PostgreSQL upgrade yesterday, today, tomorrow

Zdeněk Kotala
Solaris Revenue Product Engineer
Sun Microsystems

Agenda

- Overview
- Catalog upgrade
- Storage upgrade
- What next

Overview

Goals

- Minimal downtime
- No extra space
- Without old version
- Easy to use

Yesterday

- pg_dump/pg_restore
  - > Long downtime – depends on database size
  - > Requires extra dump
  - > Not good for big database
  - > Universal
- Slony
  - > Possible online upgrade
  - > Configuration is not easy
  - > Requires extra space or extra server

Today

- pg_migrator, pg_upgrade.sh
  - Only for 8.1 -> 8.2 or 8.3 -> 8.4
  - Based on magic and hacks

Tomorrow

Who knows?

Catalog upgrade

What catalog means

- Control file
- Flat files
- Directory structure
- System tables (pg_catalog.*)
- Configuration files

Current solutions

- Pg_migrator or pg_upgrade.sh
  - Only for 8.1->8.2, 8.3->8.4
  - Problem with tablespaces – important to keep data on same mountpoint
  - Problem with TOAST tables (TOAST pointer)
  - Problem with dropped columns (solved by hack in pg_dump)
  - Depends on hacks and magic
  - Some catalog data are lost
  - Not well tested

pg_migrator

See Bruce's presentation

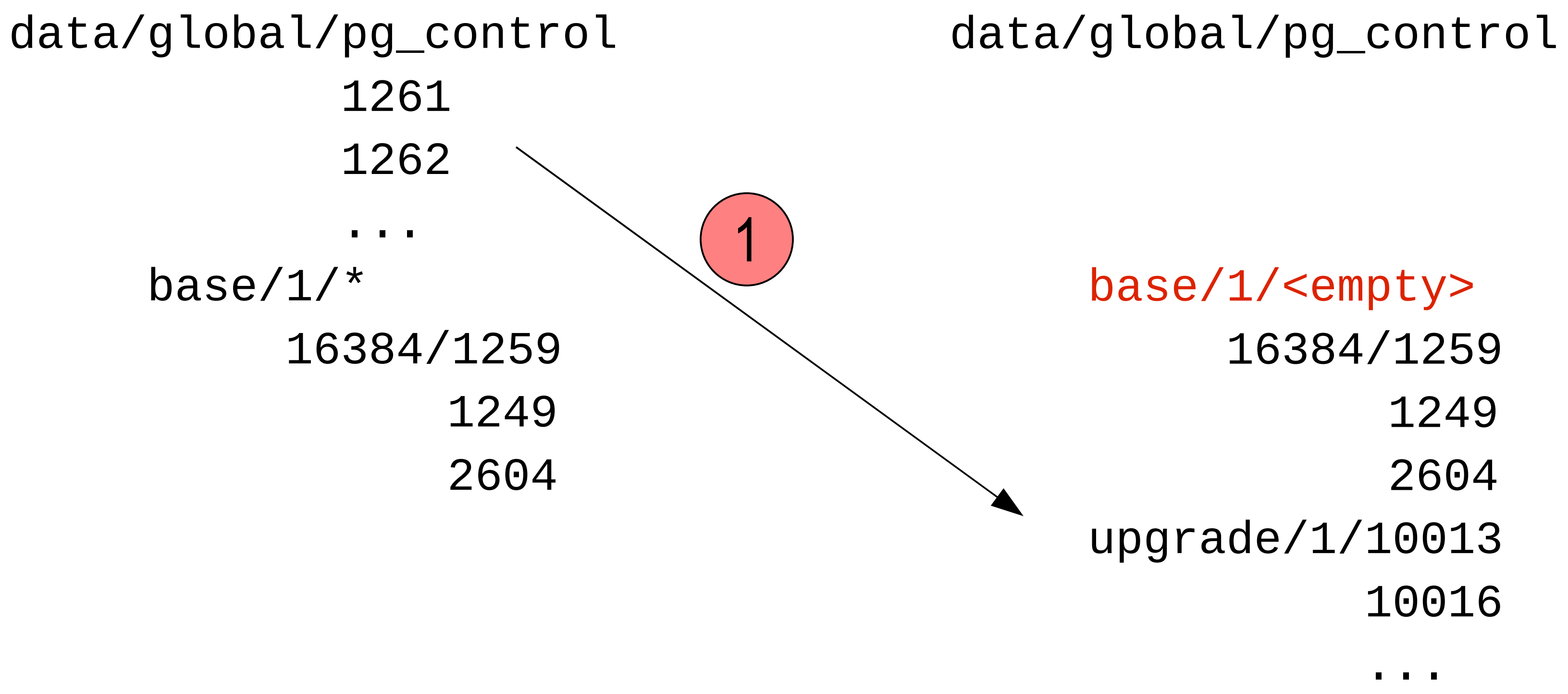
pg_upgrade (1)

- Create pg_upgrade tablespace directory
- Create directory for template1 database
- Move shared catalog tables into the directory
- Run postgres in upgrade mode (like bootstrap)
  - Convert control file
  - Process BKI and create template1 database including pg_upgrade tablespace and old catalog tables in pg_upgrade schema

pg_upgrade (2)

- Run postgres in single mode
  - Convert shared catalog tables data
  - Convert databases catalog data
    - Create database directory in pg_upgrade tablespace
    - Move old catalog tables into the directory
    - Copy new catalog tables from template1 into pg_default
    - Convert catalog tables data
    - Drop pg_upgrade schema and tablespace

pg_upgrade – workflow (1)



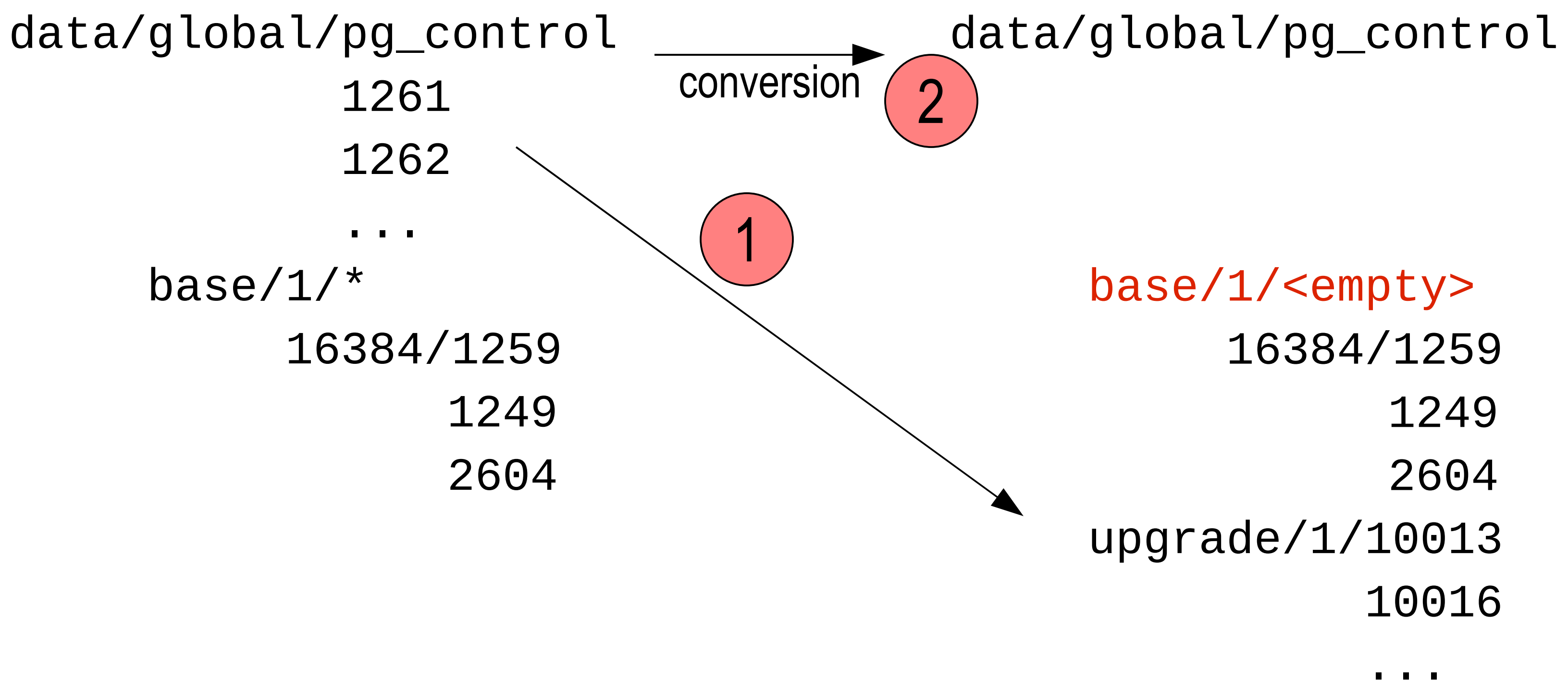
```
data/global/pg_control
            1261
            1262
            ...
     base/1/*
          16384/1259
                1249
                2604
```

```
data/global/pg_control



     base/1/<empty>
          16384/1259
                1249
                2604
     upgrade/1/10013
               10016
              ...
```

pg_upgrade – workflow (2)

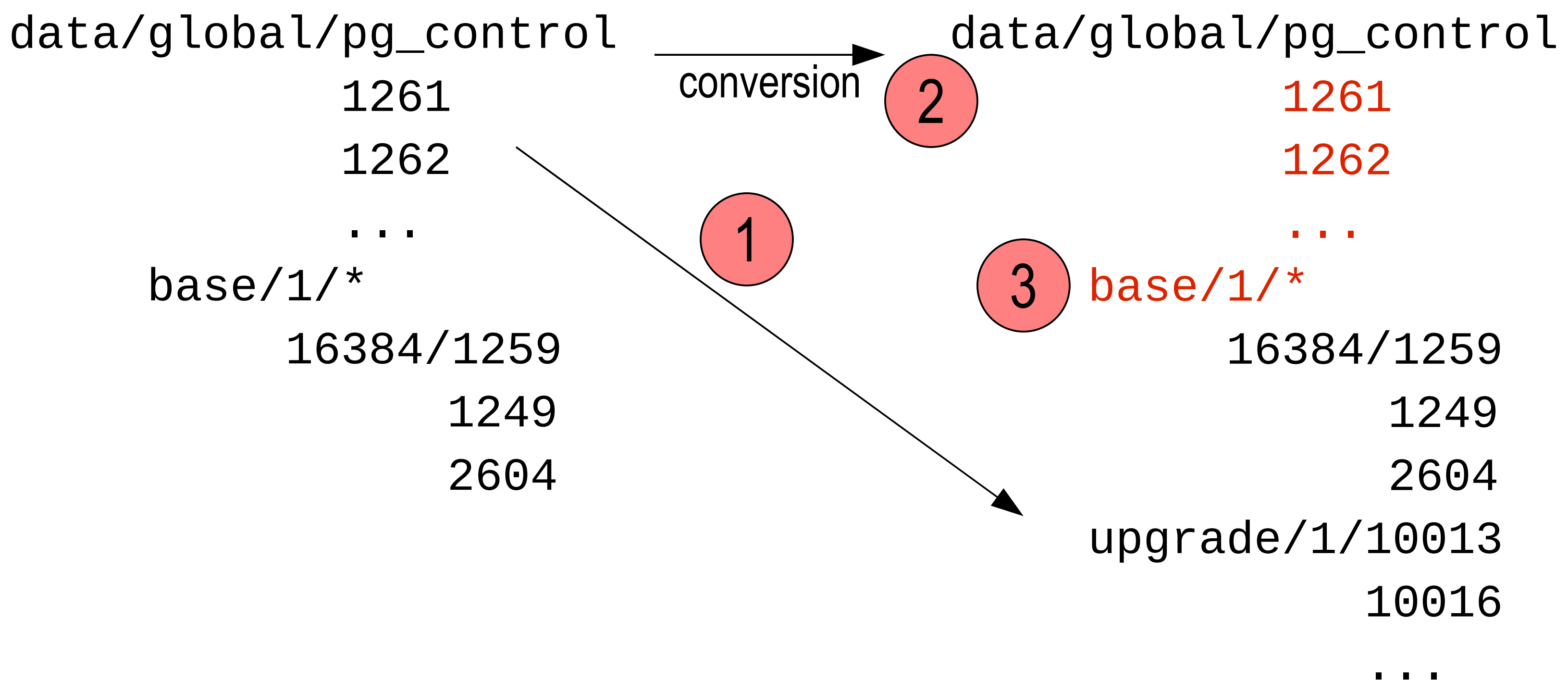


```
data/global/pg_control
              1261
              1262
              ...
     base/1/*
            16384/1259
                  1249
                  2604
```

```
data/global/pg_control


     base/1/<empty>
            16384/1259
                  1249
                  2604
     upgrade/1/10013
               10016
               ...
```

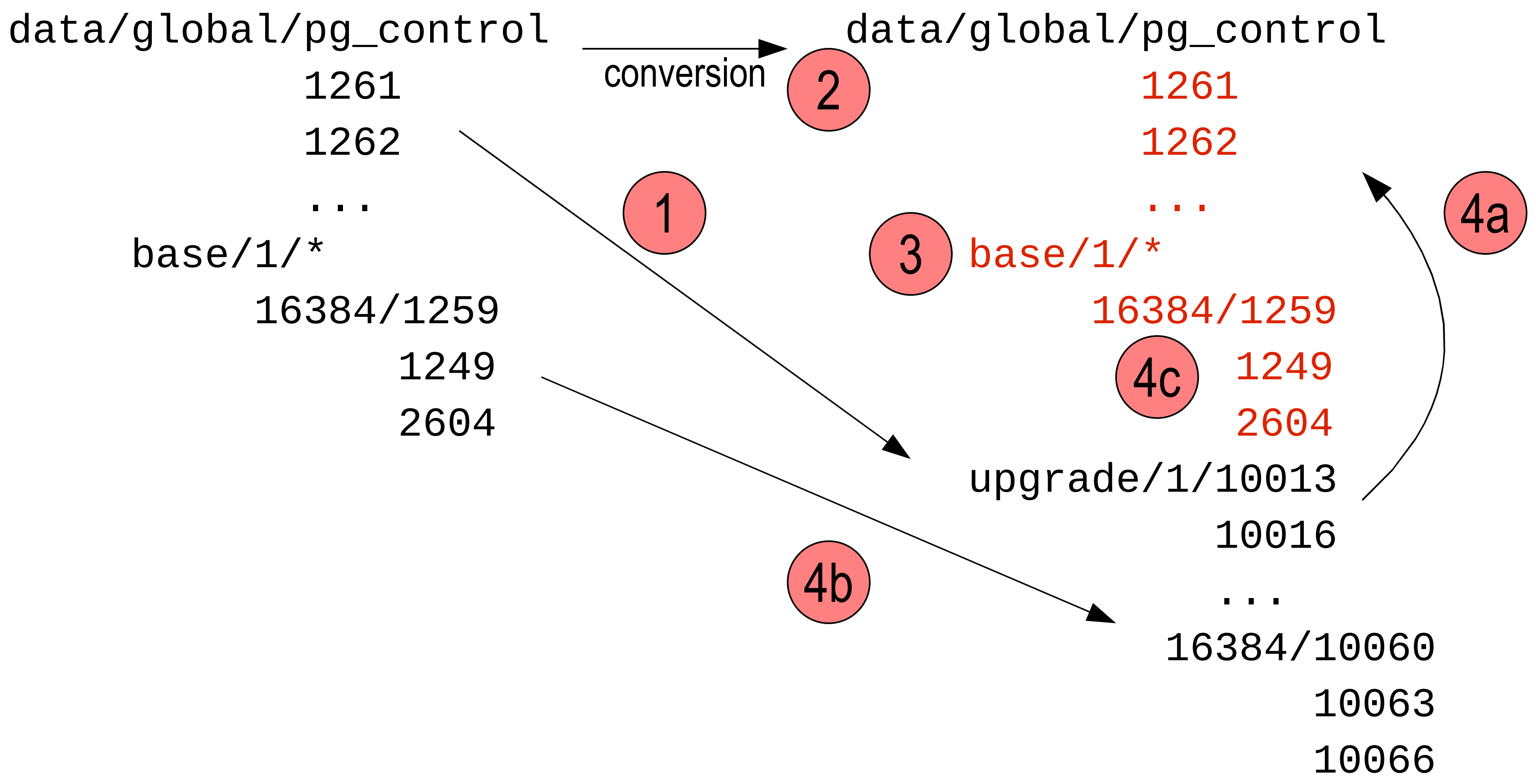
pg_upgrade – workflow (3)



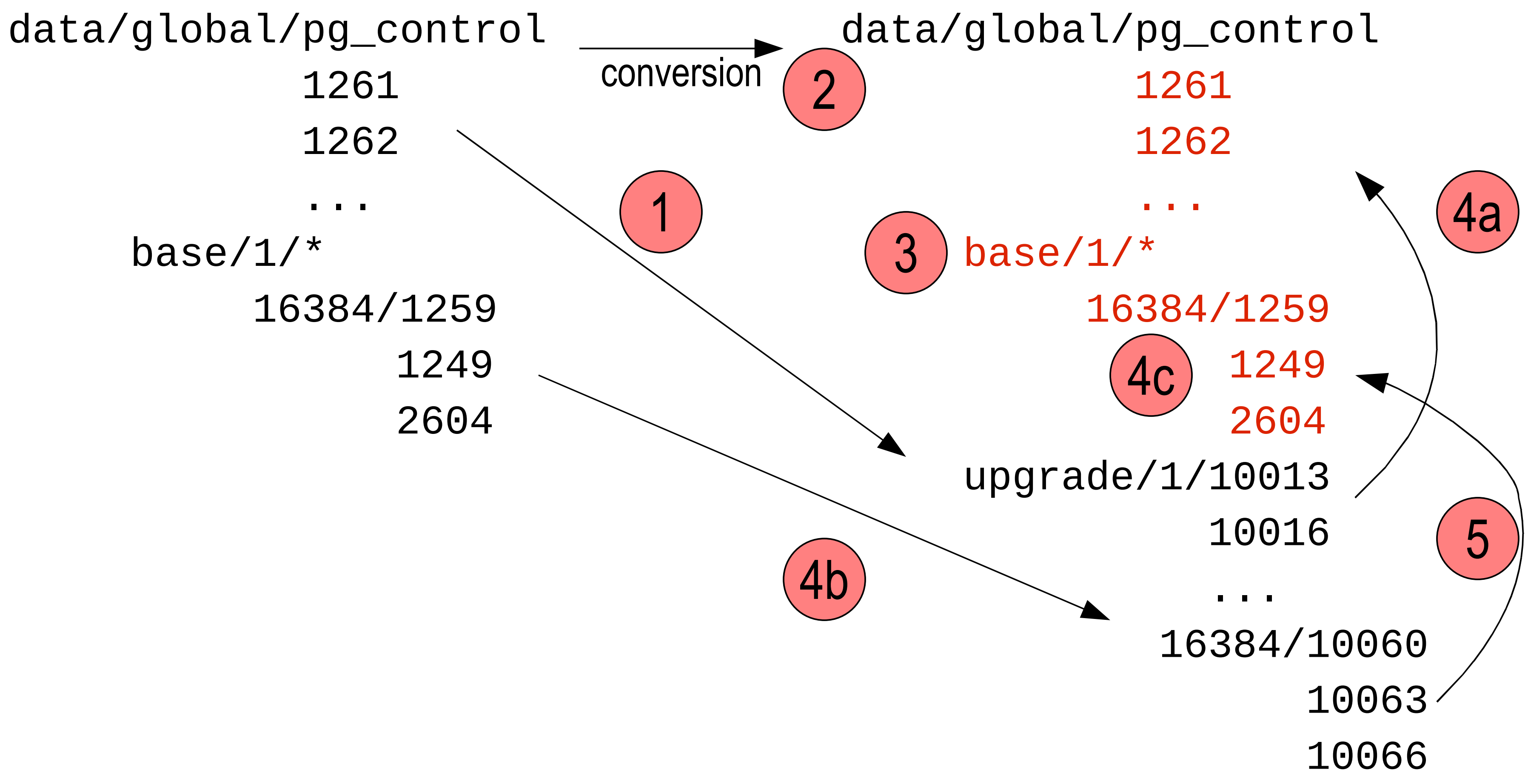
```
data/global/pg_control
            1261
            1262
            ...
     base/1/*
          16384/1259
                1249
                2604
```

```
data/global/pg_control
            1261
            1262
            ...
     base/1/*
          16384/1259
                1249
                2604
     upgrade/1/10013
               10016
               ...
```

pg_upgrade – workflow (4)



pg_upgrade – workflow (5)



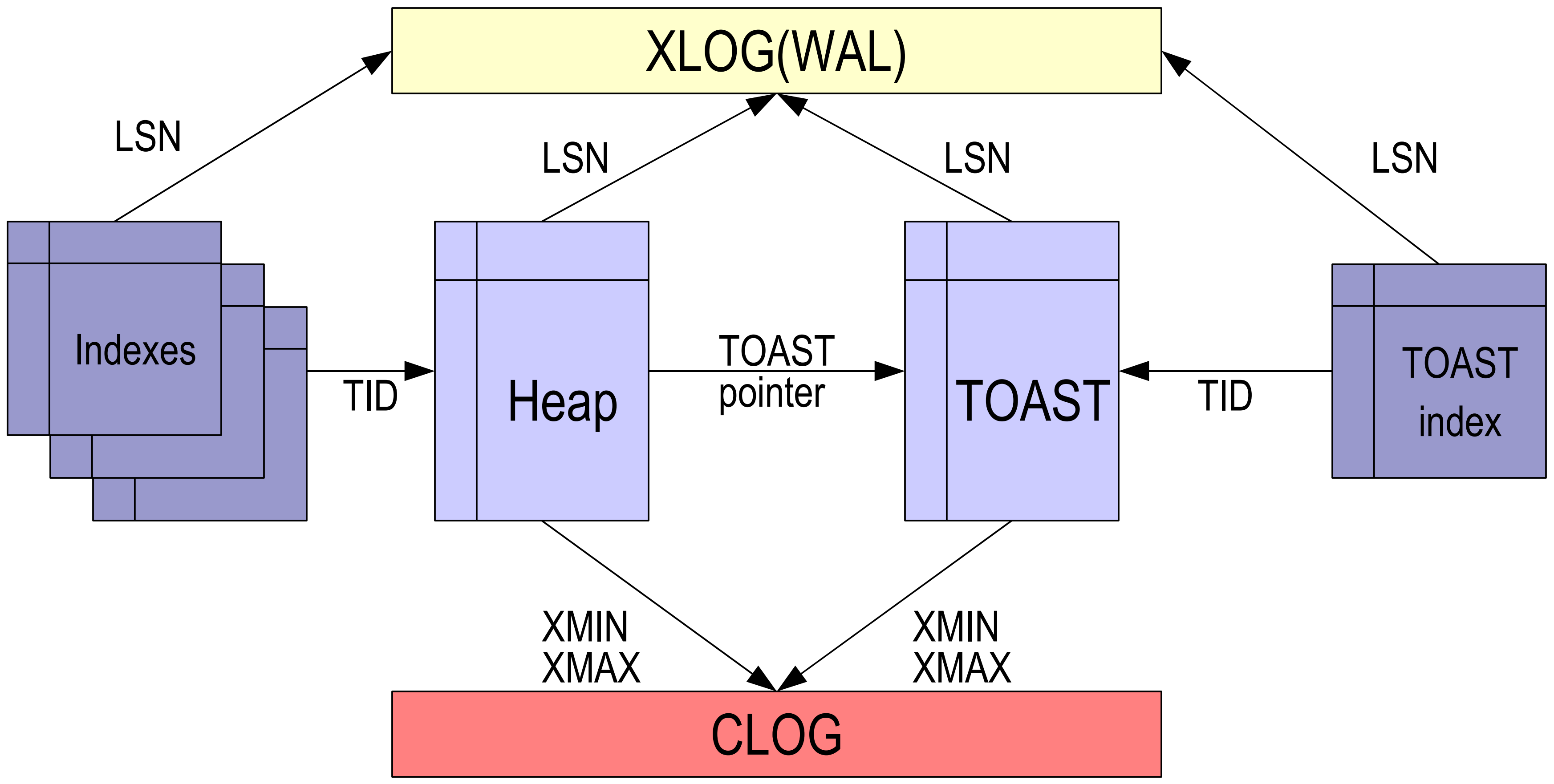
How it should work

```
pg_ctl -D /var/postgres upgrade

pg_upgrade /var/postgres
```

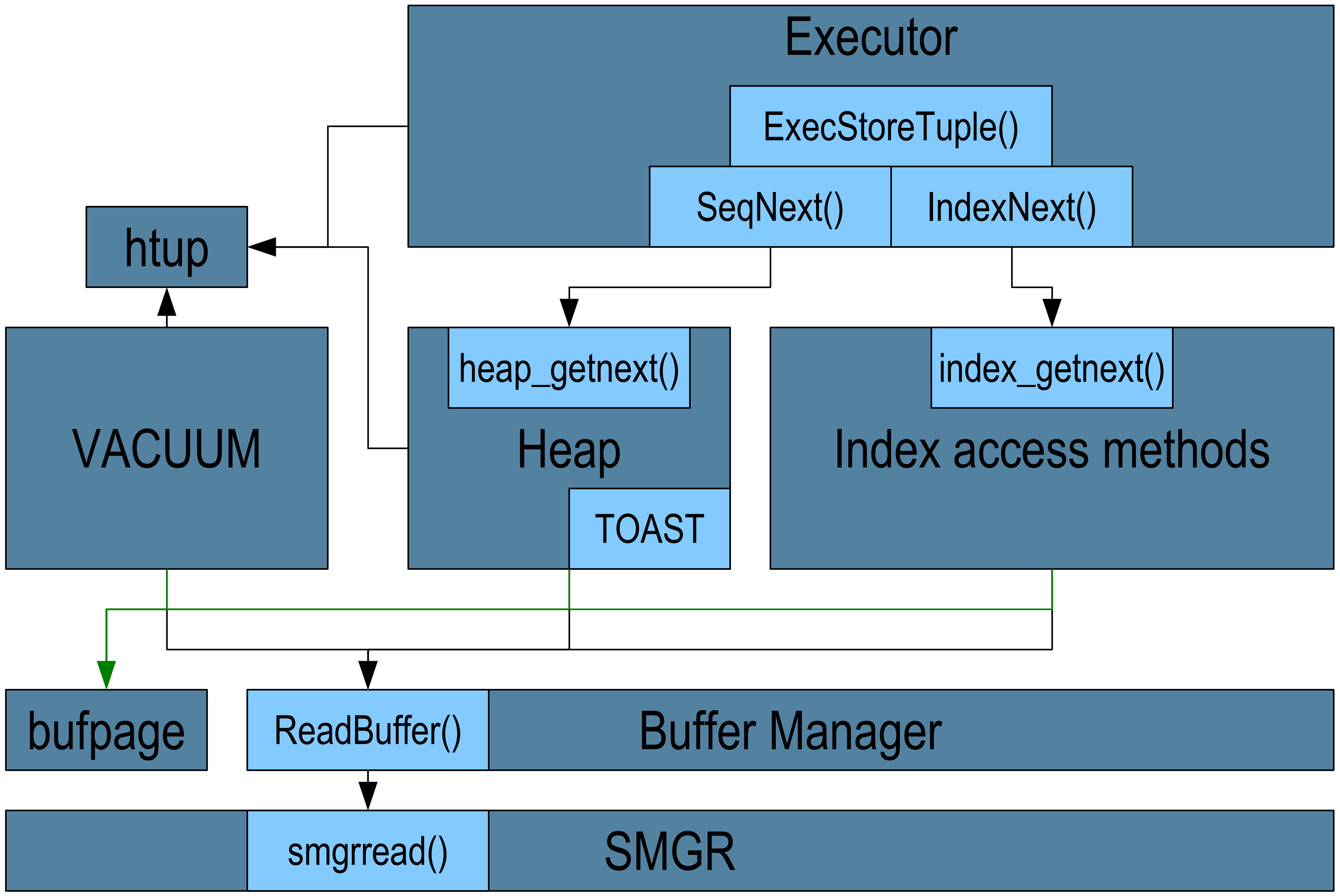
Storage upgrade

Storage dependency graph*

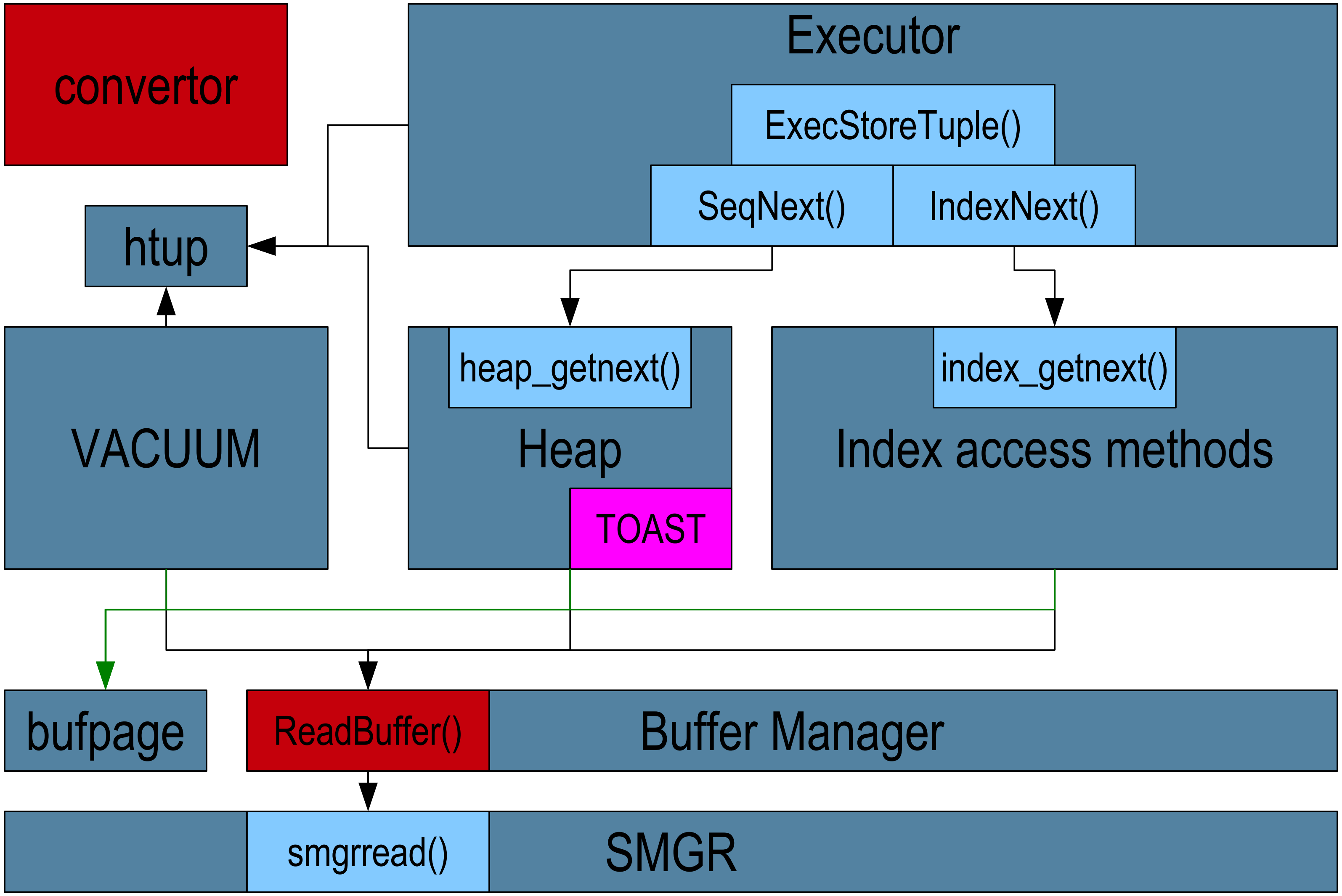


*without forks (free space map, dead space map)

Storage schema



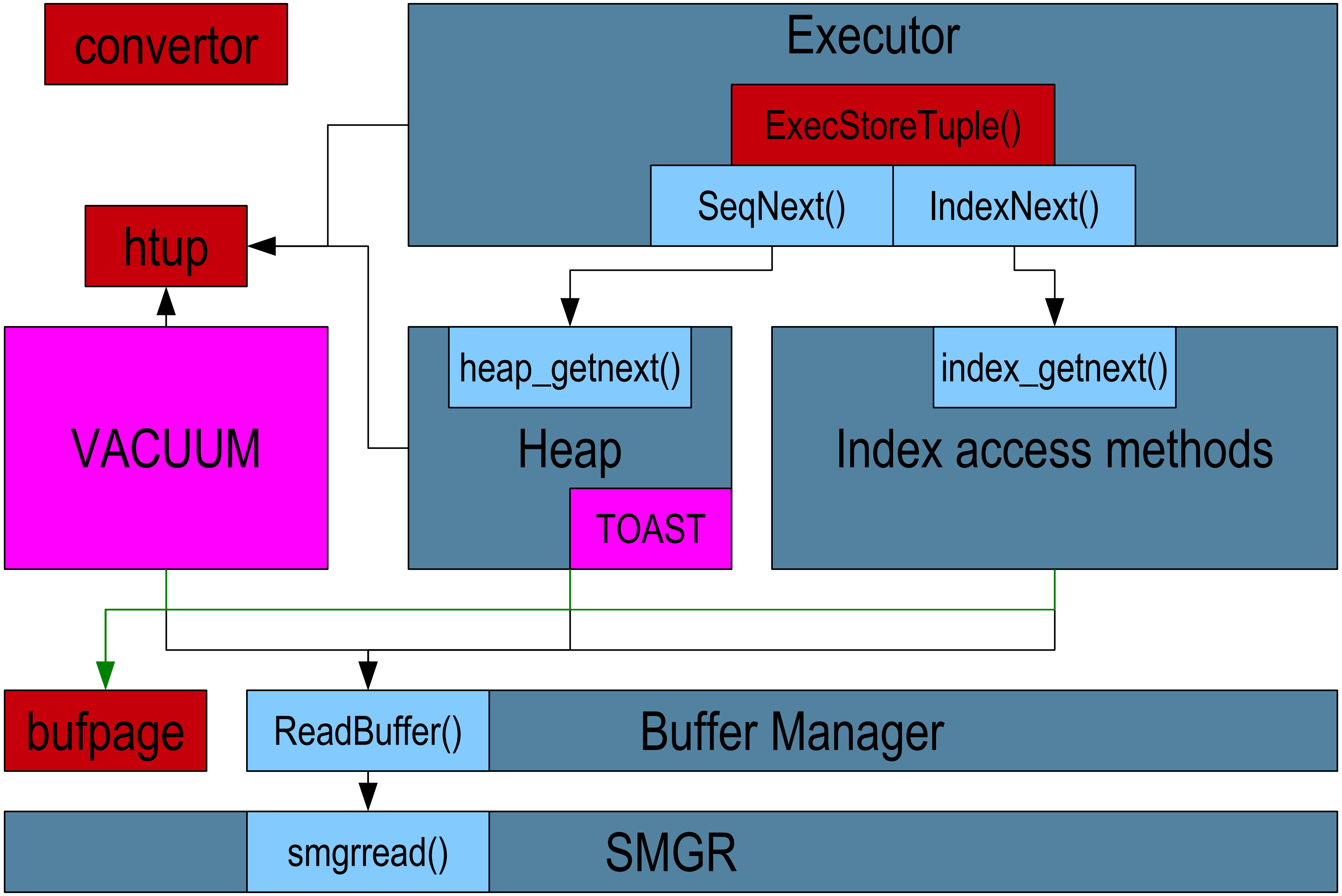
Convert on read



Convert on read - issues

- Unpredictable response time
- How to reserve space on page for conversion
- How to convert TOASTed data
- Does not solve AM incompatible changes (e.g. new version of HASH index)
- How to solve problem with too many page versions

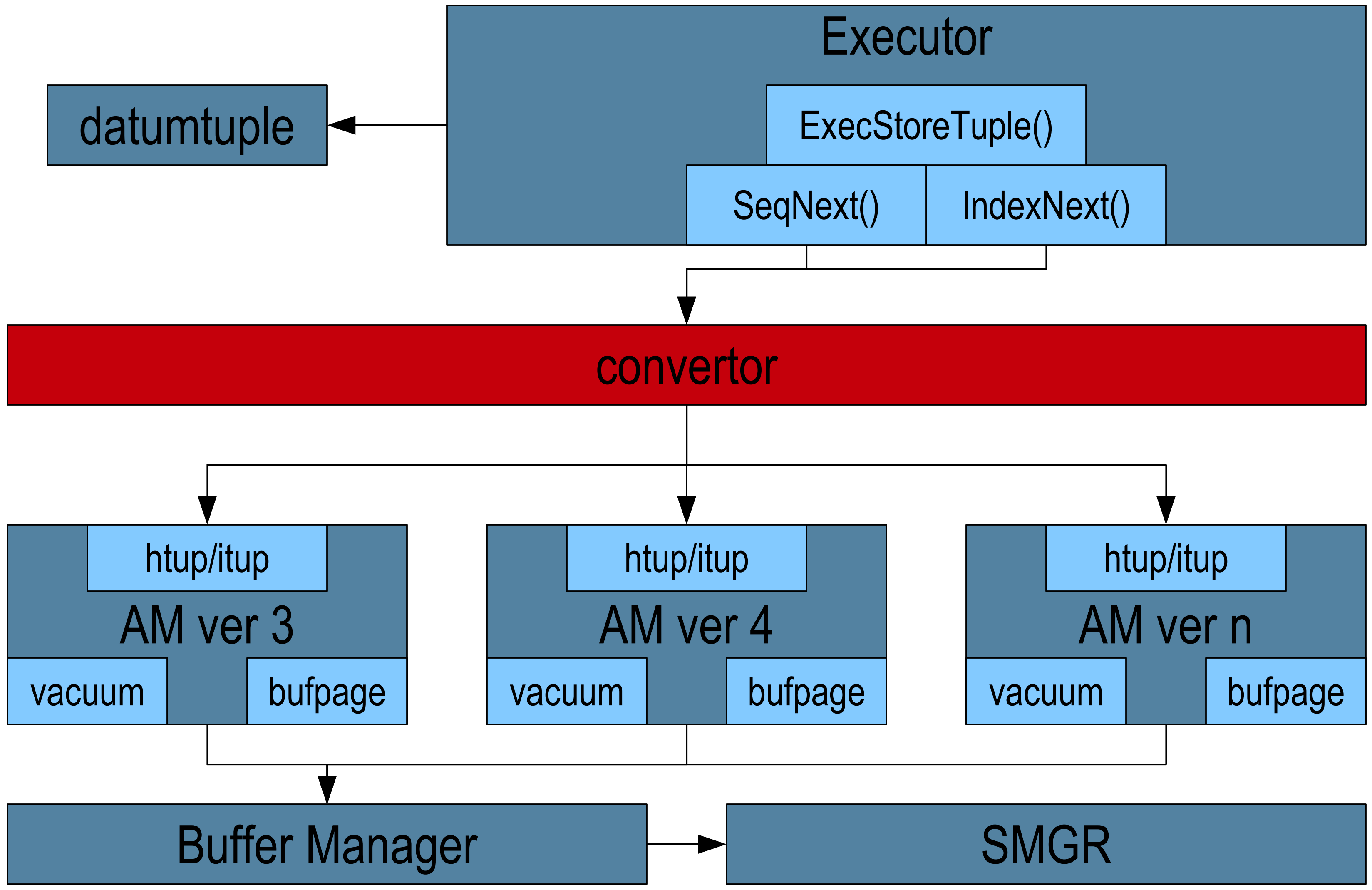
Read old, write new



Read old, write new - issues

- General performance drop (~1%)
- Huge code refactoring
- Does not solve AM incompatible changes (e.g. new version of HASH index)
- How to solve problem with too many page versions

Multi AM – modularized AM



Multi AM – modularized AM

- Huge source code repository refactoring including building
- How to deal with different cost for different AM version

What next

Issue 1

Everybody will use it
and
nobody will want to do pg_dump again

Issue 2

Never been finished
(ongoing development)

Open questions

- What methods
- Who will implemented changes
- When changes will be implemented
- Which version will be supported
- How to test it

References

http://pgfoundry.org/projects/pg-migrator/

http://src.opensolaris.org/source/xref/sfw/usr/src/cmd/postgres/postgresql-upgrade/

http://wiki.postgresql.org/wiki/In-place_upgrade

PostgreSQL upgrade yesterday, today, tomorrow

Zdeněk Kotala
zdenek.kotala@sun.com