

### Performance Tuning # ##Set this up and we will look at plans

# Setup #

drop table t1;

create table t1(x int, y int);

insert into t1 values(1,NULL);

insert into t1 values(1,2);

insert into t1 values(2,2);

insert into t1 values(3,2);

insert into t1 values(4,1);

insert into t1 select \* from t1;

insert into t1 select \* from t1;

insert into t1 select \* from t1;

insert into t1 select \* from t1;

insert into t1 select \* from t1;

insert into t1 select \* from t1;

insert into t1 select \* from t1;

insert into t1 select \* from t1;

insert into t1 select \* from t1;

insert into t1 select \* from t1;

insert into t1 select \* from t1;

insert into t1 select \* from t1;

insert into t1 select \* from t1;

insert into t1 select \* from t1;

insert into t1 select \* from t1;

insert into t1 select \* from t1;

insert into t1 select \* from t1;

insert into t1 select \* from t1;

insert into t1 select \* from t1;

alter table t1 add column a char(500);

alter table t1 add column b char(500);

alter table t1 add column c char(500);

alter table t1 add column d char(500);

alter table t1 add column e char(500);

alter table t1 add column f char(500);

alter table t1 add column g char(500);

alter table t1 add column h char(500);

alter table t1 add column i char(500);

alter table t1 add column j char(500);

alter table t1 add column k char(500);

alter table t1 add column l char(500);

update t1 set a = 'some long string that should be 500 chars',

b = 'some long string that should be 500 chars',

c = 'some long string that should be 500 chars',

d = 'some long string that should be 500 chars',

e = 'some long string that should be 500 chars',

f = 'some long string that should be 500 chars';

update t1 set g = 'some long string that should be 500 chars',

h = 'some long string that should be 500 chars',

i = 'some long string that should be 500 chars',

j = 'some long string that should be 500 chars',

k = 'some long string that should be 500 chars',

l = 'some long string that should be 500 chars';

create index idx1 on t1(x);

#### Joins

create table opening\_pairs(

team\_name varchar(20),

player\_name varchar(20));

insert into opening\_pairs values('India','Rahul'),('India','Rohit'),

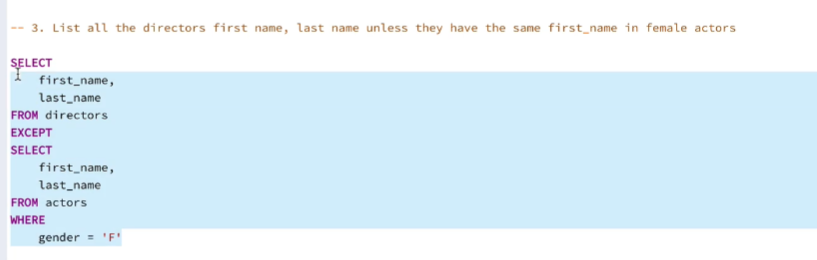
'Australia','Khwaja'),('Australia','Warner'),

('South Africa','De Kock'),('South Africa','Miller');

Output Expected

|  |  |  |
| --- | --- | --- |
| India | Rohit | Rahul |
| Australia | Warner | Khwaja |
| South Africa | Miller | De Kock |

**What is wrong in this query below**

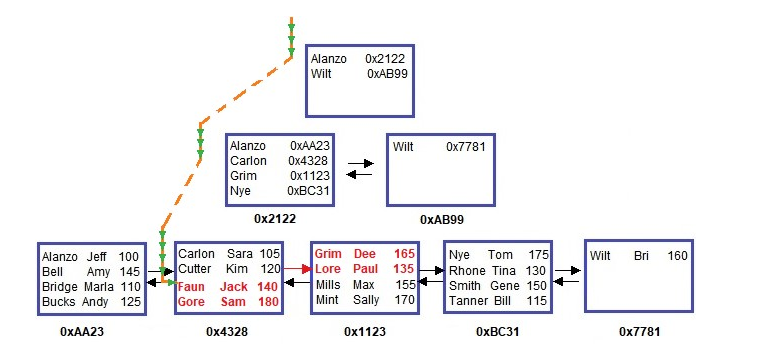


What does the pg\_dump utility do?

Write the query that compares 2 schemas? Let’s dissect the query.

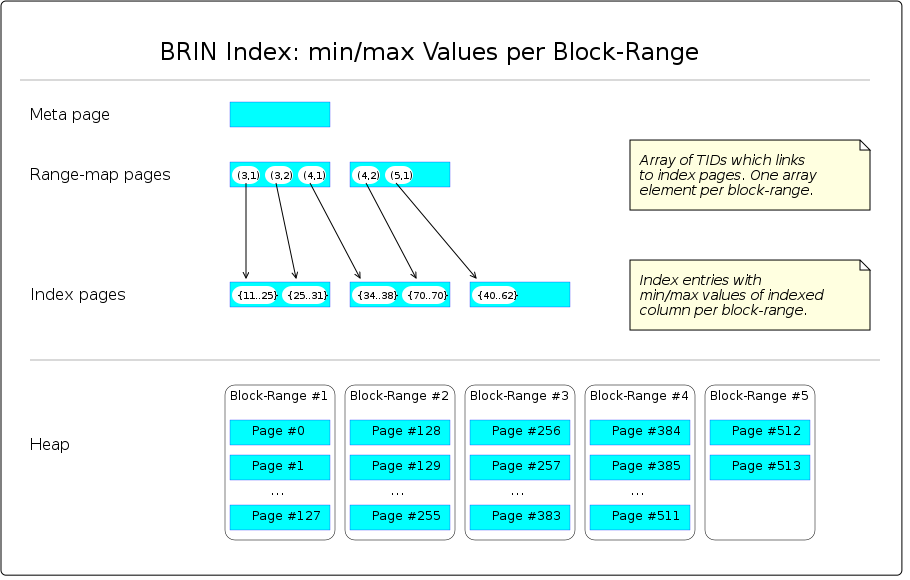
SELECT \* FROM pg\_compare\_schemas('schema1', 'schema2');

What is a clustered index?



When do you create a HASH index?

What is a partial index, how is it useful?



When a BRIN is created, the sequence of pages of the data file (heap) is virtually divided into slices, called *block-ranges*. E.g., if the file contains 600 pages, the first 128 belongs to block-range #1, the second 128 belongs to block-range #2, ... up to block-range #5, which contains the remaining number of pages. The default size for a block-range is 128 pages; it can be changed within the CREATE INDEX command. Next, all rows are scanned and the minimum and maximum values of the indexed column per block-range are saved. Please note that each min/max pair constitutes a numeric *value-range*.

The BRIN structure consists of those block-range numbers and their related value-ranges, e.g., block-range #1: min=11, max=25; block-range #2: min=25, max=31. Hence the name *Block Range Index*.

Ideally, the value-ranges don't overlap, but this is not necessary.

More Reading on GIN Index

<https://en.wikibooks.org/wiki/PostgreSQL/Index_GIN>