

---

# New Free Space Map and Visibility Map

Heikki Linnakangas

# Part 1: Free Space Map

---

- Free Space Map stores information about free space in a relation
- Used by INSERTs and UPDATEs

# Pre-8.4 Free Space Map

---

- Used a shared memory block
  - Size was fixed at startup
- `max_fsm_pages`, `max_fsm_relations`
- Saved to `global/pg_fsm.cache` at shutdown
  - Lost on crash or PITR
- Protected by a single lock (`FreeSpaceLock`)

# New Free Space Map

---

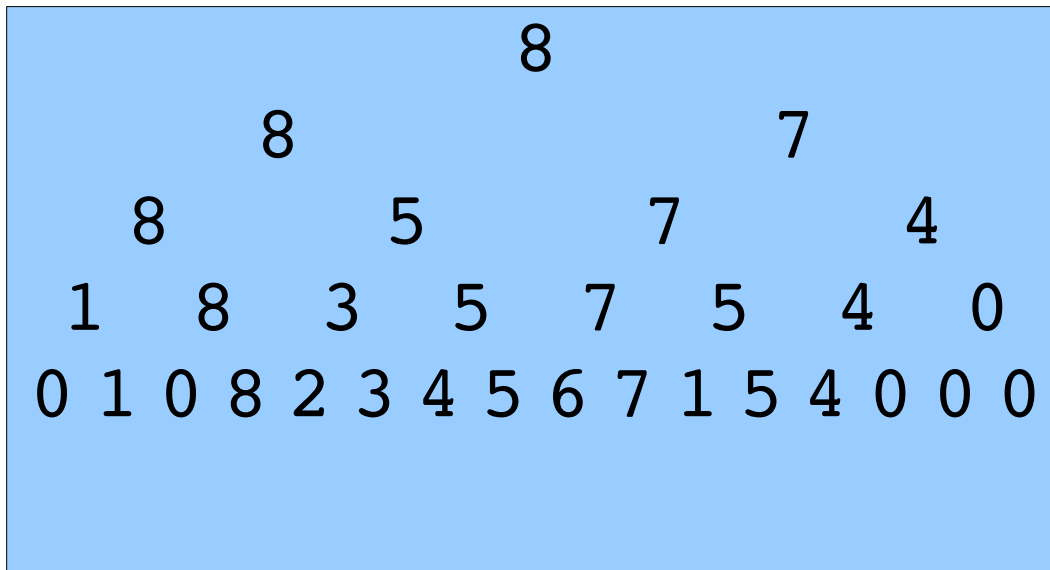
- Completely rewritten
- Stored on-disk in normal 8k pages
  - Cached in shared\_buffers
  - Goodbye max\_fsm\_pages and max\_fsm\_relations

```
$ ll data/base/11562/56553*  
-rw----- 1 hlinnaka hlinnaka 3219456 2009-02-08 00:45 56553  
-rw----- 1 hlinnaka hlinnaka   24576 2009-02-08 00:45 56553_fsm  
-rw----- 1 hlinnaka hlinnaka    8192 2009-02-08 00:45 56553_vm
```

# FSM page structure

---

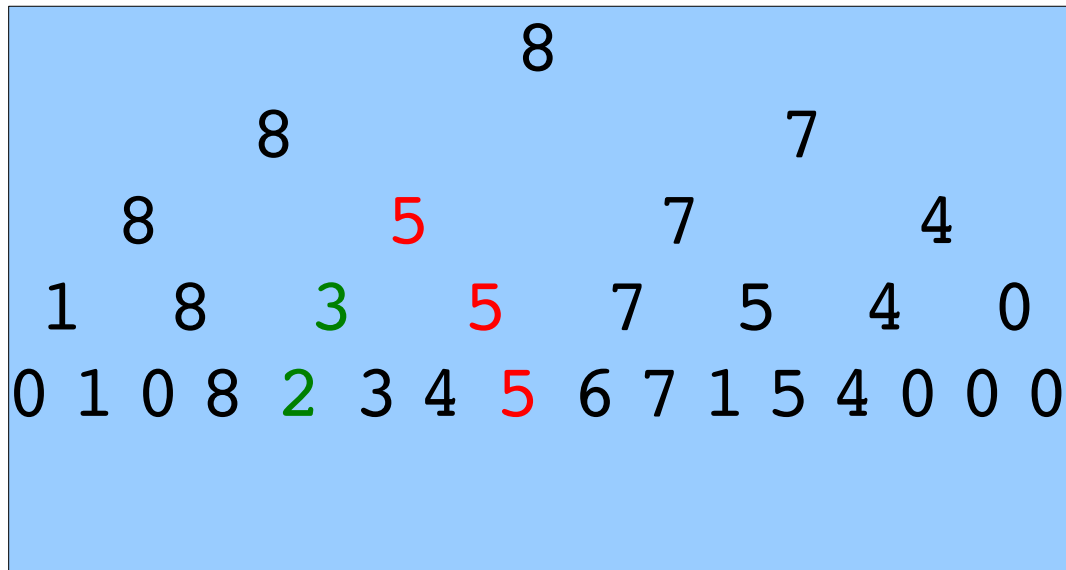
- Binary tree



# FSM page structure

---

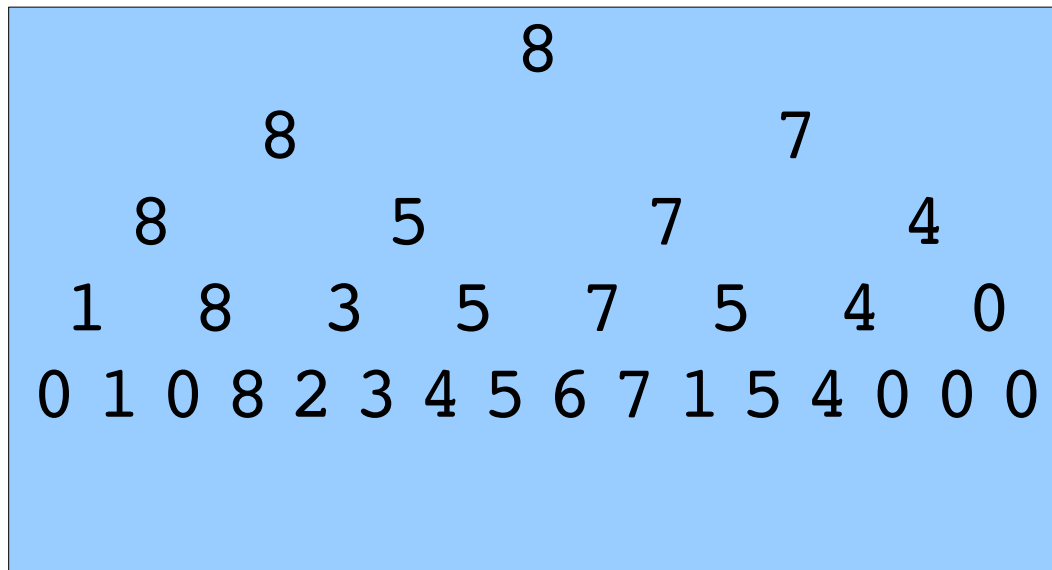
- Search for a page with 5 units of free space
  - Start from the bottom, at the green node
  - Climb up until we hit a node  $\geq 5$
  - Climb down, following the path with  $\geq 5$



# FSM page structure

---

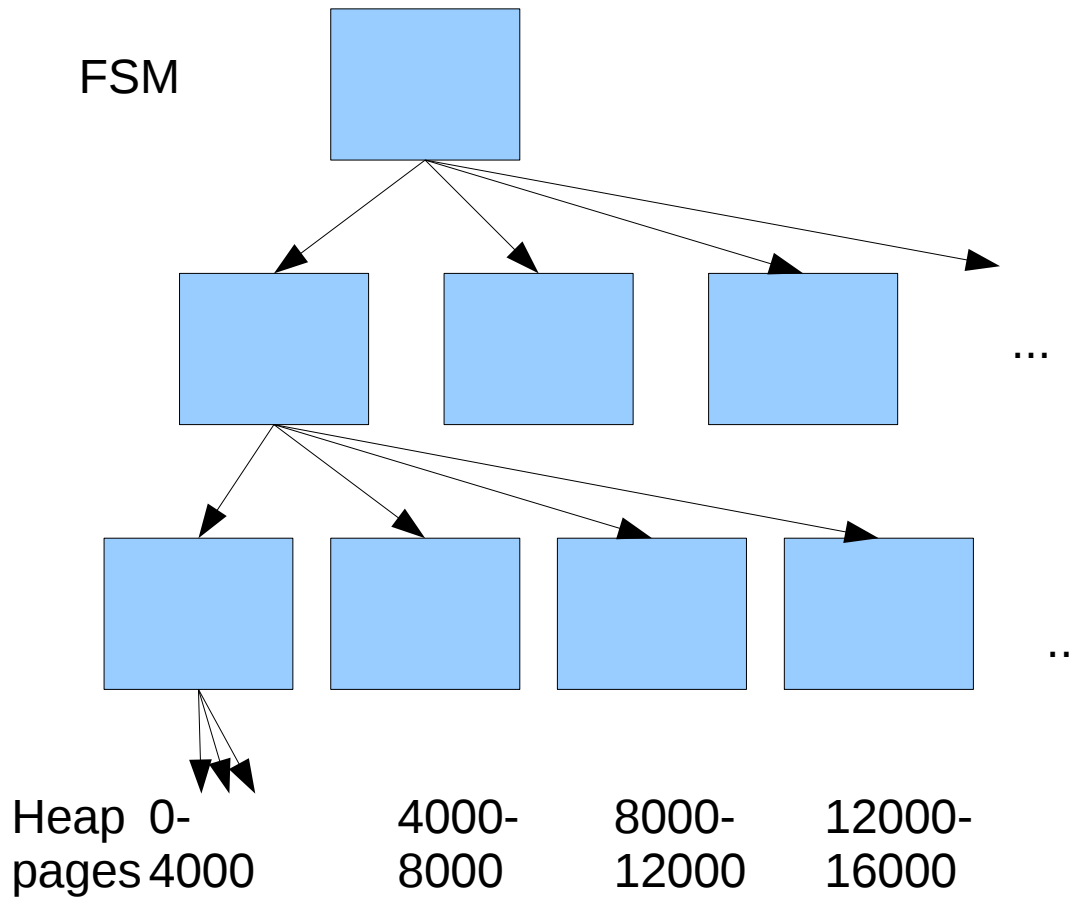
- Auto-repairing
  - Upper levels can be reconstructed from bottom
  - No pointers



# FSM high-level structure

---

- Tree of three levels





# FSM operations

---

- Search
  - Can prefer pages close to given page
- Update
  - Allows retail updates

## Part 2: Visibility Map

---

- A bitmap of heap pages
- 1 means “all tuples on page are visible to all transactions”
- Bits are set in VACUUM
- Cleared at INSERT/UPDATE/DELETE

# Partial VACUUM

---

- VACUUM can now skip pages that are already marked in visibility map
- Still needs to scan all indexes

# Partial VACUUM Example

---

```
postgres=# CREATE TABLE foo (id int4);
```

```
CREATE TABLE
```

```
postgres=# INSERT INTO foo SELECT  
generate_series(1,100000);
```

```
INSERT 0 100000
```

```
postgres=# DELETE FROM foo WHERE id <  
50000;
```

```
DELETE 49999
```

# Partial VACUUM Example (cont)

---

```
postgres=# VACUUM VERBOSE foo;
INFO:   vacuuming "public.foo"
INFO:   "foo": removed 49999 row versions in 197 pages
INFO:   "foo": found 49999 removable, 50001 nonremovable
        row versions in 393 out of 393 pages
DETAIL:  0 dead row versions cannot be removed yet.
There were 0 unused item pointers.
0 pages are entirely empty.
CPU 0.00s/0.05u sec elapsed 0.05 sec.
VACUUM
```

# Partial VACUUM Example (cont)

---

```
postgres=# VACUUM VERBOSE foo;
INFO:  vacuuming "public.foo"
INFO:  "foo": found 0 removable, 8141 nonremovable row
       versions in 228 out of 393 pages
DETAIL:  0 dead row versions cannot be removed yet.
There were 49999 unused item pointers.
0 pages are entirely empty.
CPU 0.00s/0.00u sec elapsed 0.00 sec.
VACUUM
```

# Partial VACUUM Example (cont)

---

```
postgres=# VACUUM VERBOSE foo;
INFO:  vacuuming "public.foo"
INFO:  "foo": found 0 removable, 0 nonremovable row
       versions in 31 out of 393 pages
DETAIL:  0 dead row versions cannot be removed yet.
There were 7905 unused item pointers.
0 pages are entirely empty.
CPU 0.00s/0.00u sec elapsed 0.00 sec.
VACUUM
```

# Partial VACUUM and Freezing

---

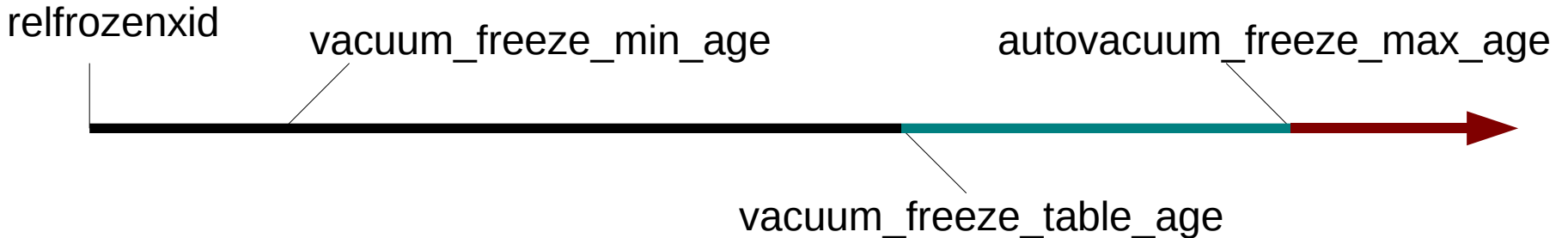
- A partial VACUUM can't advance `relfrozenxid`
- Whole-table scanning VACUUMs are still needed to avoid transaction ID wraparound
- New configuration variable:  
`vacuum_freeze_table_age`



# Configuring Freezing

---

- `autovacuum_freeze_max_age` = 200 M (200 M)
- `vacuum_freeze_min_age` = 50 M (100 M)
- `vacuum_freeze_table_age` = 150 M



# Questions?

---

- Future Works
  - Partial freezing
  - Index-only scans
- Let me know how it works