PostgreSQL SQL チューニング入門

～ Explaining Explain より～

2012 年 11 月 30 日

株式会社アシスト
田中 健一朗

アジェンダ

1.EXPLAIN とは

2. 表アクセスの基本

3. 結合の基本

4. 統計情報とは

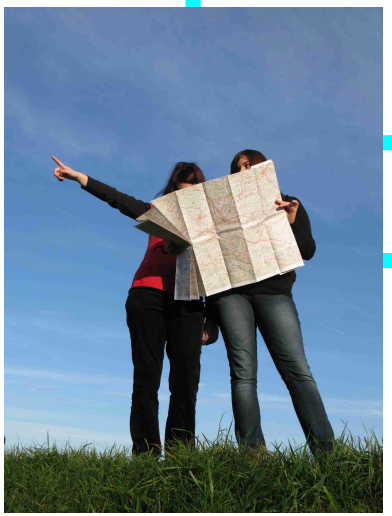
5.EXPLAIN コマンド

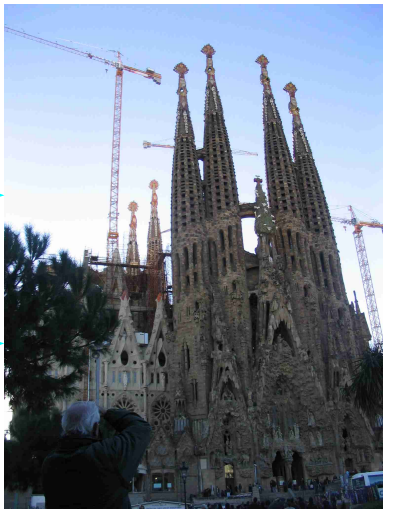
6. 問題解決例

7. まとめ

1.EXPLAIN とは

実行計画とは - 目的地は１つでもアクセス方法は複数



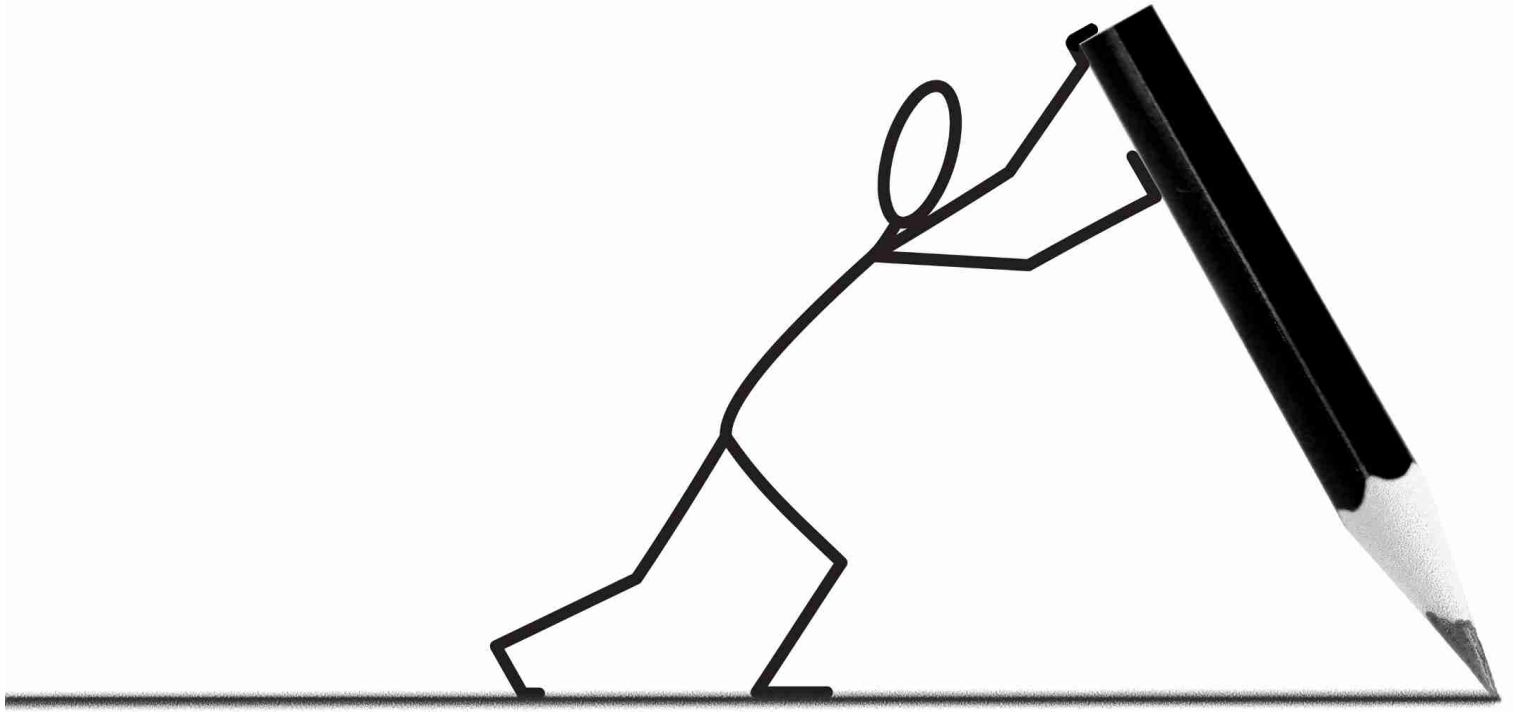


1.EXPLAIN とは

表の状態が分からなければどのパスが最適か分からない

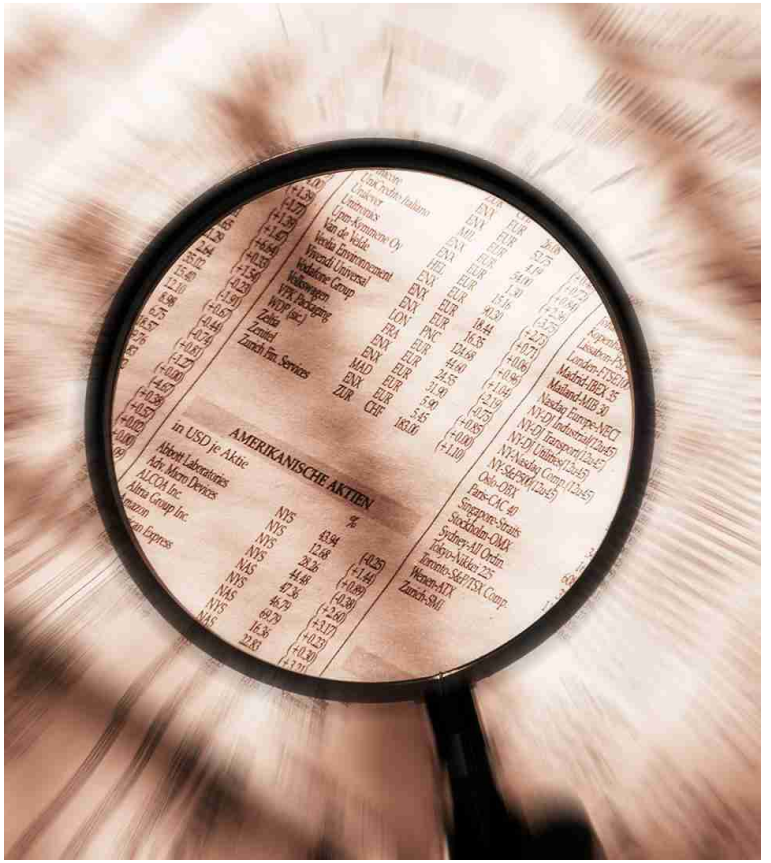
1.EXPLAIN とは

実行計画担当：プランナー　です。



1.EXPLAIN とは

プランナがどのような実行計画を作ったのかを
確認する手段が本日のテーマである
EXPLAIN コマンドです。



1.EXPLAIN とは

① どのようなアクセス方法が適切か

② どのような結合方法が適切か

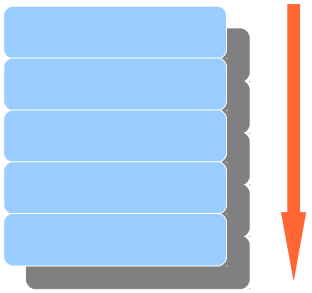
③ 統計情報を元に実行計画を作成する事
がプランナの役目

④ どのような選択が行なわれたか
どのように実行されたか
EXPLAIN コマンドで確認する

2. 表アクセス方法

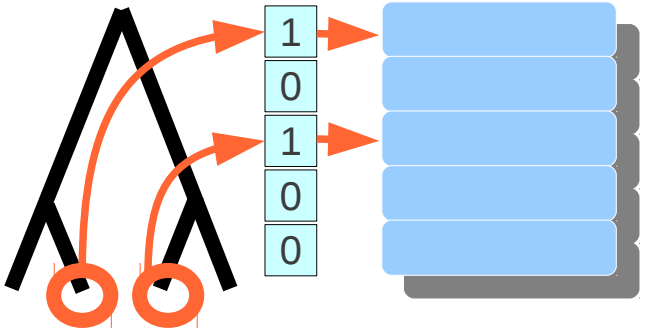
(1).Seq Scan

検索範囲：広



(3).Bit Map Scan

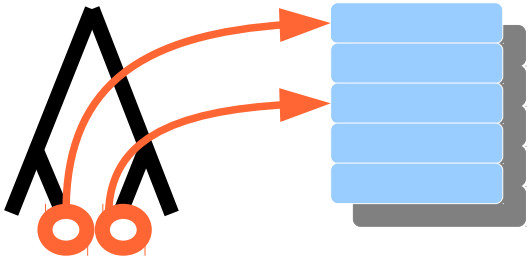
検索範囲：中 / 特殊



NEW 9.2

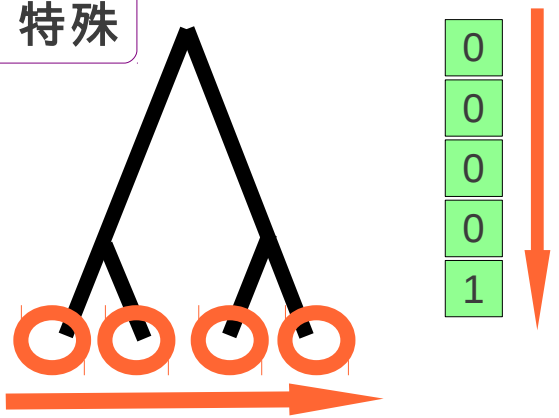
(2).Index Scan

検索範囲：狭



(4).Index Only Scan

検索範囲：特殊



3. 表結合方法

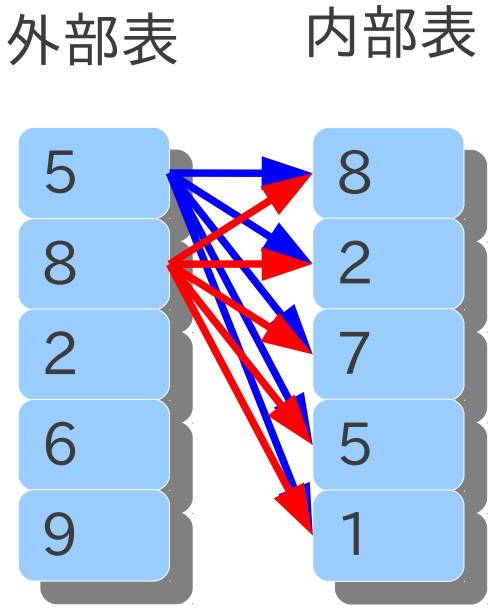
① どのようなアクセス方法が適切か

② どのような結合方法が適切か

③ 統計情報を元に実行計画を作成する事がプランナの役目

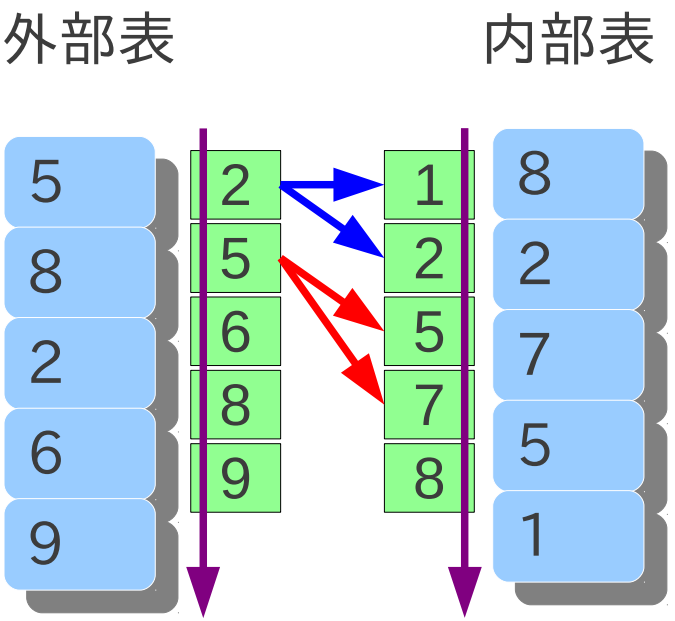
④ どのような選択が行なわれたか、どのように実行されたか、EXPLAIN コマンドで確認する

1.Nested Loop Join



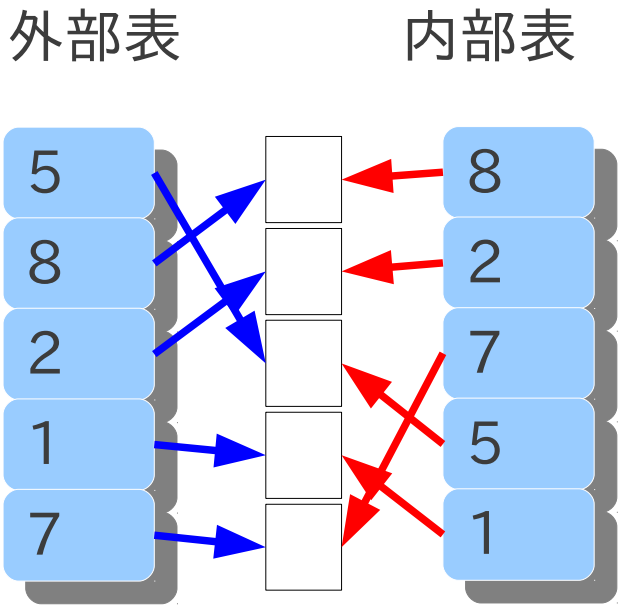
特徴：いかなる場合でも選択可能

2.Sort Merge Join



特徴：ソートが完了すれば早い

3.Hash Join



特徴：ハッシュを作成できれば早い

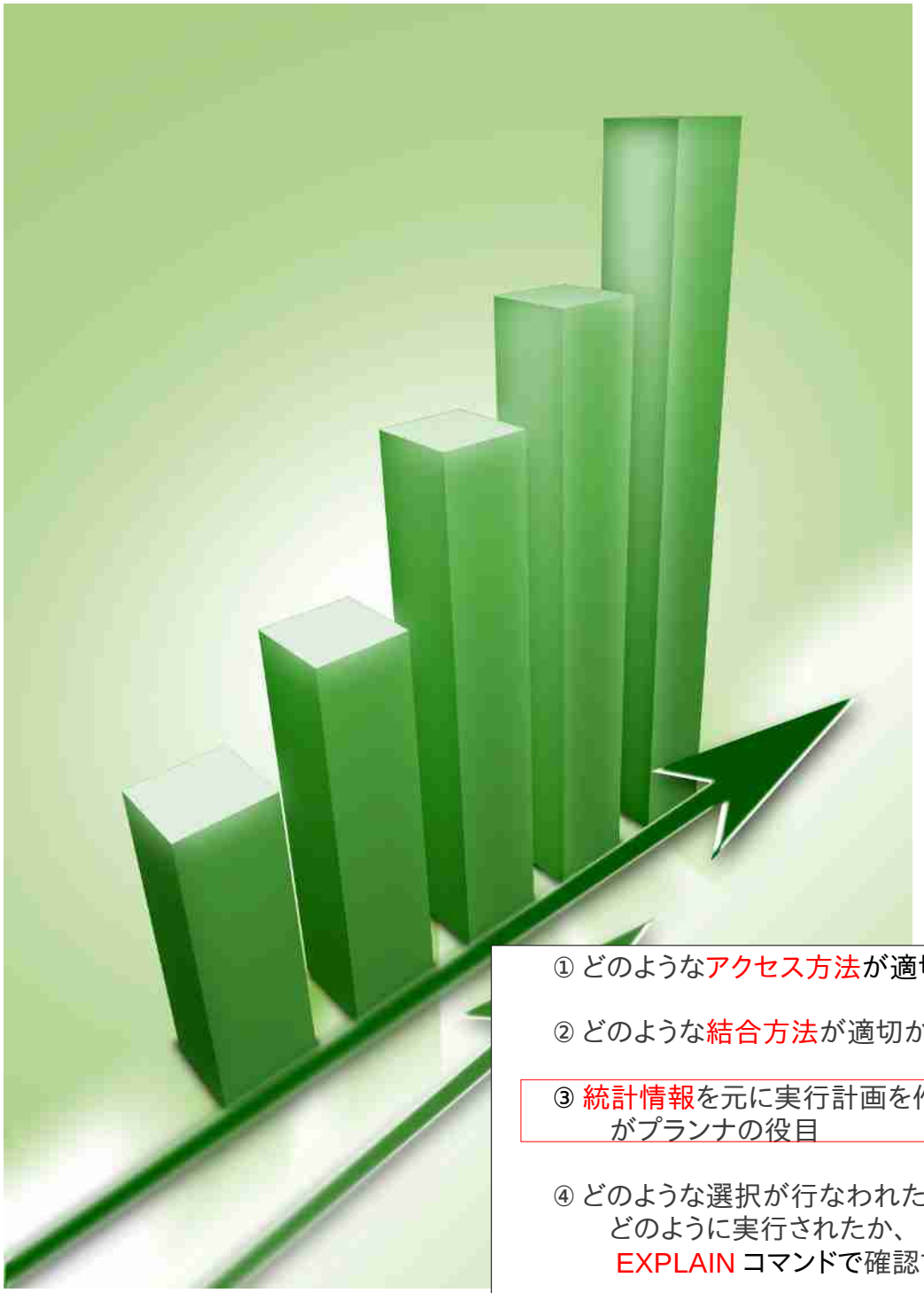
4. 統計情報

１つ１つの表の

・行数
・行サイズ平均
・相関
・ヒストグラム

などを見積もったもの。

統計情報取得コマンド
ANALYZE　表名；



① どのようなアクセス方法が適切か

② どのような結合方法が適切か

③ 統計情報を元に実行計画を作成する事
がプランナの役目

④ どのような選択が行なわれたか、
どのように実行されたか、
EXPLAIN コマンドで確認する

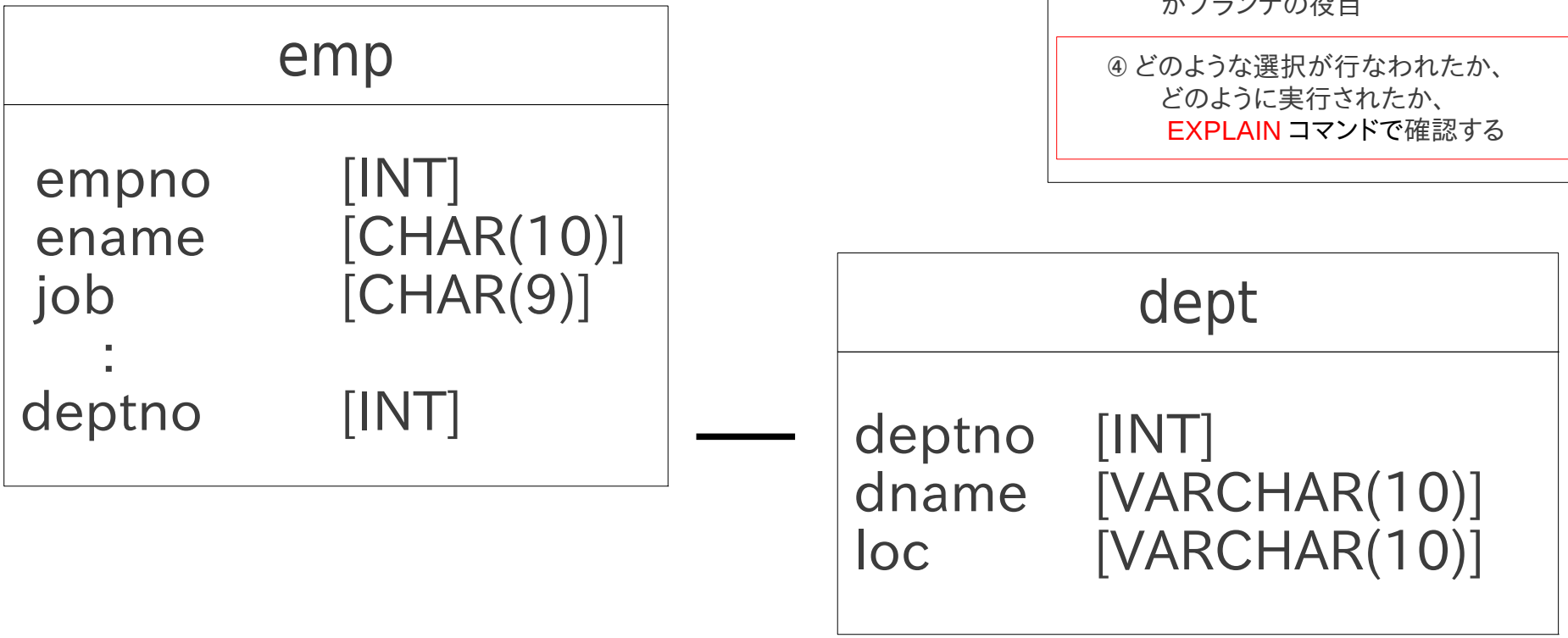
5.EXPLAIN コマンド

① どのようなアクセス方法が適切か

② どのような結合方法が適切か

③ 統計情報を元に実行計画を作成する事
がプランナの役目

④ どのような選択が行なわれたか、
どのように実行されたか、
EXPLAIN コマンドで確認する




```
SELECT d.dname,e.ename FROM emp e
JOIN dept d USING (deptno);
```

5.EXPLAIN コマンド

Explain Plan の例

```
# EXPLAIN ANALYZE SELECT d.dname,e.ename FROM emp e
JOIN dept d    USING (deptno);
                                  QUERY PLAN
-------------------------------------------------------------
 Hash Join  (cost=1.23..4101.23 rows=100000 width=66)
            (actual time=0.045..161.248 rows=90000 loops=1)
   Hash Cond: (e.deptno = d.deptno)
   ->  Seq Scan on emp e  (cost=0.00..2725.00 rows=100000 width=41)
                          (actual time=0.007..49.537 rows=100000 loops=1)
   ->  Hash  (cost=1.10..1.10 rows=10 width=37)
             (actual time=0.025..0.025 rows=10 loops=1)
         Buckets: 1024  Batches: 1  Memory Usage: 1kB
         ->  Seq Scan on dept d  (cost=0.00..1.10 rows=10 width=37)
                                 (a
 Total runtime: 196.524 ms
(7 rows)
```

ANALYZE オプションを付けることで
実際に SQL が実行され、actual time の情報が出力される
システムへの影響を考慮すること

5.EXPLAIN コマンド　（アクセス方法）

Explain Plan の例

```
# EXPLAIN ANALYZE SELECT d.dname,e.ename FR
JOIN dept d    USING (deptno);
                                QUERY PLAN
-----------------------------------------------------------------
 Hash Join  (cost=1.23..4101.23 rows=100000 width=66)
            (actual time=0.045..161.248 rows=90000 loops=1)
   Hash Cond: (e.deptno = d.deptno)
   ->  Seq Scan on emp e  (cost=0.00..2725.00 rows=100000 width=41)
                          (actual time=0.007..49.537 rows=100000 loops=1)
   ->  Hash  (cost=1.10..1.10 rows=10 width=37)
             (actual time=0.025..0.025 rows=10 loops=1)
         Buckets: 1024  Batches: 1  Memory Usage: 1kB
         ->  Seq Scan on dept d  (cost=0.00..1.10 rows=10 width=37)
                                 (actual time=0.003..0.013 rows=10 loops=1)
 Total runtime: 196.524 ms
(7 rows)
```

① どのようなアクセス方法が適切か

② どのような結合方法が適切か

③ 統計情報を元に実行計画を作成する事がプランナの役目

④ どのような選択が行なわれたか、どのように実行されたか、EXPLAIN コマンドで確認する

インデックススキャンの場合の表記
Index Scan using emp_pkey on emp e

5.EXPLAIN コマンド　（結合方法）

Explain Plan の例

```
# EXPLAIN ANALYZE SELECT d.dname,e.ename FR
JOIN dept d    USING (deptno);
                                QUERY PLAN
-----------------------------------------------------------------
 Hash Join  (cost=1.23..4101.23 rows=100000 width=66)
            (actual time=0.045..161.248 rows=90000 loops=1)
   Hash Cond: (e.deptno = d.deptno)
   ->  Seq Scan on emp e  (cost=0.00..2725.00 rows=100000 width=41)
                          (actual time=0.007..49.537 rows=100000 loops=1)
   ->  Hash  (cost=1.10..1.10 rows=10 width=37)
             (actual time=0.025..0.025 rows=10 loops=1)
         Buckets: 1024  Batches: 1  Memory Usage: 1kB
         ->  Seq Scan on dept d  (cost=0.00..1.10 rows=10 width=37)
                                 (actual time=0.003..0.013 rows=10 loops=1)
 Total runtime: 196.524 ms
(7 rows)
```

① どのようなアクセス方法が適切か

② どのような結合方法が適切か

③ 統計情報を元に実行計画を作成する事がプランナの役目

④ どのような選択が行なわれたか、どのように実行されたか、EXPLAIN コマンドで確認する

5.EXPLAIN コマンド　(統計情報)

Explain Plan の例

① どのようなアクセス方法が適切か

② どのような結合方法が適切か

③ 統計情報を元に実行計画を作成する事がプランナの役目

④ どのような選択が行なわれたか、どのように実行されたか、EXPLAIN コマンドで確認する

プランナが推定したコストと行数

```
# EXPLAIN ANALYZE SELECT d.dname,e.ename FR
JOIN dept d    USING (deptno);
                                QUERY PLAN
-----------------------------------------------------------------
 Hash Join  (cost=1.23..4101.23 rows=100000 width=66)
            (actual time=0.045..161.248 rows=90000 loops=1)
   Hash Cond: (e.deptno = d.deptno)
   ->  Seq Scan on emp e  (cost=0.00..2725.00 rows=100000 width=41)
                          (actual time=0.007..49.537 rows=100000 loops=1)
   ->  Hash  (cost=1.10..1.10 rows=10 width=37)
             (actual time=0.025..0.025 rows=10 loops=1)
         Buckets: 1024  Batches: 1  Memory Usage: 1kB
         ->  Seq Scan on dept d  (cost=0.00..1.10 rows=10 width=37)
                                 (actual time=0.013..0.013 rows=10 loops=1)
 Tota
(7 rows)
```

実際に SQL を実行した時間と行数

5.EXPLAIN コマンド　(統計情報)

見積もられた平均列長

(cost=0.00..2725.00 rows=100000 width=41)

取り出される行数の見積もり

表アクセスにかかるコストの見積もり
・ディスクからのデータ読み込み
・メモリ上のスキャン
・ CPU を使用する処理

繰り返し実行された回数

(actual time=0.007..49.537 rows=100000 loops=1)

実際に取り出された行数

実際に表アクセスにかかった時間 (ミリ秒)

5.EXPLAIN コマンド　(統計情報見方のコツ)

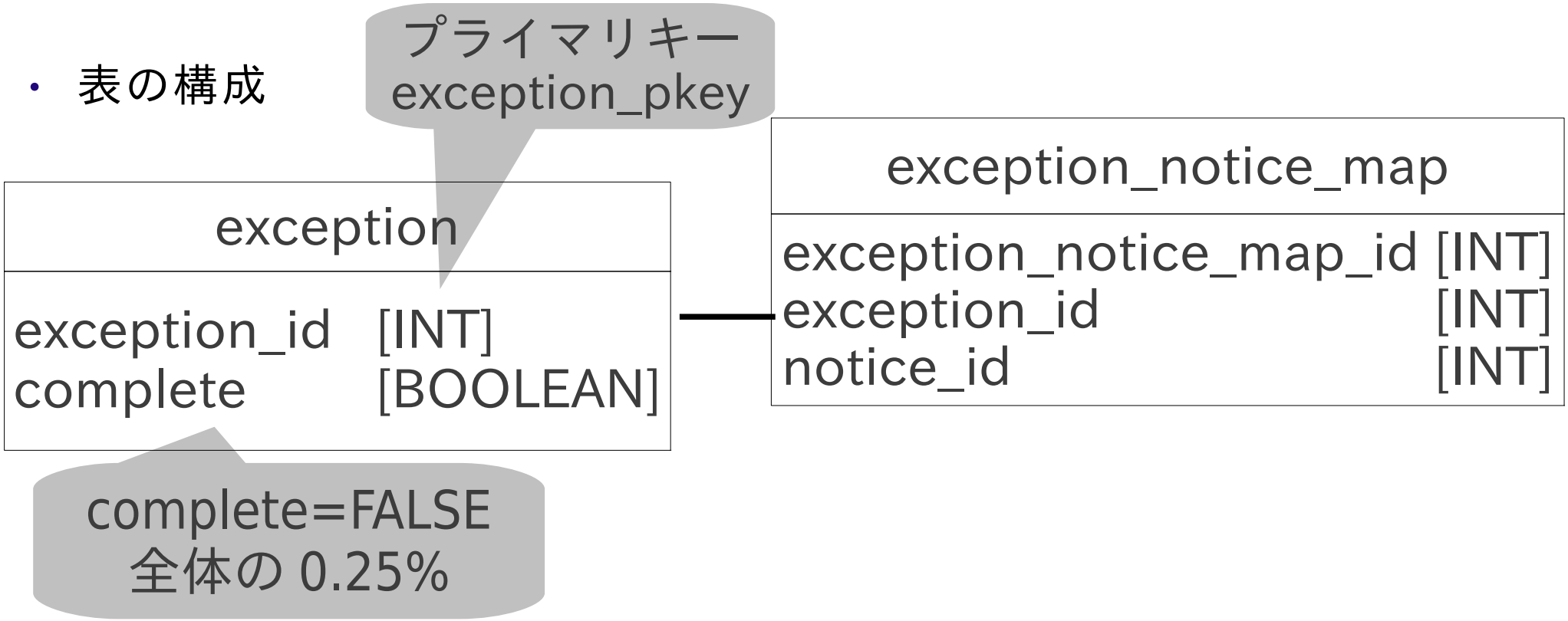
1. 統計情報は「誤差」が最も少なくなるであろう、
   下 (インデントが下のもの) から見ていく

2. 共通するパラメータは rows

```
# EXPLAIN ANALYZE SELECT d.dname,e.ename FROM emp e
JOIN dept d    USING (deptno);
                                  QUERY PLAN
-------------------------------------------------------------------
 Hash Join  (cost=1.23..4101.23 rows=100000 width=66)
            (actual time=0.045..161.248 rows=90000 loops=1)
   Hash Cond: (e.deptno = d.deptno)
   ->  Seq Scan on emp e  (cost=0.00..2725.00 rows=100000 width=41)
                          (actual time=0.007..49.537 rows=100000 loops=1)
   ->  Hash  (cost=1.10..1.10 rows=10 width=37)
             (actual time=0.025..0.025 rows=10 loops=1)
         Buckets: 1024  Batches: 1  Memory Usage: 1kB
         ->  Seq Scan on dept d  (cost=0.00..1.10 rows=10 width=37)
                                 (actual time=0.003..0.013 rows=10 loops=1)
 Total runtime: 196.524 ms
(7 rows)
```

6. 問題解決演習 (1)

- 表の構成




```
SELECT exception_id FROM exception
JOIN exception_notice_map USING (exception_id)
WHERE complete IS FALSE AND notice_id = 3;
```

6. 問題解決演習 (1)

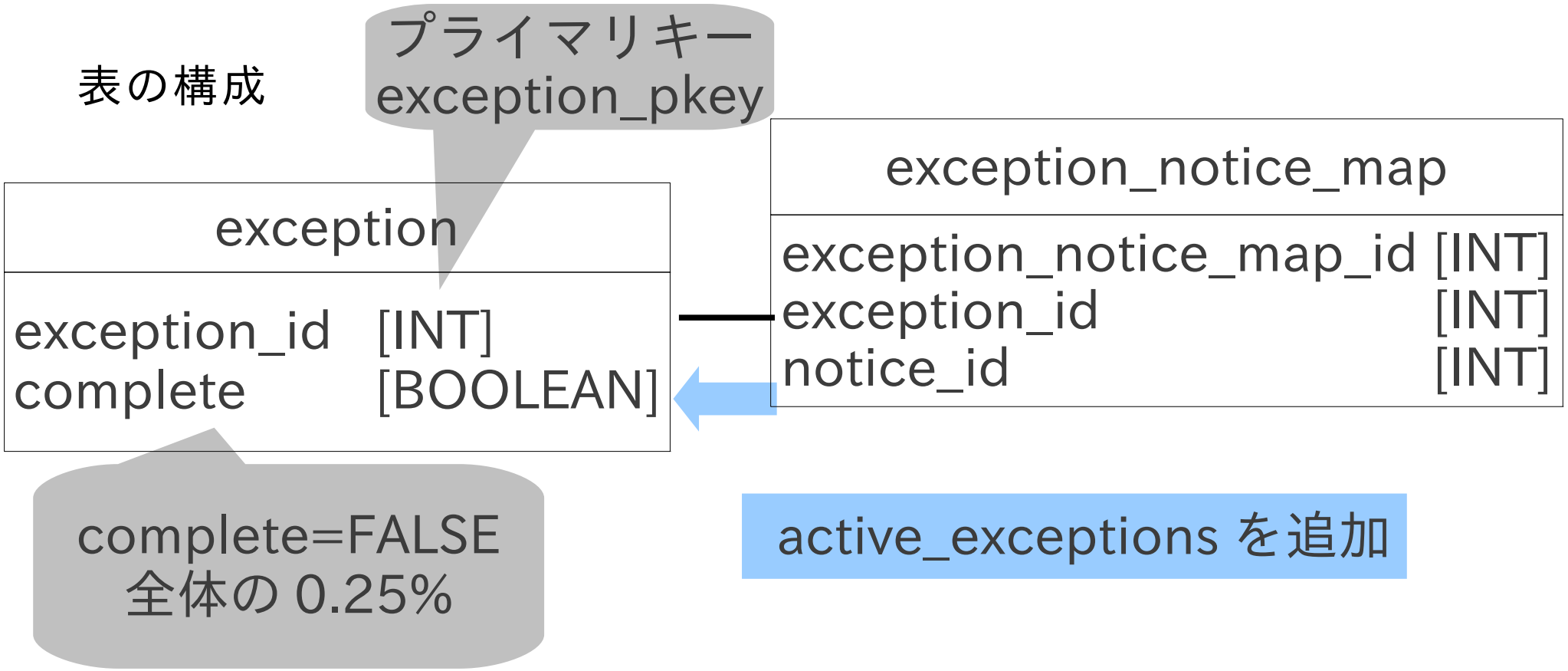
```
=# EXPLAIN ANALYZE SELECT exception_id FROM exception
-# JOIN exception_notice_map USING (exception_id)
-# WHERE complete IS FALSE AND notice_id = 3;

                                                    QUERY PLAN


-------------------------------------------------------------------------------------------------
 Hash Join  (cost=14428.34..22873.52 rows=7 width=4)
                (actual time=147.192..246.654 rows=251 loops=1)
   Hash Cond: (exception_notice_map.exception_id = exception.exception_id)
   ->  Seq Scan on exception_notice_map  (cost=0.00..8352.77 rows=24623 width=4)
                                                (actual time=0.011..88.084 rows=24800 loops=1)
         Filter: (notice_id = 3)
   ->  Hash  (cost=14425.00..14425.00 rows=267 width=4)
                                                                                  ・・・
         Buckets: 1024  Batches: 1  Memory Usage: 8kB
         ->  Seq Scan on exception  (cost=0.00..14425.00 rows=267 width=4)
                                    (actual time=0.007..147.017 rows=251 loops=1)

               Filter: (complete IS
 Total runtime: 246.807 ms
(9 rows)
```

Seq Scan on exception (cost=0.00..14425.00

exception 表に "WHERE complete IS False" という
条件はわずかなのに全てのデータにアクセスしている

6. 問題解決演習 (1)

表の構成




```
SELECT exception_id FROM exception
JOIN exception_notice_map USING (exception_id)
WHERE complete IS FALSE AND notice_id = 3;
```

6. 問題解決演習 (1)

```
=# CREATE INDEX active_exceptions ON exception(complete)
    WHERE complete IS false;

=# EXPLAIN ANALYZE SELECT exception_id FROM exception
-# JOIN exception_notice_map USING (exception_id)
-# WHERE complete IS FALSE AND notice_id = 3;

                                    QUERY PLAN

-------------------------------------------------------------------------------
=# EXPLAIN ANALYZE SELECT exception_id FROM exception
-# JOIN exception_notice_map USING (exception_id)
-# WHERE complete IS FALSE AND notice_id = 3;
 Hash Join  (cost=16.26..8461.42 rows=5 width=4) (actual time=0.566..112.103 rows=251 loops=1)
   Hash Cond: (exception_notice_map.exception_id = exception.exception_id)
   ->  Seq Scan on exception_notice_map  (cost=0.00..8352.77 rows=24623 width=4)
         (actua
          Filter:
   ->  Hash  (cost=13.76..13.76 rows=200 width=4) (actual time=0.536..0.536 rows=251 loops=1)
          Buckets: 1024  Batches: 1  Memory Usage: 6kB
          ->  Index Scan using active_exceptions on exceptio
idth=4) (actual time=0.025..0.283 rows=251 loops=1)
                Index Cond: (complete = false)
 Total runtime: 112.323 ms
(9 rows)
```

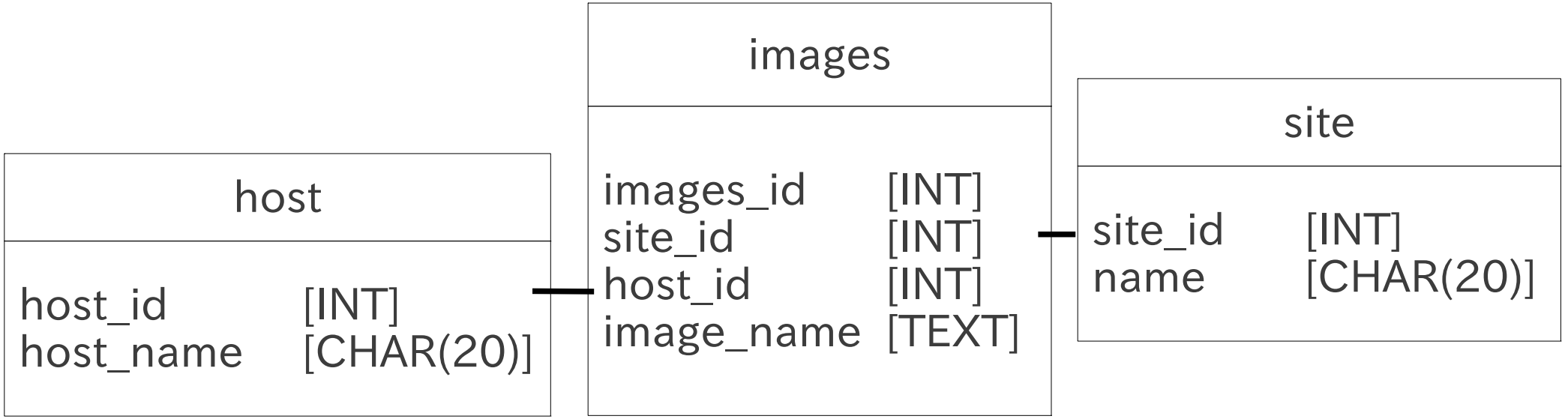
Index Scan using active_exceptions on exception ・・

インデックスを使ってくれた

INDEX 作成前
Total runtime: 246.807 ms

例 **1)** のまとめ

6. 問題解決演習 (2)




```
SELECT h.host_name,s.name,i.image_name
 FROM images i
 JOIN host h USING (host_id) JOIN site s USING (site_id)
 WHERE images_id > 2212;
```

6. 問題解決演習 (2)

host 表の Seq Scan 時間が site と比べて効率が悪い

```
=#explain analyze SELECT h.host_name,s.name
-#              JOIN host h USING (host_id) JOIN si
-#              WHERE images_id > 2212;
```

(actual time=1188.441..1236.629 rows=100000 loops=1)

```
  Hash Cond: (h.host_id = i.host_id)
  ->  Seq Scan on host h  (cost=0.00..10167.00 rows=100000 width=4)
                  (actual time=1188.441..1236.629 rows=100000 loops=1)
  ->  Hash  (cost=12
        (actual time=5.481..5.481 rows=788 loops=1)
      Buckets: 1024  Batches: 1  Memory Usage: 75kB
      ->  Hash Join  (cost=46.89..121.02 rows=788 wid
              (actual time=3.589..4.928 rows=788 loo
          Hash Cond: (s.site_id = i.site_id)
          ->  Seq Scan on site s  (cost=0.00..55.00 rows
```

1 ミリ秒に 84 行抽出

host

host_id	[INT]
host_name	[CHAR(20)]

(actual time=0.025..1.685 rows=3000 loops=1)

```
                  (actual time=1.254..1.254 rows=788 loops=1)
                Buckets: 1024
                ->  Index Scan u
                  (cost=0.00..37.04 rows=788 width=4
                  (actual time=0.065..0.758 rows=788
                    Index Cond: (images_id > 2212)
Total runtime: 1290.995 ms
```

1 ミリ秒で 120000 行抽出

site

site_id	[INT]
name	[CHAR(20)]

6. 問題解決演習 (2)

host	
host_id	[INT]
host_name	[CHAR(20)]

1 行のサイズは？
⇒INT 4byte ＋ CHAR 20byte ＋ Tupleheader 23+ α byte= 約 50bytes

ブロックヘッダは ？
⇒23bytes

1 ブロックのサイズは？
⇒8192bytes

1 ブロックに入る最大行数
(8192-32)/50　＝　約 163 行

6. 問題解決演習 (2)

```
=#explain analyze SELECT h.host_name,s.name,i.imag
-#            JOIN host h USING (host_id) JOIN site s US
-#            WHERE images_id > 2212;

 Hash Join  (cost=130.87..10680.75 rows=788 width
         (actual time=1196.263..1290.620 rows=788 l
   Hash Cond: (h.host_id = i.host_id)
   ->  Seq Scan on host h  (cost=0.00..10167.00 rows=100000 width=4)
                   (actual time=1188.441..1236.629 rows=100000 loops=1)
         Buckets: 1024  Batches: 1  Memory Usage: 75kB
         ->  Hash Join  (cost=46.89..121.02 rows=788 width=74)
                    (actual time=3.589..4.928 rows=788 loops=1)
               Hash Cond: (s.site_id = i.site_id)
               ->  Seq Scan on site s  (cost=0.00..55.00 rows=3000 width=37)
                           (actual time=0.025..1.685 rows=3000 loops=1)
                    Index Cond: (images_id > 2212)
 Total runtime: 1290.995 ms
```

host	
host_id	[INT]
host_name	[CHAR(20)]

Seq Scan on host h (cost=0.00..10167.00 rows=100000 width=4)

１ブロックに 10 行しか格納できていない
↓
削除フラグが立った行が多数あるのではないか

6. 問題解決演習 (2)

```
=#vacuum full host;
=#explain analyze SELECT h.host_name,s.name,i.image_name FROM images i
-#              JOIN host h USING (host_id) JOIN site s USING (site_id)
-#              WHERE images_id > 2212;
 Hash Join  (cost=130.87..2360.32 rows=788 width=70)
            (actual time=11.701..112.387 rows=788 loops=1)
   Hash Cond: (h.host_id = i.host_id)
   ->  Seq Scan on host h  (cost=0.00..1843.14 rows=100914 width=4)
                           (actual time=0.025..51.975 rows=100000 loops=1)
   ->  Hash  (cost=121.02..121.02 rows=788 w
               (actual time=8.148..8.148 rows=7
         Buckets: 1024  Batches: 1  Memory Usa
         ->  Hash Join  (cost=46.89..121.02 rows=
                        (actual time=5.123..7.252 rows=788 loops=1)
                  Hash Cond: (s.site_id = i.site_id)
                  (actual time=0.005..2.274 rows=3000 loops=1)
                  (actual time=1.701..1.701 rows=788 loops=1)
               Buckets: 1024  Batches: 1  Memory Usage: 50kB
               ->  Index Scan using images_pkey on images i
                     (cost=
                     (actua
                      Index
Total runtime: 112.932 ms
```

1 行あたりにかかる時間が大幅に改善

対処前　(actual time=1188.441..1236.629 rows=100000 loops=1)

対処後 (actual time=0.025..51.975 rows=100000 loops=1)

VACUUM 前 Total runtime: 1290.995 ms

VACUUM 後 Total runtime: 112.932 ms

6. 問題解決演習 (2) まとめ

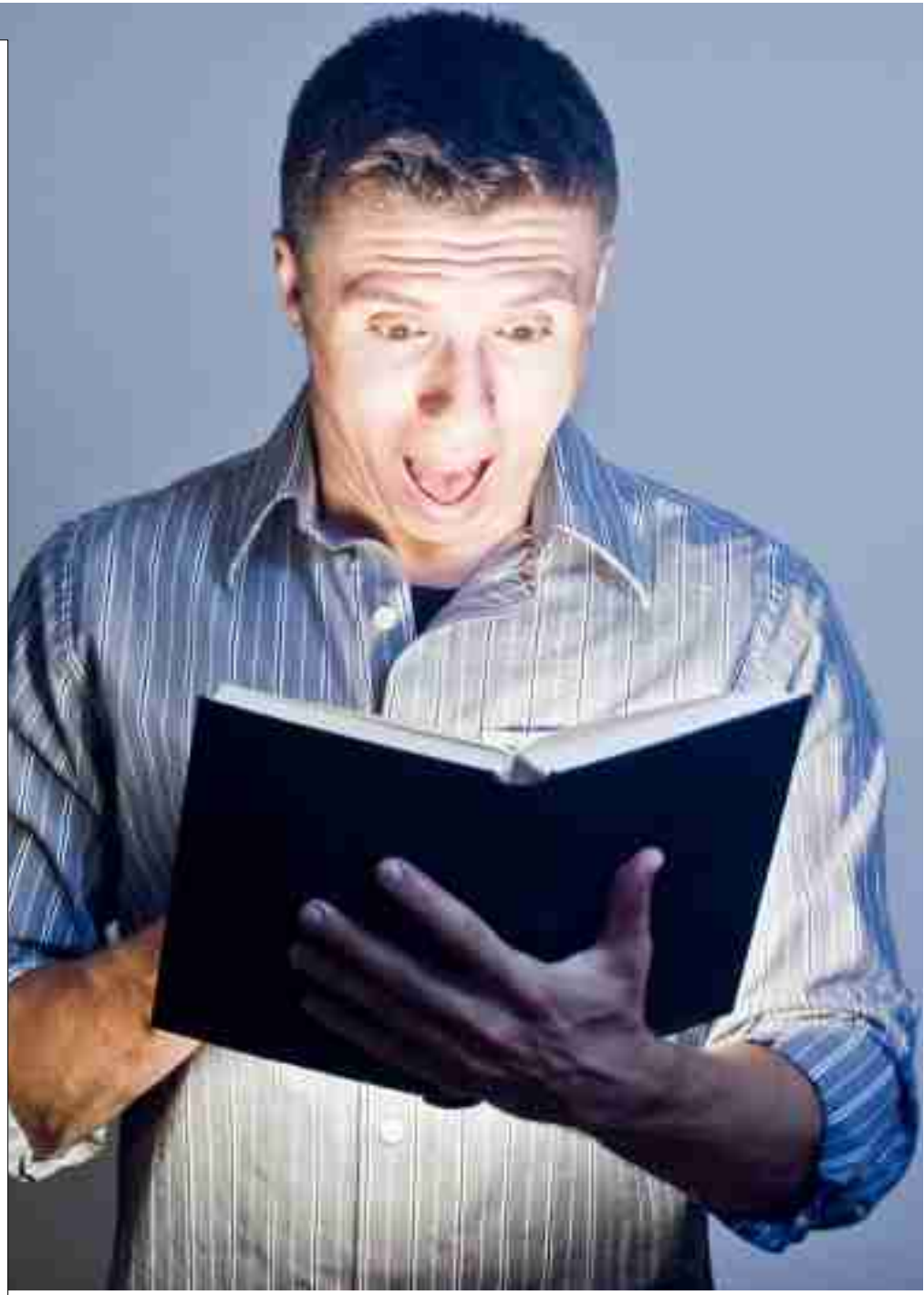
VACUUM FULL がいらない設計、運用を。
EXPLAIN を見れば、メンテナンスの必要性も分かる

まとめ

どのようなアクセス方法が適切か
どのような結合方法が適切か
統計情報を元に実行計画を
作成する事がプランナの役目
どのような選択が行なわれたかを
EXPLAIN コマンドで確認できる

EXPLAIN には ANALYZE をつける
インデントの下の方から時間が
かかっているものを見つける
対処例)

-INDEX を作成する
-VACUUM　FULL を実行する



ご静聴ありがとうございました。

参考資料

Explaining Explain ～ PostgreSQL の実行計画を読む ～

http://lets.postgresql.jp/documents/technical/query_tuning/explaining_explain_ja.pdf/view

内部を知って業務に活かす　PostgreSQL 研究所第 4 回

http://www2b.biglobe.ne.jp/~caco/webdb-pdfs/vol29.pdf

Robert Haas blog

http://rhaas.blogspot.com/2011/10/index-only-scans-weve-got-em.html

問合せ最適化インサイド

http://www.slideshare.net/ItagakiTakahiro/ss-4656848

象と戯れ

http://postgresql.g.hatena.ne.jp/umitanuki/20110425/1303752697

スライドの画像

http://www.sxc.hu/