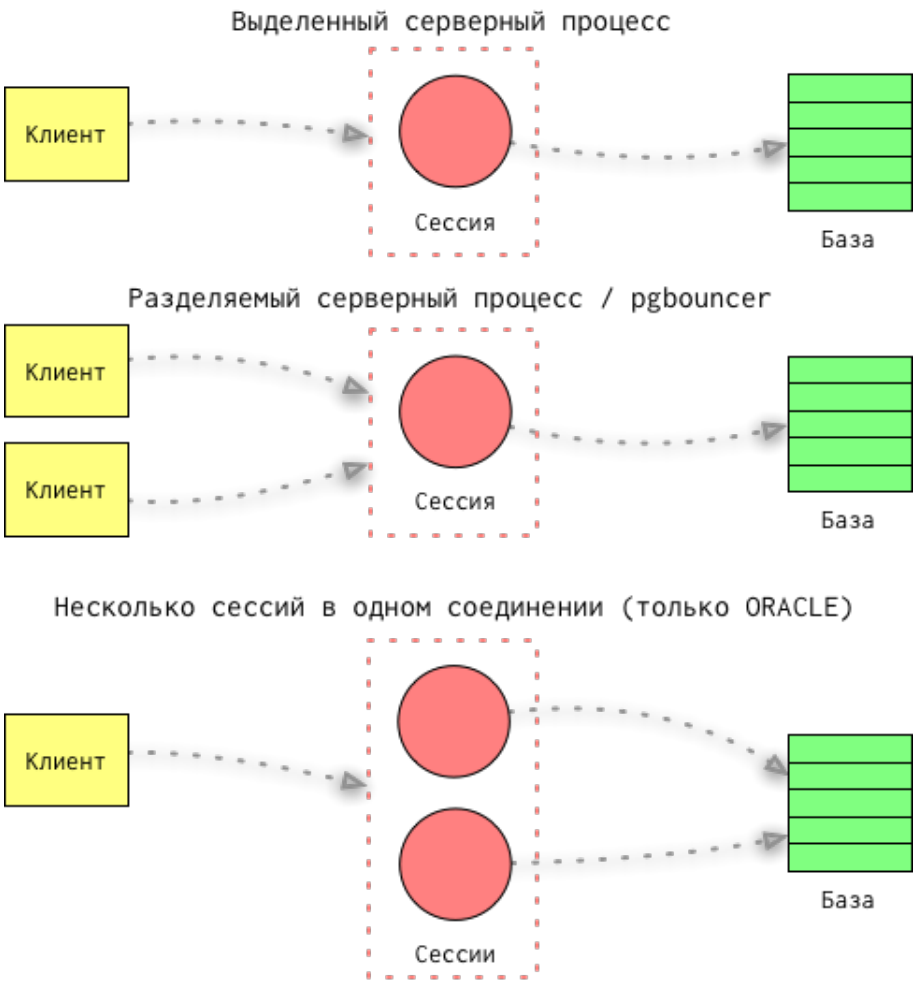
```
$ ps fx|grep -v idle
/usr/lib/postgresql/9.6/bin/postgres -D /var/lib/postgresql/9.6/main
 \_ postgres: 9.6/main: checkpointer process
 \_ postgres: 9.6/main: writer process
 \_ postgres: 9.6/main: wal writer process
 \_ postgres: 9.6/main: autovacuum launcher process
 \_ postgres: 9.6/main: stats collector process
```


```
$ ps -aef | grep ora_....._$ORACLE_SID | grep -v grep
ora_pmon_ORA12CR1   ora_dbrm_ORA12CR1   ora_p000_ORA12CR1
ora_lreg_ORA12CR1   ora_dia0_ORA12CR1   ora_p001_ORA12CR1
ora_smon_ORA12CR1   ora_lg00_ORA12CR1   ora_p002_ORA12CR1
ora_ckpt_ORA12CR1   ora_lg01_ORA12CR1   ora_p003_ORA12CR1
ora_dbw0_ORA12CR1   ora_reco_ORA12CR1   ora_p004_ORA12CR1
ora_lgwr_ORA12CR1   ora_mmnl_ORA12CR1   ora_p005_ORA12CR1
ora_diag_ORA12CR1   ora_tmon_ORA12CR1   ora_p006_ORA12CR1
ora_mman_ORA12CR1   ora_smco_ORA12CR1   ora_p007_ORA12CR1
ora_mmon_ORA12CR1   ora_fbda_ORA12CR1   ora_qm02_ORA12CR1
ora_psp0_ORA12CR1   ora_aqpc_ORA12CR1   ora_w000_ORA12CR1
ora_vktm_ORA12CR1   ora_cjq0_ORA12CR1   ora_w001_ORA12CR1
ora_gen0_ORA12CR1   ora_tt00_ORA12CR1   ora_q002_ORA12CR1
                                        ora_q003_ORA12CR1
```


ORACLE	PostgreSQL	Комментарий
pmon, *smon*, *lreg*, *listener*	головной процесс	В ORACLE нет иерархии для фоновых процессов
ckpt, *dbw**	*checkpointer*	*ckpt* не пишет грязные блоки, а только помечает загловки файлов
*dbw**	*bgwriter*	в Postgres'е блоки сохраняют *bgwriter*, *checkpointer* и индивидуальные сессии
*p0***	*parallel worker*	в Postgres'е параллельные обработчики запускаются по необходимости
lgwr, *lg**	*wal writer*	ORACLE умеет писать redo в несколько потоков

- каждое клиентское соединение обслуживается отдельным серверным процессом
- ORACLE позволяет настроить shared servers, для обслуживания многих соединений одним процессом
- для Postgres’a рекомендуется использовать *pgbouncer*

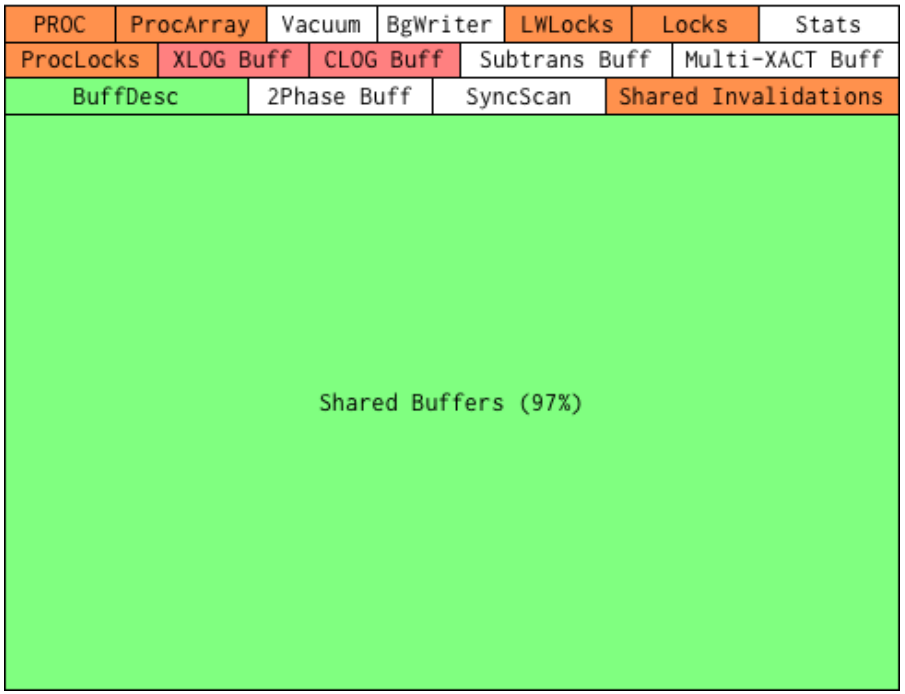
5. Типы подключений к базе



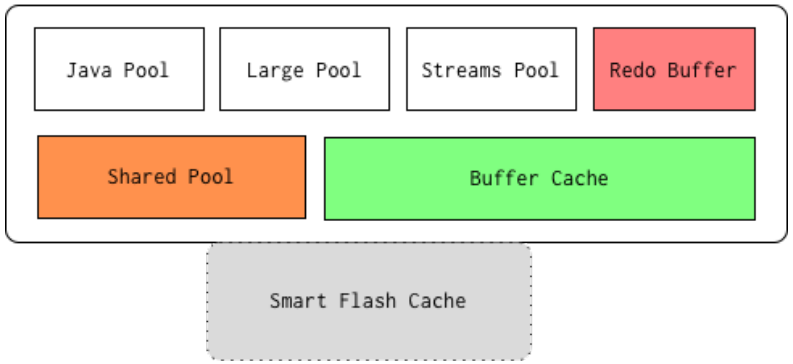
В ORACLE’е понятия “сессия” и “соединение” разделены.

В частности, *autotrace* создаёт независимую сессию через то же соедение, что используется для выполнения трассируемого запроса. Автономные процедуры пользуются этим же механизмом.

PostgreSQL



ORACLE SGA



- головной процесс отвечает за создание структур в паямти
- таблицы процессов, блокировок, кэши транзакционного и коммит логов, структуры для фоновых процессов, статитсика
- основное место занимают *shared_buffers* — кэш блоков с данными
- блоки *всегда* читаются через *shared_buffers*
- “Inside PostgreSQL Shared Memory”, Bruce Momjian
- “Модели разделяемой памяти в PostgreSQL”, Дмитрий Кремер

- SGA управляется базой автоматически (11g и выше)
- кэш *redo*, кэш блоков и ряд пулов (*pool*) — *shared*, *large*, *java*, *streams* и пр.
- можно подключить “Smart Flash Cache”, который будет кэшем 2-го уровня для основного *BufferCache*
- *shared_pool* содержит планы, код процедур, системный каталог и критичен для общей производительности системы
- база может читать с диска минуя кэш
- поддерживается работа с блоками разных размеров

- PGA не является “общим” участком памяти, однако ORACLE поддерживает суммарню память в заданных *pga_aggregate_target* рамках
- Postgres хранит планы запросов в памяти сессий
- память контролируется *work_mem* для *узла* в плане запроса

- любые изменения фиксируются в транзакционном логе *до* изменений в блоках
- Postgres умеет Direct IO для WAL логов
- Postgres записывает WAL в сегменты по 16MB в *PGDATA/pg_xlog* (переименовано в *pg_wal* в 10.0)
- ORACLE создаёт лог-группы с одним и более файлами в группе
- обе базы поддерживают архивацию транзакционного лога

- Postgres обеспечивает версионность на уровне записией в “куче”, создавая “копию” записи
- индексы в Postgres’е не версионируются
- Postgres’у требуется вакуумирование для избавления от устаревших версий
- ORACLE версионирует на уровне блоков, старые версии складываются в UNDO сегмент
- ORACLE позволяет работать с UNDO посредством Flashback Query

- горячие бэкапы, PITR восстановление
- база “не знает” о бэкапах, восстанавливать надо кластер целиком
- архивация через “внешние” утилиты
- репликационные слоты с гарантией доставки транзакционных логов
- hot standby, потоковая репликация, каскадная репликация, синхронная репликация
- логическая репликация и декодирование WAL
- утилиты для управления (*pg_barman*, *wal-e*) и снятия бэкапов (стандартная *pg_basebackup* и *pgBackRest*)

- горячие бэкапы, PITR восстановление
- RMAN, интегрирован в базу, поддержка инкрементальных бэкапов и BCT
- восстановление базы, табличной области, файла данных или блока
- PITR для экземпляра или для табличной области
- transportable tablespace, pluggable database
- резервный экземпляр — Active Data Guard, RAC (не полная резервация)

- Flashback Query / Table / Database
- ORACLE CDC / Streams / Golden Gate

- инструментарий для мониторинга
- PITR для табличных областей / баз
- переносимые табличные области / базы
- “осознание” бэкапов базой (backup slots?)