ハッシュ結合の特徴
ハッシュ結合を使用すべきデータ状況
ハッシュ結合の使い方　（ヒント句）

テーブル結合

Select 文の Where 条件において、比較項目の左辺と右辺の両方がテーブルの列で指定している場合は、テーブルの連結が指定条件での結合キーで行われ、一致レコードの出力が行われる

例 ）

```
Select * e. ename , d.dname from emp e , dept d
where e.deptno = d.deptno ;
```

この比較条件で、１つの比較値（deptno）に対してのレコードの存在は、単数 or 複数という２種類のテーブル状態が出来る

（列に、一意制約がなければ、同一の値のレコードの作成は可能）

すなわち、deptno＝10 のレコードが、emp テーブルに３件、dept テーブルに５件ということが考えられる

よって、テーブル間の一致パターンで考えると、以下の４通りになる

1 ： 1
1 ： 多
多 ： 1
多 ： 多

【~~外側表~~ハッシュ表】

実行計画で、上段に表示されたテーブルがハッシュ結合の~~外側表~~ハッシュ表になります

どちらの表を外側表にすれば、ハッシュ結合を使った検索処理（＝ハッシュパケット表を作成）が効率的に行われるかは、比較する２つの表の列値の種類数の少ない方になります

外部表の列値の種類が少ないということは、ハッシュパケット表を小さくできるということになります

	表 A	表 B
データ件数：	500000 件	3000 件
比較列の値の種類：	40 種	1500 種
それぞれの表外部表にした場合のハッシュパケット表のレコード件数 ：	40 レコード	1500 レコード

ハッシュ結合の特徴

ハッシュ結合では、ハッシュパケット表をメモリ内部に作成することから始まる

ハッシュパケット表の作成は、実行計画の上段テーブル（ハッシュ表）を**一度全件検索し**作成する。

この後は、次に下段テーブルを１件ずつ読み込んで、下段テーブルの関連項目の値に一致する内容を**ハッシュパケット表から読み込み**処理する

「ハッシュパケットのイメージ」

外部表

Rec 番号	データ項目	関連項目
1		2
2		3
3		1
4		3
5		2

ハッシュ

パケット　表

＃A
＃B
＃C

内側表

関連項目	Rec 番号
1	1
2	2
3	3
16	4
23	5

ハッシュ表のレコード・メモリ			ハッシュ パケット値
		レコード３	＃A
	レコード１	レコード５	＃B
	レコード２	レコード４	＃C

【注 意】

ハッシュパケット表のハッシュパケット値は、Where 条件で指定された列の値を算術計算して、ハッシュ値を作成する

ハッシュパケット表のレコード件数には制限があり、比較テーブルとなる外側表と内側表の列値の種類数の和が、このレコード制限数を上回った場合、異なる列値でも同じハッシュ値が割り当てることになる

外側表と内側表の列値の種類数の和 ＞ ハッシュテーブルのレコード件数制限

すなわち、ハッシュ値１個に対して、列値が複数存在するような状態になる

同一のハッシュ値でも列値が異なるレコードが存在する場合、

１件のハッシュパケット・レコードを処理するときに、元となった『列の値が幾つだったか』を、元の外側表、内側表のレコードの列値を辿って＝条件の成立がどのレコード対象なのかを見極めることとなる

このような余分な処理が加わったとしても、ハッシュパケット値で対象レコードが事前に限定されるので、テーブル全件の比較を行うよりは ずっとブロック・アクセ

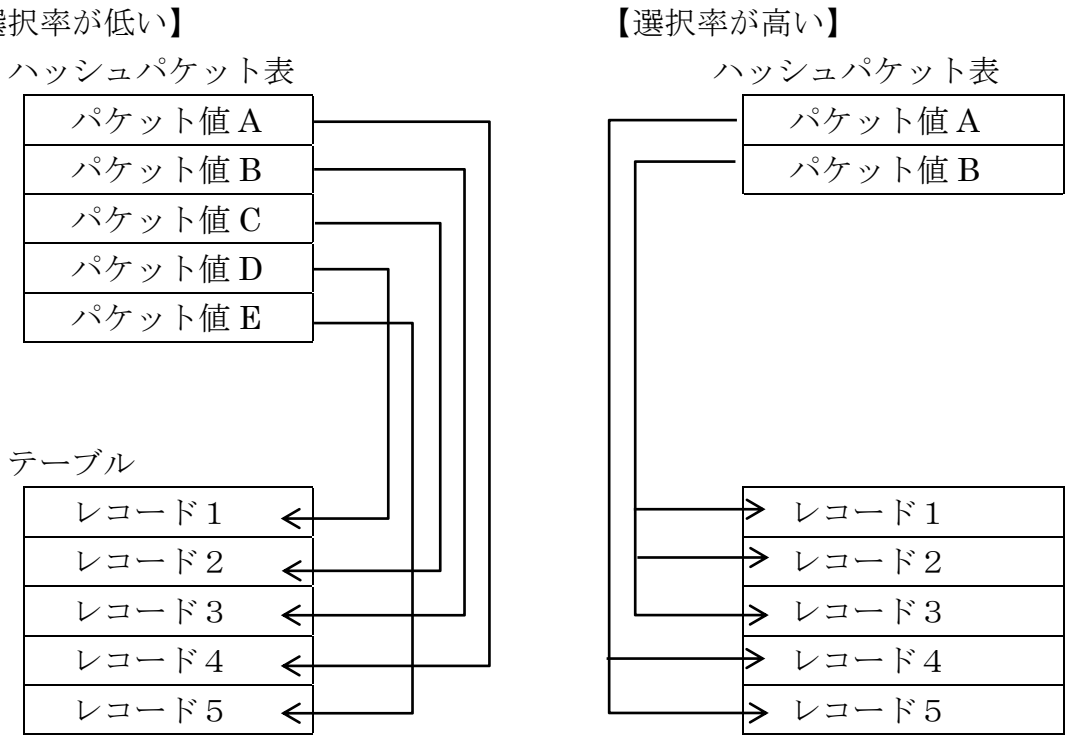
ス効率は上がる

　なお、前ページに書いたように比較する基準となった関連項目にも一意制約がなければ、比較するテーブルには同一の値のレコードが複数存在することになる

選択率

選択率とは、ハッシュパケットの１つの値が示すレコード数の割合である、

前ページの例では、ハッシュ値がパケット＃C は、外部表、内部表ともに２件のレコードが存在していることを示している



ハッシュ結合が効果を発揮できるケース

外部表の選択率が高い（列の値が同一の件数が多い）状況では、ハッシュ結合の使用が有効です

※ 選択率が低い外部表を使用すると、Nested Loops より効率が落ちる
内部表の選択率の高さは、無関係である（ハッシュ値に対する内部表のレコードは、内部表のインデックスを利用すれば影響は出ない）

ネステッドループ結合が効果を発揮できるケース

外部表の選択率が低い（列の値が同一の件数が少ない）状況では、ネステッドループ結合が有効です

このとき、内部表の比較列の値のばらつきが大きい（インデックス自体の選択率が低い）場合には、内部表の INDEX SCAN を行うとより、アクセス効果が上がります

逆に内部表の比較列の値のばらつきが小さい（インデックス自体の選択率が高い）場合には、テーブル・フルアクセスが有効です

※ どちらの結合が良いか判断するためには、**実行統計**の consistents gets に示されるアクセス・ブロック数**を調査して比較する**

また、**結合列にインデックスを作成する**のも一案である

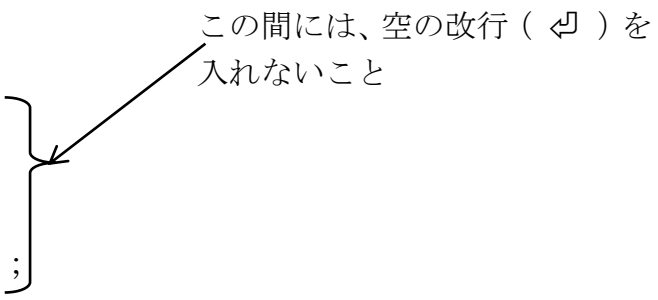
explain plan for SQL 文と@?/rdbms/admin/utlxpls.sql を使って、SQL 文の実行計画を検証する

Select 文

```
explain plan for

Select /*+ ヒント句 */
   e. ename , d.dname
   from emp e , dept d
   where e.deptno = d.deptno ;

@?/rdbms/admin/utlxpls.sql
```



※ Nested Loops 結合の実行には、ヒント句を使う

```
/*+ use_nl( e ,d ) */
```

Hash Join 結合の実行には、ヒント句を使う

```
/*+ use_hash( e ,d ) */
```

Sort Merge 結合の実行には、ヒント句を使う

```
/*+ use_merge( e ,d ) */
```

ハッシュ結合と Nested Loops 結合の実行計画と実行統計の比較

［ A　ハッシュ結合の場合 ］

A ①　実行計画の確認

ID	OPERATION	Name
0	SELECT STATEMENT	
1	❸ HASH JOIN　ハッシュジョイン結合	
2	❶ TABLE ACCESS FULL	EMPLOYEES
3	❷ TABLE ACCESS FULL	DEPARTMENTS

A ②　アクセス・ブロック数の確認

autotrace を使って、Select 文の実行する

503　consistent gets　（アクセスしたブロック数(バッファ文＋ディスク I／O 文)）

［ B　Nested Loops 結合の場合 ］

B ①　実行計画の確認

explain　plan　for　文を使って、SQL 文の実行計画を検証する

ID	OPERATION	Name
0	SELECT STATEMENT	
1	❹ NESTED LOOPS	
2	❷ TABLE ACCESS FULL	EMPLOYEES
3	❸ TABLE ACCESS BY INDEX ROWID	DEPARTMENTS
4	❶ INDEX UNIQUE SCAN	DEPT_ID_PK

B ②　アクセス・ブロック数の確認

autotrace を使って、Select 文の実行する

839　consistent gets　（アクセスしたブロック数(バッファ文＋ディスク I／O 文)）

比較結論

それぞれのアクセス・ブロック数を比較する

A(HASH　JOIN)の方が、アクセス・ブロック数が少ないので効率が良いことになる