第44回PostgreSQLアンカンファレンス

oracle_fdwのソースコードを読んでみた

日本電子計算株式会社
技術本部クラウド技術部　古川　一之輔
2023年12月26日

自己紹介

・名前

古川 一之輔（フルカワ カズノスケ）

・所属

日本電子計算株式会社

技術本部 クラウド技術部 クラウド構築担当

・業務

データベースマイグレーション支援

IaC導入支援　Etc.

本日のネタ

oracle_fdwのソースコードを読んでみた

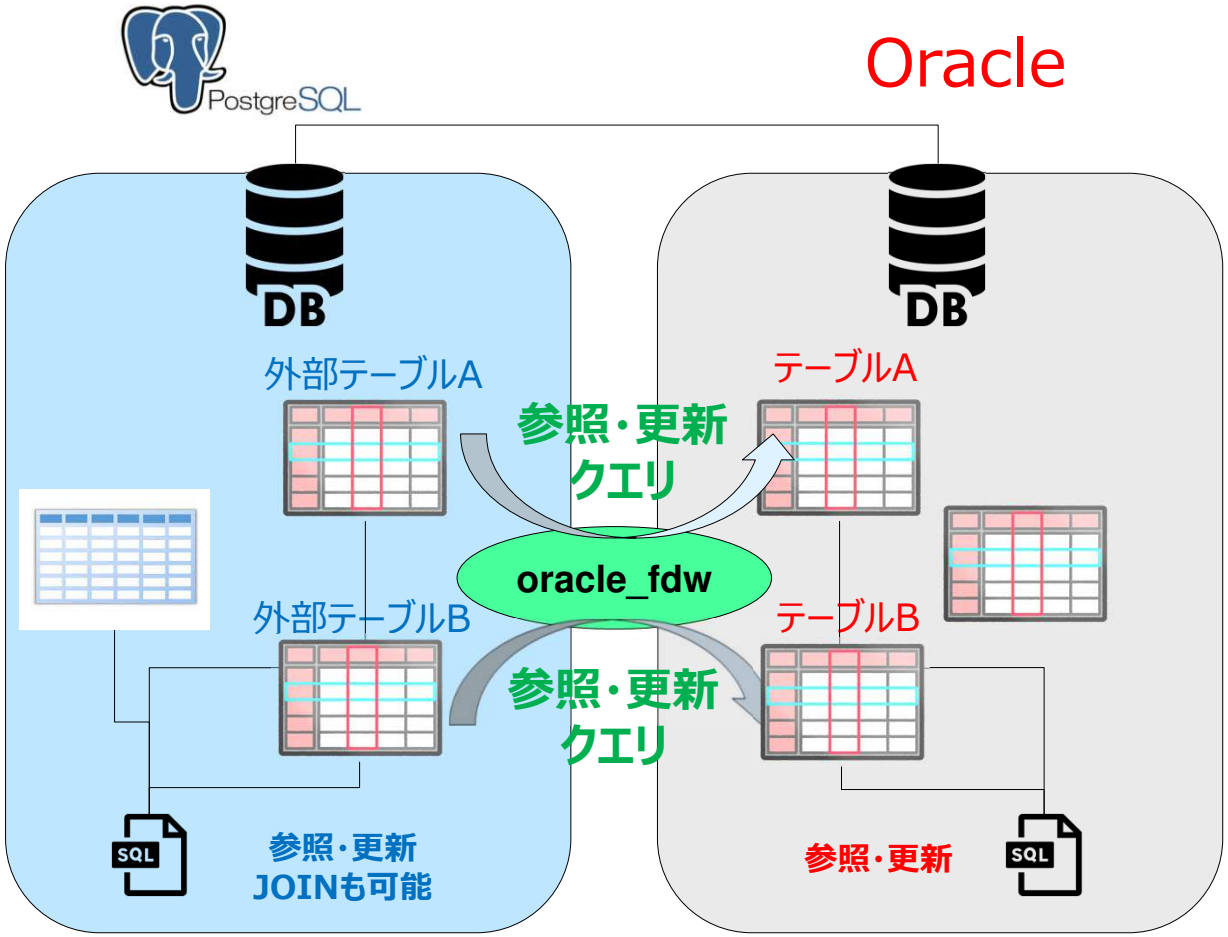
どのようにOracleへクエリを実行しているのかが見えてきた

特にプッシュダウンの発動条件を中心に読んでみた

発動条件を知ることでパフォーマンスチェック時の留意事項が分かるかも

oracle_fdwとは

SQLを利用してOracle内のデータにアクセスできるようにするためのPostgreSQLの拡張機能



OracleテーブルをPostgreSQL外部テーブルとして定義

外部テーブルにアクセスすると自動的にOracleへクエリが実行され結果がPostgreSQLに帰ってくる

【プッシュダウンとは】

クライアントから問合せのあったSQL文に含まれるWHERE句（検索条件）、ORDER BY句（ソート条件）、および、JOIN句（結合条件）の**処理をOracle側で実行させること。**なおWHERE句とJOIN句については、**ローカルとリモートとの間のデータ転送量を抑え**、通信におけるボトルネックを減少させる効果がある。

きっかけ

自社にPostgreSQLの利用を普及し根付かせたい

基幹システムのRDBMSはOracleの採用がほとんど

周辺システムでPostgreSQLを使うのはどうか

基幹システムとインターフェースを介して簡単に連携出来れば。。。

⇒ oracle_fdwがあるじゃないか！

ソースを読んでみた

どうやってOracleへクエリを投げているか

⇒

deparseExpr()という関数でOracleへのクエリを作っている

一度PostgreSQL内でparseされたSQL文をdeparseしてOracleに送るSQLを構築

parse：分解・解析

deparse：再構成

ゆえに

PostgreSQLとしてparseを通ったSQL文でないと実行されない

OracleにあってPostgreSQLに無い関数や処理はエラーとなる

プッシュダウン発動条件をまとめてみた

SELECT a.id, b.name FROM

f_tbl1 a JOIN f_tbl2 b ON a.id = b.id

INNER/OUTER JOIN 以外は発動しない
すべてoracle_fdwテーブルかつ2テーブル以下でないと発動しない
Cross Join は発動しない

WHERE a.id > to_number(b.cd)

oracle_fdwテーブルは全てOracleのSQLに変換され発動する
関数などは内部でOracleの関数に変換される（substring⇒SUBSTR等）
OracleとPostgreSQLで動作仕様が違う関数はOracleの仕様で動作する

ORDER BY a.id

以下のカラムがOrderByに設定された場合のみ発動
数値 int float numeric
日付/時刻 date timestamp 等

Limit 5;

GroupBy句や集計関数が無いかつWhere句がすべてプッシュダウン可能な場合発動
Limit値が無い（Limit All）場合発動しない
Offset値は設定されていれば加味される

プッシュダウン発動条件を検証してみた

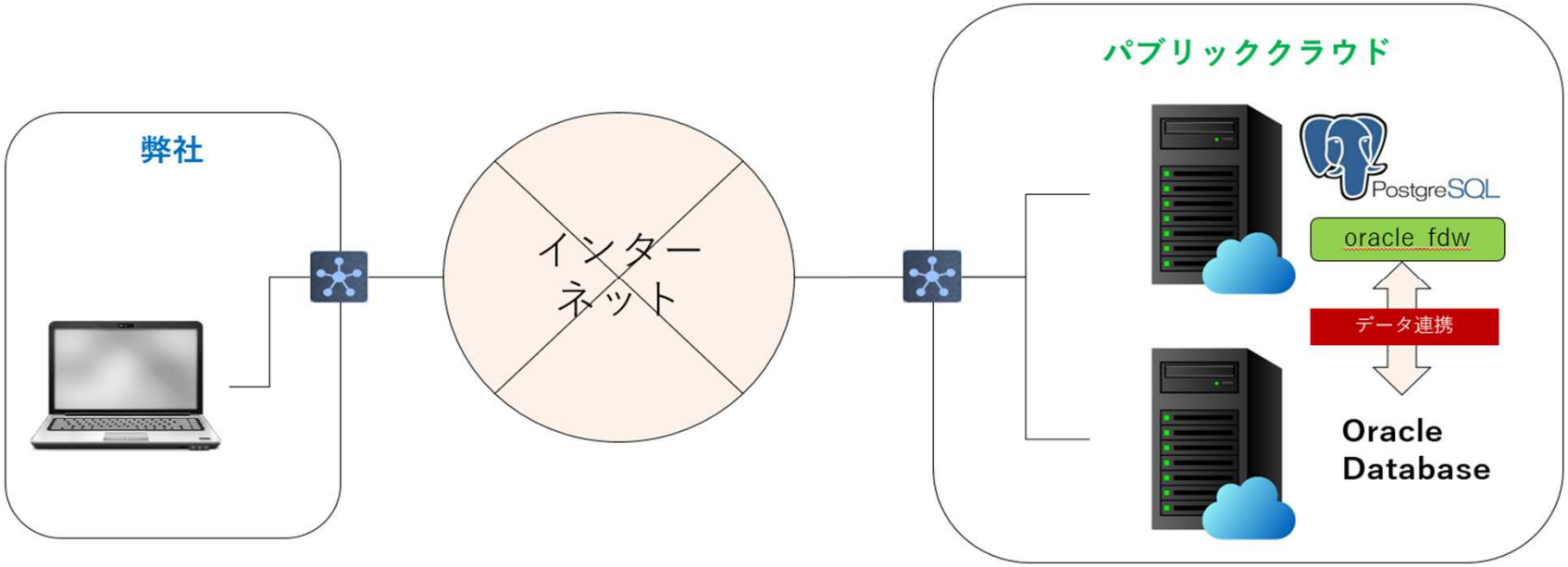
本当にそう動くのか、こうした場合は発動するのか、という観点で検証してみた

⇒EXPLAIN ANALYZE VERBOSE で取得した実行計画からプッシュダウン発動有無を判断

例）Order By 句のプッシュダウン発動検証
Oracleで実行されるSQLが記載されるOracle query:の箇所に**ORDER BY句が記載**されている。
Oracle側の実行計画であるOracle plan:の箇所に**「SORT ORDERY BY」と記載**されている。
以上より当該SQLのORDER BYは**プッシュダウンが発動**しOracle側で処理される、と判断できる。

```
db1=> EXPLAIN ANALYZE VERBOSE select * from ftbl_ora_pfm3 order by id, upd_date;
                                                    QUERY PLAN
------------------------------------------------------------------------------------------------------------------
 Foreign Scan on pguser.ftbl_ora_pfm3  (cost=10000.00..20000.00 rows=1000 width=230) (actual time=1.121..1.150 rows=6
loops=1)
   Output: id, item, upd_date
   Oracle query: SELECT r1."ID", r1."ITEM", r1."UPD_DATE" FROM "ORAUSER"."TBL_ORA_PFM3" r1 ORDER BY r1."ID" ASC NULLS LAST,
r1."UPD_DATE" ASC NULLS LAST
   Oracle plan: SELECT STATEMENT
   Oracle plan:   SORT ORDER BY
   Oracle plan:     TABLE ACCESS FULL TBL_ORA_PFM3
 Planning Time: 1.048 ms
 Execution Time: 1.179 ms
```

検証に使用した環境構成



パブリッククラウド：AWS EC2 インスタンス × 2
Oracle バージョン：23c Free　PostgreSQL バージョン：15.2
oracle_fdw バージョン：2.5.0　pg_hint_plan バージョン：15.1.5.1

⇒対象は2023年04月時点の最新バージョン
⇒PostgreSQL および oracle_fdw はソースからコンパイルしインストール
⇒対応するOracleクライアントが存在しないため、23cFreeのクライアントライブラリを使用して構築

Limit句のプッシュダウン発動検証

Limit句にLimit値を伴わないSQL文のプッシュダウン発動有無

Oracleで実行されるSQLが記載されるOracle query:の箇所に**Limit Allの記載が無い**。
Oracle側の実行計画であるOracle plan:の箇所に**Limit関連のと記載が無い**。
以上より当該SQLのLimitは**プッシュダウンが発動しなかった**、と判断できる。

```
db1=> EXPLAIN ANALYZE VERBOSE select * from ftbl_ora_pfm3 limit all;
                                                          QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------
 Foreign Scan on pguser.ftbl_ora_pfm3  (cost=10000.00..20000.00 rows=1000 width=230) (actual time=0.709..0.737 rows=6 loops=1)
   Output: id, item, upd_date
   Oracle query: SELECT /*692c58b1c18f98bb*/ r1."ID", r1."ITEM", r1."UPD_DATE" FROM "ORAUSER"."TBL_ORA_PFM3" r1
   Oracle plan: SELECT STATEMENT
   Oracle plan:   TABLE ACCESS FULL TBL_ORA_PFM3
 Planning Time: 1.230 ms
 Execution Time: 0.771 ms
(7 rows)
```

Join句のプッシュダウン発動検証

Join句を Cross Join としたSQL文のプッシュダウン発動有無

PostgreSQLの処理として行のJoinに必要とされる**Nested Loop 処理の記載**がある。
Oracleで実行されるSQLが記載されるOracle query:の箇所に**Join句の記載が無い**。
Oracle側の実行計画であるOracle plan:の箇所に**Joinに必要とされる処理の記載が無い**。
以上より当該SQLのJoinは**プッシュダウンが発動しなかった**、と判断できる。

```
db1=> EXPLAIN ANALYZE VERBOSE select c.id, c.item, d.item from ftbl_ora_pfm3 c cross join ftbl_ora_pfm4 d;
                                                               QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------------------
 Nested Loop  (cost=20000.00..20090.23 rows=18 width=46) (actual time=1.343..1.378 rows=18 loops=1)
   Output: c.id, c.item, d.item
   ->  Foreign Scan on pguser.ftbl_ora_pfm3 c  (cost=10000.00..10060.00 rows=6 width=25) (actual time=0.693..0.706 rows=6 loops=1)
         Output: c.id, c.item, c.upd_date
         Oracle query: SELECT /*1172644a0eb88162*/ r1."ID", r1."ITEM" FROM "ORAUSER"."TBL_ORA_PFM3" r1
         Oracle plan: SELECT STATEMENT
         Oracle plan:   TABLE ACCESS FULL TBL_ORA_PFM3
   ->  Materialize  (cost=10000.00..10030.01 rows=3 width=21) (actual time=0.108..0.111 rows=3 loops=6)
         Output: d.item
         ->  Foreign Scan on pguser.ftbl_ora_pfm4 d  (cost=10000.00..10030.00 rows=3 width=21) (actual time=0.642..0.657 rows=3 loops=1)
               Output: d.item
               Oracle query: SELECT /*3ea2fc6947453145*/ r2."ITEM" FROM "ORAUSER"."TBL_ORA_PFM4" r2
               Oracle plan: SELECT STATEMENT
               Oracle plan:   TABLE ACCESS FULL TBL_ORA_PFM4
 Planning Time: 1.604 ms
 Execution Time: 1.424 ms
```

【番外編】関数を伴うSelect句のプッシュダウン発動検証

単独外部テーブルへのSelect句にsubstring関数を使用したSQL文のプッシュダウン発動有無

PostgreSQLの処理であるOutput:の箇所に**substring関数の記載**。
Oracleで実行されるSQLが記載されるOracle query:の箇所に**substring関数の記載が無い（除去されている）**。
以上より当該SQLのSelect句に設定されたsubstring関数は**プッシュダウンが発動しなかった**、と判断できる。
⇒Oracleで実行されるSQLから除去し、PostgreSQL側で処理させるような仕様となっている？なぜ？
⇒OracleとPostgreSQLで動作仕様が異なる関数があるため、あくまでPostgreSQLの仕様で結果を出力させるため？（想像）

```
db1=> EXPLAIN ANALYZE VERBOSE select a.id, substring(a.item, 5, 3) a_item from ftbl_ora_pfm1 a where a.id < 10;
                                                                                  QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 Foreign Scan on pguser.ftbl_ora_pfm1 a  (cost=10000.00..10100.02 rows=10 width=36) (actual time=0.825..0.849 rows=10 loops=1)
   Output: id, "substring"((item)::text, 5, 3)
   Oracle query: SELECT /*e5876bac84fb5ac8*/ r1."ID", r1."ITEM" FROM "ORAUSER"."TBL_ORA_PFM1" r1 WHERE (r1."ID" < 10)
   Oracle plan: SELECT STATEMENT
   Oracle plan:   TABLE ACCESS BY INDEX ROWID BATCHED TBL_ORA_PFM1
   Oracle plan:     INDEX RANGE SCAN PK_ORA_PFM1 (condition "R1"."ID"<10)
 Planning Time: 1.178 ms
 Execution Time: 0.883 ms
(8 rows)
```

【番外編】ヒントを伴ったSQL文の挙動検証

pg_hint_planを導入しoracle_fdwテーブルを絡めたSQL文のプッシュダウン挙動検証

仮説：

parseしたものをdeparseしてSQLを再構築してOracleに飛ばすのなら
コメントはparseした時点で除外されているはずなのでOracle側にヒントは飛ばないはず
よってoracle_fdwテーブルへのヒント句は効果が無いのでは？

結論：

oracle_fdwテーブル単体（ForeignScanより下のレイヤ）はヒントで変化させることは出来ないが一方でテーブル結合や処理順序の変更などは効果があった

oracle_fdwテーブル単体へのSQLにヒントを追加

ヒントなしでまずは実行

```
db1=> EXPLAIN ANALYZE VERBOSE select * from ftbl_ora_hnt1 a where a.id = 10;
                                                            QUERY PLAN
--------------------------------------------------------------------------------------------------------------------------------
 Foreign Scan on pguser.ftbl_ora_hnt1 a  (cost=10000.00..10010.00 rows=1 width=33) (actual time=0.740..0.751 rows=1 loops=1)
   Output: id, item, upd_date
   Oracle query: SELECT /*e1c8b125f57203da*/ r1."ID", r1."ITEM", r1."UPD_DATE" FROM "ORAUSER"."TBL_ORA_HNT1" r1 WHERE (r1."ID" = 10)
   Oracle plan: SELECT STATEMENT
   Oracle plan:   TABLE ACCESS BY INDEX ROWID TBL_ORA_HNT1
   Oracle plan:     INDEX UNIQUE SCAN PK_ORA_HNT1 (condition "R1"."ID"=10)
 Planning Time: 1.086 ms
 Execution Time: 0.783 ms
(8 rows)
```

Oracle側の実行計画であるOracle plan:の箇所に TABLE ACCESS BY INDEX ROWID とあり、インデックススキャンを行っている。

実行SQLにヒント句 /*+ SeqScan(a) */ を追加し、実行計画が変化するかを確認してみる。

ヒントが効いた場合、該当箇所が TABLE ACCESS FULL に変更されるはず。

oracle_fdwテーブル単体へのSQLにヒントを追加

ヒント /*+ SeqScan(a) */ 付きで実行

```
db1=> EXPLAIN ANALYZE VERBOSE select /*+ SeqScan(a) */ * from ftbl_ora_hnt1 a where a.id = 10;
                                                          QUERY PLAN
--------------------------------------------------------------------------------------------------------------------------------
 Foreign Scan on pguser.ftbl_ora_hnt1 a  (cost=10000.00..10010.00 rows=1 width=33) (actual time=0.566..0.576 rows=1 loops=1)
   Output: id, item, upd_date
   Oracle query: SELECT /*e1c8b125f57203da*/ r1."ID", r1."ITEM", r1."UPD_DATE" FROM "ORAUSER"."TBL_ORA_HNT1" r1 WHERE (r1."ID" = 
10))
   Oracle plan: SELECT STATEMENT
   Oracle plan:   TABLE ACCESS BY INDEX ROWID TBL_ORA_HNT1
   Oracle plan:     INDEX UNIQUE SCAN PK_ORA_HNT1 (condition "R1"."ID"=10)
 Planning Time: 1.037 ms
 Execution Time: 0.611 ms
(8 rows)
```

Oracle側の実行計画であるOracle plan:の **TABLE ACCESS BY INDEX ROWID 変わらず。**
Oracleで実行されるSQLが記載されるOracle query:の箇所に**ヒント句の記載が無い（除去されている）**。
　⇒仮説のとおりPostgreSQL上のparseの段階でコメントとして処理され除外されているように見える。

実行SQLにOracleのヒント句の書き方 /*+ FULL(a) */ に変更し、実行計画が変化するかを確認してみる。

ヒントが効いた場合、該当箇所が TABLE ACCESS FULL に変更されるはず。

oracle_fdwテーブル単体へのSQLにヒントを追加

ヒントをOracleの記法 /*+ FULL(a) */ に変更して実行

```
db1=> EXPLAIN ANALYZE VERBOSE select /*+ FULL(a) */ * from ftbl_ora_hnt1 a where a.id = 10;
INFO:  pg_hint_plan: hint syntax error at or near "FULL(a) "
DETAIL:  Unrecognized hint keyword "FULL".
                                                              QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------------
 Foreign Scan on pguser.ftbl_ora_hnt1 a  (cost=10000.00..10010.00 rows=1 width=33) (actual time=0.650..0.660 rows=1 loops=1)
   Output: id, item, upd_date
   Oracle query: SELECT /*e1c8b125f57203da*/ r1."ID", r1."ITEM", r1."UPD_DATE" FROM "ORAUSER"."TBL_ORA_HNT1" r1 WHERE (r1."ID" =
10))
   Oracle plan: SELECT STATEMENT
   Oracle plan:   TABLE ACCESS BY INDEX ROWID TBL_ORA_HNT1
   Oracle plan:     INDEX UNIQUE SCAN PK_ORA_HNT1 (condition "R1"."ID"=10)
 Planning Time: 0.892 ms
 Execution Time: 0.687 ms
(8 rows)
```

実行計画以前に pg_hit_plan がOracle記法のヒント /*+ FULL(a) */ をシンタックスエラーと判断。
Oracle側の実行計画であるOracle plan:の **TABLE ACCESS BY INDEX ROWID 変わらず。**
Oracleで実行されるSQLが記載されるOracle query:の箇所に**ヒント句の記載が無い（除去されている）**。
⇒仮説のとおりPostgreSQL上のparseの段階でコメントとして処理され除外されているように見える。

つまり、**oracle_fdwテーブル単体（ForeignScanより下のレイヤ）はヒントで変化させることは出来ない**と分かった。

oracle_fdwテーブルとjoinした場合のローカルテーブルのヒント句の挙動

ヒントなしでまずは実行

```
db1=> EXPLAIN ANALYZE VERBOSE select a.id, b.item, c.item from ftbl_ora_hnt1 a join ftbl_ora_hnt2 b on a.id = b.id join
tbl_pg_hnt1 c on a.id= c.id;
                                                            QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------
 Hash Join  (cost=10001.27..10111.30 rows=1 width=43) (actual time=0.859..0.893 rows=10 loops=1)
   Output: a.id, b.item, c.item
   Inner Unique: true
   Hash Cond: (a.id = c.id)
   ->  Foreign Scan  (cost=10000.00..10110.00 rows=11 width=29) (actual time=0.821..0.854 rows=11 loops=1)
         Output: a.id, b.item, b.id
         Oracle query: SELECT /*84935cb46f96da0f*/ r1."ID", r2."ITEM", r2."ID" FROM ("ORAUSER"."TBL_ORA_HNT1" r1 INNER JOIN
"ORAUSER"."TBL_ORA_HNT2" r2 ON (r1."ID" = r2."ID"))
         Oracle plan: SELECT STATEMENT
         Oracle plan:   NESTED LOOPS
         Oracle plan:     TABLE ACCESS FULL TBL_ORA_HNT2
         Oracle plan:     INDEX UNIQUE SCAN PK_ORA_HNT1 (condition "R1"."ID"="R2"."ID")
   ->  Hash  (cost=1.12..1.12 rows=12 width=22) (actual time=0.026..0.026 rows=12 loops=1)
         Output: c.item, c.id
         Buckets: 1024  Batches: 1  Memory Usage: 9kB
         ->  Seq Scan on pguser.tbl_pg_hnt1 c  (cost=0.00..1.12 rows=12 width=22) (actual time=0.008..0.010 rows=12 loops=1)
               Output: c.item, c.id
 Planning Time: 1.728 ms
 Execution Time: 0.936 ms
(18 rows)
```

oracle_fdwテーブルとjoinした場合のローカルテーブルのヒント句の挙動

ヒントなしでまずは実行

Oracleで実行されるSQLが記載されるOracle query:の箇所から、
oracle_fdwテーブル a および b が join 句のプッシュダウンにてOracle内で結合していることが分かる。
ローカルテーブル c は Hash Join にて oracle_fdwテーブル a と結合している。

これらをヒント句 /*+ Leading(((b c) a)) MergeJoin(b c) */ にて実行し、
oracle_fdwテーブル b とローカルテーブル c を先に結合させ、
ローカルテーブル c は Merge Join にて oracle_fdwテーブル b と結合させるようにしてみる。

oracle_fdwテーブルとjoinした場合のローカルテーブルのヒント句の挙動

ヒント /*+ Leading(((b c) a)) MergeJoin(b c) */ 付きで実行

```
db1=> EXPLAIN ANALYZE VERBOSE select /*+ Leading(((b c) a)) MergeJoin(b c) */ a.id, b.item, c.item from ftbl_ora_hnt1 a join
ftbl_ora_hnt2 b on a.id = b.id join tbl_pg_hnt1 c on a.id = c.id;
                                                                    QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------------------------------
```

実行計画は次のスライドで！

実行計画の記載から**oracle_fdwテーブル b とローカルテーブル c の結合**が行われている。

Merge Join および Merge Cond: の記載から**ローカルテーブル c は Merge Join にて oracle_fdw テーブル b と結合**している。

Oracleで実行されるSQLが記載されるOracle query:の箇所から、
oracle_fdwテーブル a および b はOracle内での結合がされず個々にSELECT実行している。

つまり、**テーブル結合や処理順序の変更などはローカルテーブルやoracle_fdwテーブルに関わらずヒント句で制御できる**と分かった。

oracle_fdwテーブルとjoinした場合のローカルテーブルのヒント句の挙動

ヒント /*+ Leading(((b c) a)) MergeJoin(b c) */ 付きで実行

```
db1=> EXPLAIN ANALYZE VERBOSE select /*+ Leading(((b c) a)) MergeJoin(b c) */ a.id, b.item, c.item from ftbl_ora_hnt1 a join
ftbl_ora_hnt2 b on a.id = b.id join tbl_pg_hnt1 c on a.id = c.id;
                                                      QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------
 Nested Loop  (cost=20111.53..21138.66 rows=1 width=43) (actual time=1.309..1.500 rows=10 loops=1)
   Output: a.id, b.item, c.item
   Join Filter: (b.id = a.id)
   Rows Removed by Join Filter: 1000
   ->  Merge Join  (cost=10111.53..10111.74 rows=11 width=47) (actual time=0.734..0.742 rows=10 loops=1)
         Output: b.item, b.id, c.item, c.id
         Inner Unique: true
         Merge Cond: (b.id = c.id)
         ->  Sort  (cost=10110.19..10110.22 rows=11 width=25) (actual time=0.704..0.707 rows=11 loops=1)
               Output: b.item, b.id
               Sort Key: b.id
               Sort Method: quicksort  Memory: 25kB
               ->  Foreign Scan on pguser.ftbl_ora_hnt2 b  (cost=10000.00..10110.00 rows=11 width=25) (actual time=0.673..0.696
rows=11 loops=1)
                     Output: b.item, b.id
                     Oracle query: SELECT /*396ce61e6d875de1*/ r2."ID", r2."ITEM" FROM "ORAUSER"."TBL_ORA_HNT2" r2
                     Oracle plan: SELECT STATEMENT
                     Oracle plan:   TABLE ACCESS FULL TBL_ORA_HNT2
         （続く）
```

oracle_fdwテーブルとjoinした場合のローカルテーブルのヒント句の挙動

ヒント /*+ Leading(((b c) a)) MergeJoin(b c) */ 付きで実行

```
          （続き）

->  Sort  (cost=1.34..1.37 rows=12 width=22) (actual time=0.025..0.026 rows=10 loops=1)
              Output: c.item, c.id
              Sort Key: c.id
              Sort Method: quicksort  Memory: 25kB
              ->  Seq Scan on pguser.tbl_pg_hnt1 c  (cost=0.00..1.12 rows=12 width=22) (actual time=0.010..0.012 rows=12
loops=1)
                    Output: c.item, c.id
   ->  Materialize  (cost=10000.00..11010.50 rows=101 width=4) (actual time=0.057..0.069 rows=101 loops=10)
         Output: a.id
         ->  Foreign Scan on pguser.ftbl_ora_hnt1 a  (cost=10000.00..11010.00 rows=101 width=4) (actual time=0.563..0.616
rows=101 loops=1)
              Output: a.id
              Oracle query: SELECT /*133d18f86329f794*/ r1."ID" FROM "ORAUSER"."TBL_ORA_HNT1" r1
              Oracle plan: SELECT STATEMENT
              Oracle plan:   INDEX FAST FULL SCAN PK_ORA_HNT1
 Planning Time: 1.770 ms
 Execution Time: 1.599 ms
(32 rows)
```

こんなこともありました

Limit句の実行計画取得時エラー「oracleQueryPlan internal error: statement handle is not NULL 」

⇒以下の観点から**oracle_fdwの不具合**ではないか？

・VERBOSEを外すとエラーにならない（代わりにOracle側実行計画であるOracle planの表示はされなくなる）

・Oracle側でSQLトレースを行い、実行されたSQLを取得しOracle側で実行計画を取得すると問題なく取れる

・ローカルテーブルに対するLimit句付きSQLの実行計画は取得できる

Join句のプッシュダウンは条件を合わせても当初発動しなかった

⇒ソースに同梱ののREADME.oracle_fdwを見ると以下の記載あり、テーブル統計を取得再実行で発動するようになった

```
It is important that table statistics for both foreign tables have been collected with ANALYZE for PostgreSQL to determine the best join strategy.
```

訳）

両方の外部テーブルのテーブル統計が有効であることが重要です。ANALYZE for PostgreSQL で収集され、最適な結合戦略が決定されます。

Where句にPostgreSQL上に存在しない関数が使用されていた場合エラー「function func(xxx) does not exist」

⇒Oracleへ送られるSQLは**一度parseされたものからdeparseし再構成して作られる**ことが実証


ご清聴ありがとうございました。