SRA OSS

PostgreSQL でインデックスはどう使われるのか

PostgreSQL Conference Japan 2024

2024-12-06

長田悠吾（株式会社SRA OSS　OSS事業本部 技術開発室）

自己紹介

- 長田 悠吾（ながたゆうご）
- 株式会社SRA OSS　 OSS事業本部 技術開発室
- PostgreSQL Contributor
- pg_ivm
  - マテリアライズドビューの増分更新を提供する拡張モジュール

© 2024 SRA OSS K.K.

本日の内容

- インデックス
  - データベースの検索性能を向上させる一般的手法

→ インデックスが使われるまでに、PostgreSQLの中で何が起こっているのか？

  - クエリの実行開始からインデックスが使われるまでの流れ
  - 演算子クラス・演算子族
  - インデックスが使われない原因

© 2024 SRA OSS K.K.

話さないこと

- インデックスのデータ構造
- 各インデックスアクセスメソッドの説明
  - B-tree、Hash, Gist, SP-Gist, GIN, BRIN, …
- インデックスの保守運用
  - REINDEX, VACUUM

© 2024 SRA OSS K.K.

インデックスとは

© 2024 SRA OSS K.K.

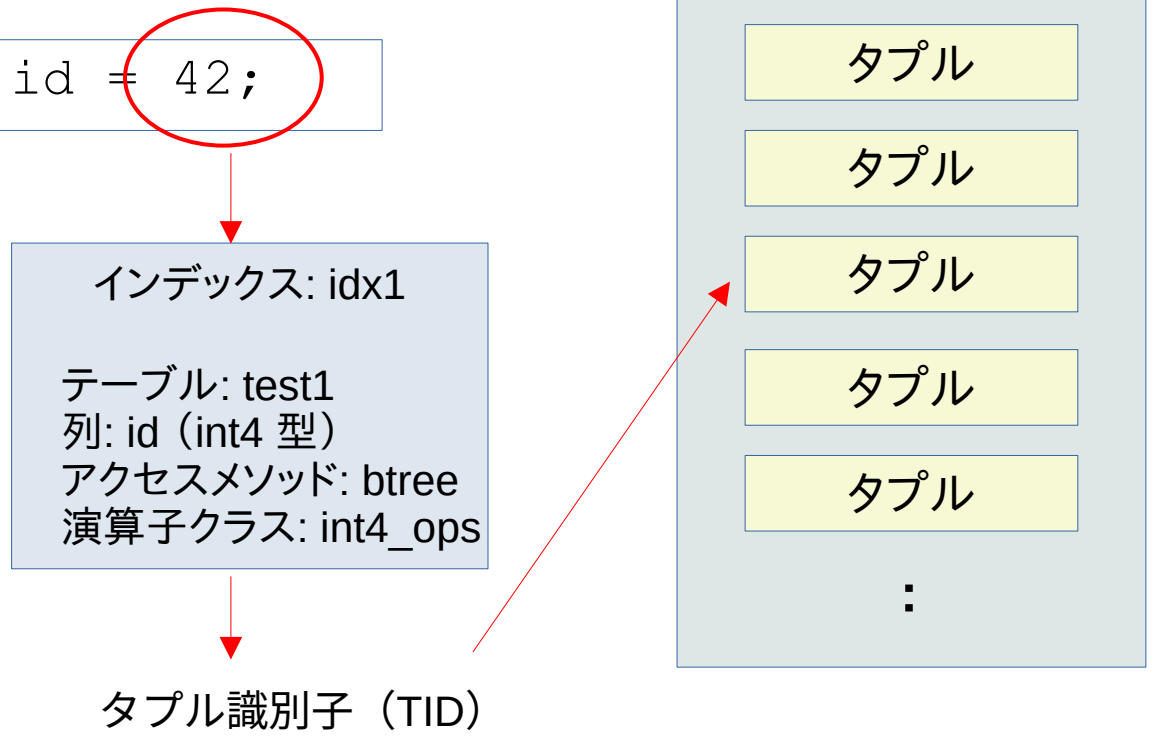
インデックス

- データベースの検索性能を向上させる一般的手法

```
SELECT * FROM test1 WHERE id = 42;
```

```
CREATE INDEX idx1 ON test1(id);
```

キー値からタプル識別子へのマッピング



© 2024 SRA OSS K.K.

インデックスの定義構文　（抜粋）

- アクセスメソッド
- 対象
  - テーブル
  - カラム or 式
- 演算子クラス
- 照合順序（collate）

```
CREATE [ UNIQUE ] INDEX ON [table_name [ USING method ]
    ( { column_name | ( expression ) } [ COLLATE collation ]
[ opclass [ ( opclass_parameter = value [, ... ] ) ] ]
    [ TABLESPACE tablespace_name ]
    [ WHERE predicate ]
```

主にこれらの情報が
「インデックスがどう使われるのか」
を決めている

- その他
  - 一意インデックス、プライマリキー
  - テーブル空間
  - 部分インデックスの条件
  - …

© 2024 SRA OSS K.K.

インデックスの定義の例（1）

```
CREATE TABLE test1 (id int, col text);

CREATE INDEX idx1 ON test1(id);
```

- アクセスメソッド
  - btree （デフォルト）
- 対象
  - テーブル：test1、カラム：id (int)
- 演算子クラス
  - int4_ops （デフォルト）

© 2024 SRA OSS K.K.

インデックスの定義の例（2）

```
CREATE TABLE test1 (id int, col text);

CREATE INDEX idx2 ON test1 USING hash(id);
```

- アクセスメソッド
  - hash
- 対象
  - テーブル：test1、カラム：id (int)
- 演算子クラス
  - int4_ops　（デフォルト）

© 2024 SRA OSS K.K.

インデックスの定義の例(3)

```
CREATE TABLE test1 (id int, col text);

CREATE INDEX idx3 ON test1 (col varchar_pattern_ops);
```

- アクセスメソッド
  - btree （デフォルト）
- 対象
  - テーブル:test1、カラム:col (text)
- 演算子クラス
  - varchar_pattern_ops

© 2024 SRA OSS K.K.

インデックスの定義の例（4）

```
CREATE TABLE test1 (id int, col text);

CREATE INDEX idx3 ON test1 (col COLLATE "ja_JP" varchar_pattern_ops);
```

- アクセスメソッド
  - btree （デフォルト）
- 対象
  - テーブル：test1、カラム：col (text)
- 演算子クラス
  - varchar_pattern_ops

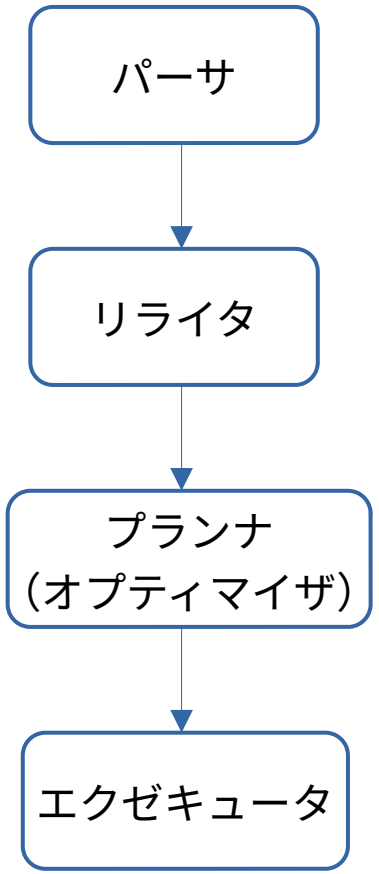
照合順序（文字のソート順や比較のきまり）：
"ja_JP" を指定

© 2024 SRA OSS K.K.

クエリ実行の流れから見たインデックス

© 2024 SRA OSS K.K.

クエリ実行の流れ



構文解析 / 意味解析

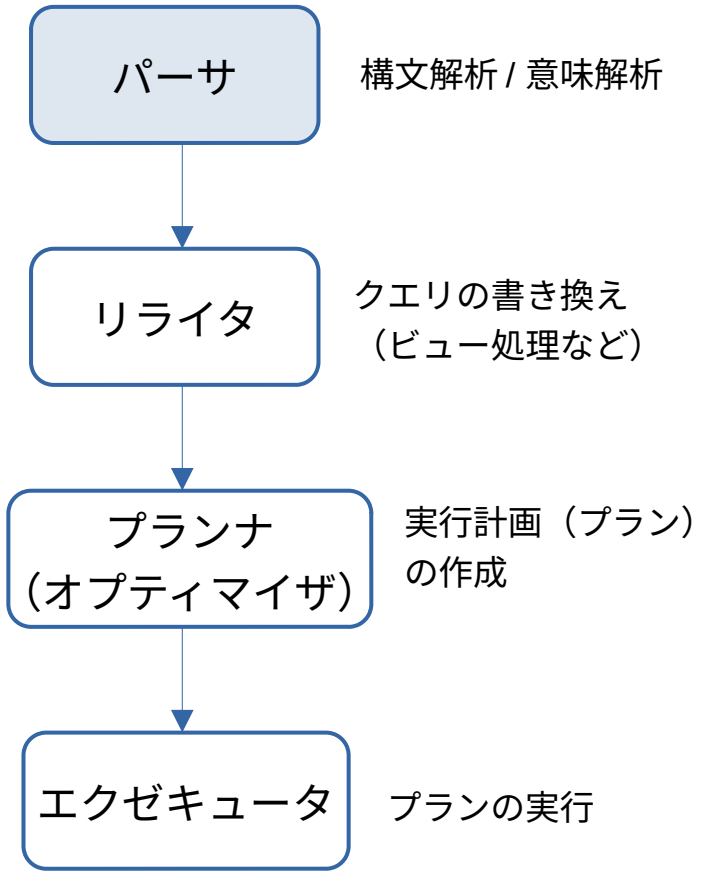
クエリの書き換え（ビュー処理など）

実行計画（プラン）の作成

プランの実行

© 2024 SRA OSS K.K.

クエリ実行の流れ(1):パーサ

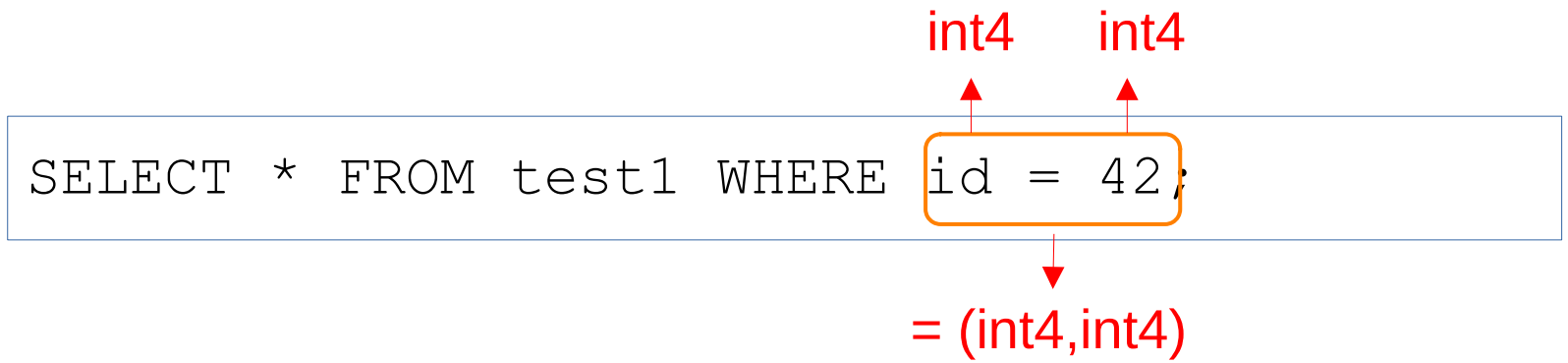


- 構文解析
  - クエリ文字列が正しいSQL構文を満たす場合に、ローパースツリー構造に変換
- 意味解析(アナライズ処理)
  - システムカタログを参照し、テーブルや演算子、型などの情報を追加
    - テーブルの OID
    - クエリの中の条件式の解析
    - 値の型や演算子の決定
    - 型キャスト
  - これらの情報を含むクエリツリーを作成

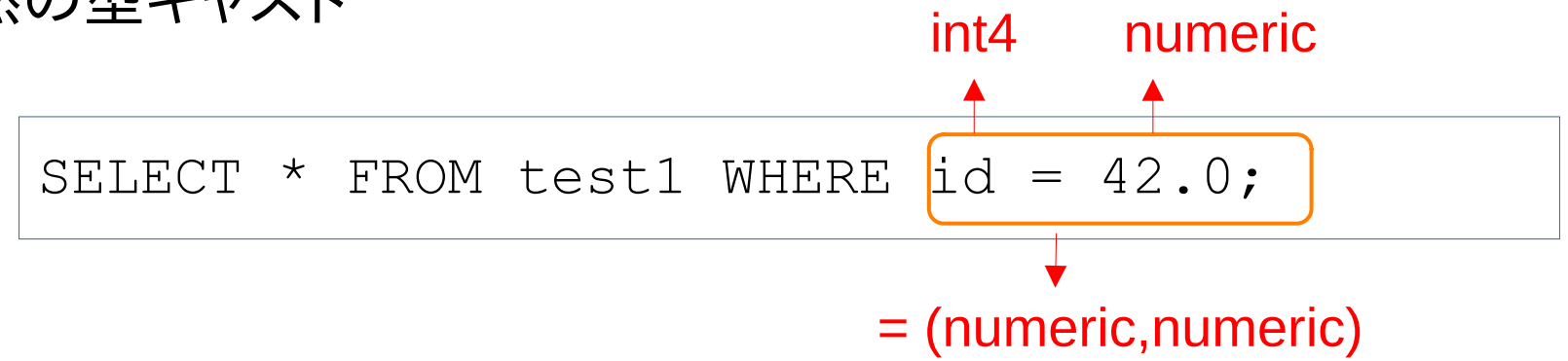
© 2024 SRA OSS K.K.

演算子の決定

- 一定の手続きのもとで、どの演算子を使用するかを判断する

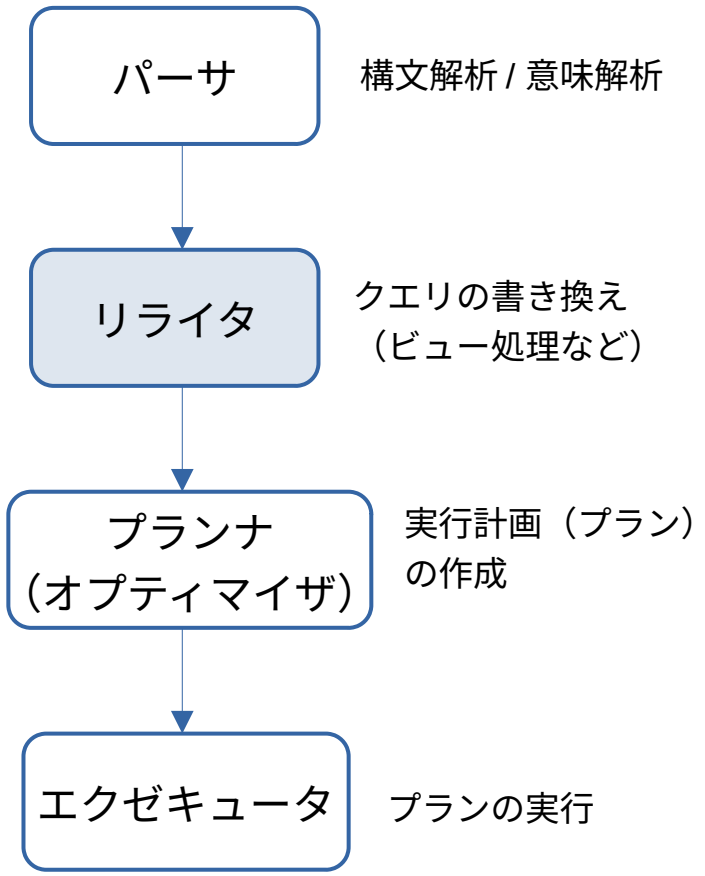


- 暗黙の型キャスト



© 2024 SRA OSS K.K.

クエリ実行の流れ（2）：リライタ



- クエリツリーを書き換えて VIEW や RULE を実現
  - ビューをサブクエリに展開
    - 「ビューへのSELECT」が、ビュー定義クエリの「サブクエリへのSELECT」に書き換えられる
  - 行レベルセキュリティ（RLS）
    - テーブルで RLS が有効になっている場合には、その情報がクエリツリーに追加される

© 2024 SRA OSS K.K.

ビューの書き換え

- ビュー定義

```
CREATE VIEW v AS SELECT c1, c2 FROM tbl WHERE c2 = 42;
```

- サブクエリに書き換え

```
SELECT * FROM v WHERE c1 = 43;
```

```
SELECT * FROM
  (SELECT c1, c2 FROM tbl WHERE c2 = 42) v
  WHERE c1 = 43;
```

© 2024 SRA OSS K.K.

行レベルセキュリティ（Row Level Security: RLS）

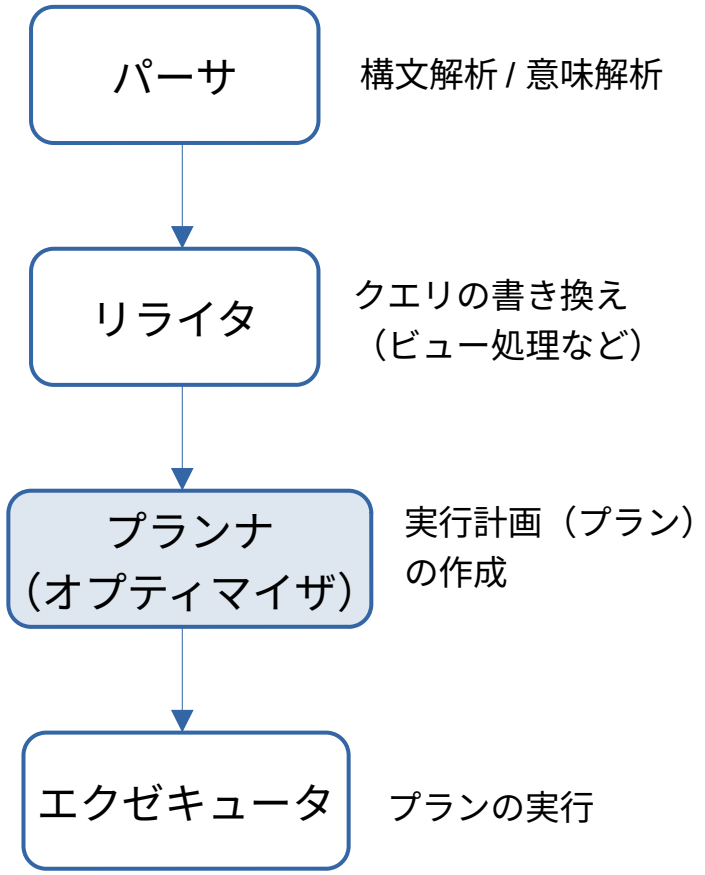
```
-- テーブル定義
CREATE TABLE accounts
  (manager text, company text, contact_email text);

-- RLS を有効にする
ALTER TABLE accounts ENABLE ROW LEVEL SECURITY;

-- セキュリティー条件（ポリシー）を定義
CREATE POLICY account_managers ON accounts
    USING (manager = current_user);
```

ユーザ名が manager カラムの値と一致する行しか読めなくなる

WHERE 句の条件に加えて、この条件もテーブル検索時に評価される

© 2024 SRA OSS K.K.

クエリ実行の流れ（3）：プランナ（オプティマイザ）



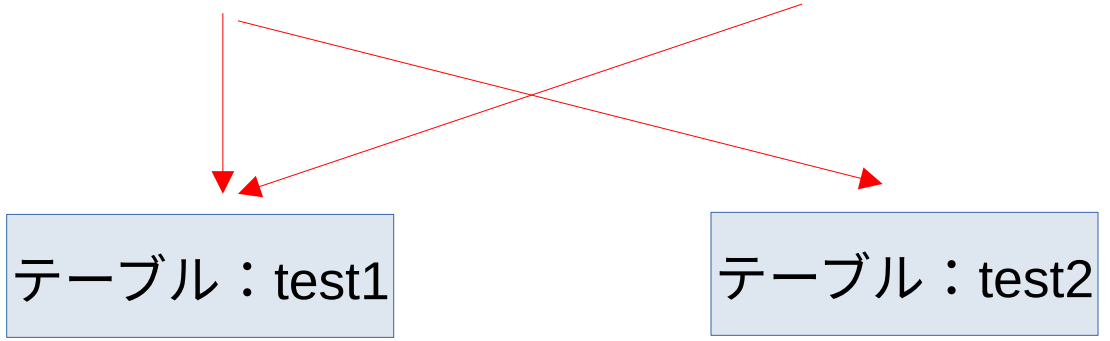
- クエリツリーから、クエリの「実行計画」（プランツリー）を生成
  - WHERE や JOIN ON で指定された条件式の分配
  - 様々なテーブルスキャン方法を探索
    - 使用可能なインデックスを調べる
  - 最もコストが少なく、実行時間が短いと思われるプランを選択する
  - インデックスを使ったクエリ最適化
    - ORDER BY
    - min/max

© 2024 SRA OSS K.K.

条件式の分配

- WHERE や JOIN ON の中の条件式が、
  関係するテーブルやサブクエリに分配される

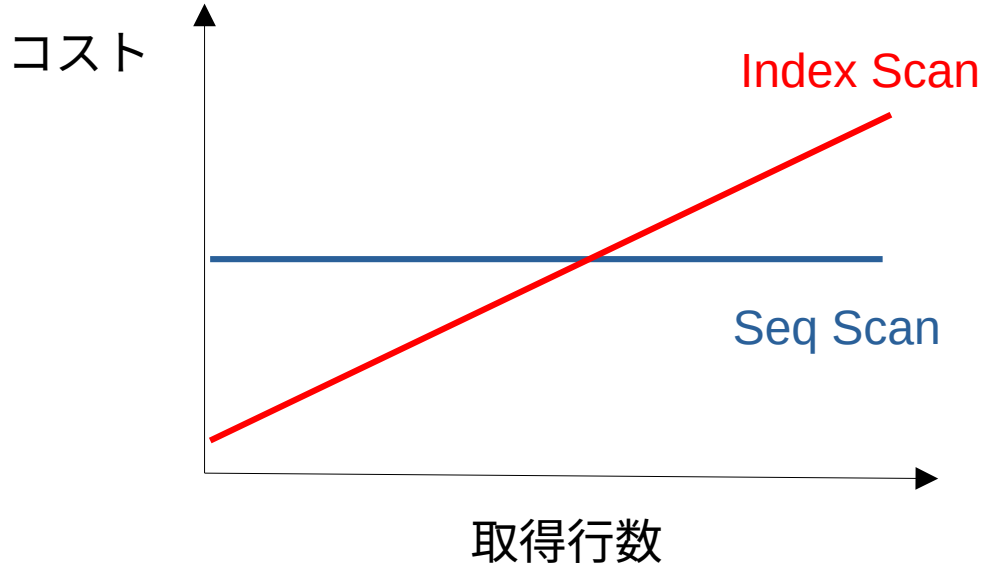
```
SELECT * FROM test1 AS t1, test2 AS t2,
 WHERE t1.id = t2.id AND t1.val > 42;
```



© 2024 SRA OSS K.K.

スキャン経路の探索とプランの選択

- 選択度（Selectivity）= 指定された条件でテーブル全体のどの程度の行が取得されるか
  - 統計情報（値の分布）を元に計算
  - 取得行数が選択度に基づいて見積もられる



- 順スキャン（Seq Scan）
  - テーブルの先頭から順にスキャンする
  - 取得する行の数が多い場合に高速

- インデックススキャン（Index Scan）
  - 指定された条件式で**「使用可能なインデックス」**がある場合に考慮される
  - 取得する行の数が少ない場合に高速

© 2024 SRA OSS K.K.

演算子クラス

© 2024 SRA OSS K.K.

演算子クラス

- 「ある演算子でどのインデックスが使用可能か」を決めるのが「演算子クラス」
- 演算子クラス
  - ある型についてインデックスを使用可能な演算子（+サポート関数）の集まり
    - サポート関数：インデックスアクセスメソッド毎に必要な内部処理用の関数
    - 例）Btree インデックスの場合、2つの値を比較した結果を返す関数が必要

例） btree, int4 型

演算子クラス:
int4_ops
< (int4,int4)
<= (int4,int4)
= (int4,int4)
>= (int4,int4)
> (int4,int4)

```
SELECT * FROM test1 WHERE id = 42;
```

= (int4,int4)

含まれる

この演算子ならば、演算子クラス int4_ops のインデックスが使用可能！

© 2024 SRA OSS K.K.

演算子族

- 演算子族
  - 互換性のある演算子クラス + 異なる型をまたがる演算子（+サポート関数）の集まり
    = インデックスが使用可能な演算子全体の集まり

```
SELECT * FROM test1 WHERE id = 4200000000;
```

= (int4,int8)

例）btree, 整数型

演算子族: integer_ops

演算子クラス: int2_ops	演算子クラス: int4_ops	演算子クラス: int8_ops
< (int2,int2)	< (int4,int4)	< (int8,int8)
<= (int2,int2)	<= (int4,int4)	<= (int8,int8)
= (int2,int2)	= (int4,int4)	= (int8,int8)
>= (int2,int2)	>= (int4,int4)	>= (int8,int8)
> (int2,int2)	> (int4,int4)	> (int8,int8)

互換性のある異なる型をまたがる演算子

< (int4,int2)	< (int4,int8)	< (int2,int4)	…
<= (int4,int2)	<= (int4,int8)	<= (int2,int4)	…
= (int4,int2)	**= (int4,int8)**	= (int2,int4)	…
>= (int4,int2)	>= (int4,int8)	>= (int2,int4)	…
> (int4,int2)	> (int4,int8)	> (int2,int4)	…

演算子族 integer_ops のインデックスが使用可能

© 2024 SRA OSS K.K.

演算子クラス:ユーザ定義の型や演算子を作る場合(1)

- 新しいユーザ型(基本型)
  - CREATE TYPE 文を使って作成
    - 例)複素数を表す complex 型
    - 「文字列表現」と「内部表現」を相互に変換する関数を作成する必要がある　(詳細は省略!)
- 新しい演算子
  - CREATE OPERATOR 文を使って作成
    - 例)
      - complex 同士の絶対値の比較 (< <=, = > >=)
    - 実際の演算、比較、選択率計算などを行う関数を作成する必要がある　(詳細は省略!!)

→ しかし、これだけでは比較演算子でインデックスが使えない

```
CREATE TYPE complex (
    internallength = 16,
    input = complex_in,
    output = complex_out,
    receive = complex_recv,
    send = complex_send,
    alignment = double
);
```

```
CREATE OPERATOR < (
    leftarg = complex,
    rightarg = complex,
    procedure = complex_abs_lt,
    commutator = > ,
    negator = >= ,
    restrict = scalarltsel,
    join = scalarltjoinsel
);
```

© 2024 SRA OSS K.K.

演算子クラス:ユーザ定義の型や演算子を作る場合(2)

- 演算子クラスの作成
  - complex 同士の絶対値の比較演算子 (< <=, = > >=) でインデックスを使えるようにするには、さらに演算子クラスを作成する必要がある
  - CREATE OPERATOR CLASS 文を使って作成
    - 演算子クラスに含まれる演算子と、サポート関数を指定する

→ これでインデックスが使用可能になる

演算子クラス:
complex_abs_ops

< (compex,compex)

<= (complex,complex)
= (complex,complex)
>= (complex,complex)
> (complex,complex)

```
CREATE OPERATOR CLASS complex_abs_ops
    DEFAULT FOR TYPE complex USING btree AS
        OPERATOR  1  < ,
        OPERATOR  2  <= ,
        OPERATOR  3  = ,
        OPERATOR  4  >= ,
        OPERATOR  5  > ,
        FUNCTION  1  complex_abs_cmp(complex, complex);
```

© 2024 SRA OSS K.K.

演算子クラス:ユーザ定義の型や演算子を作る場合(3)

- 演算子クラスを指定してインデックスを作成

```
CREATE TABLE test_complex (a complex, b complex);

CREATE INDEX test_cplx_ind ON test_complex
   USING btree(a complex_abs_ops);
```

演算子クラス:
complex_abs_ops

< (compex,compex)

<= (complex,complex)

= (complex,complex)

>= (complex,complex)

> (complex,complex)

```
SELECT * FROM test_complex WHERE a = '(56.0,-22.5)';
```

= (complex,complex)

含まれる

演算子クラス complex_abs_ops のインデックスが使用可能！

© 2024 SRA OSS K.K.

インデックスを使ったクエリ最適化

© 2024 SRA OSS K.K.

インデックスを使ったクエリ最適化（1）

- Btree インデックスの内部でキー値がソート順で格納されていることを利用した最適化
  - ORDER BY
    - インデックスのエントリを順番に出力することで、明示的なソートが不要となる
    - 降順（DESC）での出力も可能

```
EXPLAIN (costs off) SELECT * FROM tbl ORDER BY i DESC;

                QUERY PLAN
------------------------------------------
 Index Scan Backward using tbl_i_idx on tbl
(1 row)
```

© 2024 SRA OSS K.K.

インデックスを使ったクエリ最適化（2）

- Btree インデックスの内部でキー値がソート順で格納されていることを利用した最適化
  - min/max
    - 一番最初/最後のインデックスエントリのキー値を出力することで、集約が必要なくなる

```
SELECT min(i) FROM tbl
  WHERE i < 100;
```

→

```
SELECT (SELECT i FROM tbl
         WHERE i IS NOT NULL AND i < 100
         LIMIT 1);
```

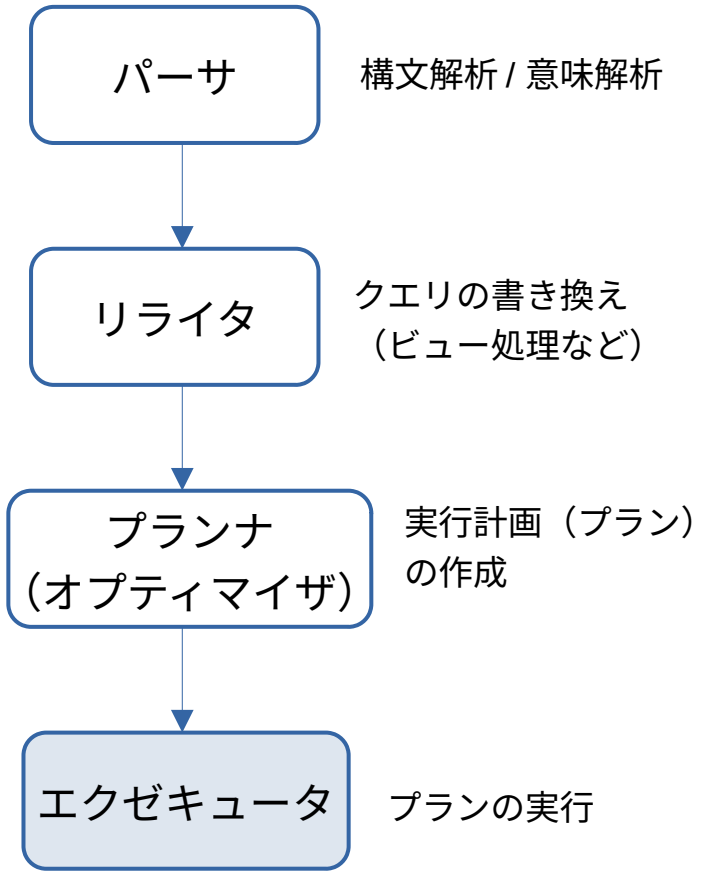
```
EXPLAIN (costs off) SELECT max(i) FROM tbl WHERE i > 100;

                          QUERY PLAN
---------------------------------------------------------------
 Result
   InitPlan 1
     ->  Limit
           ->  Index Only Scan Backward using tbl_i_idx on tbl
                 Index Cond: ((i IS NOT NULL) AND (i < 100))
(5 rows)
```

© 2024 SRA OSS K.K.

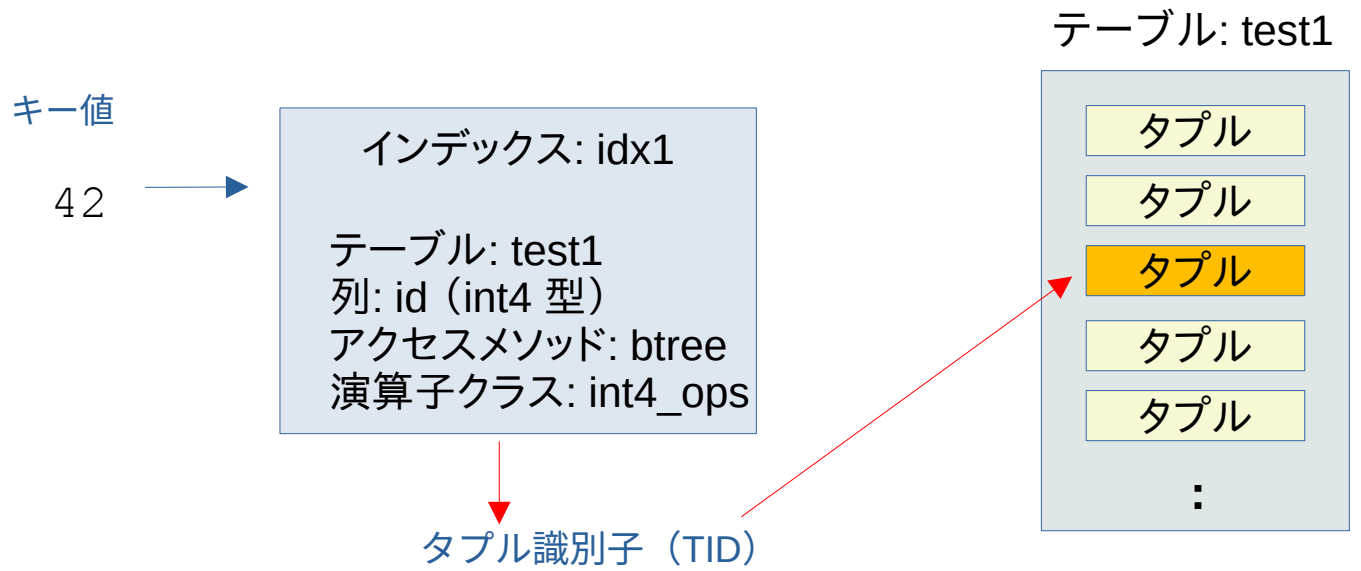
クエリ実行の流れから見たインデックス（再開）

© 2024 SRA OSS K.K.

クエリ実行の流れ（4）：エクゼキュータ

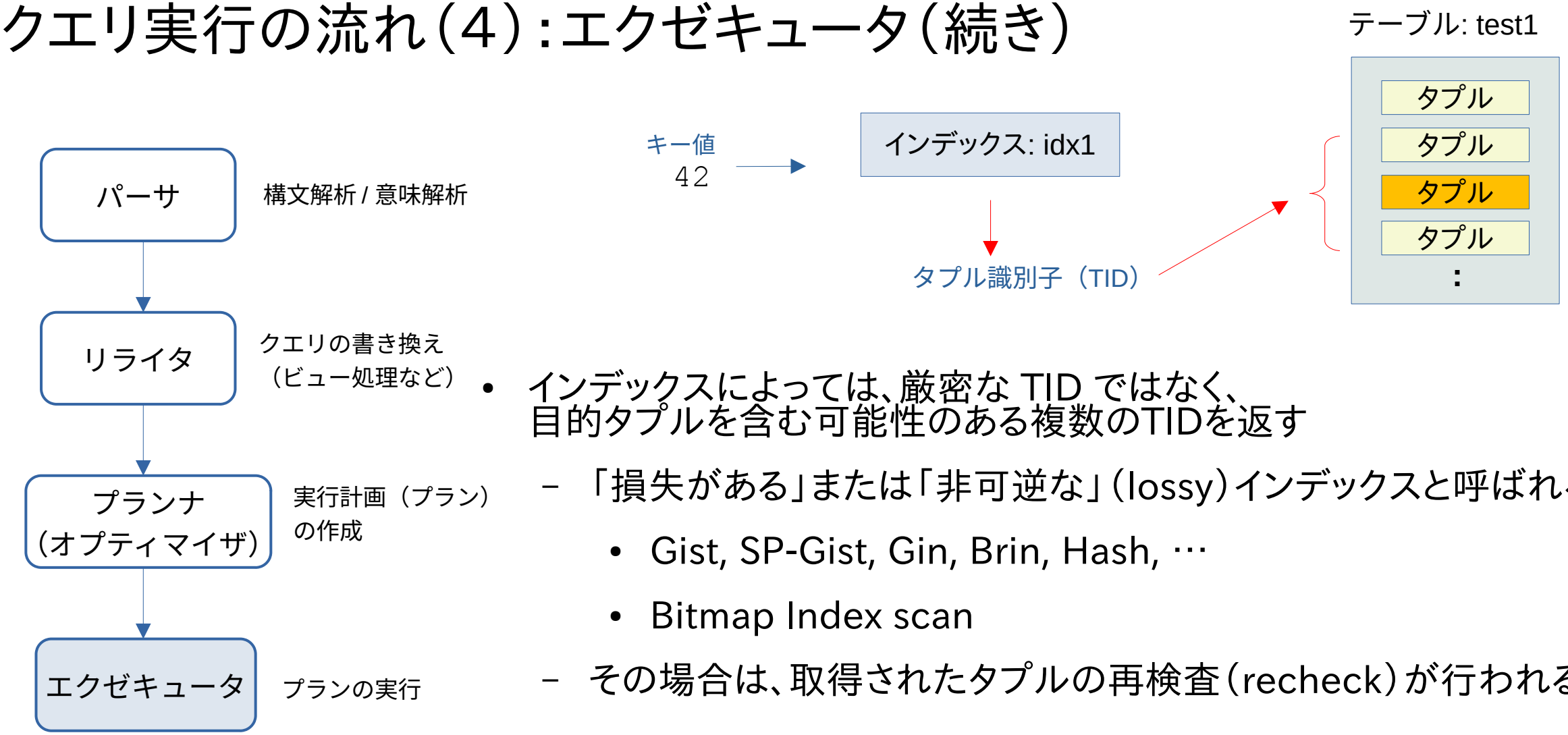


- 実行計画」（プランツリー）を実行してテーブルから行を取得する
  - インデックススキャンの場合：
    - キー値からタプルを特定する識別子を取得



© 2024 SRA OSS K.K.

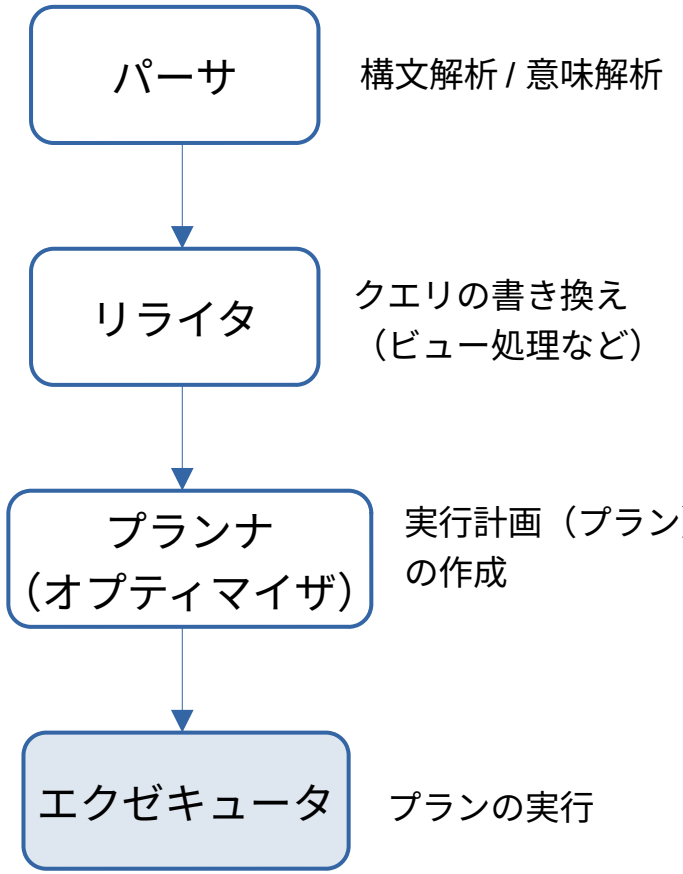
クエリ実行の流れ（4）：エクゼキュータ（続き）

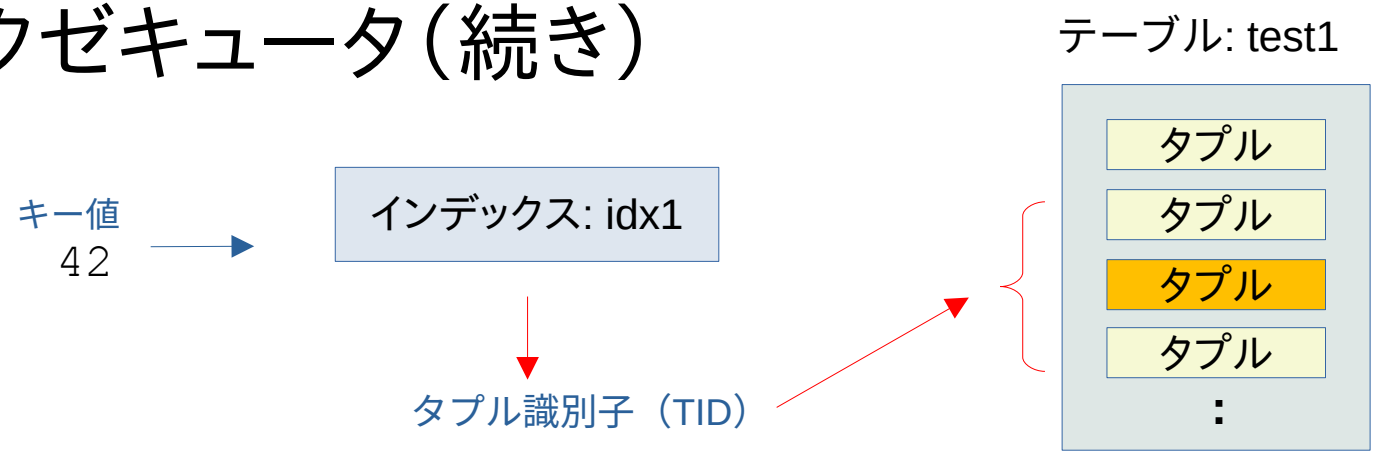


- インデックスによっては、厳密な TID ではなく、
  目的タプルを含む可能性のある複数のTIDを返す
  - 「損失がある」または「非可逆な」（lossy）インデックスと呼ばれる
    - Gist, SP-Gist, Gin, Brin, Hash, …
    - Bitmap Index scan
  - その場合は、取得されたタプルの再検査（recheck）が行われる

© 2024 SRA OSS K.K.

クエリ実行の流れ（4）：エクゼキュータ（続き）





- 再検査（recheck）：Brin インデックスの例

```
                        QUERY PLAN
-----------------------------------------------------
 Bitmap Heap Scan on brin_test
   Recheck Cond: (a = 1)
   ->  Bitmap Index Scan on brin_test_a_idx
         Index Cond: (a = 1)
(4 rows)
```

© 2024 SRA OSS K.K.

インデックスが使われない

© 2024 SRA OSS K.K.

インデックスが使われない（1）

主にプランナ（オプティマイザ）の過程に起因

- インデックススキャンプランが選択されない
  - 例）順スキャンの方が推定コストが低い
- 条件式の形
  - 例）インデックス列の式が使われている（式インデックスではない場合）
- 型の不一致
  - 条件式で使用されている演算子がインデックスの演算子族に含まれていない
- 照合順序の不一致
  - インデックスは列当たり1つの照合順序のみをサポート
- 行レベルセキュリティ（RLS）
  - 条件式で使用されている演算子や関数がLEAKPROOFでない

© 2024 SRA OSS K.K.

インデックスが使われない（2）

- 条件式の形
  - 例）式が使われている
    - 式インデックスが定義されていればインデックススキャンが使用可能

```
SELECT * FROM test1 WHERE id + 1 = 42;
```

```
                          QUERY PLAN
-----------------------------------------------------------
 Seq Scan on test1  (cost=0.00..20.00 rows=5 width=8)
   Filter: ((id + 1) = 42)
(2 rows)
```

© 2024 SRA OSS K.K.

インデックスが使われない(3)

- 型の不一致
  - 条件式で使用されている演算子がインデックスの演算子族に含まれていない

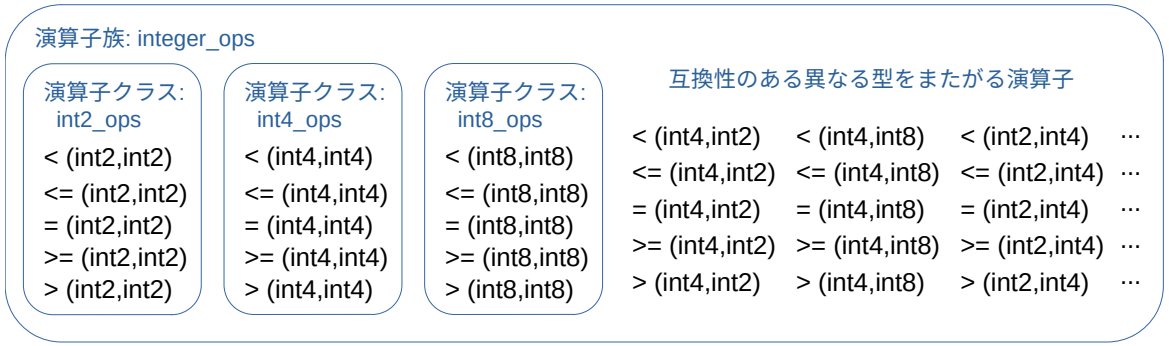
int4　numeric

```
SELECT * FROM test1 WHERE id = 42.0;
```

暗黙の型キャスト

= (numeric,numeric)

演算子族： integer_ops




```
                        QUERY PLAN
-----------------------------------------------------------
 Seq Scan on test1  (cost=0.00..20.00 rows=5 width=8)
   Filter: ((id)::numeric = 42.0)
(2 rows)
```

© 2024 SRA OSS K.K.

インデックスが使われない(4)

- 型の不一致
  - 条件式で使用されている演算子がインデックスの演算子族に含まれていない

```
CREATE TABLE tbl_float (x float);
CREATE INDEX ON tbl_float(x);

SELECT * FROM tbl_float WHERE x = 42.0::numeric;
```

float

numeric

演算子族： float_ops

含まれている

暗黙の型キャスト

= (double precision,double precision)

※ double precision = float

```
                        QUERY PLAN
-----------------------------------------------------------
 Index Only Scan using tbl_float_x_idx on tbl_float
   Index Cond: (x = '42'::double precision)
(2 rows)
```

この場合はインデックスが使える！

© 2024 SRA OSS K.K.

インデックスが使われない（5）

- 型の不一致
  - 条件式で使用されている演算子がインデックスの演算子族に含まれていない

```
CREATE TABLE tbl_numeric (x numeric);
CREATE INDEX ON tbl_float(x);

SELECT * FROM tbl_numeric WHERE x = 42.0::float;
```

numeric

float

演算子族： numeric_ops

含まれていない

暗黙の型キャスト

= (double precision,double precision)

※ double precision = float

```
                          QUERY PLAN
-----------------------------------------------------------
 Seq Scan on tbl_numeric
   Filter: ((x)::double precision = '42'::double precision)
```

この場合はインデックスが使えない！！

© 2024 SRA OSS K.K.

インデックスが使われない（6）

- 照合順序の不一致
  - インデックスは列当たり1つの照合順序のみをサポート

```
CREATE TABLE tbl (t text COLLATE "C");
```

```
CREATE INDEX ON tbl (t COLLATE "ja_JP");
```

```
SELECT * FROM tbl WHERE t = 'abc';
```

```
                QUERY PLAN
-------------------------------------
 Seq Scan on tbl
   Filter: (t = 'abc'::text)
(2 rows)
```

照合順序 C のカラムに対して、
照合順序 ja_JP のインデックスは使用できない

© 2024 SRA OSS K.K.

インデックスが使われない(7)

- 照合順序の不一致
  - インデックスは列当たり1つの照合順序のみをサポート

```
CREATE TABLE tbl (t text COLLATE "C");
```

```
CREATE INDEX ON tbl (t COLLATE "ja_JP");
```

```
SELECT * FROM tbl WHERE t = 'abc' COLLATE "ja_JP";
```

```
                        QUERY PLAN
-----------------------------------------------------
 Index Only Scan using tbl_t_idx1 on tbl
   Index Cond: (t = 'abc'::text COLLATE "ja_JP")
(2 rows)
```

検索時に照合順序を
明示的に指定すると
インデックスが使用可能

© 2024 SRA OSS K.K.

インデックスが使われない（8）

- 行レベルセキュリティ（RLS）
  - 条件式で使用されている演算子や関数がLEAKPROOFでない
    - LEAKPROOF = エラーメッセージなどでデータの情報を漏らす可能性がない

```
CREATE POLICY account_managers ON accounts
    USING (manager = current_user);
```

情報保護のため、まずこのポリシー条件が優先的に適用される

```
CREATE INDEX on accounts ((textlen(company)));
SELECT * FROM accounts WHERE textlen(company) > 10;
```

関数 textlen が LEAKPROOF でないため、この条件は後回し
この式インデックスは使用されない

© 2024 SRA OSS K.K.

インデックスが使われない（9）

- 行レベルセキュリティ（RLS）

```
CREATE POLICY account_managers ON accounts
    USING (manager = current_user);
```

情報保護のため、このポリシー条件が優先的に適用されるが・・・

```
CREATE INDEX on accounts (company);
SELECT * FROM accounts WHERE company = 'SRA OSS';
```

この演算子（ =(text,text)）は LEAKPROOF で情報を漏らす心配がないため、
ポリシー条件の適用に先立ってインデックススキャンが適用可能

© 2024 SRA OSS K.K.

インデックスが使われない（10）

- その他
  - セキュリティバリアビュー
    - 特定の行を隠蔽するためのビュー

```
CREATE VIEW phone_number WITH (security_barrier) AS
    SELECT person, phone FROM phone_data
      WHERE phone NOT LIKE '412%';
```

    - Row Level Secuity と同様、LEAKPROOF ではない演算子・関数を使っている検索条件に対してはインデックスが使用されない

  - LIKE演算子を使った前方一致検索
    - インデックスを使用可能するには、以下のいずれかが必要
      - 照合順序を “C” にする
      - text_pattern_ops 演算子クラスを使う
      - SP-Gist を使う

© 2024 SRA OSS K.K.

まとめ:クエリ実行の流れとインデックス

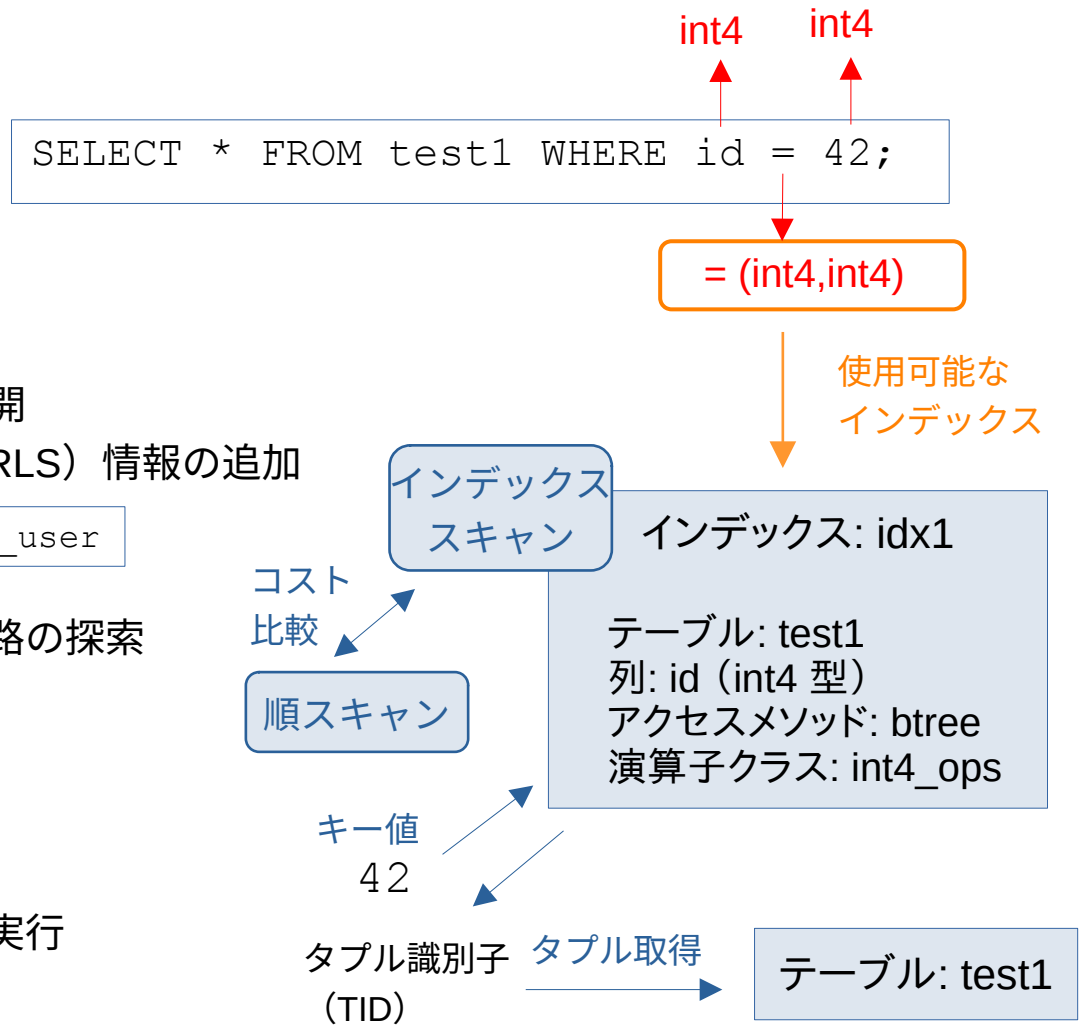
パーサ — 構文解析 / 意味解析
- 条件式の解析
- 型キャスト
- 値の型や演算子の決定

リライタ — クエリの書き換え（ビュー処理など）
- ビューをサブクエリに展開
- 行レベルセキュリティ（RLS）情報の追加

例: `user = current_user`

プランナ（オプティマイザ） — 実行計画（プラン）の作成
- インデックスを用いた経路の探索
- ORDER BY, min/max
- 選択率とコストの計算

エクゼキュータ — プランの実行
- インデックススキャンの実行



© 2024 SRA OSS K.K.

まとめ：演算子クラスとインデックス

- 演算子クラス
  - ある型についてインデックスを使用可能な演算子（+サポート関数）の集まり
- 演算子族
  - 互換性のある演算子クラス + 異なる型をまたがる演算子（+サポート関数）の集まり
    = インデックスが使用可能な演算子全体の集まり

例）btree, 整数型

演算子族: integer_ops

演算子クラス: int2_ops	演算子クラス: int4_ops	演算子クラス: int8_ops
< (int2,int2)	< (int4,int4)	< (int8,int8)
<= (int2,int2)	<= (int4,int4)	<= (int8,int8)
= (int2,int2)	= (int4,int4)	= (int8,int8)
>= (int2,int2)	>= (int4,int4)	>= (int8,int8)
> (int2,int2)	> (int4,int4)	> (int8,int8)

互換性のある異なる型をまたがる演算子

< (int2,int4)	< (int2,int8)	< (int4,int2)	…
<= (int2,int4)	<= (int2,int8)	<= (int4,int2)	…
= (int2,int4)	= (int2,int8)	= (int4,int2)	…
>= (int2,int4)	>= (int2,int8)	>= (int4,int2)	…
> (int2,int4)	> (int2,int8)	> (int4,int2)	…

© 2024 SRA OSS K.K.

まとめ:インデックスが使われない

- インデックススキャンプランが選択されない
  - 例)順スキャンの方が推定コストが低い
- 条件式の形
  - 例)インデックス列の式が使われている(式インデックスではない場合)

```
SELECT * FROM test1 WHERE id + 1 = 42;
```

```
                        QUERY PLAN
-----------------------------------------------------------
 Seq Scan on test1  (cost=0.00..20.00 rows=5 width=8)
   Filter: ((id + 1) = 42)
(2 rows)
```

- 型の不一致
  - 条件式で使用されている演算子がインデックスの演算子族に含まれていない
  - 例)

```
SELECT * FROM test1 WHERE id = 42.0;
```

```
                        QUERY PLAN
-----------------------------------------------------------
 Seq Scan on test1  (cost=0.00..20.00 rows=5 width=8)
   Filter: ((id)::numeric = 42.0)
(2 rows)
```

- 照合順序の不一致
  - インデックスは列当たり1つの照合順序のみをサポート
- 行レベルセキュリティ(RLS)
  - 条件式で使用されている演算子や関数がLEAKPROOFでない
- その他
  - セキュリティバリアビュー、LIKE 演算子

© 2024 SRA OSS K.K.

Thank you!

© 2024 SRA OSS K.K.