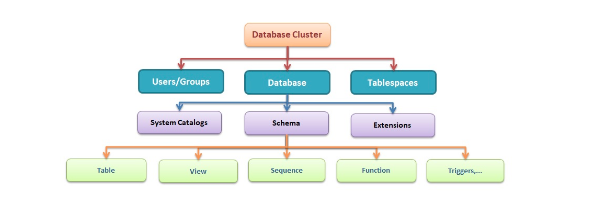
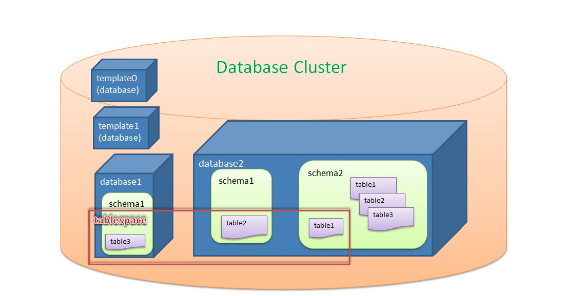
**Lưu trữ dữ liệu trên PostgreSQL**

**1. Giới thiệu**

Trong postgreSQL, table là nơi lưu trữ dữ liệu. Table được lưu trong đơn vị lớn hơn là database, các database lại được nằm trong một đơn vị lớn hơn là database cluster. Database cluster là đơn vị lưu trữ lớn nhất của một PostgreSQL database server. Ta có thể quan sát rõ hơn ở hình bên dưới.



Mô hình phân cấp trong PostgreSQL.

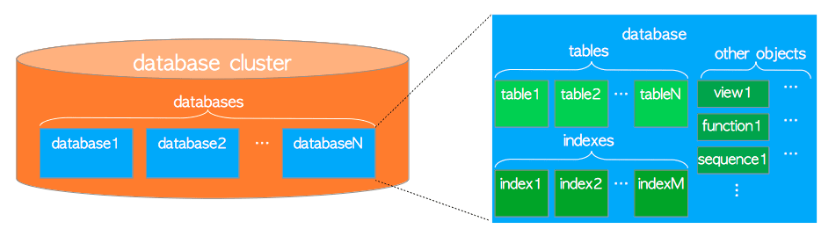


Các đối tượng lưu trữ trong PostgreSQL

Nhìn mình họa, ta thấy table nằm dưới Schema và tablespace. Schema là đơn vị tập hợp các bảng theo hình thức logic. Tablespace là đơn vị tập hợp các bảng và indexes theo hình thức vật lý.

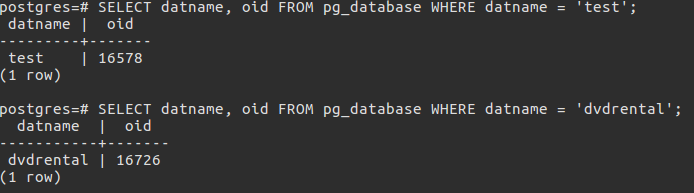
**2. Cấu trúc logic của database cluster**

Một postgreSQL server chạy trên một single host và quản lý một single database cluster. Một database cluster là một tập hợp nhiều database quản lý bởi một PostgreSQL server. Chúng ta có thể quan sát hình minh họa phía bên dưới.



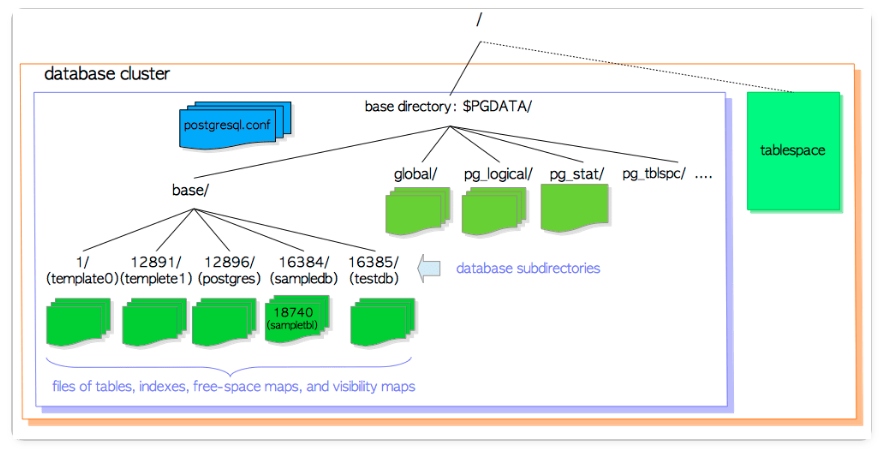
Cấu trúc vật lý của một database cluster

Tất cả các đối tượng database trong PostgreSQL được quản lý bởi định danh OIDs (object identifiers). OIDs là một số nguyên 4-byte. Chúng ta có thể tìm định danh OIDs của database và heap table bằng câu query sau:



**3. Cấu trúc vật lý của database cluster**

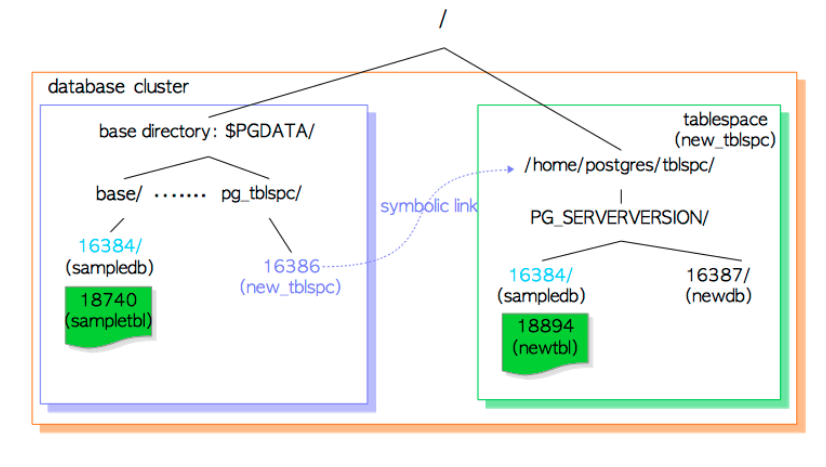
Một database cluster cơ bản là một thư mục gọi là base directory. Nó chứa những thư mục con và rất nhiều tập tin. Để tạo một database cluster, ta sử dụng lệnh initdb (https://www.postgresql.org/docs/current/app-initdb.html). Sau đó, một base directory sẽ được tạo ra bên dưới thư mục chỉ định. Đường dẫn cho base directory thường được đặt trong biến môi trường PGDATA. Ta quan sát một ví dụ về cấu trúc vật lý của một database cluster ở hình bên dưới.



Quan sát ảnh minh họa, ta thấy thư mục con base chứa nhiều thư mục con. Mỗi thư mục con là một database. Table và index được lưu ở những file nằm trong thư mục database tương ứng. Ngoài ra, có một số thư mục con chứa dữ liệu cụ thể và tệp cấu hình (tham khảo: <https://www.postgresql.org/docs/current/storage-file-layout.html>). PostgreSQL từ phiên bản 8.0 hỗ trợ thêm tablespace.

**4. Tablespaces**

Tablespace trong postgreSQL là một vùng dữ liệu bổ sung bên ngoài base directory. Thông thường dữ liệu vật lý được lưu trữ tại thư mục con được đề cập ở trên phần 1.3. Nhưng postgreSQL có một phương pháp lưu trữ dữ liệu ngoài phân vùng này, nhờ sử dụng chức năng tablespace. Khi tạo một tablespace tức là ta đã tạo ra một vùng lưu trữ dữ liệu mới độc lập với base directory. Chúng ta có thể quan sát rõ hơn về bố cục bên trong tablespace và mối quan hệ với vùng dữ liệu chính ở hình bên dưới.



Chúng ta có thể tạo ra một tablespace bằng lệnh CREATE TABLESPACE (<https://www.postgresql.org/docs/current/sql-createtablespace.html>). Sau đó, một tablespace sẽ được tạo ra ở thư mục chỉ định và bên dưới sẽ có một thư mục con hiển thị version chi tiết (PG\_SERVERVERSION). Tên của thư mục này theo format là PG\_'version postgreSQL'\_'Catalogue version number'. Như hình minh họa, ta tạo tablespace 'new\_tblspc' tại thư mục '/home/postgres/tblspc' có oid là 16386. PostgresSQL sẽ tạo một symbolic link có tên giống với giá trị oid của tablespace nằm dưới pg\_tblspc của database cluster. Nó trỏ tới thư mục chỉ định lưu trữ tablespace. Nếu tạo một table cho một database ở base directory thì bên tablespace cũng sẽ tạo ra một table tương ứng như vậy.

**5. Cấu trúc database subdirectory**

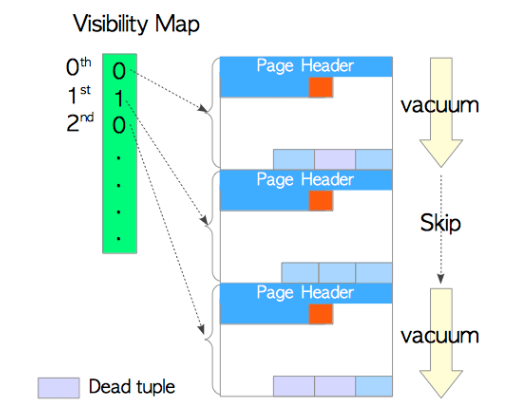
Một database là một thư mục con dưới thư mục con base, và tên thư mục database này giống với oid tương ứng. Trong thư mục database, chứa những file index và table có kích thước dưới 1GB. Chúng được quản lý bởi một biến là relfilenode. Tên những file này được đặt theo giá trị của biến relfilenode.

Khi kích thước file vượt quá 1GB, PostgreSQL sẽ tạo ra một file mới có tên là relfilenode.1. Nếu file mới đầy, file tiếp theo được tạo ra có tên là relfilenode.2 và tương tự cho các file tiếp theo. Ví dụ: relfilenode = 12345 thì tên các file có thể là 12345, 12345.1, 12345.2,…

## Ngoài ra, tương ứng với mỗi file table hay index liên kết đến 2 file với hậu tố là ‘\_fsm’ và ‘\_vm’:

## FSM là free space map lưu trữ không gian còn trống trong các page. Khi insert một heap tuple hay index tuple, PostgreSQL sử dụng FSM để chọn page phù hợp để insert.

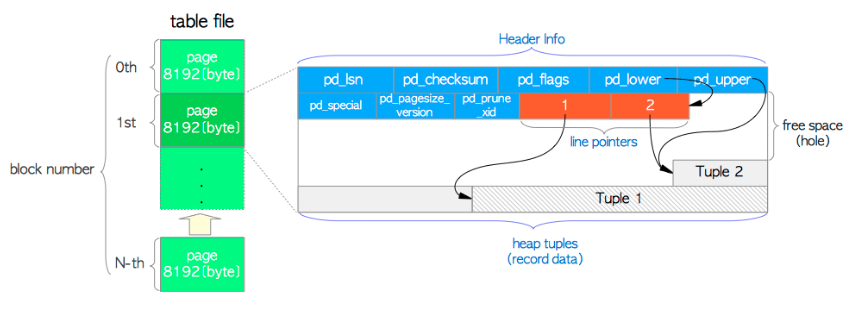
## VM là visibility map chứa khả năng hiển thị của từng page. Khả năng hiển thị của page được xác định xem mỗi page có dead tuple hay không. Dựa vào file này, VACUUM biết được file nào nên thực hiện dọn dẹp.



Quan sát hình minh họa phía trên, ta thấy page 1 có giá trị là 1, tức là không có dead tuple nên Vacuum có thể bỏ qua không cần dọn dẹp ở page này. Page 0 và page 2 có giá trị là 0, tức là có chứa dead tuple nên vacuum cần thực hiện dọn dẹp. Nhờ vào đó, tiến trình vacuum được giảm bớt chi phí.

**6. Cấu trúc Heap Table File**

Bên trong data file, nó được chia thành các page (hay block) có độ dài cố định, mặc định là 8192 byte (8 KB). Các page trong mỗi file được đánh số bắt đầu từ 0, các số này được gọi là block number. Nếu file bị đầy, PostgreSQL thêm một page mới vào cuối file để tăng kích thước file.



**Bố cục của page trong một heap table file**

Quan sát ảnh minh họa bên trên, một page trong table chứa 3 loại sau:

1. heap tuple(s): Một heap tuple là một record data chứa dữ liệu. Chúng được sắp xếp từ dưới lên.

2. line pointer(s)- Một line pointer có độ dài 4 byte và giữ một con trỏ đến một heap tuple. Nó còn được gọi là item pointer. Những line pointer tổ chức thành mảng, nó đóng vai trò là index cho những tuple. Mỗi index được đánh số tuần tự từ 1 và được gọi là offset number. Khi một tuple mới được thêm vào page, một line pointer mới cũng được thêm vào mảng để trỏ đến tuple mới được thêm vào.

3. header data có độ dài 24 byte và chứa thông tin chung về page. Header data được xác định bởi cấu trúc PageHeaderData nằm ở đầu mỗi page. Các biến chính trong cấu trúc này là:

+ pd\_lsn là số nguyên 8 byte, sử dụng cơ chế WAL(Write-Ahead Logging). Nó lưu trữ LSN (Log Sequence Number là vị trí của record được ghi trên transction log) của XLOG record (XLOG record ghi lại tất cả các sửa đổi của page) được viết bởi thay đổi mới nhất trong page.

+ pd\_checksum là biến chứa checksum của giá trị trong page.

+ pd\_lower là offset chỉ đến vị trí cuối line pointer.

+ pd\_upper là offset chỉ đến vị trí đầu của heap tuple mới nhất.

+ pd\_special là biến giành cho index. Trong page chứa table, nó chỉ đến vị trí cuối page. Trong page chứa index, nó chỉ đến vị trí bắt đầu của một không gian đặc biệt. Không gian đặc biệt này là nơi lưu trữ data được giữ bởi index and chứa dữ liệu cụ thể theo lọai index như B-tree, GiST, GiN,…

Để xác định một tuple trong table, PostgreSQl dùng tuple identifier (TID). Một tid bao gồm một cặp giá trị: block number của page chứa tuple và offset number của line pointer chỉ đến tuple.

**7. Tuple**

Một thiết kế cốt lõi trong PostgreSQL là immutable row table. Immutable row này được Postgres gọi là "tuple". Mỗi tuple là duy nhất được xác định bằng ctid. ctid về mặt khái niệm đại diện cho vị trí trên disk (physical disk offset) của tuple. Nhiều tuple có khả năng mô tả một hàng trên table.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| xmin | xmax | cid | ctid | User data |

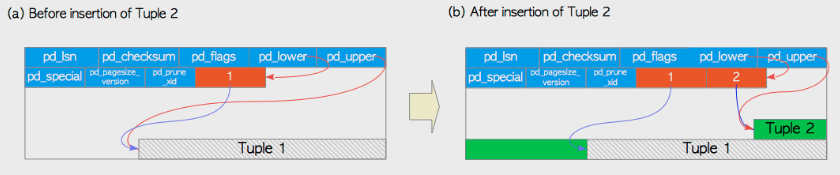
+ xmin chứa txid (transaction id) của transaction insert tuple.

+ xmax chứa txid của transaction delete hoặc update của tuple. Nếu tuple không có delete hay update, t\_xmax có giá trị là 0 .

+ cid chứa command id, có nghĩa là có bao nhiêu tuple thực thi trước tuple hiện tại trong cùng một transaction (bắt đầu bằng 0).

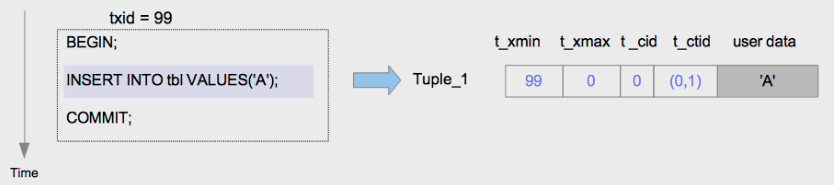
+ ctid chứa ctid của tuple mới nhất mô tả hàng đó.

**8. Phương thức ghi heap tuple**



Chúng ta quan sát hình minh họa ở trên, giả sử table hiện tại chỉ chứa một page và page chứa một heap tuple. pd\_lower của page đó chỉ đến vị trí cuối line pointer và pd\_upper chỉ đến vị trí của tuple 1. Khi tuple 2 được thêm vào, nó sẽ nằm ở sau tuple 1. Line pointer của tuple 2 được thêm vào kế tiếp tuple 1, và chỉ đến vị trí của tuple 2. Lúc này, vị trí cuối của line pointer thay đổi nên pd\_lower sẽ thay đổi theo, và pd\_upper sẽ chỉ đến tuple 2. Theo logic, nó sẽ thực hiện như sau:

- Khi insert



Giả sử insert Tuple\_1 có transaction id là txid = 99.

Tuple\_1:

xmin = 99 => txid = 99.

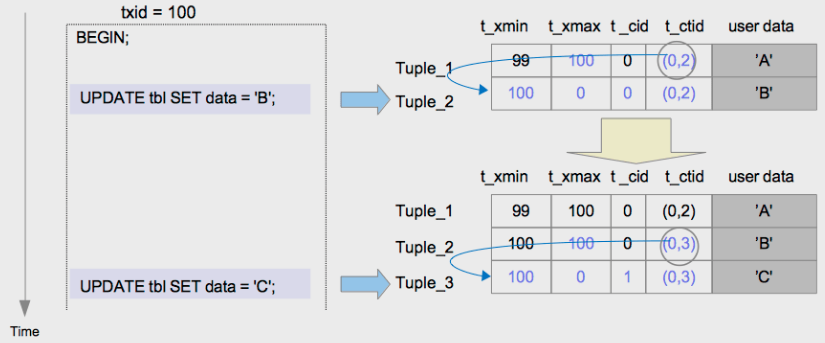
max = 0 => tuple chưa có update hay delete

cid = 0 => tuple này là tuple đầu tiên được insert vào.

ctid = (0,1) => tuple\_1 là tuple mới nhất mô tả hàng đó.

- Khi Update

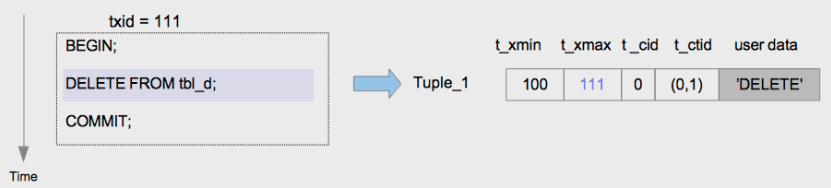
Ví dụ thực hiện 2 lần update dữ liệu trong một transaction có transaction id là txid = 100.



Khi chạy lệnh update đầu tiên, theo logic thì tuple\_1 bị delete và tuple\_2 được insert vào. Sau đó, ctid của tuple\_1 sẽ được viết lại chỉ đến tuple\_2.

Khi chạy lệnh update thứ hai, theo logic thì tuple\_2 bị delete và tuple\_3 được insert vào. Sau đó, ctid của tuple\_2 sẽ được viết lại chỉ đến tuple\_3.

- Khi delete



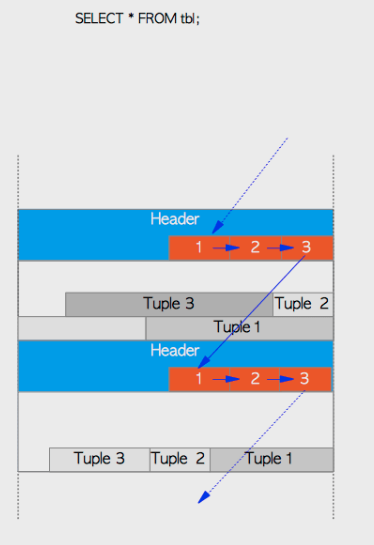
Giả sử detete Tuple\_1 có transaction id là txid = 111.

Nếu transaction commit thì tuple\_1 là dead tuple. Ngược lại, tuple\_2 và tuple\_3 là dead tuple.

**9. Phương thức đọc heap tuple**

Postgres có 2 cách thục hiện là quét tuần tự và quét B-tree index:

- Đọc tuần tự: Tất cả các tuple tất cả các page được đọc tuần tự bằng cách quét qua tất cả line pointer trong mỗi page.



- Quét B-tree index: Một file index chứa một bộ index tuple, mỗi bộ bao gồm một index key and một tid trỏ đến heap tuple tương ứng.

