SQL のパフォーマンスを調査するには

準備

```
PLAN_TABLE の作成
1. ユーザー sys でログイン。
2. utlxplan.sql を実行。（オラクルホーム \rdbmsxx\admin\ の下にあります）
3. 次の権限を実行。
    GRANT SELECT ON SYS.PLAN_TABLE TO PUBLIC;
    GRANT INSERT ON SYS.PLAN_TABLE TO PUBLIC;
    GRANT UPDATE ON SYS.PLAN_TABLE TO PUBLIC;
    GRANT DELETE ON SYS.PLAN_TABLE TO PUBLIC;
4. パブリックシノニムの作成。
    CREATE PUBLIC SYNONYM PLAN_TABLE FOR SYS.PLAN_TABLE;
```

調査

```
explain plan  for (select 文 )
```

でパフォーマンス情報を取得し

```
SELECT PLAN_TABLE_OUTPUT FROM TABLE(DBMS_XPLAN.DISPLAY());
```

で情報を表示する。
または、

```
delete from plan_table
/
explain plan set statement_id = '1'
 for
sql 文
/
SELECT LPAD(' ', 4 * (LEVEL - 1)) ||
operation || ' ' || options || ' ' || object_name || ' ' ||
DECODE(id, 0, 'Cost = ' || position) || ' ' || cost AS "Query Plan"
FROM plan_table
START WITH id = 0
CONNECT BY PRIOR id = parent_id
ORDER BY id, position
```

調査２

http://oracle.na7.info/tuning1.html

■索引

テーブル（表）内のデータは、物理入力順に格納されている。
そのため、表内の特定の列に「索引」を作成して、よりよい検索パフォーマンスを引き出すことができる。
●索引には、

1. 単一型索引－－表内の単一の列から構成された索引
2. 連結型索引－－複数の列から構成された索引。最大３２列まで指定できる。

※連結型索引でも、先頭列だけは単一型索引と同様に使用される。

索引１	地域コード	部署コード　←地域コードをキーに検索すると、索引１が使用される
索引２	地域コード	―

●索引列選択のガイドライン

1. **WHERE 句の条件として頻繁に使用する列で、かつ検索対象となる行数が表全体に占める割合が低い場合**
2. **列の値が比較的一意な列**
3. **表の結合で使用する列**

※オラクルでは、表に対して作成できる索引の数に制限はない。
しかし、更新頻度が高い表では、同時に索引も更新されるためオーバヘッドが大きくなるので、考慮が必要である。

■表の結合操作

Oracle では、複数の表を結合する際に以下の結合方法の１つを選択し、実行する。

1. **ネストループ結合**
2. **ソートマージ結合**
3. **ハッシュ結合**

ソート種別	内部動作
ネストループ結合	オプティマイザが、どちらかひとつの表を「外部表」として選択し、もう一方を「内部表」とする。外部表の各行について、結合条件を満たす内部表の行を見つけるを満たす各行のペアでデータを結合し、結果を返すの場合、各表に対応するアクセス・パスのランクが低いほうが外部表になる
ソートマージ結合	結合条件で使用される列をソートキーとして、各表をソートする２つの表をマージして、結果を戻す。※ソート処理がメモリで行なえる程度のデータ量だと速い
ハッシュ結合	一方の表をハッシュ関数を用いてメモリ上にロードし、ハッシュテーブルを作成するハッシュテーブルを利用して、ディスク上のもう一方の表の各行に対応する行を結合し、結果を返す。※ハッシュ結合は、コストベースの時で、各表が等価結合のときのみ使用される

■ SQL 文の診断

１．オプティマイザ

SQL 文の実行時、表への最適なアクセス方法（実行計画）を選択する機能。
DML 文 (SELECT、INSERT、UPDATE、DELETE) を発行する際にもっとも効率よく表にアクセスする方法を選択してくれる

オプティマイザの設定は、初期化パラメータ「OPTIMIZER_MODE」で指定する。
●ルールベース・アプローチ

OPTIMIZER_MODE=RULE

オプティマイザは SQL 構文と表、索引定義を情報として、アクセス･パスの優先順位に基づいて実行計画を作成する。
WHERE 文中の条件が複数ある場合、もっとも優先順位の高いアクセス・パスを採用する。

ルール・ベースによるアクセスパスの採用順位ランク	アクセス・パス
1	ROWID による単一行
2	クラスタ結合による単一行
3	一意キーまたは主キーを持つハッシュ・クラスタ・キーによる単一行
4	一意キーまたは主キーによる単一行
5	クラスタ結合
6	ハッシュ・クラスタ・キー
7	索引付きクラスタ・キー
8	複合索引
9	単一列索引
10	索引列の境界付きの範囲検索（＜＞、BETWEEN）
11	索引列の境界無しの範囲検索（＜、または＞のみ）
12	ソート／マージ結合
13	索引付き列の MAX または MIN
14	索引付き列の ORDER BY
15	全表走査

実行例

```
SELECT * FROM EMP  ←（ランク 15：全表走査）

  WHERE EMPNO = 1   ←（ランク 4：一意キーまたは主キーによる単一行）
    AND DEPTNO > 20;   ←（ランク 11：索引列の境界無しの範囲検索）
```

アクセス・パス⇒４を採用

●コストベース・アプローチ

OPTIMIZER_MODE=CHOOSE

オプティマイザは ANALYZE 文を使用して収集した統計情報に基づいて、最適なアクセス・パスを採用する。
初期化パラメータでコストベースが選択されていても、統計情報が無ければルールベースで実行計画が立てられる。

計算（全表走査）による統計情報の取得

```
ANALYZE TABLE 表名 COMPUTE STATISTICS;
例）ANALYZE TABLE emp COMPUTE STATISTICS;
```

推定による統計情報の取得

```
ANALYZE TABLE 表名 ESTIMATE STATISTICS
        [ SAMPLE n ROWS | PERCENT ]
                （デフォルト値は 1064 行）
```

統計情報の削除

```
ANALYZE TABLE 表名 DELETE STATISTICS;
```

■ SQL 文の改良

１．索引を使用する SQL 文を記述する

検索で実際に使用される索引は、Explain Plan を用いて検証する。実行計画を見て、索引を効率的に使用していない場合は、以下の指標に沿ってＳＱＬ文を改善するとよい。

●索引列での計算の回避

```
×  SELECT EMPNAME,SAL FROM EMP WHERE SAL * 12 > 5000000;
○    SELECT EMPNAME,SAL FROM EMP WHERE SAL > 5000000 / 12;
```

SAL 列に索引がついている場合、この列を計算式や関数に含めてしまうと索引が使用されないので、 下のように記述する。

※暗黙の型変換に注意する
Oracle では、SQL で明示的に指定しなくても数値型⇔文字型の変換を行なう。 索引列では関数が使用されたのと同じことになり、索引が使用されない。 暗黙の変換を避けるには、比較の対象となる値の書き方に注意すること
（CHAR 型：'12345' NUMBER 型：12345）

● NOT の回避

```
×  SELECT EMPNAME,SAL FROM EMP WHERE DEPTNO != 0;
○    SELECT EMPNAME,SAL FROM EMP WHERE DEPTNO > 0;
```

否定演算子を使用した場合、全表走査が行なわれるので、下のように記述する。

● NULL の回避

```
×  SELECT EMPNAME,SAL FROM EMP WHERE DEPTNO IS NOT NULL;
○    SELECT EMPNAME,SAL FROM EMP WHERE DEPTNO >= 0;
```

NULL 列は索引に入らないので、索引を使用するように下のように記述する。

● UNIQUE 索引の優先

EMPNO に UNIQUE 索引、DEPTNO に非 UNIQUE 索引が作成されている EMP 表を検索する場合

```
SELECT * FROM EMP WHERE EMPNO = '000012' /* ←この条件に索引が使われる */
  AND DEPTNO = '0001';
```

Oracle では、表に利用可能な索引が２つ以上存在し、１つが UNIQUE 索引、もう一つが非 UNIQUE 索引である場合、 UNIQUE 索引のみを表検索に利用し、もう一方の索引は無視される。

●複合索引使用の注意

DEPTNO ＋ AREA に複合索引が作成されている EMP 表を検索する場合

```
○  SELECT EMPNAME,SAL FROM EMP WHERE DEPTNO >= 0 AND AREA = 'TOKYO';
○  SELECT EMPNAME,SAL FROM EMP WHERE DEPTNO >= 0;
×  SELECT EMPNAME,SAL FROM EMP WHERE AREA = 'TOKYO';
```

複数の列を組み合わせた複合索引は、構成列全てを条件で指定している場合、および複合索引の構成列の先頭の列を単独条件で使用している場合に利用される。

操作説明表

操作	オプション	説明
AND-EQUAL	-	複数の ROWID のセットを受け取り、重複をなくして、そのセットの共通部分を戻す処理。この処理は単一列索引のアクセス・パスに対して使用されます。
BITMAP	CONVERSION	TO ROWIDS は、ビットマップ表現を、表にアクセスするために使用できる実際の ROWID に変換します。FROM ROWIDS は、ROWID をビットマップ表現に変換します。COUNT は、実際の値を必要としない場合に ROWID の数を戻します。
BITMAP	INDEX	SINGLE VALUE は、索引内の単一のキー値のビットマップを参照します。RANGE SCAN は、ある範囲のキー値のビットマップを取り出します。FULL SCAN は、開始キーまたは終了キーがない場合にビットマップ索引の全体スキャンを実行します。

BITMAP	MERGE	レンジ・スキャンの結果の複数のビットマップを 1 つのビットマップにマージします。
BITMAP	MINUS	片方のビットマップのビットを、もう一方のビットマップから減算します。行ソースは否定述語に対して使用されます。減算が発生する可能性があるビットマップを作成する非否定述語がある場合にのみ使用できます。「EXPLAIN PLAN によるビットマップ索引の表示」で例を示します。
BITMAP	OR	2 つのビットマップのビット単位の OR を計算します。
BITMAP	AND	2 つのビットマップのビット単位の AND を計算します。
BITMAP	KEY ITERATION	表の行ソースから各行を取り出し、ビットマップ索引から対応するビットマップを検索します。その後、このビットマップのセットは、次の BITMAP MERGE 操作で 1 つのビットマップにマージされます。
CONNECT BY	-	CONNECT BY 句を含んでいる問合せについて階層順に行を取り出します。
CONCATENATION	-	複数の行のセットを受け取り、そのセットの UNION-ALL を戻す処理。
COUNT	-	表から選択された行の数をカウントする処理。
COUNT	STOPKEY	戻される行数を WHERE 句の ROWNUM 式によって制限するカウント処理。
DOMAIN INDEX	-	ドメイン索引からの 1 つ以上の ROWID の取出し。オプション列には、ユーザー定義ドメイン・インデックス・コスト関数から与えられた情報が含まれています。

FILTER	-	行のセットを受け取り、そのいくつかを取り除き、残りを戻す処理。
FIRST ROW	-	問合せで選択される最初の行のみの取出し。
FOR UPDATE	-	FOR UPDATE 句が含まれている問合せによって選択される行を取り出し、ロックする処理。
HASH	GROUP BY	GROUP BY 句を持つ問合せで、行のセットを複数のグループにハッシュする処理。
HASH JOIN	-	2 つのセットの行を結合し、結果を戻す操作。この結合方法は、データのラージ・データ・セット（DSS やバッチなど）の結合に役立ちます。この結合条件は、第 2 の表にアクセスする場合に有効です。問合せオプティマイザは、2 つの表またはデータ・ソースの小さいほうを使用して、メモリー内に結合キーについてのハッシュ表を作成します。次に、大きいほうの表をスキャンし、ハッシュ表を調べて結合された行を見つけます。
（これらは結合操作です。）	--	--
HASH JOIN	ANTI	ハッシュ（左側）アンチ結合。
HASH JOIN	SEMI	ハッシュ（左側）セミ結合。
HASH JOIN	RIGHT ANTI	ハッシュ右側アンチ結合。
HASH JOIN	RIGHT SEMI	ハッシュ右側セミ結合。
HASH JOIN	OUTER	ハッシュ（左側）外部結合。
HASH JOIN	RIGHT OUTER	ハッシュ（右側）外部結合。
INDEX	UNIQUE SCAN	索引からの単一の ROWID の取出し。
（これらはアクセス方法です。）	--	--
INDEX	RANGE SCAN	索引からの 1 つ以上の ROWID の取出し。索引値は昇順でスキャンされます。

INDEX	RANGE SCAN DESCENDING	索引からの 1 つ以上の ROWID の取出し。索引値は降順でスキャンされます。
INDEX	FULL SCAN	スタート・キーおよびストップ・キーがない場合の、索引からのすべての ROWID の取得。索引値は昇順でスキャンされます。
INDEX	FULL SCAN DESCENDING	スタート・キーおよびストップ・キーがない場合の、索引からのすべての ROWID の取得。索引値は降順でスキャンされます。
INDEX	FAST FULL SCAN	マルチブロック READ を使用した全 ROWID（および列の値）の取得。ソート順は定義できません。索引付けされた列に対してのみ、全表スキャンと比較されます。コストベース・オプティマイザでのみ使用可能です。
INDEX	SKIP SCAN	索引内の先頭列を使用しない、連結索引からの ROWID の取得。Oracle9i で導入されています。コストベース・オプティマイザでのみ使用可能です。
INLIST ITERATOR	-	IN リスト述語内の各値に対して、計画内の次の操作を反復します。
INTERSECTION	-	2 つの行のセットを受け取り、重複をなくして、そのセットの共通部分を戻す処理。
MERGE JOIN	-	2 つの行のセットを受け取り、それぞれを特定の値でソートし、一方のセットの各行を他方の行と突き合せて結合し、その結果を戻す処理。
（これらは結合操作です。）	--	--
MERGE JOIN	OUTER	外部結合文を実行するマージ結合処理。
MERGE JOIN	ANTI	マージ・アンチ結合。

MERGE JOIN	SEMI	マージ・セミ結合。
MERGE JOIN	CARTESIAN	文中に他の表への結合条件を持たない 1 つ以上の表について発生する操作です。結合とともに発生する可能性がありますが、計画内では CARTESIAN とフラグが付かないことがあります。
CONNECT BY	-	CONNECT BY 句を含んでいる問合せに対する、階層順での行の取出し。
MAT_VIEW REWITE ACCESS	FULL	マテリアライズド・ビューのすべての行の取出し。
（これらはアクセス方法です。）	--	--
MAT_VIEW REWITE ACCESS	SAMPLE	マテリアライズド・ビューのサンプル行の取出し。
MAT_VIEW REWITE ACCESS	CLUSTER	索引クラスタのキーの値に基づいた、マテリアライズド・ビューからの行の取出し。
MAT_VIEW REWITE ACCESS	HASH	ハッシュ・クラスタのキーの値に基づいた、マテリアライズド・ビューからの行の取出し。
MAT_VIEW REWITE ACCESS	BY ROWID RANGE	ROWID 範囲に基づいたマテリアライズド・ビューからの行の取出し。
MAT_VIEW REWITE ACCESS	SAMPLE BY ROWID RANGE	ROWID 範囲に基づいたマテリアライズド・ビューからのサンプル行の取出し。
MAT_VIEW REWITE ACCESS	BY USER ROWID	ユーザー指定の ROWID を使用してマテリアライズド・ビューの行が指定される場合。
MAT_VIEW REWITE ACCESS	BY INDEX ROWID	マテリアライズド・ビューがパーティション化されておらず、索引を使用して行が指定される場合。
MAT_VIEW REWITE ACCESS	BY GLOBAL INDEX ROWID	マテリアライズド・ビューがパーティション化されており、グローバル索引のみを使用して行が指定される場合。

MAT_VIEW REWITE ACCESS	BY LOCAL INDEX ROWID	マテリアライズド・ビューがパーティション化されており、1 つ以上のローカル索引と、場合によってはいくつかのグローバル索引を使用して行が指定される場合。パーティション区間 : パーティション区間は次のようにして計算されている可能性があります。前の PARTITION ステップによって決定される場合。この場合、PARTITION_START 列の値と PARTITION_STOP 列の値は PARTITION ステップ内の値をレプリケートし、PARTITION_ID には PARTITION ステップの ID が組み込まれます。PARTITION_START および PARTITION_STOP に使用できる値は、NUMBER(n)、KEY、INVALID です。MAT_VIEW REWRITE ACCESS または INDEX ステップ自体で決定される場合。この場合、PARTITION_ID にはそのステップの ID が組み込まれます。PARTITION_START および PARTITION_STOP に使用できる値は、NUMBER(n)、KEY、ROW REMOVE_LOCATION（MAT_VIEW REWRITE ACCESS のみ）および INVALID です。
MINUS	-	2 つの行のセットを受け取り、最初のセットにあって 2 番目のセットにない行を戻して、重複をなくす処理。

NESTED LOOPS	-	外側のセットと内側のセット、2 つの行のセットを受け取る処理。Oracle は、外側のセットの各行を内側のセットの各行と比較し、条件を満たす行を戻します。この結合方法は、小さいサブセットのデータを結合する場合（OLTP）に役立ちます。この結合条件は、第 2 の表にアクセスする場合に有効です。
（これらは結合操作です。）	--	--
NESTED LOOPS	OUTER	外部結合文を実行するネステッド・ループ操作。
PARTITION	-	PARTITION_START 列および PARTITION_STOP 列によって指定された範囲の各パーティションに対して、計画内の次の操作を反復します。PARTITION は、単一のパーティション・オブジェクト（表または索引）や同じ数でパーティション化されたオブジェクトのセット（パーティション表やそのローカル索引）に適用できるパーティションの区間を示します。パーティションの区間は、PARTITION の PARTITION_START および PARTITION_STOP の値で指定されます。PARTITION_START および PARTITION_STOP の有効な値は、表 19-1 を参照してください。
PARTITION	SINGLE	1 つのパーティションへのアクセス。
PARTITION	ITERATOR	多数のパーティション（サブセット）へのアクセス。
PARTITION	ALL	すべてのパーティションへのアクセス。
PARTITION	INLIST	IN リスト述語を基準にしたイテレータ。

PARTITION	INVALID	アクセスするよう設定されているパーティションが空であることを示します。
PX ITERATOR	BLOCK、CHUNK	パラレル・スレーブ・セット間でのブロックまたはチャンク範囲へのオブジェクトの分割を実装します。
PX COORDINATOR	-	パラレル問合せスレーブを使用して下位のパラレル計画を制御、スケジュールおよび実行する問合せコーディネータを実装します。また、パラレルに実行され、常に下位に PX SEND QC 操作を持つ計画部分の終わりとして、シリアライズ・ポイントを表します。
PX PARTITION	-	セマンティクスは通常の PARTITION 操作と同じですが、パラレル計画に表示されます。
PX RECEIVE	-	PX SEND ノード上で実行される送信側 / プロデューサ（QC またはスレーブ）からパーティション化されたデータを読み取る、コンシューマ / 受信側スレーブ・ノードを示します。以前は、この情報は DISTRIBUTION 列に表示されていました。表 19-2 を参照してください。
PX SEND	QC (RANDOM)、HASH、RANGE	スレーブの 2 つのパラレル・セットの間における配分方法を実装します。2 つのスレーブ・セット間の境界と、送信側 / プロデューサ側（QC またはスレーブ）でのデータのパーティション化方法を示します。以前は、この情報は DISTRIBUTION 列に表示されていました。表 19-2 を参照してください。
REMOTE	-	リモート・データベースからのデータの取出し。

SEQUENCE	-	順序値のアクセスを伴う処理。
SORT	AGGREGATE	選択した行のグループにグループ関数を適用した結果として取得される単一行の取出し。
SORT	UNIQUE	行のセットをソートし、重複をなくす処理。
SORT	GROUP BY	GROUP BY 句を持つ問合せで、行のセットを複数のグループにソートする処理。
SORT	JOIN	マージ結合の前に、一連の行をソートする操作。
SORT	ORDER BY	ORDER BY 句を持つ問合せに対して行のセットをソートする処理。
TABLE ACCESS）	FULL	表のすべての行の取出し。
（これらはアクセス方法です。	--	--
TABLE ACCESS	SAMPLE	表のサンプル採取された行の取出し。
TABLE ACCESS	CLUSTER	索引クラスタのキーの値に基づいた、表からの行の取出し。
TABLE ACCESS	HASH	ハッシュ・クラスタのキーの値に基づいた、表からの行の取出し。
TABLE ACCESS	BY ROWID RANGE	ROWID 範囲に基づいた表からの行の取出し。
TABLE ACCESS	SAMPLE BY ROWID RANGE	ROWID 範囲に基づいた表からのサンプル行の取出し。
TABLE ACCESS	BY USER ROWID	ユーザー指定の ROWID を使用して表の行が指定される場合。
TABLE ACCESS	BY INDEX ROWID	表がパーティション化されておらず、索引を使用して行が指定される場合。
TABLE ACCESS	BY GLOBAL INDEX ROWID	表がパーティション化されており、グローバル索引のみを使用して行が指定される場合。

TABLE ACCESS	BY LOCAL INDEX ROWID	表がパーティション化されており、1 つ以上のローカル索引と場合によってはいくつかのグローバル索引を使用して、行が指定される場合。パーティション区間 : パーティション区間は次のようにして計算されている可能性があります。前の PARTITION ステップによって決定される場合。この場合、PARTITION_START 列の値と PARTITION_STOP 列の値は PARTITION ステップ内の値をレプリケートし、PARTITION_ID には PARTITION ステップの ID が組み込まれます。PARTITION_START および PARTITION_STOP に使用できる値は、NUMBER(n)、KEY、INVALID です。TABLE ACCESS または INDEX ステップ自体で決定される場合。この場合、PARTITION_ID にはそのステップの ID が組み込まれます。PARTITION_START および PARTITION_STOP に使用できる値は、NUMBER(n)、KEY、ROW REMOVE_LOCATION（TABLE ACCESS のみ）および INVALID です。
UNION	-	2 つの行のセットを受け取り、重複をなくして、そのセットの連結結果を戻す処理。
VIEW	-	ビューの問合せを実行し、結果の行を別の処理に戻す処理。