HITACHI
Inspire the Next

ノンストップデータベース

HiRDB Version 10 パフォーマンスガイド

解説・手引書

3020-6-568-10

前書き

■ 対象製品

●適用 OS：AIX V7.1，AIX V7.2

P-1M62-35A1 HiRDB Server Version 10 10-06

P-1M62-1BA1 HiRDB/Run Time Version 10 10-06

P-1M62-1CA1 HiRDB/Developer's Kit Version 10 10-06

P-1M62-1DA1 HiRDB/Run Time Version 10(64) 10-06

P-1M62-1EA1 HiRDB/Developer's Kit Version 10(64) 10-06

●適用 OS：Red Hat Enterprise Linux 7 (64-bit x86_64)，Red Hat Enterprise Linux 8 (64-bit x86_64)

P-8462-35A1 HiRDB Server Version 10 10-06

P-8462-1DA1 HiRDB/Run Time Version 10(64) 10-06

P-8462-1EA1 HiRDB/Developer's Kit Version 10(64) 10-06

●適用 OS：Red Hat Enterprise Linux 7 (64-bit x86_64)，Red Hat Enterprise Linux 8 (64-bit x86_64)

P-8362-1BA1 HiRDB/Run Time Version 10 10-06

P-8362-1CA1 HiRDB/Developer's Kit Version 10 10-06

P-8362-3CA1 HiRDB Developer's Suite Version 10 10-06

●適用 OS：Windows Server 2016，Windows Server 2019，Windows Server 2022，Windows 10 Pro (x64)，Windows 10 Enterprise (x64)，Windows 11

P-2962-91A4 HiRDB Server Version 10 10-06

●適用 OS：Windows Server 2016，Windows Server 2019，Windows Server 2022，Windows 10，Windows 11

P-2662-11A4 HiRDB/Run Time Version 10 10-06

P-2662-12A4 HiRDB/Developer's Kit Version 10 10-06

P-2662-32A4 HiRDB Developer's Suite Version 10 10-06

●適用 OS：Windows Server 2016，Windows Server 2019，Windows Server 2022，Windows 10 Home (x64)，Windows 10 Pro (x64)，Windows 10 Enterprise (x64)，Windows 11

P-2962-11A4 HiRDB/Run Time Version 10(64) 10-06

P-2962-12A4 HiRDB/Developer's Kit Version 10(64) 10-06

これらのプログラムプロダクトのほかにもこのマニュアルをご利用になれる場合があります。詳細は「リリースノート」でご確認ください。

■ 輸出時の注意

本製品を輸出される場合には、外国為替及び外国貿易法の規制並びに米国輸出管理規則など外国の輸出関連法規をご確認の上、必要な手続きをお取りください。

なお、不明な場合は、弊社担当営業にお問い合わせください。

■ 商標類

■ 発行

2022 年 10 月　3020-6-568-10

変更内容

変更内容(3020-6-568-10)　HiRDB Version 10 10-06

追加・変更内容	変更個所
HP-UX に関する説明を削除しました。	－

単なる誤字・脱字などはお断りなく訂正しました。

変更内容(3020-6-568)　HiRDB Version 10 10-00

追加・変更内容
HiRDB を 10-00 にバージョンアップしました。

変更内容(3020-6-470-20)　HiRDB Version 9 09-66

追加・変更内容
HiRDB のサポートプラットフォームに次の OS を追加しました。 • AIX V7.2 • Windows Server 2016
マニュアルの体裁を変更しました。

変更内容(3020-6-470-10)　HiRDB Version 9 09-50

追加・変更内容
HiRDB SQL Tuning Advisor を使用した SQL トレースと UAP 統計レポートの調査方法を追加しました。
効果的なインデクスの作り方に関する説明を，マニュアル「HiRDB Version 9 システム導入・設計ガイド」に移動しました。
横分割表に効率良くアクセスする方法を追加しました。
第 2 編データベース設計編を追加しました。
HiRDB のサポートプラットフォームに次の OS を追加しました。 • Windows 10

はじめに

このマニュアルは，プログラムプロダクト　ノンストップデータベース　HiRDB Version 10 の性能に関連する項目について説明したものです。なお，ここに記載されていない前提情報については，マニュアル「HiRDB Version 10 解説」を参照してください。

■ 対象読者

第 1 編は，HiRDB Version 10（以降，HiRDB と表記します）でアプリケーションを開発する方を対象にしています。

第 2 編は，HiRDB でデータベースを設計する方を対象にしています。

このマニュアルは次に示す知識があることを前提に説明しています。

- Windows のシステム管理の基礎的な知識（Windows 版の場合）
- UNIX または Linux のシステム管理の基礎的な知識（UNIX 版の場合）
- SQL の基礎的な知識

なお，このマニュアルは，マニュアル「HiRDB Version 10 解説」を前提としていますので，あらかじめお読みいただくことをお勧めします。

■ パス名の表記

- パス名の区切りは「¥」で表記しています。UNIX 版 HiRDB を使用している場合は，マニュアル中の「¥」を「/」に置き換えてください。ただし，Windows 版と UNIX 版でパス名が異なる場合は，それぞれのパス名を表記しています。
- HiRDB 運用ディレクトリのパスを%PDDIR%と表記します。ただし，Windows 版と UNIX 版でパス名が異なるため，それぞれを表記する場合，UNIX 版は$PDDIR と表記します。例を次に示します。

  Windows 版：%PDDIR%¥CLIENT¥UTL¥

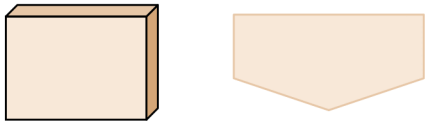
  UNIX 版：$PDDIR/client/lib/

- Windows のインストールディレクトリのパスを%windir%と表記します。

■ 図中で使用している記号

このマニュアルの図中で使用している記号を次のように定義します。

●プログラムまたはサーバ※　●作業手順



注※

一部の図では，プログラムを単に四角で囲んで（影を付けないで）記載しています。

■ HiRDB SQL Tuning Advisor の画面について

このマニュアルでは，HiRDB SQL Tuning Advisor の画面を，80 バイトで折り返し表示しています。マニュアルでの画面と，実際に表示される画面とで折り返し位置が異なるため，ご注意ください。

■ このマニュアルで使用している記号

形式および説明で使用している記号を次に示します。ここで説明する文法記述記号は，説明のための記号なので実際には記述しないでください。

記号	意味
{　　}	この記号で囲まれている複数の項目の中から 1 つを選択することを示します。 （例） PDDLKPRIO= {96 | 64 | 32} これは，PDDLKPRIO オペランドの指定値として 96，64 および 32 から選択できることを示します。
[　　]	この記号で囲まれた項目は省略できることを示します。 （例） PDHOST=システムマネジャのホスト名［，予備系システムマネジャのホスト名］ これは，予備系システムを省略できることを示します。
|（ストローク）	記号{　　}で囲まれた複数の項目を 1 つずつの項目に区切ることを示します。 （例） PDXAMODE= {0 | 1} これは，PDXAMODE オペランドの設定値として 0，および 1 の項目に区切ることを示します。
（下線）	記号{　　}で囲まれた複数の項目の中の 1 つにだけ使用され，記号内の項目をすべて省略したとき，システムが設定する標準値を示します。 （例） PDDFLNVAL= {USE | NOUSE} これは，PDDFLNVAL オペランドを省略したときに PDDFLNVAL=NOUSE と仮定されることを示します。

記号	意味
～	この記号の後にユーザ指定値の属性を示します。
《　》	ユーザが指定しなかった場合に仮定される値を示します。
＜　＞	ユーザ指定値の構文要素を示します。
((　))	ユーザ指定値の指定範囲を示します。

■ このマニュアルで使用している構文要素記号

このマニュアルで使用している構文要素記号を次に示します。

構文要素記号	意味
<英字>	アルファベット（A～Z，a～z）と下線（_）
<英字記号>	アルファベット（A～Z，a～z）と#，@，¥
<英数字>	英字と数字（0～9）
<英数字記号>	英字記号と数字
<符号なし整数>	数字
<符号なし 10 進数>※1	数字（0～9 の並び）ピリオド（.）数字（0～9 の並び）
<識別子>※2	先頭がアルファベットの英数字列
<文字列>	任意の文字の配列
<記号名称>	先頭が英字記号の英数字記号 UNIX 版の場合，¥は使用できません。
<パス名>※3	UNIX 版の場合：/，英数字，ピリオド（.），#，および@ Windows 版の場合：¥，英数字，ピリオド（.），空白，丸括弧，#，および@

注

すべて半角文字を使用してください。また，英字の大文字と小文字は区別されます。さらに，パス名は使用している OS に依存します。

注※1

ピリオドの前の数字がすべて 0 の場合，ピリオドより前の 0 を省略できます。また，ピリオドの後ろの数字がすべて 0 の場合，ピリオド以降を省略できます。

例 1：0.008 → .008

例 2：15.000 → 15

注※2

RD エリア名の場合は，先頭が英字記号で始まる英数字記号，下線（_），ハイフン（-），および空白となります。また，RD エリア名に空白が含まれる場合は，引用符（"）で囲んでください。

ホスト名の場合は，アルファベット（A～Z，a～z），数字，ピリオド（.），ハイフン（-），下線（_），および@で構成される文字列となります。

注※3

パス名に空白，または丸括弧を含む場合は，前後を引用符（"）で囲んでください。

なお，Windows 版の場合，コロン（:）をドライブ名に使用できます。

■ HiRDB のデータベース言語の出典

このマニュアルで記述する HiRDB のデータベース言語仕様は，次に示す規格を基に日立製作所独自の解釈と仕様を追加したものです。原開発者に謝意を表するとともに，仕様の出典を示します。

- JIS X 3005 規格群　データベース言語 SQL
- ISO/IEC 9075　Information technology － Database languages － SQL －

注

JIS：日本工業規格（Japanese Industrial Standard）

ISO：国際標準化機構（International Organization for Standardization）

IEC：国際電気標準会議（International Electrotechnical Commission）

目次

1 アプリケーション開発編の概要

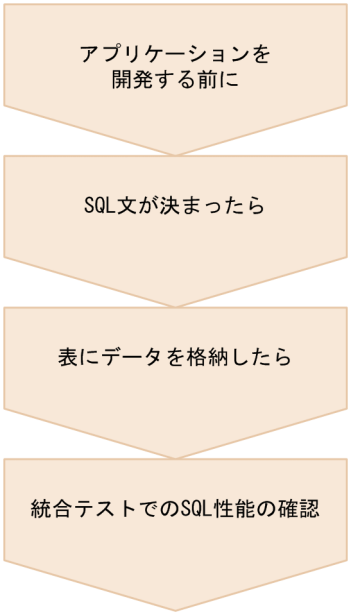
この章では，アプリケーション開発編の概要について説明します。

1.1　アプリケーション開発編の読み方

1.1.1　アプリケーション開発編の読み方の説明

この編は，HiRDB にアクセスするアプリケーションが，よりよいパフォーマンスを出すために，アプリケーション開発工程のどの時期に，何を確認すればよいか，また注意することは何かについて，説明しています。該当する工程に応じた章の説明をお読みください。

図 1-1　アプリケーション開発編の読み方



(1)　アプリケーションを開発する前に

「アプリケーションを開発する前に」では，アプリケーション開発前に理解しておきたい性能上の考慮点について説明します。

(2)　SQL 文が決まったら

「SQL 文が決まったら」では，アクセスパスの確認方法について説明します。

- 対策が必要なアクセスパスの一覧は，「SQL 文が決まったら」の「ここは必ず対策しよう」を参照してください。
- アクセスパスのチューニング例も説明しています。「チューニング例の一覧」を参照してください。

(3)　表にデータを格納したら

「表にデータを格納したら」では，SQL トレースと UAP 統計レポートを使って SQL 性能を確認する方法について説明します。

- SQL トレースと UAP 統計レポートで最初に確認する項目の一覧は，「表にデータを格納したら」の「ここは必ず確認しよう」を参照してください。
- 中間結果情報のチューニング例も説明しています。「チューニング例の一覧」を参照してください。

(4) 統合テストでの SQL 性能の確認

「統合テストでの SQL 性能の確認」では，SQL オブジェクト用バッファの統計情報を使って SQL 性能を確認する方法について説明します。

- SQL オブジェクト用バッファの統計情報で最初に確認する項目の一覧は，「統合テストでの SQL 性能の確認」の「ここは必ず確認しよう」を参照してください。

1.2　SQL 性能調査で使用する製品および機能

この編で使用する HiRDB の関連製品および機能について紹介します。

1.2.1　HiRDB SQL Tuning Advisor

HiRDB SQL Tuning Advisor は，SQL の性能上のボトルネックとなっている所を特定する作業，およびその対策を支援する製品です。詳細は，マニュアル「HiRDB 解説」の「HiRDB SQL Tuning Advisor」および HiRDB SQL Tuning Advisor のヘルプを参照してください。

この編で，説明に使用している章を次に示します。

- 「SQL 文が決まったら」
- 「表にデータを格納したら」※

注※

この章では，HiRDB SQL Tuning Advisor 08-04 以降を使用してください。

1.2.2　HiRDB SQL Executer

HiRDB SQL Executer は，SQL を対話形式で実行できる製品です。詳細は，マニュアル「HiRDB 解説」の「HiRDB SQL Executer」および HiRDB SQL Executer のヘルプを参照してください。

この編で，説明に使用している章を次に示します。

- 「SQL 文が決まったら」

1.2.3　SQL トレース機能

実行した SQL 文のトレース情報をファイルに出力する機能です。SQL トレースでは SQL 文ごと実行時間などがわかります。詳細は，マニュアル「HiRDB UAP 開発ガイド」の「SQL トレース機能」を参照してください。

この編で，説明に使用している章を次に示します。

- 「表にデータを格納したら」

1.2.4　UAP 統計レポート機能

アプリケーション実行時の SQL に関するさまざまな情報を提供する機能です。提供する情報は SQL トレース，UAP 実行に関する統計情報，SQL 実行時のデータ処理件数，アクセスパス情報です。詳細は，マニュアル「HiRDB UAP 開発ガイド」の「UAP 統計レポート機能」を参照してください。

この編で，説明に使用している章を次に示します。

- 「SQL 文が決まったら」
- 「表にデータを格納したら」

1.2.5　SQL オブジェクト用バッファの統計情報

SQL オブジェクト用バッファに格納されている SQL オブジェクトの統計情報です。処理時間の掛かる SQL や入出力の多い SQL を特定できます。詳細は，マニュアル「HiRDB コマンドリファレンス」の「pdobils（SQL オブジェクト用バッファの統計情報表示）」を参照してください。

この編で，説明に使用している章を次に示します。

- 「統合テストでの SQL 性能の確認」

2 アプリケーションを開発する前に

この章では，アプリケーションを開発する前に理解しておきたい性能上の考慮点について説明します。

2.1 パフォーマンスを効率良く確認するために

ここでは，パフォーマンスの確認を効率良く行うためのポイントを説明します。

1. クライアント環境定義 PDCLTAPNAME の指定

   アプリケーションを実行する環境には，クライアント環境定義 PDCLTAPNAME を指定してください。SQL 性能を確認するための統計情報やトレース情報では，アプリケーションを識別する名称として PDCLTAPNAME の設定値が表示されます。PDCLTAPNAME を省略していると，どのアプリケーションが実行した情報なのか特定するのが難しくなるため，必ず指定してください。

2. ?パラメタの使用

   SQL 文中に指定する定数が可変である場合は，SQL 文に定数を直接指定するのではなく，?パラメタを使った SQL 文に変更してください。SQL 文の種類を削減することで，統合テストでの SQL オブジェクト用バッファの統計情報の確認が容易になります。また，SQL オブジェクトを共通化することで，SQL 文の前処理時間も削減できます。?パラメタを使った SQL 文に変更する例を次に示します。

   変更前

   ```
   SELECT * FROM T1 WHERE C1=1
   SELECT * FROM T1 WHERE C1=2
   ```

   変更後※

   ```
   SELECT * FROM T1 WHERE C1=?
   ```

   注※

   ?パラメタには 1 または 2 を設定します。

2.2 効果的なインデクスの作り方

SQL 文がよりよいパフォーマンスを出すためには，SQL 文に適したインデクスを定義することが重要です。効果的なインデクスの作り方は，次の説明を参照してください。

- インデクスの効果
  インデクスの効果については，マニュアル「HiRDB 解説」の「インデクスの基本構造」を参照してください。
- インデクスの設計ポイント
  インデクスの設計ポイントについては，マニュアル「HiRDB システム導入・設計ガイド」の「インデクスの設計」を参照してください。

［参照先］

- インデクス定義に適している列，インデクス定義に適さない列を知りたい
  マニュアル「HiRDB システム導入・設計ガイド」の「インデクスの作成」を参照してください。
- インデクス構成列の組み合わせや構成列の順序の検討方法を知りたい
  マニュアル「HiRDB システム導入・設計ガイド」の「インデクス構成列の検討」を参照してください。
- 1 つの表に複数のインデクスを定義した場合，使用するインデクスの優先順位を知りたい
  マニュアル「HiRDB システム導入・設計ガイド」の「インデクスの優先順位」を参照してください。

2.3 横分割表に効率良くアクセスする方法

横分割した表に効率良くアクセスする方法について，説明します。

- 分割キーに対する条件を探索条件または結合条件に指定してください。これによって，アクセスするRD エリアを限定できます。

3 SQL 文が決まったら

この章では，SQL 文が決定した時点で確認する内容について説明します。

3.1　アクセスパスを確認しよう

SQL 文が決定したら，アクセスパスに問題がないか確認してください。

アクセスパスとは，データベースへのアクセス手順のことです。処理効率の悪いアクセスパスになっていると，その SQL 文は処理に時間を要してしまいます。効率の良いアクセスパスにするためには，SQL 文の修正が必要になる場合もあります。SQL 文が決まった時点で，アクセスパスに問題がないか，必ず確認しておきましょう。

［こんなときは］

フレームワークを使用している場合など，SQL 文を直接指定していないアプリケーションでは，アプリケーションが動作した時点で，アクセスパスを確認してください。

［注意事項］

確認する際に接続する HiRDB サーバは，次に示す点を本番環境と同じ構成にしてください。

- HiRDB サーバの種別（HiRDB/シングルサーバまたは HiRDB/パラレルサーバ）
  HiRDB/シングルサーバと HiRDB/パラレルサーバでは，アクセスパスが変わることがあります。
- 表の分割数および分割条件
  表の分割条件が変わると，アクセスパスが変わることがあります。
- HiRDB/パラレルサーバの場合は，バックエンドサーバの数と RD エリアの配置
  RD エリアを管理するバックエンドサーバが変わると，アクセスパスが変わることがあります。

3.2　アクセスパスの出力方法

SQL 文が決定した時点で確認する場合は，SQL 文を入力情報としてアクセスパスを出力します。これによって，アプリケーションを作成する前にアクセスパスを確認できます。アクセスパスの出力方法について，次に示します。確認時期や使用できる製品を確認の上，出力方法を選択してください。

表 3-1　アクセスパスの出力方法

出力時期	方法選択の目安	出力方法
SQL 文が決定した時点で確認する場合	HiRDB SQL Tuning Advisor が使用できる場合は，この方法を選択します。HiRDB SQL Tuning Advisor では，アクセスパスの出力とともに，対策が必要なアクセスパスに対して，ガイダンスメッセージを出力します。	HiRDB SQL Tuning Advisor を使ったアクセスパスの出力方法
	HiRDB SQL Tuning Advisor が使用できない場合は，この方法を選択します。	HiRDB SQL Executer で SQL 文を前処理してアクセスパスを出力する方法
アプリケーションが動作した時点で確認する場合	SQL 文を直接指定していないアプリケーションの場合は，この方法を選択します。	アプリケーション実行時にアクセスパスを出力する方法

3.2.1　HiRDB SQL Tuning Advisor を使ったアクセスパスの出力方法

HiRDB SQL Tuning Advisor を使ったアクセスパスの出力方法について説明します。

(1)　HiRDB サーバ側の準備

HiRDB サーバ側で準備することを，次に説明します。

- HiRDB サーバを起動して，HiRDB SQL Tunning Advisor が接続できる状態にします。
- 表とインデクスの定義をしておきます。この時，データは格納されていなくても問題ありません。

(2)　SQL ファイルの作成

SQL 文を記述したファイルを作成します。1 つの SQL ファイルに複数の SQL 文を記述できます。SQL ファイルの規則を次に示します。

- SQL 文の終わりには半角のセミコロン（;）を付け，改行してください。
- ファイル名の拡張子は"sql"にしてください（例：sample.sql）。

[こんなときは]

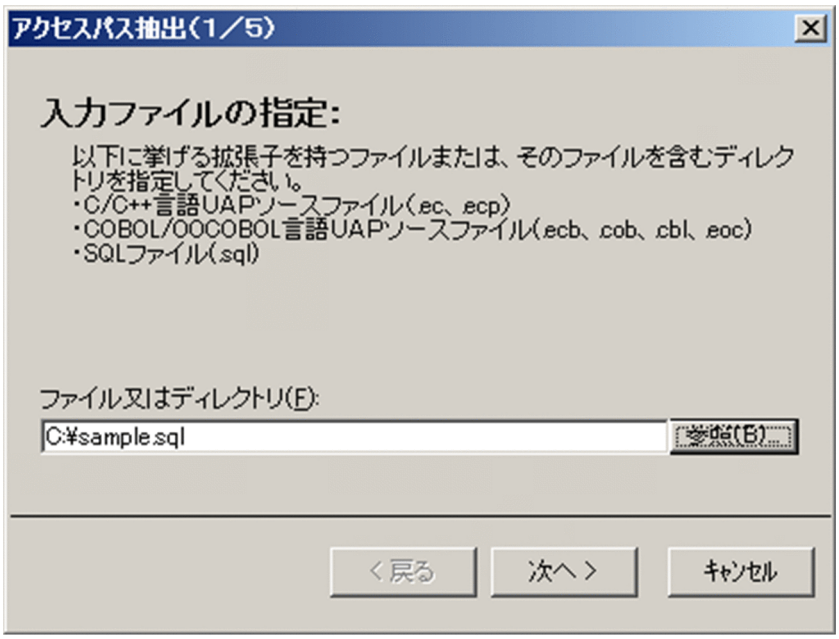
HiRDB SQL Tuning Advisor の SQL 抽出機能を使用すると，埋め込み型アプリケーションのソースファイル（.ec，.ecb など）から SQL 文を SQL ファイルに抽出できます。詳細は，HiRDB SQL Tuning Advisor のヘルプを参照してください。

(3) アクセスパス抽出の実行

実行手順を次に説明します。詳細は，HiRDB SQL Tuning Advisor のヘルプを参照してください。

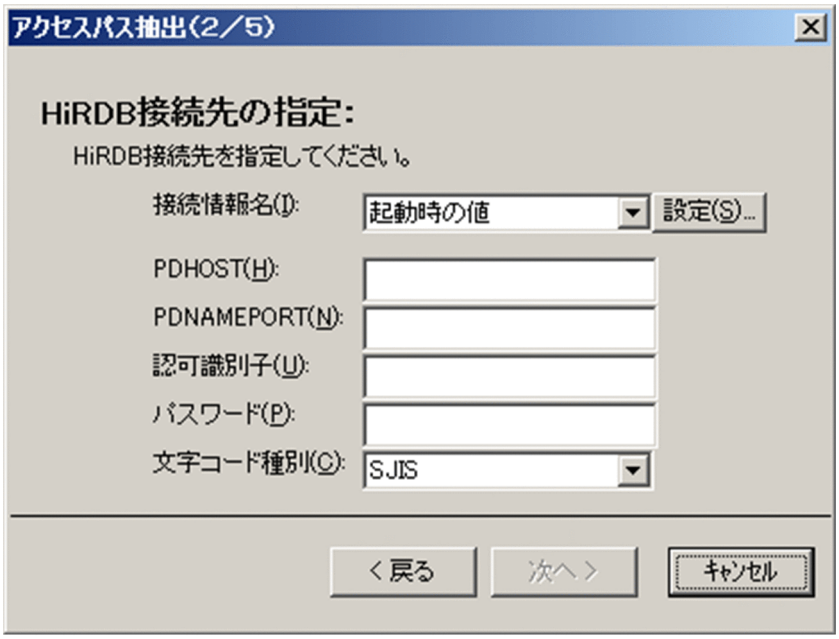
1. 「ファイル」－「実行」－「アクセスパス抽出...」のメニューを起動します。
2. 「入力ファイルの指定」画面で SQL ファイルを指定します。

図 3-1　入力ファイルの指定の画面



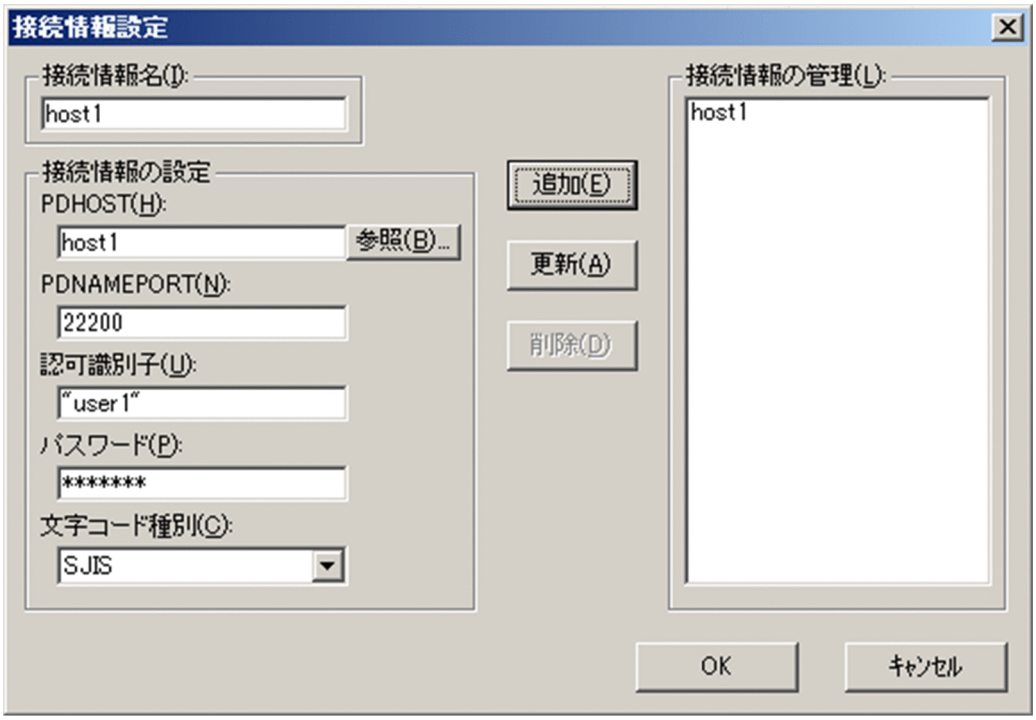
3. 「HiRDB 接続先の指定」画面で，接続する HiRDB サーバの接続情報名を選択します。接続情報名を登録していない場合は，「設定」ボタンを押して登録します。登録済みの場合は，手順 5 に進みます。

図 3–2　HiRDB 接続先の指定の画面



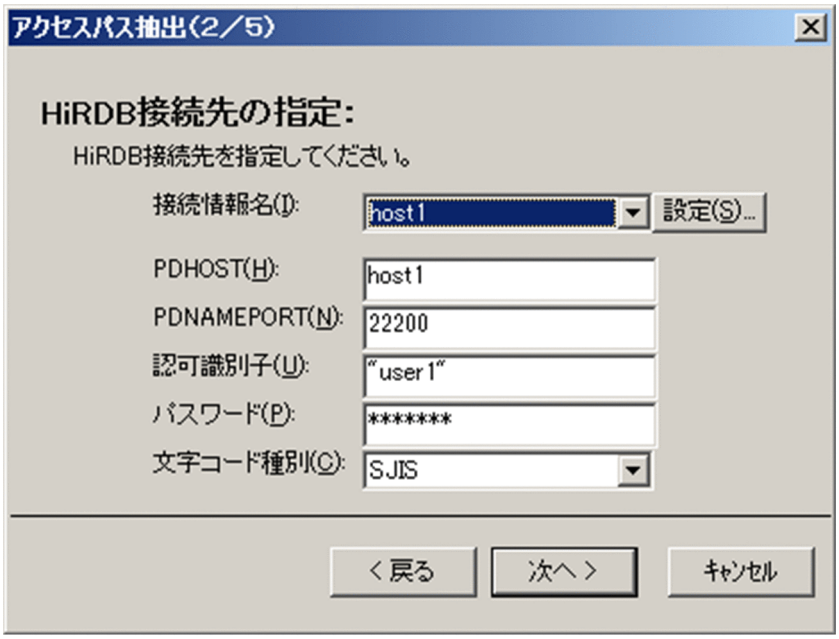
4.「接続情報設定」画面で，接続情報名と接続情報を入力します。「追加」ボタンを押して登録します。

図 3–3　接続情報設定の画面



5.「HiRDB 接続先の指定」画面で，接続情報を指定し，「次へ」ボタンを押します。

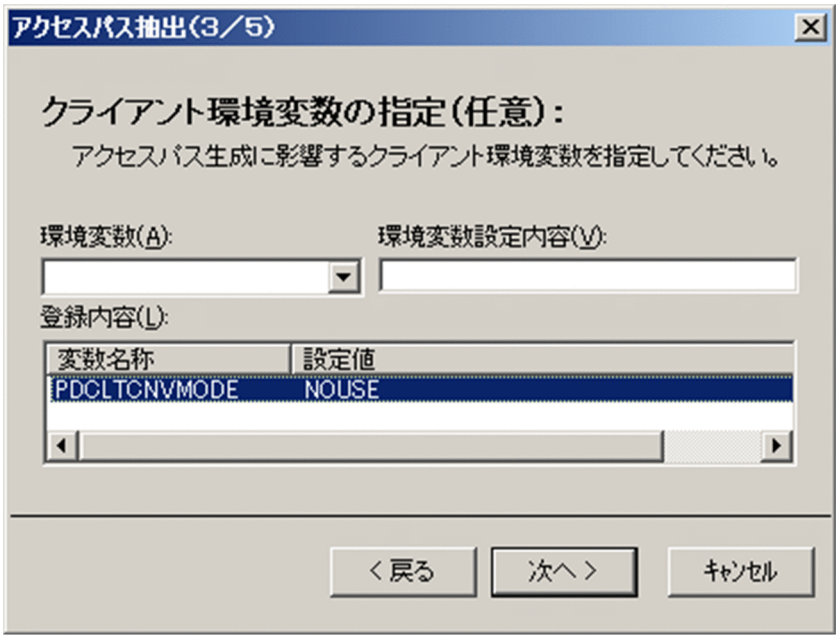
図 3-4　HiRDB 接続先の指定の画面



6. 次の条件に当てはまる場合は，「クライアント環境変数の指定（任意)」画面で，該当するクライアント環境定義を指定してください。

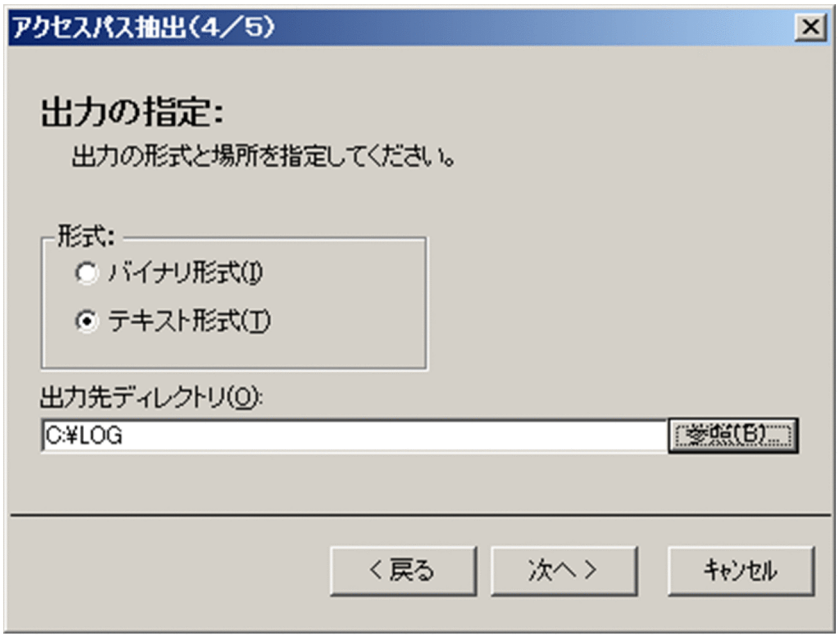
- SQL 最適化オプションに省略値以外の値を適用する場合は，クライアント環境定義 PDSQLOPTLVL または PDADDITIONALOPTLVL に適用する値を指定してください。
- HiRDB サーバの文字コード種別が SJIS 以外の場合は，クライアント環境定義 PDCLTCNVMODE に AUTO を指定してください。

図 3-5　クライアント環境変数の指定の画面



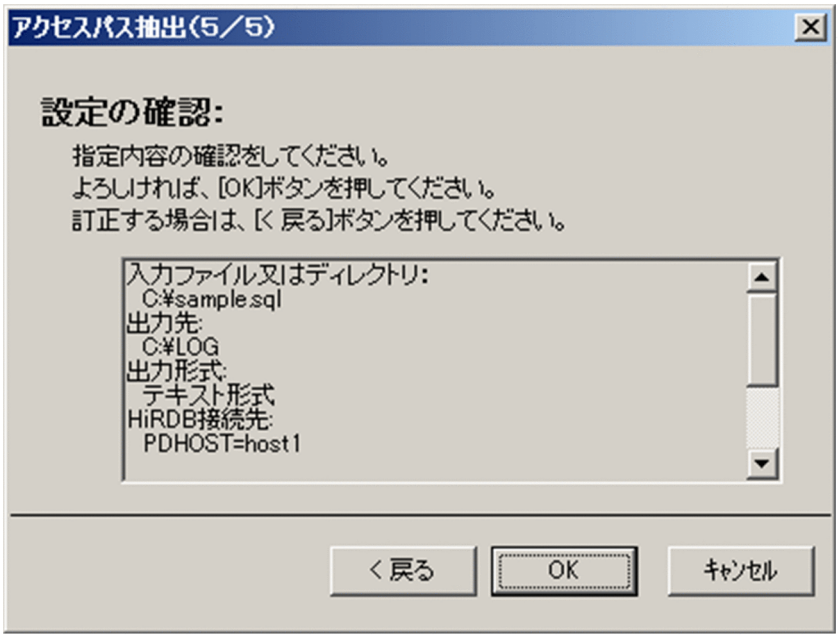
7. 「出力の指定」画面で出力形式と出力先ディレクトリを指定します。出力形式は，ここではテキスト形式を選択します。HiRDB SQL Tuning Advisor のアクセスパス解析機能を使ってアクセスパスを調査する場合は，バイナリ形式を選択します。

図 3-6　出力の指定の画面



8. 「設定の確認」画面で設定内容を確認し，「OK」ボタンを押します。

図 3-7　設定の確認の画面



9. 出力先ディレクトリに，アクセスパスのファイルが出力されます。テキスト形式で出力した場合，ファイル名は，入力ファイル名の拡張子を"txt"に変更した名前になります。

3.2.2 HiRDB SQL Executer で SQL 文を前処理してアクセスパスを出力する方法

HiRDB SQL Executer で SQL 文を前処理してアクセスパスを出力する方法について説明します。

(1) HiRDB サーバ側の準備

HiRDB サーバ側で準備することを，次に説明します。

- HiRDB サーバを起動して，HiRDB SQL Executer が接続できる状態にします。
- 表とインデクスの定義をしておきます。この時，データは格納されていなくても問題ありません。

(2) SQL ファイルの作成

SQL 文を記述したファイルを作成します。1 つの SQL ファイルに複数の SQL 文を記述できます。SQL ファイルの規則を次に示します。

- SQL 文の終わりには半角のセミコロン（;）を付け，改行してください。
- ファイル名の拡張子は"txt"が推奨です（例：sample.txt）。"txt"以外の拡張子のファイルでも入力できます。

(3) クライアント環境定義の設定

HiRDB SQL Executer を実行する環境に，次のクライアント環境定義を設定してください。

表 3-2 設定するクライアント環境定義

目的	環境定義名	設定値
UAP 統計レポートを取得する設定	PDCLTPATH	UAP 統計レポート出力先ディレクトリを指定します。
	PDSQLTRACE	UAP 統計レポートのファイルサイズの最大値を指定します。サイズの上限を指定しない「0」を設定することを推奨します。
	PDUAPREPLVL	アクセスパスを出力するレベルである「p」を設定してください。
SQL 最適化オプションに関する設定	PDSQLOPTLVL PDADDITIONALOPTLVL	SQL 最適化オプションに省略値以外の値を適用する場合は，その値を指定してください。
文字コード変換に関する設定	PDCLTCNVMODE	HiRDB サーバと HiRDB クライアントの文字コード種別が異なる場合は，AUTO を指定してください。

クライアント環境定義の指定方法の詳細は，マニュアル「HiRDB UAP 開発ガイド」の「クライアント環境定義の設定内容」を参照してください。

(4) SQL 文の前処理

HiRDB SQL Executer でアクセスパスを出力する場合は，SQL の前処理だけを実行するモードで実行します。これによって，データを操作しないでアクセスパスを確認できます。実行方法を次に説明します。

GUI 版 HiRDB SQL Executer の場合

1. HiRDB SQL Executer を起動して，HiRDB サーバに接続します。
2. SQL の前処理だけを実行するモードに変更します。「SQL 入力画面」で次のコマンドを実行します。

```
SET ANALYSIS MODE ON;
```

3.「ファイル」－「ファイルから実行...」のメニューから SQL ファイルを指定し，実行します。

ラインモード版の場合

1. HiRDB SQL Executer を起動して，HiRDB サーバに接続します。
2. SQL の前処理だけを実行するモードに変更します。次のコマンドを実行します。

```
COMMAND ?     +----2----+----3----+----4----+----5----+----6----+----7----+

SET ANALYSIS MODE ON;
```

3. SQL ファイルを読み込むコマンドを実行します。コマンド実行例を次に示します。

```
COMMAND ?     +----2----+----3----+----4----+----5----+----6----+----7----+

< sample.sql;
```

実行方法の詳細は HiRDB SQL Executer のヘルプを参照してください。

SQL を実行すると，UAP 統計レポートのファイルにアクセスパスを出力します。出力ファイル名は，マニュアル「HiRDB UAP 開発ガイド」の「SQL トレース機能」を参照してください。

3.2.3 アプリケーション実行時にアクセスパスを出力する方法

アプリケーション実行時にアクセスパスを出力する方法について説明します。

(1) HiRDB サーバ側の準備

HiRDB サーバ側で準備することを，次に説明します。

- HiRDB サーバを起動して，アプリケーションが接続できる状態にします。
- 表とインデクスの定義をしておきます。
- アプリケーションで必要なデータを格納します。

[注意事項]

アプリケーション実行時にアクセスパスを出力する方法では，SQL 文が実行されるため，更新 SQL 文を含む場合はデータが更新されます。

(2) クライアント環境定義の設定

アプリケーションを実行する環境に，UAP 統計レポートを取得するクライアント環境定義を設定してください。設定内容は，「HiRDB SQL Executer で SQL 文を前処理してアクセスパスを出力する方法」の表「設定するクライアント環境定義」を参照してください。

(3) アプリケーションの実行

アプリケーションを実行すると，UAP 統計レポートのファイルにアクセスパスを出力します。出力ファイル名は，マニュアル「HiRDB UAP 開発ガイド」の「SQL トレース機能」を参照してください。

3.3　アクセスパスの確認方法の概要

アクセスパスの情報のうち，最初に確認する項目を次に示します。

表 3-3　アクセスパスの確認項目

確認項目	説明
検索方法（Scan Type）	表の検索方法です。詳細は，マニュアル「HiRDB UAP 開発ガイド」の「検索方式」を参照してください。
インデクスのサーチ条件（SearchCnd）	インデクスのサーチ範囲を決定する条件です。詳細は，マニュアル「HiRDB コマンドリファレンス」の「アクセスパス表示ユティリティ（pdvwopt)」「サーチ条件」を参照してください。
結合方法および転送方法（Join Type）	• 結合方法は，表を結合する方法です。結合方法の種類および詳細は，マニュアル「HiRDB UAP 開発ガイド」の「結合方式」を参照してください。 • 転送方法は，HiRDB/パラレルサーバで表を結合する際の，バックエンドサーバ間のデータ転送方法です。詳細は，マニュアル「HiRDB コマンドリファレンス」の「アクセスパス表示ユティリティ（pdvwopt)」「転送方法の種類」を参照してください。
副問合せの実行方式（Sub Query Type）	副問合せを実行する方式です。詳細は，マニュアル「HiRDB UAP 開発ガイド」の「外への参照のない副問合せの実行方式」および「外への参照のある副問合せの実行方式」を参照してください。

これらの確認項目のうち，効率の悪いアクセスパスについて「ここは必ず対策しよう」で説明します。

アクセスパスを対策する際の注意事項を，次に説明します。

[注意事項]

アクセスパスの対策方法として，新しいインデクスの追加を提案している個所があります。ただし，1つの表に定義するインデクスの数が多いと，更新処理が遅くなります。インデクスの数が多くならないように，性能を重要視する SQL 文で使うインデクスを優先的に定義してください。

[注意事項]

アクセスパスの対策によって，インデクスの定義内容を変更した場合は，再度すべての SQL 文のアクセスパスを確認してください。インデクスを変更すると，その表にアクセスする別の SQL 文のアクセスパスが変わることがあります。

[ポイント]

インデクス定義を変更する前後のアクセスパスの差分は，「アクセスパスの出力方法」で説明した方法によって出力したファイルの差分を取り，確認してください。

3.4　ここは必ず対策しよう

効率の悪いアクセスパスを次に示します。このアクセスパスは必ず対策してください。

表 3-4　効率の悪いアクセスパスの一覧

確認項目	アクセスパス	HiRDB SQL Tuning Advisor のガイダンスメッセージ ID
検索方法（Scan Type）	TABLE SCAN	KFPX29601-I
インデクスのサーチ条件（SearchCnd）	FULL SCAN	KFPX29604-I KFPX29608-I KFPX29984-I KFPX29985-I
結合方法（Join Type）	CROSS JOIN	KFPX29994-I
	MERGE JOIN	KFPX29995-I
	効率の悪い NESTED LOOPS JOIN	－
副問合せの実行方式（Sub Query Type）	WORK TABLE SUBQ	－
	NESTED LOOPS WORK TABLE SUBQ	－

（凡例）－：該当メッセージなし

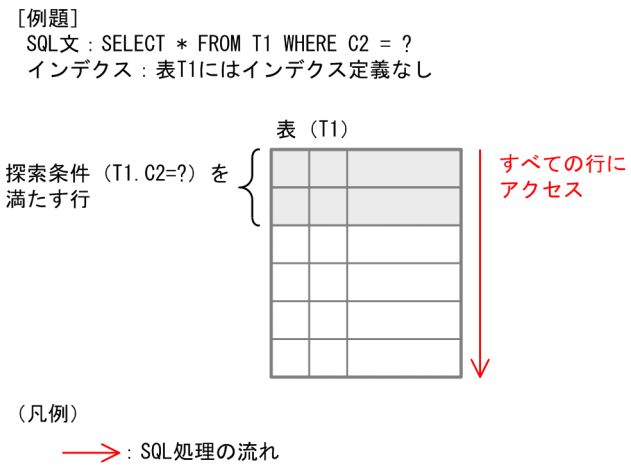
対策方法の詳細について，項目ごとに説明します。

3.4.1　TABLE SCAN の対策

(1)　TABLE SCAN とは

TABLE SCAN は，探索条件の内容に関わらず，検索対象表の全行をシーケンシャルにアクセスする方法のため，処理効率が悪いです。

図 3-8　TABLE SCAN の処理方式



［こんなときは］

次の場合は，TABLE SCAN のままでも問題ありません。

- 全件抽出を目的とする場合
- 表に格納する件数が極端に少ない場合

  採番目的の表など格納件数が 1 件の表が該当します。

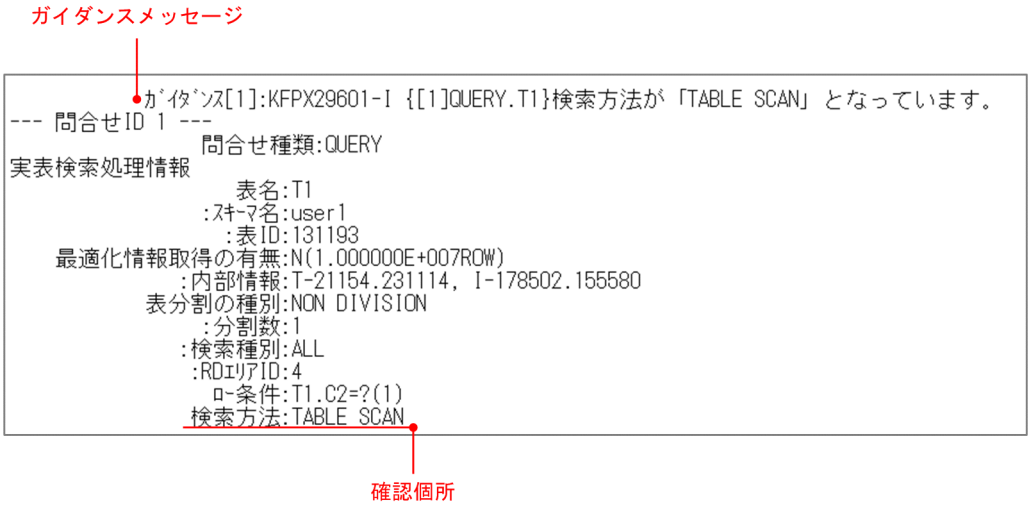
(2)　確認方法

TABLE SCAN の確認方法を次に示します。

- HiRDB SQL Tuning Advisor の場合

  アクセスパス情報の「検索方法」に「TABLE SCAN」と表示されます。TABLE SCAN の出力例を次に示します。

  図 3-9　HiRDB SQL Tuning Advisor の出力例（TABLE SCAN）

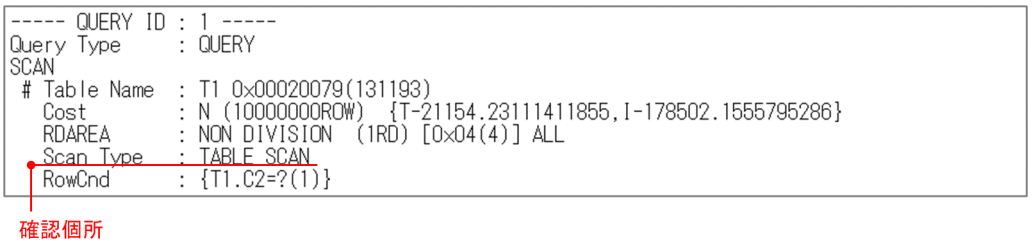


- UAP 統計レポートの場合

アクセスパス情報の「Scan Type」に「TABLE SCAN」と表示されます。

TABLE SCAN の出力例を次に示します。

図 3-10　UAP 統計レポートの出力例（TABLE SCAN）



(3)　対策方法

探索条件に指定した列にインデクスが定義されていないため，TABLE SCAN になっています。インデクスを追加して，インデクスを使用した検索方法（INDEX SCAN または KEY SCAN）に変更してください。

［ポイント］

次の個所に表示された列にインデクスを定義することで，インデクスを使用した検索方法に変更できます。

- HiRDB SQL Tuning Advisor の場合

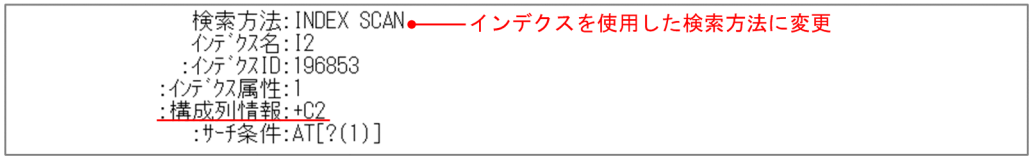
  「TABLE SCAN」の直前にある「ﾛｰ条件」または「If Then 条件」に表示された列

  図 3-11　HiRDB SQL Tuning Advisor の出力例

  対策前のアクセスパス

  対策後のアクセスパス

- UAP 統計レポートの場合

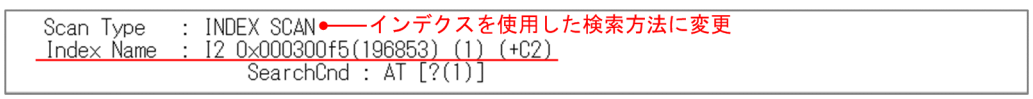
  「TABLE SCAN」の直下にある「RowCnd」または「IfThenCnd」に表示された列

  図 3-12　UAP 統計レポートの出力例

  対策前のアクセスパス

  対策後のアクセスパス

[こんなときは]

「ﾛｰ条件（RowCnd）」または「If Then 条件（IfThenCnd）」に表示された列にインデクスを定義しているにも関わらず，SQL 文の作り方によっては，TABLE SCAN になってしまうケースもあります。この要因について，次に示します。

1. インデクスを利用できない探索条件を OR 論理演算している場合
2. NOT 演算子（<>，^=，!=）で比較している場合

それぞれの要因について，対策方法を次に説明します。

[項番 1 の対策方法]

インデクスを利用できる探索条件がある場合でも，インデクスを利用できない探索条件※を OR 論理演算していると，TABLE SCAN になります。この場合は，探索条件に指定した列が第 1 構成列であるインデクスを追加して，すべての探索条件でインデクスが利用できるようにします。これによって，複数インデクス利用（OR PLURAL INDEXES SCAN または AND PLURAL INDEXES SCAN）に変更できます。

次に対策方法の例を示します。

[例題]

- インデクス定義列：C1
- SQL 文：SELECT * FROM T1 WHERE C1=? OR C2=?

  下線はインデクスに含まれない列に対する探索条件のため，インデクスは利用できません。

[対策方法]

C2 が第 1 構成列であるインデクスを追加します。

注※

次の SQL 文の下線部分のように，表にアクセスしない探索条件もインデクスを利用できません。この場合，表にアクセスしない探索条件はアプリケーション側で判定し，SQL 文からこの探索条件を外すことができないか，検討してください。

```
SELECT * FROM T1 WHERE C1=? OR 1=?
```

[項番 2 の対策方法]

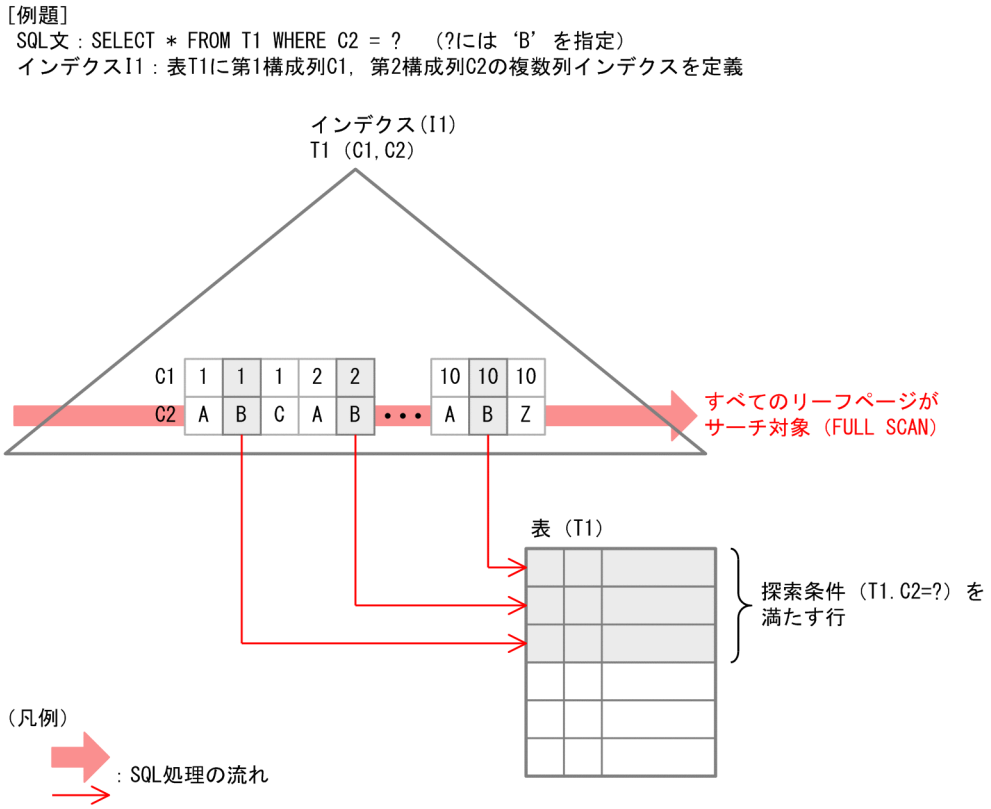
NOT 演算子（<>，^=，!=）で比較している場合は，NOT 演算子は使用しないように探索条件の指定方法を変更してください。

3.4.2　FULL SCAN の対策

(1)　FULL SCAN とは

FULL SCAN とは，インデクスを使用した検索方法（INDEX SCAN または KEY SCAN）で，インデクスのすべてのリーフページがサーチ対象となるため，効率が悪いです。

図 3-13　FULL SCAN の処理方式

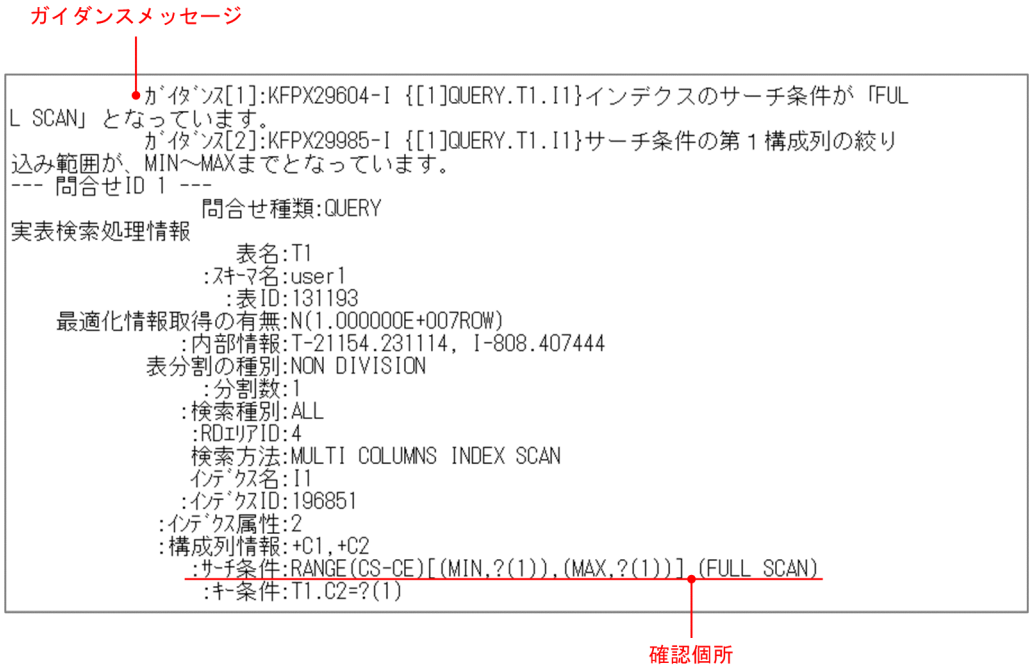


(2)　確認方法

FULL SCAN の確認方法を次に示します。

- HiRDB SQL Tuning Advisor の場合
  アクセスパス情報の「ｻｰﾁ条件」の行に「FULL SCAN」と表示されます。
  FULL SCAN の出力例を次に示します。

図 3-14　HiRDB SQL Tuning Advisor の出力例（FULL SCAN）

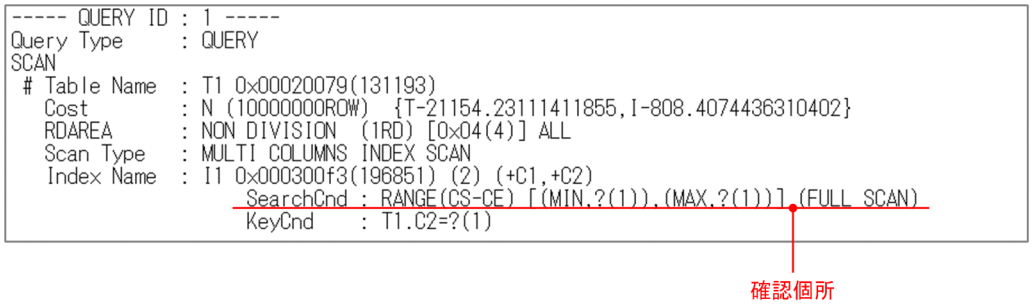


- UAP 統計レポートの場合

  アクセスパス情報の「SearchCnd」の行に「FULL SCAN」と表示されます。

  FULL SCAN の出力例を次に示します。

図 3-15　UAP 統計レポートの出力例（FULL SCAN）



(3)　対策方法

FULL SCAN になる要因について，次の表に示します。

表 3-5　FULL SCAN の要因

項番	要因		HiRDB SQL Tuning Advisor のガイダンスメッセージ ID	UAP 統計レポートの表示
1	インデクス第 1 構成列のサーチ条件が最小値から最大値である	インデクスの第 1 構成列に対する探索条件が指定されていない場合	KFPX29604-I KFPX29985-I	「SearchCnd」の行が次に示す内容を含む場合 • 「RANGE」と行の末尾に「(FULL SCAN)」

項番	要因		HiRDB SQL Tuning Advisor のガイダンスメッセージ ID	UAP 統計レポートの表示
				• 「RANGES」と行の末尾に「(FULL SCAN)」
2	インデクスのサーチ条件がない	インデクスを定義した列に対して，スカラ演算を使用した探索条件を指定した場合	KFPX29604-I KFPX29984-I	「SearchCnd」の行に「NONE(FULL SCAN)」と表示
3		複数列インデクスの各構成列に対する探索条件を，OR 論理演算している場合	KFPX29604-I KFPX29608-I KFPX29984-I	「SearchCnd」の行に「NONE(FULL SCAN)」と表示

それぞれの要因について，対策方法を説明します。

(a) インデクスの第 1 構成列に対する探索条件が指定されていない場合

インデクスの第 1 構成列に対する探索条件が指定されていないため，第 1 構成列のサーチ条件が最小値から最大値となり，FULL SCAN になっています。次のどちらかの対策方法を検討してください。

表 3-6 第 1 構成列に対する探索条件が指定されていない場合の対策方法

対策方法	説明
SQL 文の変更	第 1 構成列に対する探索条件の指定が漏れている場合は，追加してください。
インデクス構成列順序の変更	=条件など最も絞り込める探索条件を指定している列を第 1 構成列にできないか，インデクスの構成列の順序を見直してください。
新しいインデクスの追加	=条件など最も絞り込める探索条件を指定している列が第 1 構成列である新しいインデクスの追加を検討してください。

(b) インデクスを定義した列に対してスカラ演算を使用した探索条件を指定した場合

インデクスを定義した列に対して，スカラ演算を使用した探索条件を指定すると，インデクスのサーチ条件がないアクセスパスとなり，FULL SCAN になります。スカラ演算を行わないよう SQL 文を修正してください。

表 3-7 スカラ演算を使用した探索条件と対策方法の例

スカラ演算を使用した探索条件	対策後の探索条件
C1 + 100 > ?	C1 > ? - 100
SUBSTR(C1,1,3) = 'abc'	C1 LIKE 'abc%'
YYYY || MMDD = '20110203'	(YYYY , MMDD) = ('2011' , '0203')

(c) 複数列インデクスの各構成列に対する探索条件を OR 論理演算している場合

複数列インデクスの各構成列に対する探索条件を OR 論理演算している場合，インデクスのサーチ条件がないアクセスパスとなり，FULL SCAN になります。探索条件に指定した列が第 1 構成列であるインデクスを追加して，すべての探索条件でインデクスが利用できるようにします。これによって，複数インデクス利用（OR PLURAL INDEXES SCAN または AND PLURAL INDEXES SCAN）に変更できます。

次に対策方法の例を示します。

[例題]

- 複数列インデクスの定義列：C1,C2,C3
- SQL 文：SELECT * FROM T1 WHERE C1=? OR C2=? OR C3=?

探索条件に指定した列はすべてインデクスに含まれていますが，探索条件を OR 論理演算しています。

[対策方法]

次に示すインデクスを追加します。

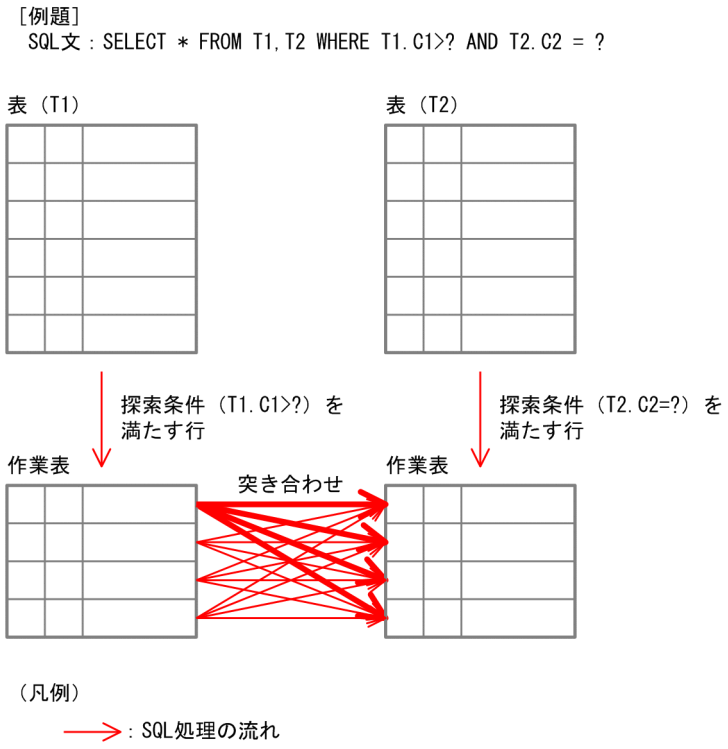
- C2 が第 1 構成列であるインデクス
- C3 が第 1 構成列であるインデクス

3.4.3 CROSS JOIN の対策

(1) CROSS JOIN とは

CROSS JOIN（直積）は，結合する表ごとに探索条件を満たす行を作業表に取り出します。そして，すべての行をそれぞれに組み合わせて結合します。このため，処理効率が悪いです。

図 3-16　CROSS JOIN の処理方式

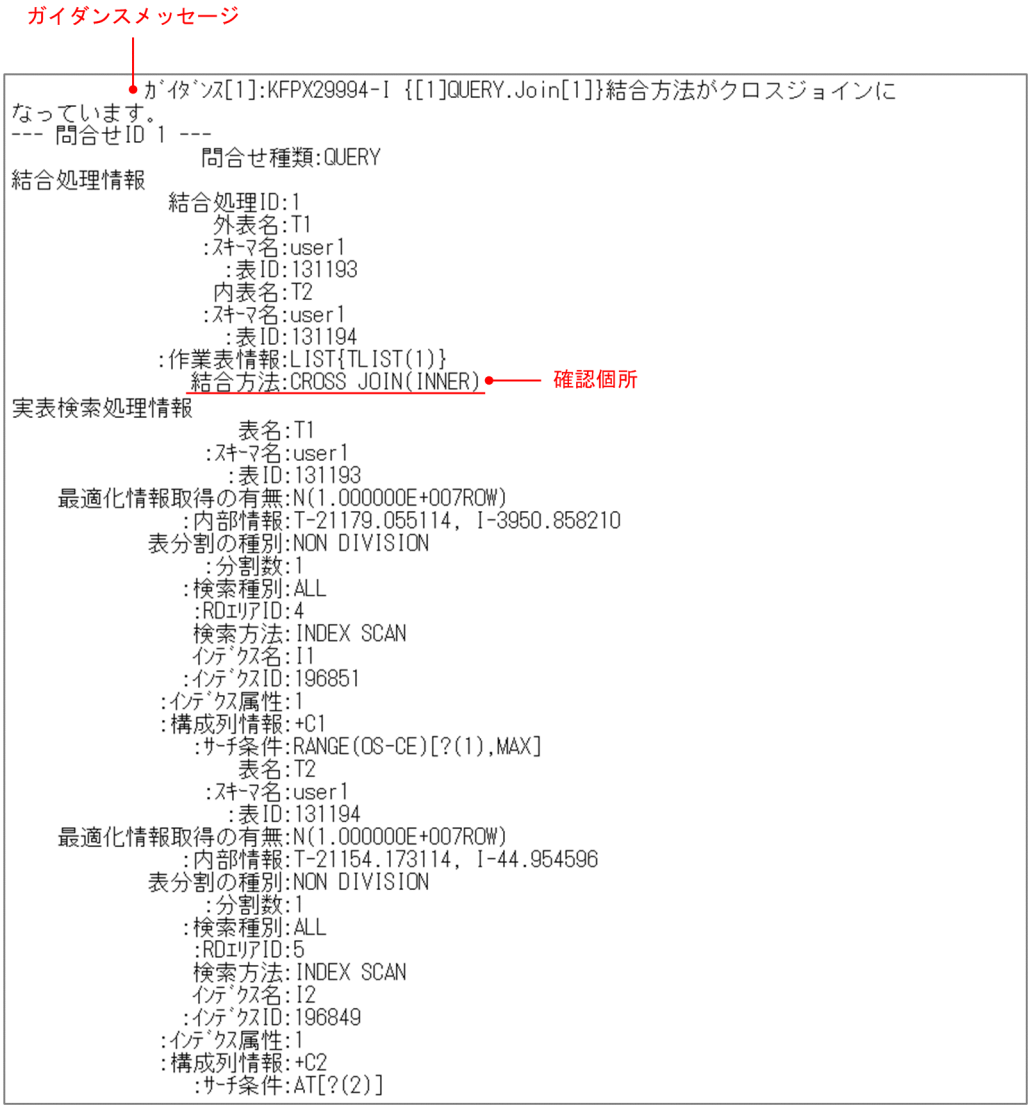


(2)　確認方法

CROSS JOIN の確認方法を次に示します。

- HiRDB SQL Tuning Advisor の場合
  アクセスパス情報の「結合方法」に「CROSS JOIN」と表示されます。
  CROSS JOIN の出力例を次に示します。

図 3-17　HiRDB SQL Tuning Advisor の出力例（CROSS JOIN）

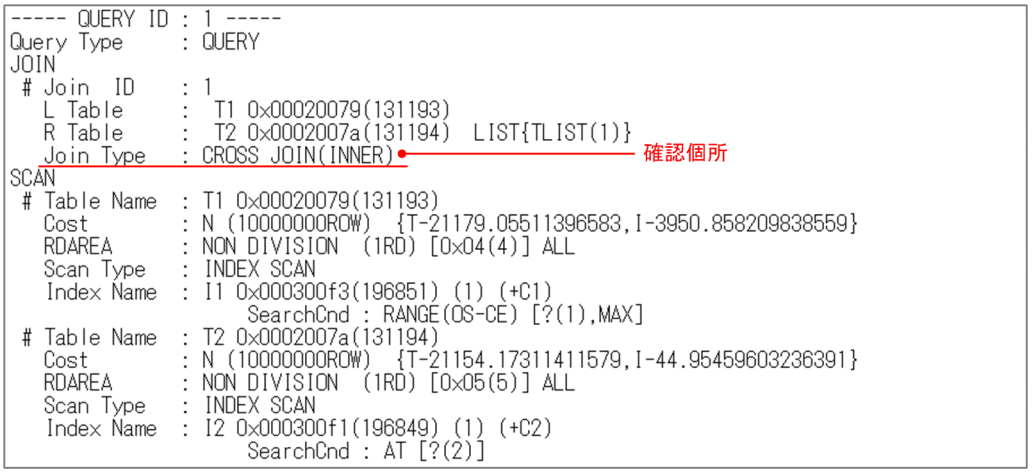


- UAP 統計レポートの場合

  アクセスパス情報の「Join Type」に「CROSS JOIN」と表示されます。

  CROSS JOIN の出力例を次に示します。

図 3-18　UAP 統計レポートの出力例（CROSS JOIN）



(3) 対策方法

SQL 文に結合条件が指定されていないため，CROSS JOIN になっています。結合条件とは，結合する表間の関係を示す条件のことです。SQL 文の探索条件に，結合処理に使用できる結合条件を追加してください。そして，効率の良い結合方法に変更してください。

［効率の良い結合方法について］

SQL 文のヒット件数が少ない場合は NESTED LOOPS JOIN，SQL 文のヒット件数が多い場合は HASH JOIN に変更してください。それぞれの結合方法への変更方法は，「MERGE JOIN の対策」の「対策方法」を参照してください。

［こんなときは］

結合する表間の関係を示す条件を指定していても，SQL 文によっては，結合処理に使用できない場合があります。使用できない例について，次に示します。

・結合する列が，両方ともどちらか片方の辺に指定されている場合

［例題］

```
SELECT * FROM T1 INNER JOIN BY NEST T2 ON T1.C1+T2.C1=10 WHERE T1.C2=? AND T2.C2=?
```

結合する列が両方とも結合条件の左辺に指定されています。

［対策方法］

左辺と右辺に分けて指定してください。

```
SELECT * FROM T1 INNER JOIN BY NEST T2 ON T2.C1=10-T1.C1 WHERE T1.C2=? AND T2.C2=?
```

対策前後のアクセスパスについて，次に示します。

図 3-19　HiRDB SQL Tuning Advisor の出力例

対策前のアクセスパス

```
結合処理情報
                結合処理ID:1
                    外表名:T1
                   :ｽｷｰﾏ名:user1
                     :表ID:131193
                    内表名:T2
                   :ｽｷｰﾏ名:user1
                     :表ID:131194
               :作業表情報:LIST{TLIST(1)}
                  結合方法:CROSS JOIN(INNER)
 SQL最適化指定の有効無効:SPECIFICATION IGNORED
    :IF-THEN条件(外結合):T1.C1+T2.C1=10
```

NESTED LOOPS JOINの指定は無効となり
CROSS JOINとなっている

対策後のアクセスパス

```
結合処理情報
                結合処理ID:1
                    外表名:T1
                   :ｽｷｰﾏ名:user1
                     :表ID:131193
                    内表名:T2
                   :ｽｷｰﾏ名:user1
                     :表ID:131194
                  結合方法:1-CLM  NESTED LOOPS JOIN(INNER)
 SQL最適化指定の有効無効:AS SPECIFIED
```

NESTED LOOPS JOINに変更

```
                    表名:T2
                   :ｽｷｰﾏ名:user1
                     :表ID:131194
    最適化情報取得の有無:N(1.000000E+007ROW)
                :内部情報:T-21154.289114, I-45.054596
            表分割の種別:NON DIVISION
                   :分割数:1
                  :検索種別:ALL
                   :RDｴﾘｱID:5
                     ﾛｰ条件:T2.C2=?(2)
                  検索方法:INDEX SCAN
                  ｲﾝﾃﾞｸｽ名:I21
                 :ｲﾝﾃﾞｸｽID:196861
               :ｲﾝﾃﾞｸｽ属性:1
               :構成列情報:+C1
                  :ｻｰﾁ条件:AT[10-T1.C1]
```

内表のインデクスで結合条件を評価

図 3-20　UAP 統計レポートの出力例

対策前のアクセスパス

```
JOIN
 # Join  ID    : 1
   L Table     :  T1 0x00020079(131193)
   R Table     :  T2 0x0002007a(131194)  LIST{TLIST(1)}
   Join Type   : CROSS JOIN(INNER)  [SPECIFICATION IGNORED]
   IfThenCnd   : {T1.C1+T2.C1=10}
```

NESTED LOOPS JOINの指定は無効となり
CROSS JOINとなっている

対策後のアクセスパス

```
JOIN
 # Join  ID    : 1
   L Table     :  T1 0x00020079(131193)
   R Table     :  T2 0x0002007a(131194)
   Join Type   : 1-CLM NESTED LOOPS JOIN(INNER)  [AS SPECIFIED]
```

NESTED LOOPS JOINに変更

```
 # Table Name  : T2 0x0002007a(131194)
   Cost        : N (10000000ROW)  {T-21154.2891141213,I-45.05459603711365}
   RDAREA      : NON DIVISION  (1RD) [0x05(5)] ALL
   Scan Type   : INDEX SCAN
   Index Name  : I21 0x000300fd(196861) (1) (+C1)
                   SearchCnd : AT [10-T1.C1]
   RowCnd      : {T2.C2=?(2)}
```

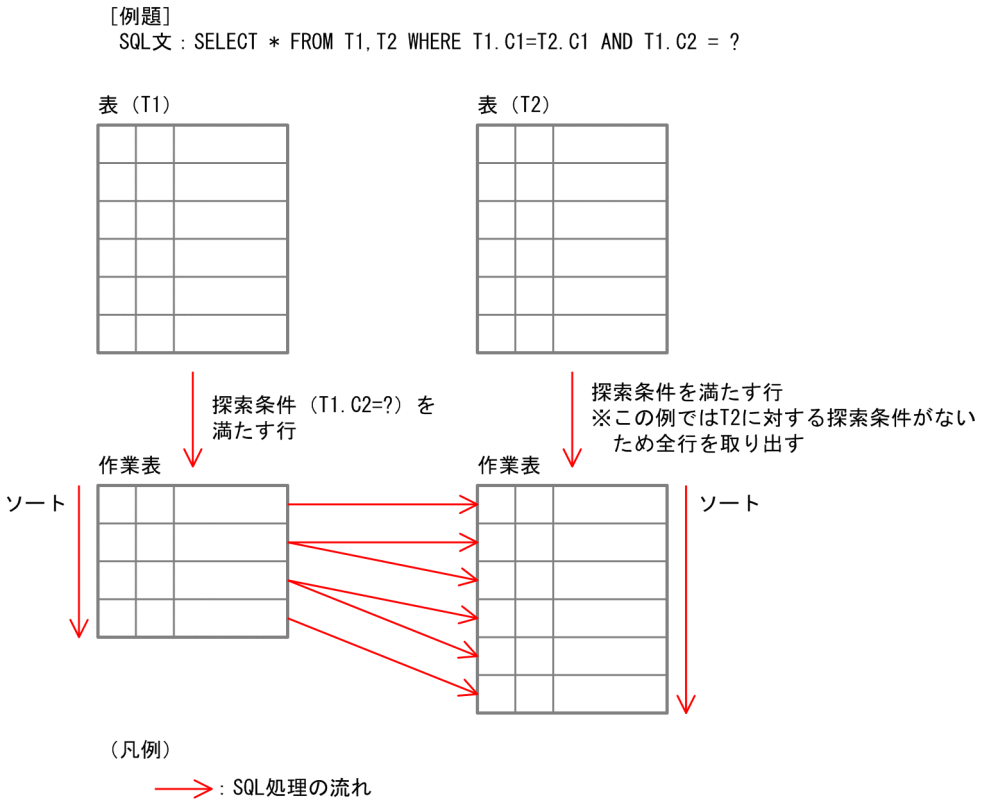
内表のインデクスで結合条件を評価

3.4.4　MERGE JOIN の対策

(1)　MERGE JOIN とは

MERGE JOIN は，結合列でソートして，結合列の値が小さいものから順に突き合わせるため，効率が悪いです。MERGE JOIN の中でも，SORT MERGE JOIN は，結合するそれぞれの表から行を取り出す際に，作業表を作成しソートするため，特に処理効率が悪いです。

図 3-21　SORT MERGE JOIN の処理方式

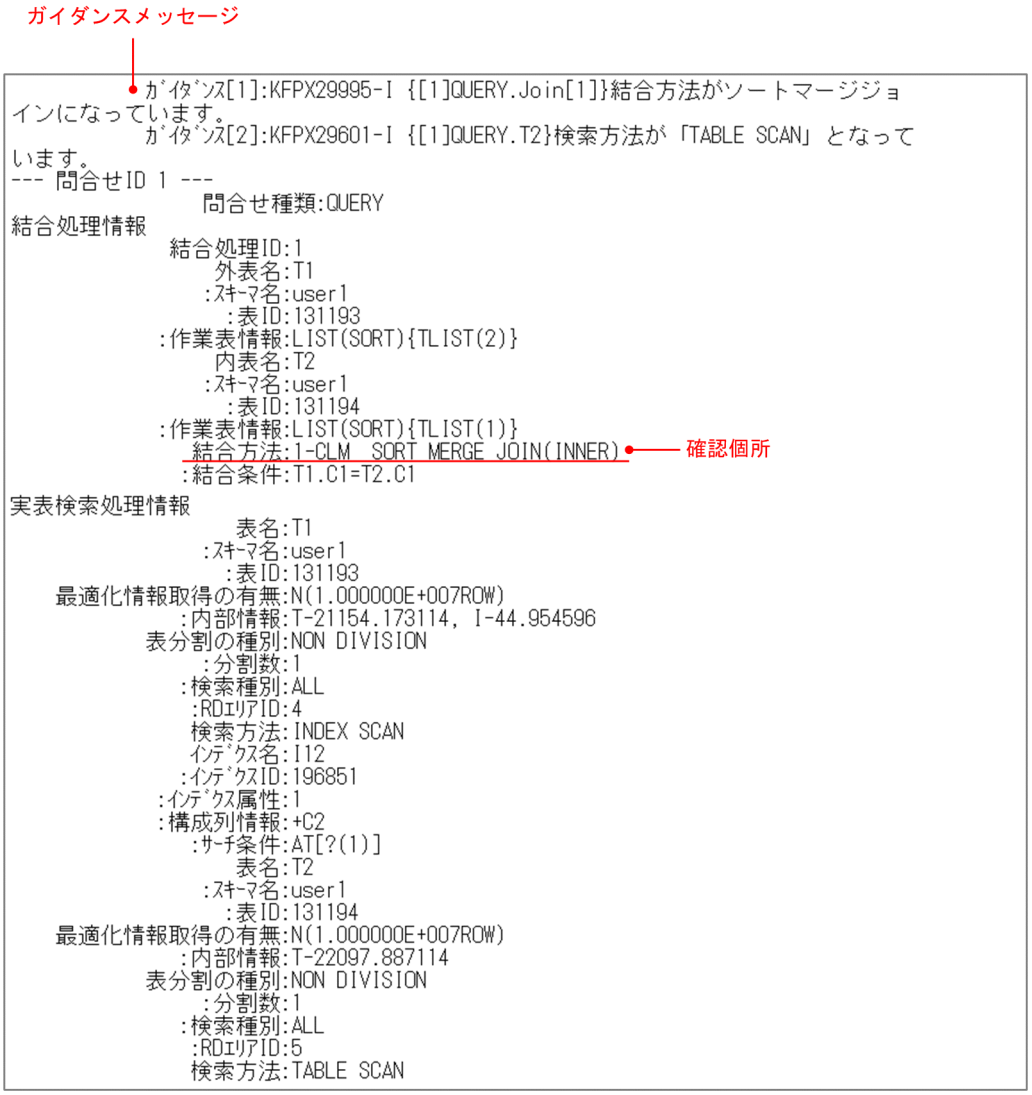


(2)　確認方法

MERGE JOIN の確認方法を次に示します。

- HiRDB SQL Tuning Advisor の場合
  アクセスパス情報の「結合方法」に「MERGE JOIN」と表示されます。
  SORT MERGE JOIN の出力例を次に示します。

図 3-22　HiRDB SQL Tuning Advisor の出力例（SORT MERGE JOIN）

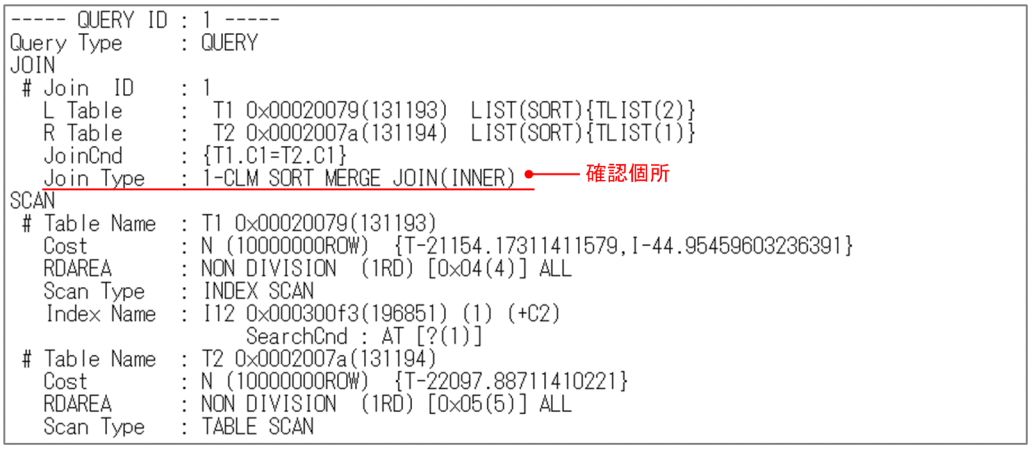


- UAP 統計レポートの場合

  アクセスパス情報の「Join Type」に「MERGE JOIN」と表示されます。

  SORT MERGE JOIN の出力例を次に示します。

図 3-23　UAP 統計レポートの出力例（SORT MERGE JOIN）



(3) 対策方法

SQL 文のヒット件数が少ない場合は NESTED LOOPS JOIN，SQL 文のヒット件数が多い場合は HASH JOIN に変更します。それぞれの結合方法への変更方法を次に説明します。

[NESTED LOOPS JOIN にするには]

結合方法を NESTED LOOPS JOIN にするときの手順を次に示します。

1. 内表の結合条件の列にインデクスを定義してください。この時，インデクス構成列には，結合条件の列をすべて含めてください。内表の結合条件の列は，アクセスパス情報の「結合条件（JoinCnd)」に表示された「内表名（R Table)」の列になります。
2. SQL 文の SQL 最適化指定で結合方式（BY NEST）を指定します。指定方法は，マニュアル「HiRDB SQL リファレンス」の「結合方式の SQL 最適化指定」を参照してください。

[こんなときは]

上記の手順を実施しても，NESTED LOOPS JOIN に変更できない場合は，次の変更を実施してください。

- 外表に結合条件以外の探索条件が指定できないか検討してください。
- HiRDB/パラレルサーバで，内表を横分割している場合は，分割キーを結合条件にいれてください。

[HASH JOIN にするには]

結合方法を HASH JOIN にするときの手順を次に示します。

1. HASH JOIN を実行できる環境設定をします。詳細は，マニュアル「HiRDB UAP 開発ガイド」の「ハッシュジョイン，副問合せのハッシュ実行を適用する場合の準備」を参照してください。
2. SQL 文の SQL 最適化指定で結合方式（BY HASH）を指定します。指定方法は，マニュアル「HiRDB SQL リファレンス」の「結合方式の SQL 最適化指定」を参照してください。

3.4.5 効率の悪い NESTED LOOPS JOIN の対策

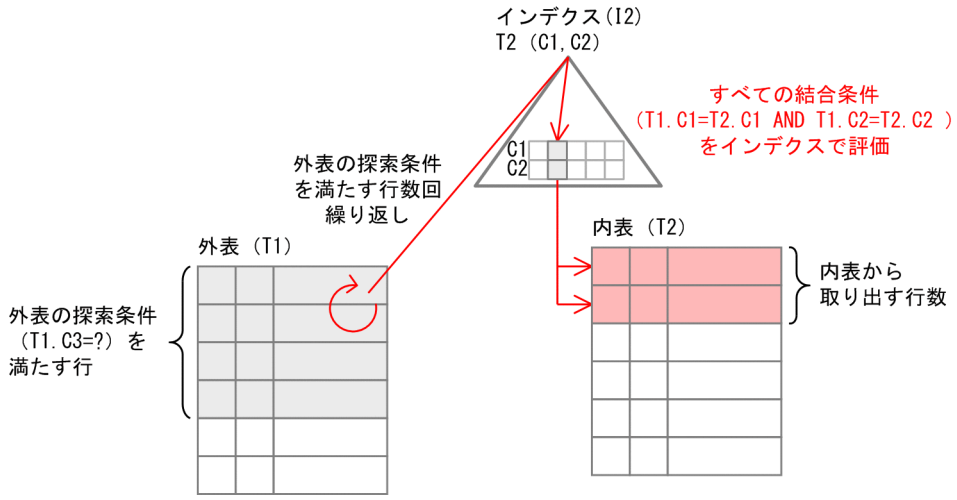
(1) 効率の悪い NESTED LOOPS JOIN とは

NESTED LOOPS JOIN とは，一方の表から 1 行ずつ行を取り出し，もう一方の表のそれぞれの行に突き合わせて，結合条件を満たす行を取り出す入れ子型のループ処理の結合方法です。先に取り出す表を「外表」，突合せる表を「内表」といいます。

内表のインデクスを利用して結合条件を評価することで，処理効率が良くなります。しかし，結合条件のすべての列がインデクスに含まれていない場合，結合条件の評価がインデクスだけでは判定できず，表データとの突合せが発生するため，処理効率が悪いです。

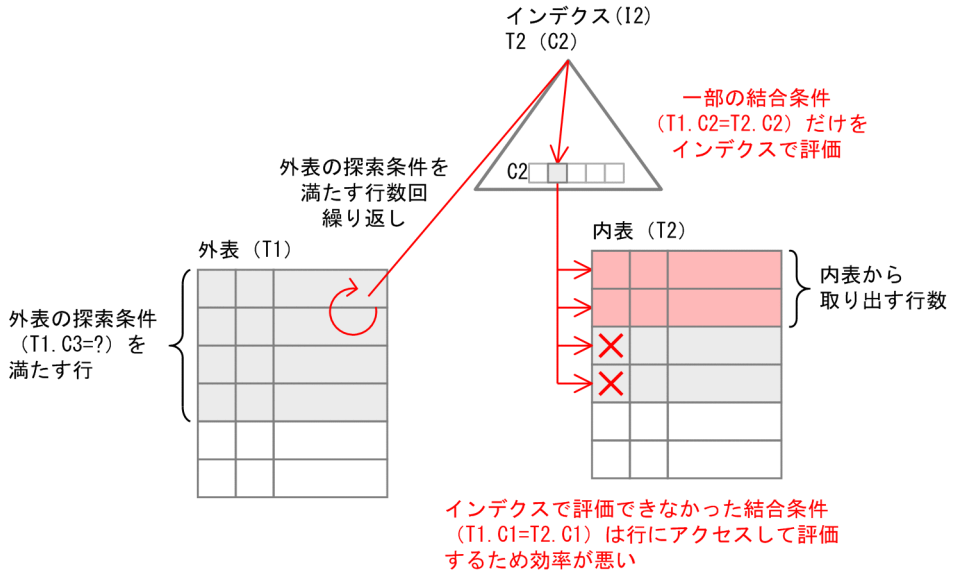
図 3-24　NESTED LOOPS JOIN の処理方式

●効率の良いNested Loops Joinの場合
［例題］
SQL文：SELECT * FROM T1,T2 WHERE T1.C1=T2.C1 AND T1.C2=T2.C2 AND T1.C3 = ?
インデクスI2：内表（T2）に第1構成列C1，第2構成列C2の複数列インデクスを定義



（凡例）
：SQL処理の流れ

●効率の悪いNested Loops Joinの場合
［例題］
SQL文：SELECT * FROM T1,T2 WHERE T1.C1=T2.C1 AND T1.C2=T2.C2 AND T1.C3 = ?
インデクスI2：内表（T2）に構成列C2の単一列インデクスを定義



（凡例）
：SQL処理の流れ

(2)　確認方法

効率の悪い NESTED LOOPS JOIN の確認方法を次に示します。

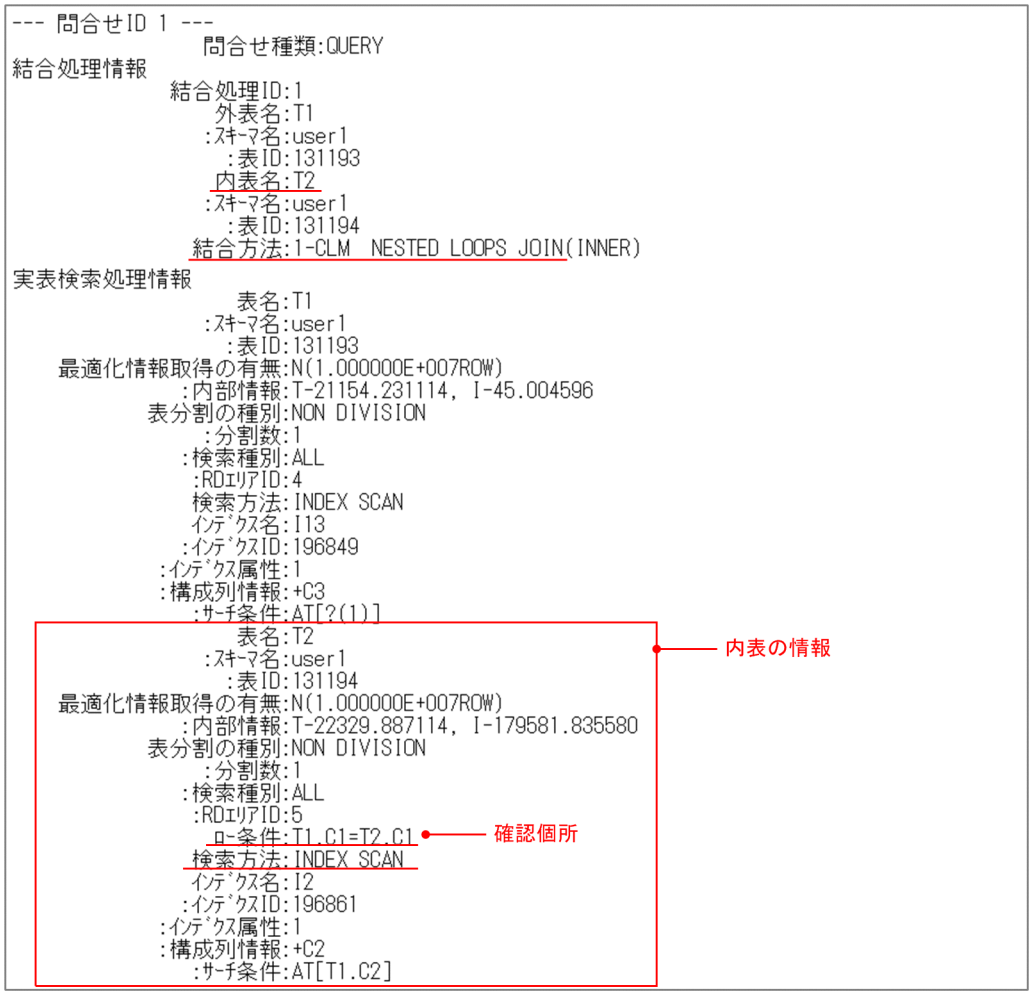
- HiRDB SQL Tuning Advisor の場合

次のすべての条件を満たすアクセスパスが該当します。

1.「結合方法」に「NESTED LOOPS JOIN」と表示されている
2.「内表名」に表示された表の「検索方法」に「INDEX SCAN」と表示されている
3.「検索方法」の上に「ﾛｰ条件」または「IF THEN 条件」の表示があり，その行に外表の列名が表示されている

HiRDB SQL Tuning Advisor の出力例を次に示します。

図 3-25　HiRDB SQL Tuning Advisor の出力例



- UAP 統計レポートの場合

次のすべての条件を満たすアクセスパスが該当します。

1.「Join Type」に「NESTED LOOPS JOIN」と表示されている
2.「R Table」に表示された表の「Scan Type」に「INDEX SCAN」と表示されている
3.「Scan Type」の上に「RowCnd」または「IfThenCnd」の表示があり，その行に外表の列名が表示されている

UAP 統計レポートの出力例を次に示します。

図 3-26　UAP 統計レポートの場合の出力例

```
----- QUERY ID : 1 -----
Query Type      : QUERY
JOIN
 # Join  ID     : 1
   L Table      :  T1 0x00020079(131193)
   R Table      :  T2 0x0002007a(131194)
   Join Type    : 1-CLM NESTED LOOPS JOIN(INNER)
SCAN
 # Table Name   : T1 0x00020079(131193)
   Cost         : N (10000000ROW)  {T-21154.23111411855,I-45.00459603473878}
   RDAREA       : NON DIVISION  (1RD) [0x04(4)] ALL
   Scan Type    : INDEX SCAN                                          内表の情報
   Index Name   : I13 0x000300f1(196849) (1) (+C3)
                      SearchCnd : AT [?(1)]
 # Table Name   : T2 0x0002007a(131194)
   Cost         : N (10000000ROW)  {T-22329.88711410221,I-179581.8355795134}
   RDAREA       : NON DIVISION  (1RD) [0x05(5)] ALL
   Scan Type    : INDEX SCAN
   Index Name   : I2 0x000300fd(196861) (1) (+C2)
                      SearchCnd : AT [T1.C2]
   RowCnd       : {T1.C1=T2.C1}          確認個所
```

(3)　対策方法

内表に定義するインデクスは，結合条件となる列をすべて構成列に含めてください。「ﾛｰ条件（RowCnd)」または「IF THEN 条件（IfThenCnd)」の行に表示されている内表の列をインデクス構成列に加えることで対策できます。この時，最も絞り込める探索条件を指定している列を，第 1 構成列にしてください。対策方法の例を次に示します。

図 3-27　対策方法の例（HiRDB SQL Tuning Advisor の場合）

対策前のアクセスパス（内表の情報）

```
                表名:T2
             :ｽｷｰﾏ名:user1
               :表ID:131194
最適化情報取得の有無:N(1.000000E+007ROW)
          :内部情報:T-22329.887114, I-179581.835580
      表分割の種別:NON DIVISION
           :分割数:1
         :検索種別:ALL
          :RDｴﾘｱID:5
            ﾛｰ条件:T1.C1=T2.C1          確認個所
          検索方法:INDEX SCAN
          ｲﾝﾃﾞｸｽ名:I2
         :ｲﾝﾃﾞｸｽID:196861
        :ｲﾝﾃﾞｸｽ属性:1
        :構成列情報:+C2
           :ｻｰﾁ条件:AT[T1.C2]
```

対策後のアクセスパス（内表の情報）

```
                表名:T2
             :ｽｷｰﾏ名:user1
               :表ID:131194
最適化情報取得の有無:N(1.000000E+007ROW)
          :内部情報:T-22329.887114, I-179581.835580
      表分割の種別:NON DIVISION
           :分割数:1
         :検索種別:ALL
          :RDｴﾘｱID:5
          検索方法:MULTI COLUMNS INDEX SCAN
          ｲﾝﾃﾞｸｽ名:I2
         :ｲﾝﾃﾞｸｽID:196861
        :ｲﾝﾃﾞｸｽ属性:2
        :構成列情報:+C1,+C2          最も絞り込めるC1列を第1構成列とする
           :ｻｰﾁ条件:AT[(T1.C1,T1.C2)]  複数列インデクスに変更
```

図 3-28　対策方法の例（UAP 統計レポートの場合）

対策前のアクセスパス（内表の情報）

```
# Table Name  : T2 0x0002007a(131194)
  Cost        : N (10000000ROW)  {T-22329.88711410221,I-179581.8355795134}
  RDAREA      : NON DIVISION  (1RD) [0x05(5)] ALL
  Scan Type   : INDEX SCAN
  Index Name  : I2 0x000300fd(196861) (1) (+C2)
                    SearchCnd : AT [T1.C2]
  RowCnd      : {T1.C1=T2.C1}          ←― 確認個所
```

対策後のアクセスパス（内表の情報）

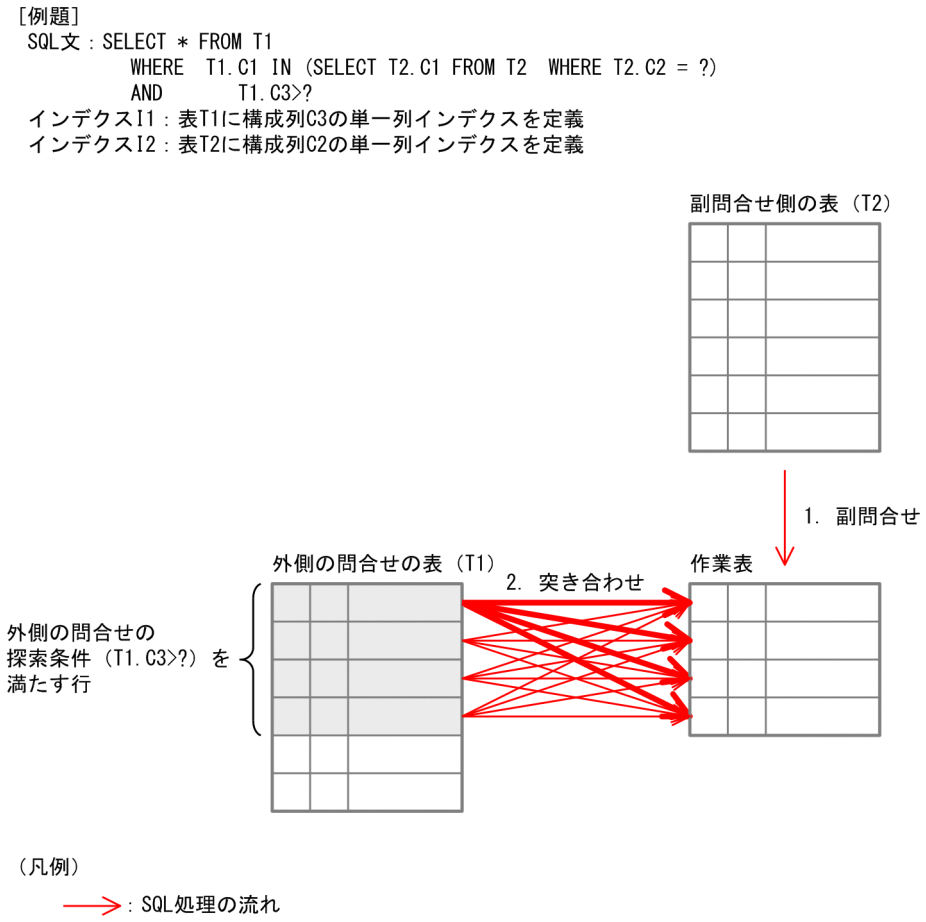
```
# Table Name  : T2 0x0002007a(131194)
  Cost        : N (10000000ROW)  {T-22329.88711410221,I-179581.8355795134}
  RDAREA      : NON DIVISION  (1RD) [0x05(5)] ALL
  Scan Type   : MULTI COLUMNS INDEX SCAN
  Index Name  : I2 0x000300fd(196861) (2) (+C1,+C2)   ←― 最も絞り込めるC1列を
                                                         第1構成列とする
                    SearchCnd : AT [(T1.C1,T1.C2)]     複数列インデクスに変更
```

3.4.6　WORK TABLE SUBQ の対策

(1)　WORK TABLE SUBQ とは

WORK TABLE SUBQ とは，限定述語および IN 述語に対する表副問合せに適用される副問合せの実行方式です。まず，副問合せの選択式の値を求めて，作業表を作成します（図中の 1)。次に，外側の問合せを検索し 1 行検索するごとに，副問合せの結果と突き合わせて探索条件を評価します（図中の 2)。このため，処理効率が悪いです。

図 3-29　WORK TABLE SUBQ の処理方式



(2)　確認方法

WORK TABLE SUBQ の確認方法を次に示します。

- HiRDB SQL Tuning Advisor の場合

  アクセスパス情報の「副問合せ実行方式」に「WORK TABLE SUBQ」と表示されます。WORK TABLE SUBQ の出力例を次に示します。

図 3-30　HiRDB SQL Tuning Advisor の出力例

```
--- 問合せID 1 ---
                問合せ種類:QUERY
実表検索処理情報
                   表名:T1
                 :ｽｷｰﾏ名:user1
                   :表ID:131193
    最適化情報取得の有無:N(1.000000E+007ROW)
              :内部情報:T-21384.159114, I-3959.658210
           表分割の種別:NON DIVISION
                :分割数:1
              :検索種別:ALL
               :RDｴﾘｱID:4
            IF-THEN条件:T1.C1=any(SUBQ(2))
               検索方法:INDEX SCAN
               ｲﾝﾃﾞｸｽ名:I1
              :ｲﾝﾃﾞｸｽID:196849
            :ｲﾝﾃﾞｸｽ属性:1
            :構成列情報:+C3
               :ｻｰﾁ条件:RANGE(OS-CE)[?(2),MAX]
--- 問合せID 2 ---
                問合せ種類:SUBQUERY
                作業表情報:LIST{TLIST(1)}
         副問合せ実行方式:WORK TABLE SUBQ ●―― 確認個所
実表検索処理情報
                   表名:T2
                 :ｽｷｰﾏ名:user1
                   :表ID:131194
    最適化情報取得の有無:N(1.000000E+007ROW)
              :内部情報:T-21154.115114, I-44.904596
           表分割の種別:NON DIVISION
                :分割数:1
              :検索種別:ALL
               :RDｴﾘｱID:5
               検索方法:INDEX SCAN
               ｲﾝﾃﾞｸｽ名:I2
              :ｲﾝﾃﾞｸｽID:196853
            :ｲﾝﾃﾞｸｽ属性:1
            :構成列情報:+C2
               :ｻｰﾁ条件:AT[?(1)]
```

- UAP 統計レポートの場合

  アクセスパス情報の「Sub Query Type」に「WORK TABLE SUBQ」と表示されます。WORK TABLE SUBQ の出力例を次に示します。

図 3-31　UAP 統計レポートの出力例

```
----- QUERY ID : 1 -----
Query Type     : QUERY
SCAN
 # Table Name  : T1 0x00020079(131193)
   Cost        : N (10000000ROW)  {T-21384.15911393818,I-3959.658209790875}
   RDAREA      : NON DIVISION  (1RD) [0x04(4)] ALL
   Scan Type   : INDEX SCAN
   Index Name  : I1 0x000300f1(196849) (1) (+C3)
                     SearchCnd : RANGE(OS-CE) [?(2),MAX]
   IfThenCnd   : {T1.C1=any(SUBQ(2))}
----- QUERY ID : 2 -----
Query Type     : SUBQUERY  LIST{TLIST(1)}
Sub Query Type : WORK TABLE SUBQ ●―― 確認個所
SCAN
 # Table Name  : T2 0x0002007a(131194)
   Cost        : N (10000000ROW)  {T-21154.11511411304,I-44.90459602998904}
   RDAREA      : NON DIVISION  (1RD) [0x05(5)] ALL
   Scan Type   : INDEX SCAN
   Index Name  : I2 0x000300f5(196853) (1) (+C2)
                     SearchCnd : AT [?(1)]
```

(3)　対策方法

適用条件を確認して，次のどちらかの方法で対策してください。

表 3-8　WORK TABLE SUBQ の対策方法

項番	対策方法	適用条件
1	インデクスを追加または変更して，副問合せの実行方式を WORK TABLE ATS SUBQ に変更してください。	• IN 述語または=ANY または=SOME の限定述語に対する副問合せである • 次の関係を満たす場合 外側の問合せのヒット件数＞副問合せのヒット件数 • インデクスの追加または構成列の変更ができる
2	EXISTS 述語を使った SQL 文への変更と，インデクスを追加または変更して，副問合せの実行方式を NESTED LOOPS ROW VALUE SUBQ に変更してください。	• 次の関係を満たす場合 外側の問合せのヒット件数＜副問合せのヒット件数 • SQL 文が変更できる • インデクスの追加または構成列の変更ができる

(a)　WORK TABLE ATS SUBQ への変更

インデクスを追加または変更して，副問合せの実行方式を WORK TABLE ATS SUBQ に変更します。これによって，外側の問合せのインデクスを使用して副問合せの結果との条件を評価するため，効率がよいです。また，副問合せのヒット件数分の値を探索条件に指定し，外側の問合せの表を検索するため，副問合せのヒット件数が少ない場合に効果があります。

次の個所に表示された列を外側の問合せで使用しているインデクスの構成列に加えることで，WORK TABLE ATS SUBQ に変更できます。外側の問合せが TABLE SCAN の場合は，次の個所に表示された列にインデクスを追加してください。

- HiRDB SQL Tuning Advisor の場合

  アクセスパス情報の外側の問合せの「IF-THEN 条件」で，副問合せ（SUBQ）の結果と条件評価している列をインデクスの構成列に追加してください。

図 3-32　HiRDB SQL Tuning Advisor の出力例

対策前のアクセスパス

```
--- 問合せID 1 ---
                問合せ種類:QUERY ←― 外側の問合せ
実表検索処理情報
                   表名:T1
                 :ｽｷｰﾏ名:user1
                   :表ID:131193
    最適化情報取得の有無:N(1.000000E+007ROW)
                :内部情報:T-21384.159114, I-3959.658210
            表分割の種別:NON DIVISION
                 :分割数:1
               :検索種別:ALL
                :RDｴﾘｱID:4
             IF-THEN条件:T1.C1=any(SUBQ(2)) ←― 確認個所
                検索方法:INDEX SCAN
                ｲﾝﾃﾞｸｽ名:I1
               :ｲﾝﾃﾞｸｽID:196849
             :ｲﾝﾃﾞｸｽ属性:1
             :構成列情報:+C3
                :ｻｰﾁ条件:RANGE(OS-CE)[?(2),MAX]
--- 問合せID 2 ---
                問合せ種類:SUBQUERY
                作業表情報:LIST{TLIST(1)}
          副問合せ実行方式:WORK TABLE SUBQ
```

対策後のアクセスパス

```
--- 問合せID 1 ---
                問合せ種類:QUERY
実表検索処理情報
                   表名:T1
                 :ｽｷｰﾏ名:user1
                   :表ID:131193
    最適化情報取得の有無:N(1.000000E+007ROW)
                :内部情報:T-21384.159114, I-71.464979
            表分割の種別:NON DIVISION
                 :分割数:1
               :検索種別:ALL
                :RDｴﾘｱID:4
                検索方法:MULTI COLUMNS INDEX SCAN
                ｲﾝﾃﾞｸｽ名:I1
               :ｲﾝﾃﾞｸｽID:196849                    最も絞り込めるC1列を
             :ｲﾝﾃﾞｸｽ属性:2                        第1構成列とする
             :構成列情報:+C1,+C3 ←――――――――――― 複数列インデクスに変更
                :ｻｰﾁ条件:RANGES(CS-CE)[((SUBQ(2)),?(2)),((SUBQ(2)),MAX)]
                 :ｷｰ条件:T1.C3>?(2)
--- 問合せID 2 ---
                問合せ種類:SUBQUERY
                作業表情報:LIST(SORT){TLIST(1)}
          副問合せ実行方式:WORK TABLE ATS SUBQ ←― WORK TABLE ATS SUBQに変更
```

- UAP 統計レポートの場合

  アクセスパス情報の外側の問合せの「IfThenCnd」で，副問合せ（SUBQ）の結果と条件評価している列をインデクスの構成列に追加してください。

図 3-33　UAP 統計レポートの出力例

対策前のアクセスパス

```
----- QUERY ID : 1 -----
Query Type      : QUERY ●―― 外側の問合せ
SCAN
 # Table Name   : T1 0x00020079(131193)
   Cost         : N (10000000ROW)  {T-21384.15911393818,I-3959.658209790875}
   RDAREA       : NON DIVISION  (1RD) [0x04(4)] ALL
   Scan Type    : INDEX SCAN
   Index Name   : I1 0x000300f1(196849) (1) (+C3)
                  SearchCnd : RANGE(OS-CE) [?(2),MAX]
   IfThenCnd    : {T1.C1=any(SUBQ(2))}●―― 確認個所
----- QUERY ID : 2 -----
Query Type      : SUBQUERY  LIST{TLIST(1)}
Sub Query Type  : WORK TABLE SUBQ
```

対策後のアクセスパス

```
----- QUERY ID : 1 -----
Query Type      : QUERY
SCAN
 # Table Name   : T1 0x00020079(131193)
   Cost         : N (10000000ROW)  {T-21384.15911393818,I-71.46497906725503}
   RDAREA       : NON DIVISION  (1RD) [0x04(4)] ALL
   Scan Type    : MULTI COLUMNS INDEX SCAN
   Index Name   : I1 0x000300f1(196849) (2) (+C1,+C3)●―― 最も絞り込めるC1列を第1構成列
                                                         とする複数列インデクスに変更
                  SearchCnd : RANGES(CS-CE) [((SUBQ(2)),?(2)),((SUBQ(2)),MAX)]
                  KeyCnd    : T1.C3>?(2)
----- QUERY ID : 2 -----
Query Type      : SUBQUERY  LIST(SORT){TLIST(1)}
Sub Query Type  : WORK TABLE ATS SUBQ●―― WORK TABLE ATS SUBQに変更
```

[注意事項]

インデクスに構成列を追加する場合は，探索条件の指定方法を考慮して，構成列の順序を決定してください。詳細は，マニュアル「HiRDB システム導入・設計ガイド」の「インデクス構成列の検討」を参照してください。

(b)　NESTED LOOPS ROW VALUE SUBQ への変更

EXISTS 述語使った SQL 文に変更して，副問合せの実行方式を NESTED LOOPS ROW VALUE SUBQ に変更します。これによって，副問合せのインデクスを使用して，外側の問合せの結果との条件を評価するため，効率がよいです。また，この方法では，外側の問合せのヒット件数分，副問合せを実行するため，外側の問合せのヒット件数が少ない場合に効果があります。SQL 文の変更方法の例を次に示します。

表 3-9　NESTED LOOPS ROW VALUE SUBQ への変更例

述語	変更前	変更後
IN 述語（IN）	SELECT * FROM T1 WHERE T1.C1 IN (SELECT T2.C1 FROM T2 WHERE T2.C2 = ?) AND T1.C3>?	SELECT * FROM T1 WHERE EXISTS(SELECT * FROM T2 WHERE T2.C2 = ? AND T1.C1 = T2.C1) AND T1.C3>?
IN 述語（NOT IN）	SELECT * FROM T1 WHERE T1.C1 NOT IN (SELECT T2.C1 FROM T2 WHERE T2.C2 = ?) AND T1.C3>?	SELECT * FROM T1 WHERE NOT EXISTS(SELECT * FROM T2 WHERE T2.C2 = ? AND T1.C1 = T2.C1) AND T1.C3>?

述語	変更前	変更後
限定述語 (=ANY または =SOME)	SELECT * FROM T1 WHERE T1.C1 = ANY(SELECT T2.C1 FROM T2 WHERE T2.C2 = ?) AND T1.C3>?	SELECT * FROM T1 WHERE EXISTS(SELECT * FROM T2 WHERE T2.C2 = ? AND T1.C1 = T2.C1) AND T1.C3>?

また，副問合せのすべての探索条件が 1 つのインデクスで評価できるようにしてください。1 つのインデクスで評価できているか確認する方法は，「効率の悪い NESTED LOOPS JOIN」の手順を参照してください。

IN 述語の例題の SQL 文について，対策後のアクセスパスの出力例を次に示します。この例では，副問合せ側の表 T2 の列 C2 と C1 に複数列インデクスを定義します。

図 3-34　HiRDB SQL Tuning Advisor の出力例

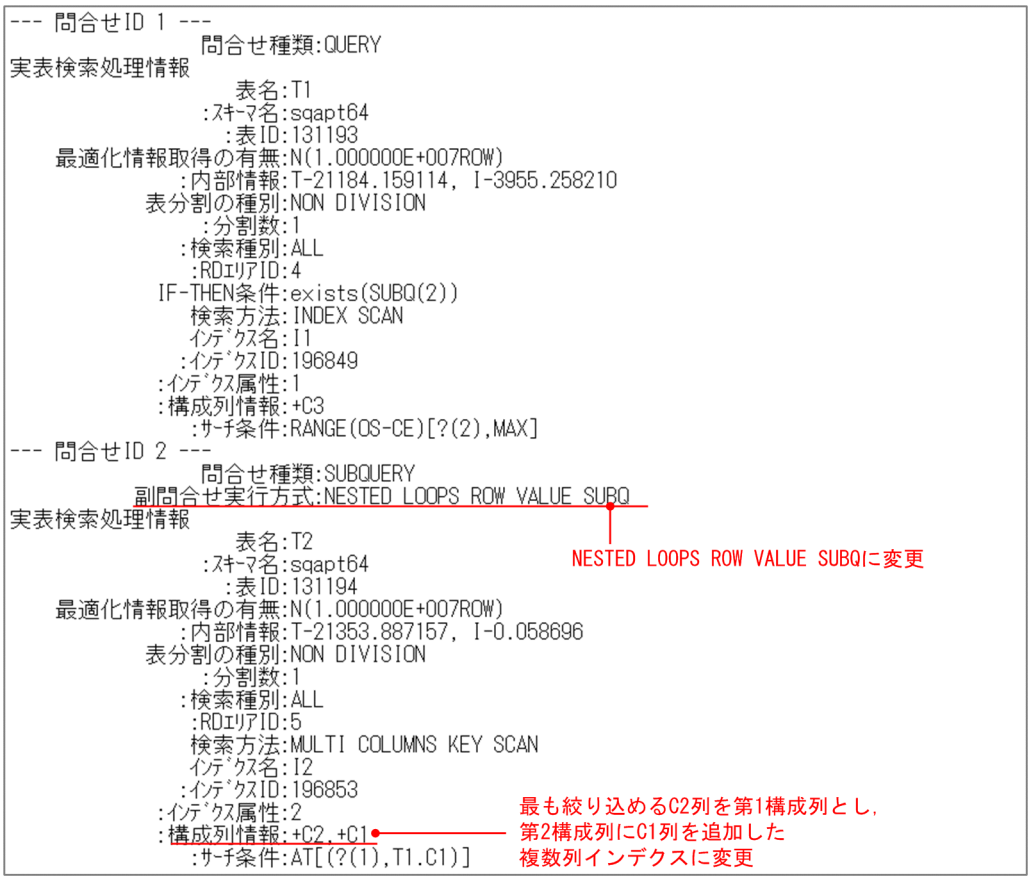


図 3-35　UAP 統計レポートの出力例

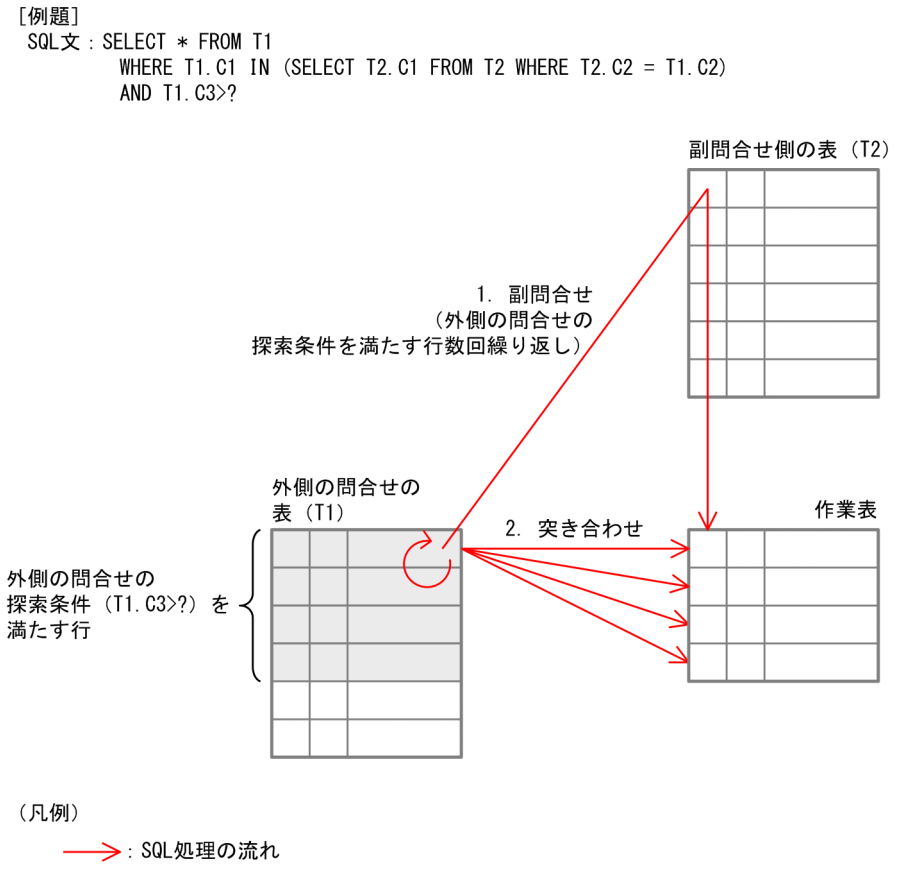
```
----- QUERY ID : 1 -----
Query Type      : QUERY
SCAN
 # Table Name   : T1 0x00020079(131193)
   Cost         : N (10000000ROW)  {T-21184.15911393818,I-3955.258209814717}
   RDAREA       : NON DIVISION  (1RD) [0x04(4)] ALL
   Scan Type    : INDEX SCAN
   Index Name   : I1 0x000300f1(196849) (1) (+C3)
                      SearchCnd : RANGE(OS-CE) [?(2),MAX]
   IfThenCnd    : {exists(SUBQ(2))}
----- QUERY ID : 2 -----
Query Type      : SUBQUERY
Sub Query Type  : NESTED LOOPS ROW VALUE SUBQ ◀── NESTED LOOPS ROW VALUE SUBQに変更
SCAN
 # Table Name   : T2 0x0002007a(131194)
   Cost         : N (10000000ROW)  {T-21353.88715660221,I-0.05869551714242759}
   RDAREA       : NON DIVISION  (1RD) [0x05(5)] ALL
   Scan Type    : MULTI COLUMNS KEY SCAN
   Index Name   : I2 0x000300f5(196853) (2) (+C2,+C1) ◀── 第2構成列にC1列を追加した
                      SearchCnd : AT [(?(1),T1.C1)]      複数列インデクスに変更
                                                         最も絞り込めるC2列を第1構成列とし，
```

3.4.7　NESTED LOOPS WORK TABLE SUBQ の対策

(1)　NESTED LOOPS WORK TABLE SUBQ とは

NESTED LOOPS WORK TABLE SUBQ とは，限定述語および IN 述語に対する表副問合せに適用されるアクセスパスです。外側の問合せを 1 行取り出すごとに外への参照列の値を使用して副問合せを実行し，副問合せの選択式の値を求めて作業表を作成します（図中の 1)。そして，副問合せから作成した作業表と突き合わせて，外側の副問合せを含む条件を評価します（図中の 2)。このため，処理効率が悪いです。

図 3-36　NESTED LOOPS WORK TABLE SUBQ の処理方式



(2)　確認方法

NESTED LOOPS WORK TABLE SUBQ の確認方法を次に示します。

- HiRDB SQL Tuning Advisor の場合

  アクセスパス情報の「副問合せ実行方式」に「NESTED LOOPS WORK TABLE SUBQ」と表示されます。NESTED LOOPS WORK TABLE SUBQ の出力例を次に示します。

図 3-37　HiRDB SQL Tuning Advisor の出力例

```
--- 問合せID 1 ---
                問合せ種類:QUERY
実表検索処理情報
                   表名:T1
                 :ｽｷｰﾏ名:user1
                   :表ID:131193
    最適化情報取得の有無:N(1.000000E+007ROW)
                 :内部情報:T-21184.159114, I-3955.258210
            表分割の種別:NON DIVISION
                 :分割数:1
               :検索種別:ALL
                :RDｴﾘｱID:4
             IF-THEN条件:T1.C1=any(SUBQ(2))
                検索方法:INDEX SCAN
                ｲﾝﾃﾞｸｽ名:I13
               :ｲﾝﾃﾞｸｽID:196849
             :ｲﾝﾃﾞｸｽ属性:1
             :構成列情報:+C3
                :ｻｰﾁ条件:RANGE(OS-CE)[?(1),MAX]
--- 問合せID 2 ---
                問合せ種類:SUBQUERY
                作業表情報:LIST{TLIST(1)}
          副問合せ実行方式:NESTED LOOPS WORK TABLE SUBQ ●── 確認個所
実表検索処理情報
                   表名:T2
                 :ｽｷｰﾏ名:user1
                   :表ID:131194
    最適化情報取得の有無:N(1.000000E+007ROW)
                 :内部情報:T-21154.115114, I-44.904596
            表分割の種別:NON DIVISION
                 :分割数:1
               :検索種別:ALL
                :RDｴﾘｱID:5
                検索方法:INDEX SCAN
                ｲﾝﾃﾞｸｽ名:I22
               :ｲﾝﾃﾞｸｽID:196862
             :ｲﾝﾃﾞｸｽ属性:1
             :構成列情報:+C2
                :ｻｰﾁ条件:AT[T1.C2]
```

- UAP 統計レポートの場合

  アクセスパス情報の「Sub Query Type」に「NESTED LOOPS WORK TABLE SUBQ」と表示されます。NESTED LOOPS WORK TABLE SUBQ の出力例を次に示します。

図 3-38　UAP 統計レポートの出力例

```
----- QUERY ID : 1 -----
Query Type     : QUERY
SCAN
 # Table Name  : T1 0x00020079(131193)
   Cost        : N (10000000ROW)  {T-21184.15911393818,I-3955.258209814717}
   RDAREA      : NON DIVISION  (1RD) [0x04(4)] ALL
   Scan Type   : INDEX SCAN
   Index Name  : I13 0x000300f1(196849) (1) (+C3)
                     SearchCnd : RANGE(OS-CE) [?(1),MAX]
   IfThenCnd   : {T1.C1=any(SUBQ(2))}
----- QUERY ID : 2 -----
Query Type     : SUBQUERY  LIST{TLIST(1)}
Sub Query Type : NESTED LOOPS WORK TABLE SUBQ ●── 確認個所
SCAN
 # Table Name  : T2 0x0002007a(131194)
   Cost        : N (10000000ROW)  {T-21154.11511411304,I-44.90459602998904}
   RDAREA      : NON DIVISION  (1RD) [0x05(5)] ALL
   Scan Type   : INDEX SCAN
   Index Name  : I22 0x000300fe(196862) (1) (+C2)
                     SearchCnd : AT [T1.C2]
```

(3)　対策方法

EXISTS 述語を使った SQL 文に変更して，副問合せの実行方式を NESTED LOOPS ROW VALUE SUBQ に変更します。これによって，作業表を使用しないため処理効率が良くなります。

変更方法の例は，「WORK TABLE SUBQ の対策」の表「NESTED LOOPS ROW VALUE SUBQ への変更例」を参照してください。

4 表にデータを格納したら

この章では，表にデータを格納した時点で確認する内容について説明します。

4.1　SQL トレースと UAP 統計レポートを確認しよう

表にデータを格納したら，アプリケーションを実行して SQL トレースと UAP 統計レポートを取得してください。そして，個々の SQL 性能について確認してください。

［前提条件］

性能を正確に調査するための前提条件について，次に示します。

- 性能を確認する際に接続する HiRDB サーバは，「アクセスパスを確認しよう」で説明した内容を本番環境と同じにしておく必要があります。また，次の設定についても同じ内容にしてください。
  ・HiRDB システム定義
  ・OS の環境変数
- アプリケーションを実行する環境は，次の設定を本番環境と同じ内容にしてください。
  ・HiRDB クライアント環境定義
  ・OS の環境変数
- 性能を確認する際に接続する HiRDB サーバには，次の点を満たすデータを格納してください。
  ・データ量が本番環境と同等である。
  ・データの種類と分布が本番環境と同等である。
  ・データの格納順序が本番環境と同等である。ただし，格納順序を同等にすることが難しい場合は，探索条件に指定する列の値の順序ではなく，無作為な順序で格納してください。

［注意事項］

性能を調査する際の注意事項について，次に説明します。

- 更新 SQL 文を使用するアプリケーションを実行する場合は，データが更新されます。

4.2　SQL トレースと UAP 統計レポートの取得方法

アプリケーション実行時に，SQL トレースと UAP 統計レポートを出力する方法について説明します。

4.2.1　HiRDB サーバ側の準備

HiRDB サーバ側で準備することを，次に説明します。

- HiRDB サーバを起動して，アプリケーションが接続できる状態にします。
- 表とインデクスの定義をしておきます。
- 性能評価で必要なデータを格納します。格納データに関する前提条件は，「SQL トレースと UAP 統計レポートを確認しよう」を参照してください。

4.2.2　SQL トレースと UAP 統計レポートを取得する設定

アプリケーションを実行する環境に，次のクライアント環境定義を設定してください。

表 4-1　設定するクライアント環境定義

目的	環境定義名	設定値
SQL トレースと UAP 統計レポートを取得する設定	PDCLTPATH	SQL トレースと UAP 統計レポートの出力先ディレクトリを指定します。
	PDSQLTRACE	SQL トレースと UAP 統計レポートのファイルサイズの最大値を指定します。サイズの上限を指定しない「0」を設定することを推奨します。
	PDUAPREPLVL	UAP 統計レポートの取得レベルは「a」または「at」を設定してください。 「a」を指定すると次の情報が取得されます。 • SQL 単位の情報 • UAP 単位またはトランザクション単位の情報 • アクセスパス情報 • SQL 実行時の中間結果情報 コネクションプーリング機能を使用しているアプリケーションは，UAP 単位の情報をトランザクション単位で出力するよう「at」を設定してください。

クライアント環境定義の指定方法の詳細は，マニュアル「HiRDB UAP 開発ガイド」の「クライアント環境定義の設定内容」を参照してください。

4.2.3　アプリケーションの実行

アプリケーションを実行すると，クライアント環境定義 PDCLTPATH に指定したのディレクトリに SQL トレースと UAP 統計レポートの情報が同じファイルに出力されます。出力ファイル名は，マニュアル「HiRDB UAP 開発ガイド」の「SQL トレース機能」を参照してください。

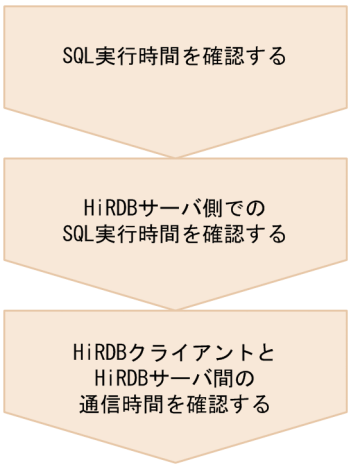
4.3　SQL トレースと UAP 統計レポートの確認方法

4.3.1　ここは必ず確認しよう

(1)　初期調査方法の流れ

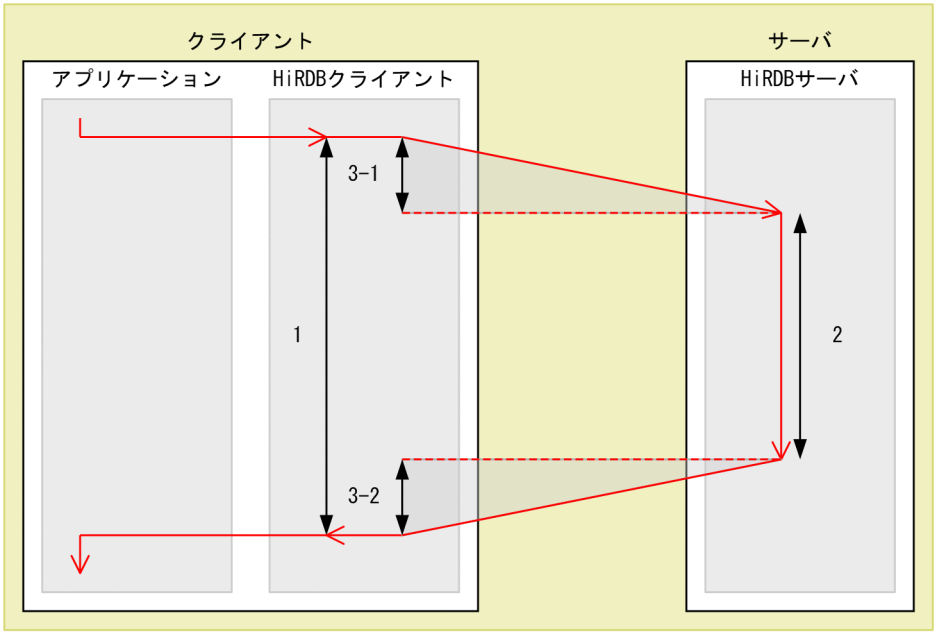
アプリケーションの性能に問題がある場合は，SQL トレースと UAP 統計レポートを使って，SQL の性能を確認してください。SQL トレースと UAP 統計レポートを使用した SQL 性能の初期調査の流れを次に示します。

図 4-1　初期調査の流れ



「SQL 実行時間」，「HiRDB サーバ側での SQL 実行時間」および「HiRDB クライアントと HiRDB サーバ間の通信時間」が SQL 処理のどの範囲に該当するか，次に示します。

図 4-2　SQL 処理での確認時間の範囲



（凡例）

→：SQL処理の流れ

[説明]

1. SQL 実行時間
2. サーバ側での SQL 実行時間
3. HiRDB クライアントと HiRDB サーバ間の通信時間（図中の 3-1 と 3-2 を足した時間）

(2)　初期調査で確認する情報の一覧

SQL トレースと UAP 統計レポートの情報のうち，初期調査で確認する情報について，次に示します。

表 4-2　初期調査で確認する情報

目的	情報の分類	情報の内容	情報名	HiRDB SQL Tuning Advisor の画面名および項目名
SQL 文を特定する	SQL 単位の情報	コネクト通番	CNCT NO	SQL トレース集計情報画面（SQL 単位）(CONNECT-NO)
		UAP のプロセス番号	CLPID	SQL トレース集計情報画面（SQL 単位）CLT-PID
		UAP のスレッド番号	CLTID	SQL トレース集計情報画面（SQL 単位）CLT-TID
		SQL カウンタ	NO	SQL トレース詳細画面 NO
		オペレーションコード	OP	SQL トレース詳細画面

目的	情報の分類	情報の内容	情報名	HiRDB SQL Tuning Advisor の画面名および項目名
			CODE	OP-CODE
		セクション番号	SEC NO	SQL トレース集計情報画面（SQL 単位）(SEC-NO)
		SQL 文	SQL	SQL トレース集計情報画面（SQL 単位）SQL
SQL 実行結果を確認する	SQL 単位の情報	SQL 文を実行した結果	SQL CODE	SQL トレース詳細画面 SQL-CODE
		警告情報	SQL WARN	SQL トレース詳細画面 WARN
SQL 実行時間を確認する	SQL 単位の情報	SQL 実行時間	EXEC-TIME	SQL トレース集計情報画面（SQL 単位）合計実行時間
				SQL トレース詳細画面 実行時間
HiRDB サーバ側での SQL 実行時間を確認する	SQL 単位の情報	サーバ側での SQL 実行時間	SVR EXEC-TIME	SQL トレース集計情報画面（SQL 単位）SVR EXEC-TIME
				SQL トレース詳細画面 SVR EXEC-TIME
	UAP 単位またはトランザクション単位の情報	データベースに対する入出力時間の累計	IOTIM	UAP 統計情報画面 IOTIM
			IOTIMM	UAP 統計情報画面 IOTIMM
		排他待ち時間	WAITT	UAP 統計情報画面 WAITT
HiRDB クライアントと HiRDB サーバ間の通信時間を確認する	SQL 単位の情報	通信処理の時間の目安	DIFF	SQL トレース詳細画面 DIFF
		処理行数	EXEC COUNT	SQL トレース詳細画面 EXEC COUNT

(3) 出力ファイルから調査したいアプリケーションの情報を特定する方法

SQL トレースと UAP 統計レポートの出力ファイル名にコネクト通番など接続ごとの情報が含まれていない場合（pd2sql1.trc，pdsql1.trc など），1 つのファイルに複数のアプリケーションの情報が出力されています。出力ファイル名の詳細は，マニュアル「HiRDB UAP 開発ガイド」の「SQL トレース機能」を参照してください。

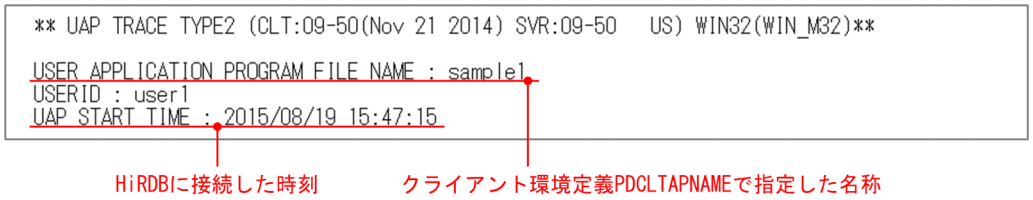
1 つのファイルに複数のアプリケーションの情報が出力されている場合は，調査したいアプリケーションの情報を次に示す手順で特定してください。

1. 出力ファイルを次のキーワードで検索して，調査したいアプリケーションが HiRDB 接続時に出力した情報を特定します。

表 4-3　接続時に出力する情報のキーワード

キーワード	表示内容
USER APPLICATION PROGRAM FILE NAME	クライアント環境定義 PDCLTAPNAME で指定した名称
UAP START TIME	HiRDB に接続した時刻

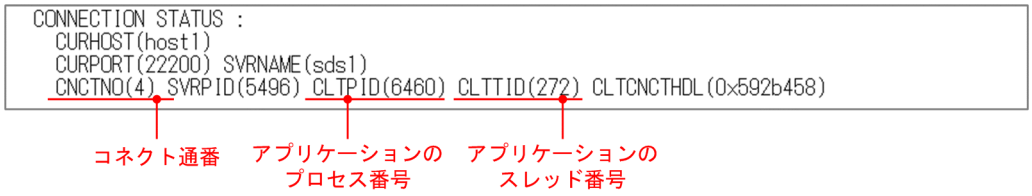
図 4-3　HiRDB 接続時に出力する情報の先頭部分



2. 特定した情報の下に表示されている CONNECTION STATUS（UAP 実行ステータス）から，次の情報を確認します。

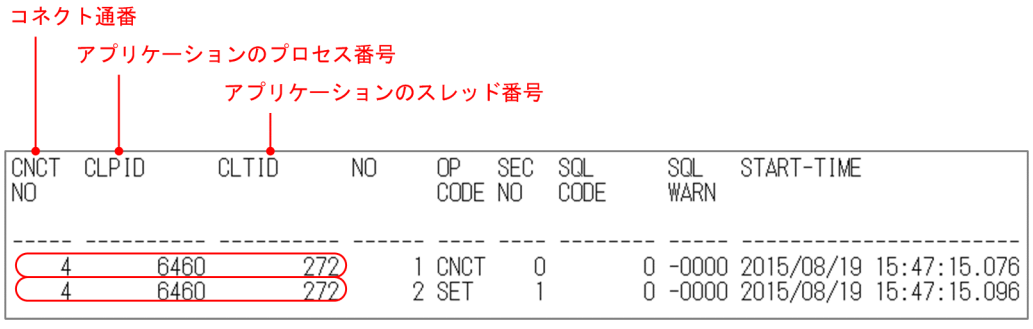
- CNCTNO（コネクト通番）
- CLTPID（UAP のプロセス番号）
- CLTTID（UAP のスレッド番号）

図 4-4　HiRDB 接続時に出力する CONNECTION STATUS



3. 調査したいアプリケーションの SQL 単位の情報は，手順 2 で確認した値が，行の冒頭に表示されています。この値を基に，SQL 単位の情報を特定してください。

図 4-5　SQL 単位の情報

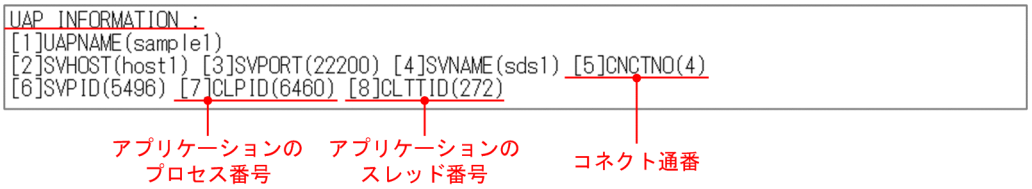


注

この図では，SQL 単位の情報の右側部分を省略しています。

4. UAP 単位またはトランザクション単位の情報の特定方法を説明します。UAP 単位の情報とトランザクション単位の情報では，最初の行に出力されるキーワードが異なります。クライアント環境定義 PDUAPREPLVL の設定値を確認して，どちらかのキーワードで検索してください。
   - UAP 単位の情報の場合：UAP INFORMATION
   - トランザクション単位の情報：TRANSACTION INFORMATION
5. UAP 単位またはトランザクション単位の情報についても，手順 2 で確認した値を基に特定してください。

図 4-6　UAP 単位の情報



4.3.2　SQL 実行時間を確認する

(1)　確認する情報

「SQL 実行時間」を確認し，性能要件を満たしているか確認してください。また，エラーや警告が発生している影響によって，実行に時間が掛かることもあります。エラーや警告が発生していないか，SQL 実行時間とあわせて確認してください。確認する情報名について，次に示します。

表 4-4　確認する情報名（SQL 実行時間の確認）

目的	情報の分類	情報の内容	情報名	HiRDB SQL Tuning Advisor の画面名および項目名
SQL 実行結果を確認する	SQL 単位の情報	SQL 文を実行した結果	SQLCODE	SQL トレース詳細画面 SQL-CODE
		警告情報	SQLWARN	SQL トレース詳細画面 WARN
SQL 実行時間を確認する		SQL 実行時間	EXEC-TIME	SQL トレース集計情報画面（SQL 単位） 合計実行時間
				SQL トレース詳細画面 実行時間

(2)　確認する情報の表示例

SQL 実行時間を確認する情報を HiRDB SQL Tuning Advisor で表示する例を次に示します。

(a) SQL トレース集計情報画面（SQL 単位）

SQL トレース集計情報画面（SQL 単位）では，「合計実行時間」を確認してください。「合計実行時間」には，各オペレーションの実行時間の合計が表示されます。時間順に表示を並び替えることもできます。

「合計実行時間」が性能要件を満たしていない SQL は，SQL トレース詳細画面で各オペレーションの実行時間や実行結果を確認してください。SQL トレース詳細画面は，SQL トレース集計情報画面（SQL 単位）で確認したい SQL 文の行をダブルクリックすると表示できます。表示方法の詳細は，HiRDB SQL Tuning Advisor のヘルプを参照してください。

SQL トレース集計情報画面（SQL 単位）の表示例を次に示します。

図 4-7 SQL トレース集計情報画面（SQL 単位）「合計実行時間」の表示例

[説明]

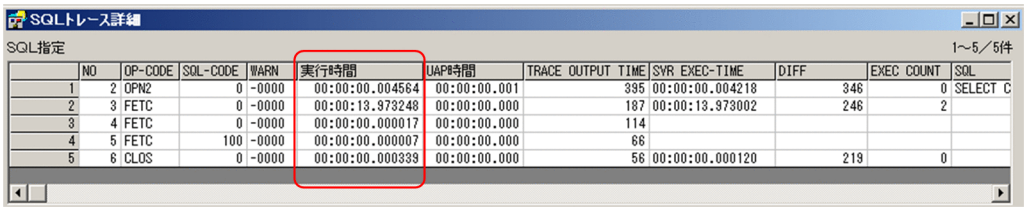
この画面では，「合計実行時間」順にソートしています。

(b) SQL トレース詳細画面

SQL トレース詳細画面では，各オペレーションの「実行時間」が確認できます。

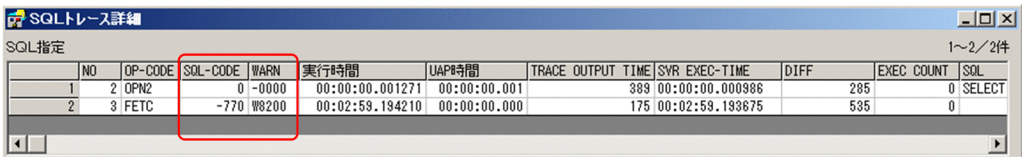
SQL トレース詳細画面の表示例を次に示します。

図 4-8 SQL トレース詳細画面「実行時間」の表示例



SQL 実行結果は，「SQL-CODE」および「WARN」を確認してください。エラーが発生している場合の SQL トレース詳細画面の表示例を次に示します。

図 4-9 SQL トレース詳細画面「SQL-CODE」および「WARN」の表示例



(3) 調査の進め方

(a) EXEC-TIME が性能要件を満たしていない場合

EXEC-TIME（SQL 実行時間）が性能要件を満たしていない場合は，HiRDB サーバ側での SQL 実行時間を確認してください。確認方法は，「HiRDB サーバ側での SQL 実行時間を確認する」を参照してください。

(b) EXEC-TIME が性能要件を満たしている場合

EXEC-TIME（SQL 実行時間）が性能要件を満たしているにも関わらず，アプリケーションが性能要件を満たしていない場合は，次に示す原因が考えられます。

1. アプリケーションの処理に時間が掛かっている
2. SQL トレースの出力処理に時間が掛かっている

次に示す情報を確認して原因を特定してください。

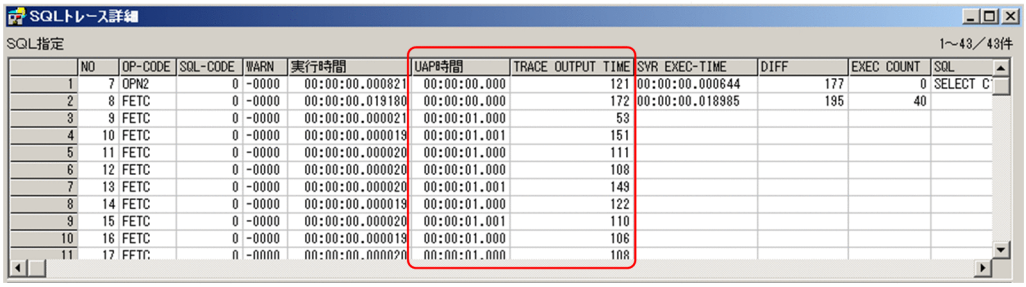
表 4-5 確認する情報名（EXEC-TIME が性能要件を満たしている場合）

目的	情報の分類	情報の内容	情報名	HiRDB SQL Tuning Advisor の画面名および項目名
アプリケーションの処理時間を確認する	SQL 単位の情報	SQL 実行要求受付時刻	START-TIME	SQL トレース詳細画面 UAP 時間
		SQL 実行要求終了時刻	END-TIME	
SQL トレースの出力処理に掛かった時間を確認する		SQL トレース出力処理時間	TRACE OUTPUT TIME	SQL トレース詳細画面 TRACE OUTPUT TIME

- アプリケーション側が処理している時間は，次の計算式で求められます。時間が掛かっている場合は，アプリケーション側の処理を調査および対策してください。
  - SQL トレースを直接確認する場合
    ((START-TIME) −（直前のオペレーションの END-TIME)) −（TRACE OUTPUT TIME)
  - HiRDB SQL Tuning Advisor を使って確認する場合
    (UAP 時間) −（TRACE OUTPUT TIME)
    注
    各項目の表示形式や時間の単位が異なるため，計算する際は形式および単位を合わせてください。表示形式の詳細は，マニュアル「HiRDB UAP 開発ガイド」の「SQL トレース機能」を参照してください。
- TRACE OUTPUT TIME の値が大きく，SQL トレースの出力処理に時間が掛かっている場合は，トレースを出力しない状態で，アプリケーションの性能を再度評価してください。

アプリケーション側で時間が掛かっている SQL トレースを，HiRDB SQL Tuning Advisor の SQL トレース詳細画面で表示した例を次に示します。

図 4-10　SQL トレース詳細画面「UAP 時間」および「TRACE OUTPUT TIME」の表示例



4.3.3　HiRDB サーバ側での SQL 実行時間を確認する

(1)　確認する情報

「SQL 実行時間」が性能要件を満たしていない場合は，「サーバ側での SQL 実行時間」を確認してください。確認する情報名を次に示します。

表 4-6　確認する情報名（HiRDB サーバ側での SQL 実行時間の確認）

目的	情報の分類	情報の内容	情報名	HiRDB SQL Tuning Advisor の画面名および項目名
HiRDB サーバ側での SQL 実行時間を確認する	SQL 単位の情報	サーバ側での SQL 実行時間	SVR EXEC-TIME	SQL トレース集計情報画面（SQL 単位） SVR EXEC-TIME
				SQL トレース詳細画面 SVR EXEC-TIME

(2)　確認する情報の表示例

サーバ側での SQL 実行時間を確認する情報を HiRDB SQL Tuning Advisor で表示する例を次に示します。

(a)　SQL トレース集計情報画面（SQL 単位）

SQL トレース集計情報画面（SQL 単位）の「SVR EXEC-TIME」には，各オペレーションの「SVR EXEC-TIME」の合計が表示されます。

「SVR EXEC-TIME」に時間が掛かっている SQL は，SQL トレース詳細画面で各オペレーションの「SVR EXEC-TIME」を確認してください。SQL トレース詳細画面は，SQL トレース集計情報画面（SQL 単位）で確認したい SQL 文の行をダブルクリックすると表示できます。表示方法の詳細は，HiRDB SQL Tuning Advisor のヘルプを参照してください。

サーバ側での SQL 実行に時間が掛かっている例を次に示します。

図 4-11　SQL トレース集計情報画面「SVR EXEC-TIME」の表示例

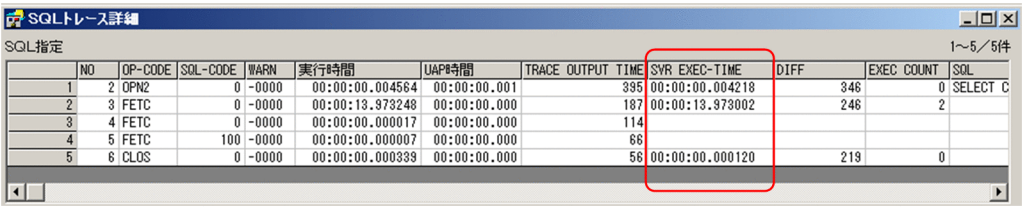
[説明]

この画面では，「合計実行時間」順にソートしています。

(b)　SQL トレース詳細画面

SQL トレース詳細画面では，各オペレーションの「SVR EXEC-TIME」が確認できます。SQL トレース詳細画面の表示例を次に示します。

図 4-12　SQL トレース詳細画面「SVR EXEC-TIME」の表示例

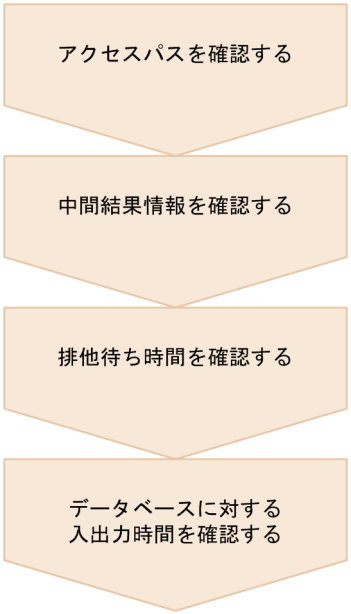


(3)　調査の進め方

SVR EXEC-TIME（サーバ側での SQL 実行時間）に問題がある場合は，次の手順で問題点を確認してください。

SVR EXEC-TIME に問題がない場合は，HiRDB クライアントと HiRDB サーバ間の通信時間を確認してください。確認方法は，「HiRDB クライアントと HiRDB サーバ間の通信時間を確認する」を参照してください。

図 4–13　調査の進め方



注

INSERT 文でアクセスパスが出力されていない SQL については，アクセスパスと中間結果情報の確認手順は不要です。

(a)　アクセスパスを確認する

アクセスパスに問題がないか確認し，対策してください。アクセスパスの確認方法は，「アクセスパスの確認方法の概要」を参照してください。

アクセスパスは，UAP 統計レポートに出力されています。アクセスパス情報の出力形式の詳細は，マニュアル「HiRDB UAP 開発ガイド」の「UAP 統計レポート機能」と「HiRDB コマンドリファレンス」の「アクセスパス表示ユティリティ（pdvwopt)」を参照してください。

アクセスパスの確認方法を次に説明します。

■ HiRDB SQL Tuning Advisor で確認する場合

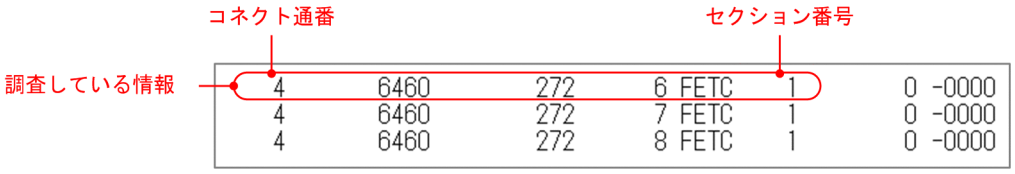
SQL トレース集計情報画面（SQL 単位）で，確認したい SQL 文を選択し，メニュー「表示」－「アクセスパス」を実行すると，アクセスパス情報が表示できます。表示方法の詳細は，HiRDB SQL Tuning Advisor のヘルプを参照してください。

■ UAP 統計レポートを直接確認する場合

次に示す手順で，確認するアクセスパス情報を特定してください。

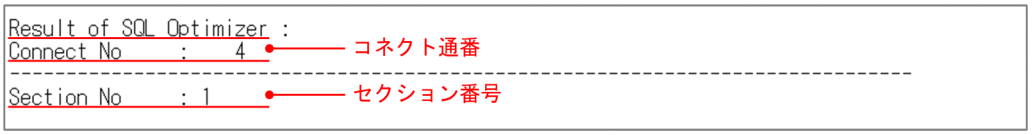
1. 調査している SQL 単位の情報からコネクト通番（CNCTNO）とセクション番号（SECNO）を確認してください。

図 4-14　SQL 単位の情報



2. アクセスパス情報を示すキーワード「Result of SQL Optimizer」で検索して，手順 1 で確認したコネクト通番とセクション番号の情報を探してください。

図 4-15　アクセスパス情報



(b)　中間結果情報を確認する

中間結果情報に問題がないか確認し，対策してください。中間結果情報の確認方法は，「中間結果情報の確認方法」を参照してください。

中間結果情報の確認方法を次に説明します。

■ HiRDB SQL Tuning Advisor で確認する場合

SQL トレース集計情報画面（SQL 単位）で，確認したい SQL 文を選択し，メニュー「表示」－「アクセスパス」を実行すると，アクセスパス情報とともに中間結果情報が表示できます。詳細は，HiRDB SQL Tuning Advisor のヘルプを参照してください。

■ UAP 統計レポートを直接確認する場合

中間結果情報の出力個所は，「中間結果情報の出力個所」を参照してください。

(c)　排他待ち時間を確認する

排他待ちが発生していないか確認してください。排他待ちが発生している場合は，排他オプションの指定方法に問題がないか確認し，対策してください。排他オプションの指定方法は，マニュアル「HiRDB UAP 開発ガイド」の「排他制御」を参照してください。

排他待ち時間は，UAP 単位またはトランザクション単位の情報で確認できます。確認する情報名を次に示します。

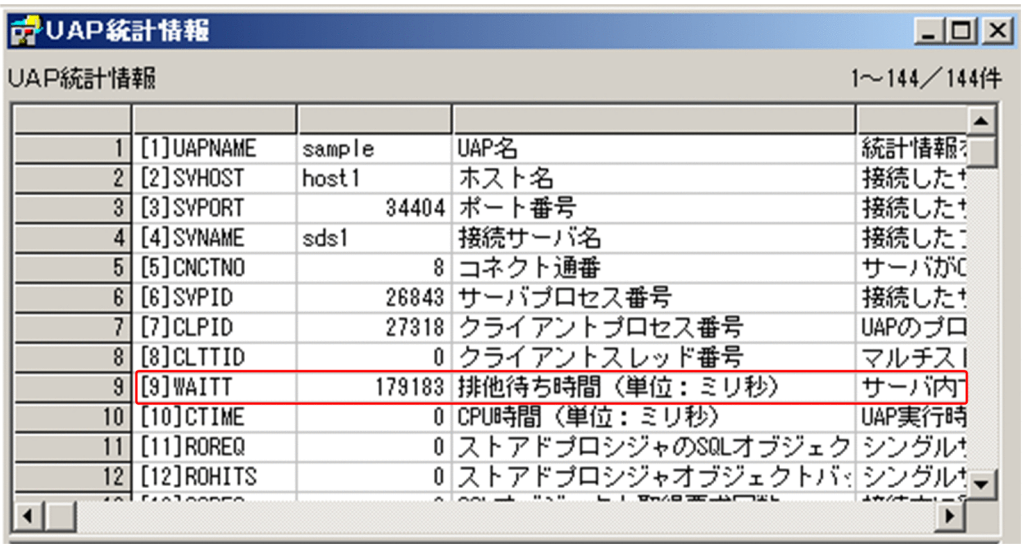
表 4-7　確認する情報名（排他待ち時間の確認）

目的	情報の分類	情報の内容	情報名	HiRDB SQL Tuning Advisor の画面名および項目名
HiRDB サーバ側での SQL 実行時間を確認する	UAP 単位またはトランザク	排他待ち時間	WAITT	UAP 統計情報画面 WAITT

目的	情報の分類	情報の内容	情報名	HiRDB SQL Tuning Advisor の画面名および項目名
	ション単位の情報			

排他待ちに時間が掛かっている UAP 統計レポートを，HiRDB SQL Tuning Advisor の UAP 統計情報画面で表示した例を次に示します。

図 4-16　UAP 統計情報画面「WAITT」の表示例



(d)　データベースに対する入出力時間を確認する

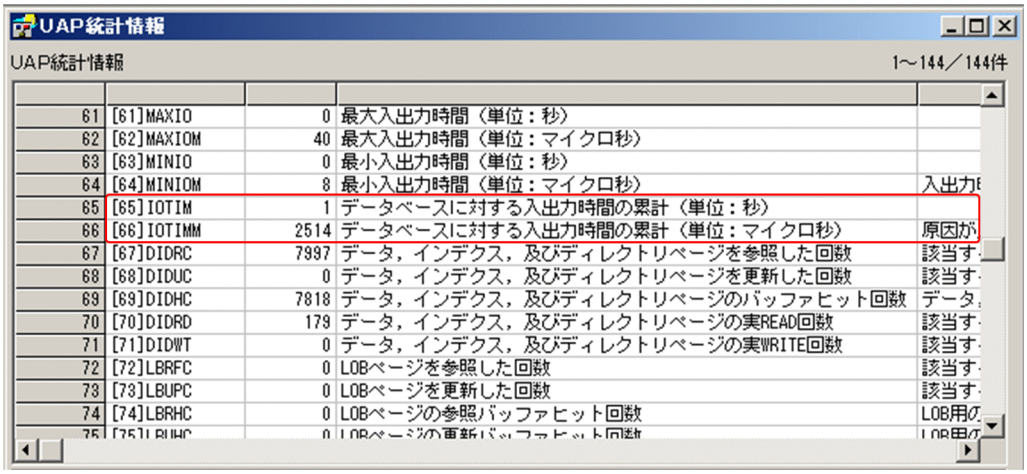
データベースに対する入出力時間に問題がないか確認し，対策してください。データベースに対する入出力時間に時間が掛かっている場合は，グローバルバッファの使用状況について，確認してください。グローバルバッファの確認方法は，マニュアル「HiRDB システム運用ガイド」の「グローバルバッファプールのチューニング」を参照してください。

表 4-8　確認する情報名（入出力時間の確認）

目的	情報の分類	情報の内容	情報名	HiRDB SQL Tuning Advisor の画面名および項目名
HiRDB サーバ側での SQL 実行時間を確認する	UAP 単位またはトランザクション単位の情報	データベースに対する入出力時間の累計	IOTIM	UAP 統計情報画面 IOTIM
			IOTIMM	UAP 統計情報画面 IOTIMM

入出力に時間が掛かっている UAP 統計レポートを，HiRDB SQL Tuning Advisor の UAP 統計情報画面で表示した例を次に示します。

図 4-17　UAP 統計情報画面「IOTIM」および「IOTIMM」の表示例



4.3.4　HiRDB クライアントと HiRDB サーバ間の通信時間を確認する

(1)　確認する情報

「HiRDB サーバ側での SQL 実行時間」に問題がない場合は，次のチューニングを実施してください。

(a)　ブロック転送機能のチューニング

ブロック転送機能は，HiRDB サーバから HiRDB クライアントへ検索結果を複数行数分まとめて転送する機能です。大量の検索結果を取得する処理で，オペレーションコード（OPCODE）が「FETC」の行が SQL 文の場合，ブロック転送機能で使用するバッファサイズをチューニングすることで，HiRDB クライアントと HiRDB サーバ間の通信回数を削減できます。ブロック転送機能の詳細は，マニュアル「HiRDB UAP 開発ガイド」の「ブロック転送機能」を参照してください。

ブロック転送機能で，一度に転送した行数は，次の情報で確認できます。転送行数を増やしたい場合は，ブロック転送機能に関するクライアント環境定義の設定を見直してください。

表 4-9　確認する情報名（ブロック転送機能のチューニング）

目的	情報の分類	情報の内容	情報名	HiRDB SQL Tuning Advisor の画面名および項目名
HiRDB クライアントと HiRDB サーバ間の通信時間を確認する	SQL 単位の情報	処理行数	EXEC COUNT	SQL トレース詳細画面 EXEC COUNT

[注意事項]

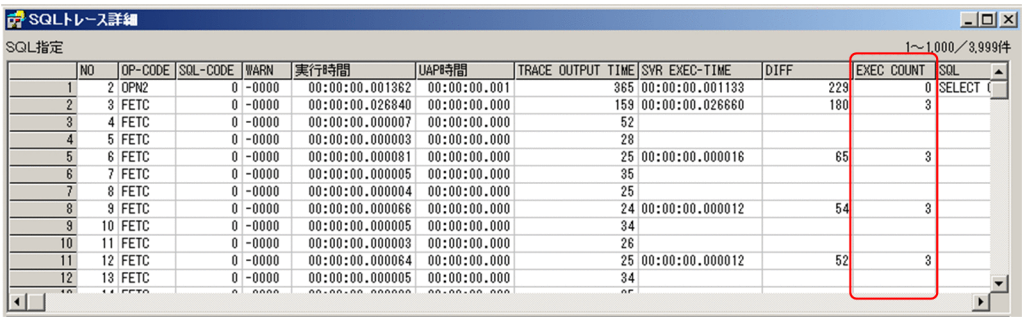
ブロック転送機能を指定しているにも関わらず，ブロック転送数機能が有効になっていない場合は，警告が発生していないか確認してください。警告が発生するとブロック転送機能は無効になります。警告がでないように SQL 文を見直ししてください。

(2)　確認する情報の表示例

転送行数を確認する情報を HiRDB SQL Tuning Advisor で表示する例を次に示します。

SQL トレース詳細画面では，各オペレーションの「EXEC COUNT」が確認できます。ブロック転送機能の転送行数は，「EXEC COUNT」を確認してください。転送行数が少ない場合の SQL トレース詳細画面の表示例を次に示します。

図 4-18　SQL トレース詳細画面「EXEC COUNT」の表示例



	NO	OP-CODE	SQL-CODE	WARN	実行時間	UAP時間	TRACE OUTPUT TIME	SVR EXEC-TIME	DIFF	EXEC COUNT	SQL
1	2	OPN2	0	-0000	00:00:00.001362	00:00:00.001	365	00:00:00.001133	229	0	SELECT (
2	3	FETC	0	-0000	00:00:00.026840	00:00:00.000	159	00:00:00.026660	180	3	
3	4	FETC	0	-0000	00:00:00.000007	00:00:00.000	52				
4	5	FETC	0	-0000	00:00:00.000003	00:00:00.000	28				
5	6	FETC	0	-0000	00:00:00.000081	00:00:00.000	25	00:00:00.000016	65	3	
6	7	FETC	0	-0000	00:00:00.000005	00:00:00.000	35				
7	8	FETC	0	-0000	00:00:00.000004	00:00:00.000	25				
8	9	FETC	0	-0000	00:00:00.000066	00:00:00.000	24	00:00:00.000012	54	3	
9	10	FETC	0	-0000	00:00:00.000005	00:00:00.000	34				
10	11	FETC	0	-0000	00:00:00.000003	00:00:00.000	26				
11	12	FETC	0	-0000	00:00:00.000064	00:00:00.000	25	00:00:00.000012	52	3	
12	13	FETC	0	-0000	00:00:00.000005	00:00:00.000	34				

4.4　中間結果情報の確認方法

中間結果情報とは，HiRDB が SQL 文を実行する中間段階で処理した行数の情報です。アクセスパスに問題がない場合でも，データの分布によっては，中間段階の処理行数が多くなり，実行時間が掛かってしまうこともあります。サーバ側での SQL 実行に時間が掛かっている場合は，中間結果情報を確認して中間段階の処理行数が多くないか確認してください。

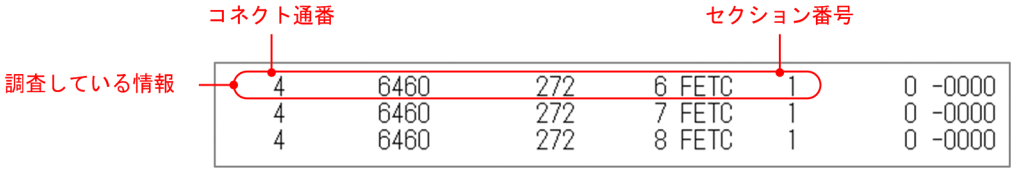
4.4.1　中間結果情報の確認方法の概要

(1)　中間結果情報の出力個所

中間結果情報は，UAP 統計レポートのファイルに出力されています。次に示す手順で，確認する中間結果情報を特定してください。中間結果情報の出力形式の詳細は，マニュアル「HiRDB UAP 開発ガイド」の「UAP 統計レポート機能」を参照してください。

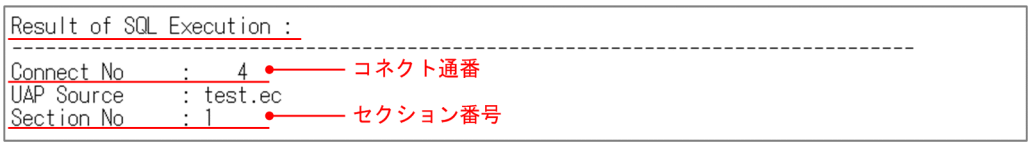
1. 調査している SQL 単位の情報からコネクト通番（CNCTNO）とセクション番号（SECNO）を確認してください。

図 4-19　SQL 単位の情報



2. 中間結果情報を示すキーワード「Result of SQL Execution」で検索して，手順 1 で確認したコネクト通番とセクション番号の情報を探してください。

図 4-20　中間結果情報



(2)　ここは必ず確認しよう

中間結果情報で，最初に確認する項目を次に示します。

表 4-10　中間結果情報の確認項目

分類	確認項目	確認内容
実表検索処理情報（SCAN）	• Row Count • Search	インデクスで絞り込めているか確認します。

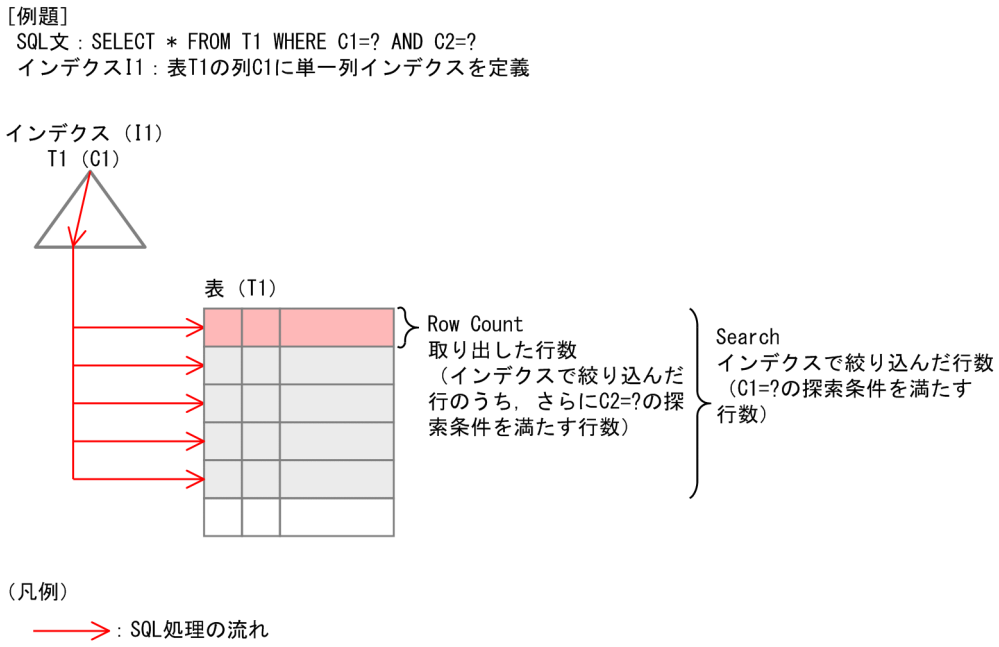
分類	確認項目	確認内容
結合処理情報（JOIN）	Join Type が NESTED LOOPS JOIN の場合 • Left • Right	NESTED LOOPS JOIN の結合方法で，外表が絞り込めているか確認します。

4.4.2　実表検索処理情報（SCAN）の Row Count と Search

(1)　インデクスで絞り込めているか確認する

インデクスを使用した検索では，インデクスのサーチ条件で取り出す行を絞り込みます（Search）。そして，絞り込んだ行に対して，インデクスで判定できない探索条件を評価して，行を取り出します（Row Count）。Search の値と Row Count の値の行数に開きがあると，実行時間が掛かってしまうことがあります。インデクスで絞り込んだ行数と取り出した行数について，次に示します。

図 4-21　インデクスで絞り込んだ行数と取り出した行数の関係



(2)　確認方法

中間結果情報の中で「SCAN」と表示されている個所が，実表検索処理情報です。この下の Row Count と Search の値に開きがないか確認してください。アクセスパスと実表検索処理情報の出力例を次に示します。

図 4-22　アクセスパスと実表検索処理情報の出力例

アクセスパス

```
----- QUERY ID : 1 -----
Query Type     : QUERY
SCAN
 # Table Name  : T1 0x00020079(131193)
   Cost        : N (10000000ROW)  {T-21353.88720010222,I-44.73467602192207}
   RDAREA      : NON DIVISION  (1RD) [0x04(4)] ALL
   Scan Type   : INDEX SCAN
   Index Name  : I1 0x000300f1(196849) (1) (+C1)
                      SearchCnd : AT [?(1)]
   RowCnd      : {T1.C2=?(2)}
```

実表検索処理情報

```
SCAN
 # Table Name  : T1 0x00020079(131193)
   Row Count   : 10 ROWS
   Index Name  : I1 0x000300f1(196849)
                 Search     : 70000 ROWS
```

取り出した行数　　インデクスで絞り込んだ行数

(3)　対策方法

サーチ条件で絞り込めていない理由としては，次のことが考えられます。

- インデクスを定義した列の値の種類が少ない
- インデクスを定義した列の値の分布に偏りがある

次のどちらかの対策方法を検討してください。

表 4-11　対策方法

対策方法	説明
インデクス構成列の追加	アクセスパスで「RowCnd」または「IfThenCnd」に表示されている列をインデクスに追加してください。この時，インデクスの構成列の順序は，最も絞り込める列を第 1 構成列にしてください。
使用するインデクスの変更	アクセスパスで「RowCnd」または「IfThenCnd」に表示されている列に別のインデクスが定義されている場合は，使用するインデクスを変更してください。使用するインデクスは SQL 文の使用インデクスの SQL 最適化指定（WITH INDEX）で指定します。指定方法は，マニュアル「HiRDB SQL リファレンス」の「使用インデクスの SQL 最適化指定」を参照してください。

インデクス構成列を追加して対策した結果の例を次に示します。

図 4-23　対策結果例

［対策前］
アクセスパス

```
----- QUERY ID : 1 -----
Query Type      : QUERY
SCAN
 # Table Name   : T1 0x00020079(131193)
   Cost         : N (10000000ROW)  {T-21353.88720010222,I-44.73467602192207}
   RDAREA       : NON DIVISION  (1RD) [0x04(4)] ALL
   Scan Type    : INDEX SCAN
   Index Name   : I1 0x000300f1(196849) (1) (+C1)
                      SearchCnd : AT [?(1)]
   RowCnd       : {T1.C2=?(2)}          確認個所
```

［対策後］
アクセスパス

```
----- QUERY ID : 1 -----
Query Type      : QUERY
SCAN
 # Table Name   : T1 0x00020079(131193)
   Cost         : N (10000000ROW)  {T-21353.88720010222,I-0.07039865460995917}
   RDAREA       : NON DIVISION  (1RD) [0x04(4)] ALL
   Scan Type    : MULTI COLUMNS INDEX SCAN
   Index Name   : I1 0x000300f1(196849) (2) (+C2,+C1)   最も絞り込めるC2列を
                                                        第1構成列とする
                      SearchCnd : AT [(?(2),?(1))]      複数列インデクスに変更
```

実表検索処理情報

```
SCAN
 # Table Name   : T1 0x00020079(131193)
   Row Count    : 10 ROWS
   Index Name   : I1 0x000300f1(196849)
                      Search    : 10 ROWS
```

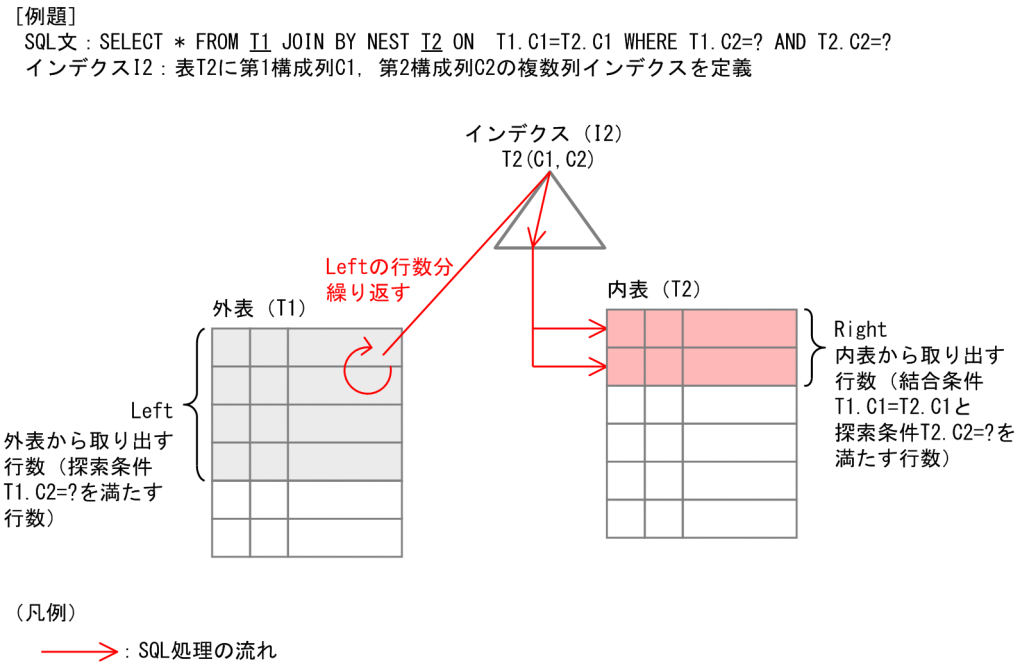
取り出した行数　　インデクスで絞り込んだ行数

4.4.3　結合処理情報（JOIN）の Left と Right

(1)　外表が絞り込めているか確認する

結合方法が NESTED LOOPS JOIN の場合，先に検索する表（外表）から取り出した行の数だけ，もう一方の表（内表）を検索します。このため，取り出す行数の少ない方を外表にした方が，内表を検索する回数が少なくなります。結合処理情報では，外表の方が取り出した行数が少ないか確認します。外表から取り出した行数は Left の値，内表から取り出した行数は Right の値になります。外表と内表から取り出す行数について，次に示します。

図 4-24　外表と内表から取り出す行数



(2)　確認方法

中間結果情報の中で「JOIN」と表示されている個所が，結合処理情報です。この下の Left の値と Right の値を比べて，外表から取り出す件数が多くないか確認してください。アクセスパスと結合処理情報の出力例を次に示します。

図 4-25　結合処理情報の出力例

アクセスパス

```
----- QUERY ID : 1 -----
Query Type     : QUERY
JOIN
 # Join  ID    : 1
   L Table     :  T1 0x00020079(131193)●── T1が外表
   R Table     :  T2 0x0002007a(131194)
   Join Type   : 1-CLM NESTED LOOPS JOIN(INNER)  [AS SPECIFIED]
SCAN
 # Table Name  : T1 0x00020079(131193)
   Cost        : N (10000000ROW)  {T-21154.2891141213,I-45.05459603711365}
   RDAREA      : NON DIVISION  (1RD) [0x04(4)] ALL
   Scan Type   : INDEX SCAN
   Index Name  : I1 0x000300f1(196849) (1) (+C2)
                     SearchCnd : AT [?(1)]
 # Table Name  : T2 0x0002007a(131194)
   Cost        : N (10000000ROW)  {T-21154.2891141213,I-45.05459603711365}
   RDAREA      : NON DIVISION  (1RD) [0x05(5)] ALL
   Scan Type   : MULTI COLUMNS INDEX SCAN
   Index Name  : I2 0x000300f5(196853) (2) (+C2,+C1)
                     SearchCnd : AT [(?(2),T1.C1)]
```

結合処理情報

```
JOIN
 # Join ID     : 1
   Row Count   : 3 ROWS
   Left        : 10000 ROWS ●── 外表から取り出した行数
   Right       : 3 ROWS●──── 内表から取り出した行数
   Join Type   : NESTED LOOPS JOIN(INNER)
```

(3)　対策方法

対策方法を次に説明します。一方の方法だけで対策できない場合は，両方の対策を合わせて実施してください。

1. 外表に，さらに絞り込める探索条件を追加してください。

2. 内表の方が探索条件で絞り込める場合は，外表と内表を入れ替えてください。SQL の結合表の指定（JOIN）を使って，外表と内表を明示的に指定することで対策できます。

［注意事項］

内表と外表を入れ替える場合は，内表に結合条件を評価できるインデクスを定義してください。詳細は，「［NESTED LOOPS JOIN にするには］」を参照してください。

内表と外表を入れ替えて対策した結果の例を次に示します。

図 4-26　対策結果例

［対策後のSQL］
```
SELECT * FROM T2 JOIN BY NEST T1 ON  T1.C1=T2.C1 WHERE T1.C2=? AND T2.C2=?
```

アクセスパス

```
----- QUERY ID : 1 -----
Query Type     : QUERY
JOIN
 # Join  ID    : 1
   L Table     :  T2 0x0002007a(131194)      ← T2が外表
   R Table     :  T1 0x00020079(131193)
   Join Type   : 1-CLM NESTED LOOPS JOIN(INNER)  [AS SPECIFIED]
SCAN
 # Table Name  : T2 0x0002007a(131194)
   Cost        : N (10000000ROW)  {T-21154.2891141213,I-45.05459603711365}
   RDAREA      : NON DIVISION  (1RD) [0x05(5)] ALL
   Scan Type   : MULTI COLUMNS INDEX SCAN
   Index Name  : I2 0x000300f5(196853) (2) (+C2,+C1)
                     SearchCnd : RANGE(CS-CE) [(?(2),MIN),(?(2),MAX)]
 # Table Name  : T1 0x00020079(131193)
   Cost        : N (10000000ROW)  {T-21154.2891141213,I-45.05459603711365}
   RDAREA      : NON DIVISION  (1RD) [0x04(4)] ALL
   Scan Type   : MULTI COLUMNS INDEX SCAN
   Index Name  : I1 0x000300f1(196849) (2) (+C2,+C1)   ← 内表T1のインデクスで
                     SearchCnd : AT [(?(1),T2.C1)]       結合条件が評価できるように
                                                         インデクスの構成列にC1列を追加
```

結合処理情報

```
JOIN
 # Join ID      : 1
   Row Count    : 3 ROWS
   Left         : 1 ROWS   ← 外表から取り出した行数
   Right        : 3 ROWS   ← 内表から取り出した行数
   Join Type    : NESTED LOOPS JOIN(INNER)
```

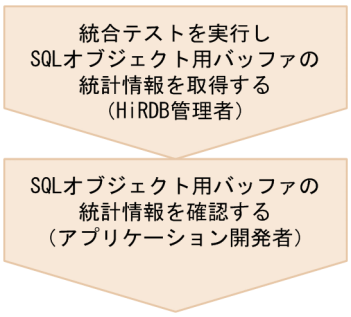
5 統合テストでの SQL 性能の確認

この章では，統合テストでの SQL 性能の確認方法について説明します。

5.1　SQL オブジェクト用バッファの統計情報を確認しよう

統合テストでは，SQL オブジェクト用バッファの統計情報を確認して，性能上問題のある SQL 文を特定します。そして，その SQL の問題点を個別に調査し，対策してください。確認の流れを次に示します。

図 5-1　確認の流れ



[説明]

1. 統合テストを実行し SQL オブジェクト用バッファの統計情報を取得する

   SQL オブジェクト用バッファの統計情報は，HiRDB サーバ側で取得する情報です。HiRDB 管理者に取得を依頼してください。SQL オブジェクト用バッファの統計情報は，pdobils コマンドで取得できます。pdobils コマンドの詳細は，マニュアル「HiRDB コマンドリファレンス」の「pdobils（SQL オブジェクト用バッファの統計情報表示)」を参照してください。

2. SQL オブジェクト用バッファの統計情報を確認する

   手順 1.で取得した統計情報をアプリケーション開発者が確認します。

[注意事項]

統合テストでは本番と同等の性能評価をするために，SQL トレースと UAP 統計レポートは取得しないでテストしてください。

5.2 SQL オブジェクト用バッファの統計情報の取得方法

SQL オブジェクト用バッファの統計情報を取得する手順について，次に説明します。この手順は，HiRDB 管理者が実行してください。

1. システム定義の設定を確認してください。次の条件に当てはまる場合は，システム共通定義（pdsys）の pd_sqlobject_stat_timing に tran を指定してください。
   - JDBC を使ったアプリケーションの場合
   - ROLLBACK したトランザクションの SQL 統計情報を取得する場合
2. 統合テストを開始する前に，pdobils コマンドを実行して，SQL オブジェクト用バッファの統計情報の値を初期化します。コマンド実行例を次に示します。

   UNIX の場合

   ```
   pdobils -C -H -R -e -NR > /dev/null
   ```

   Windows の場合

   ```
   pdobils -C -H -R -e -NR > nul
   ```

3. 統合テストを実行します。
4. 統合テストが終了したら，pdobils コマンドを実行して，SQL オブジェクト用バッファの統計情報を取得します。統合テストを繰り返し実行する場合は，出力ファイル名を変更して，実行してください。コマンド実行例を次に示します。

   ```
   pdobils -C -H -R -e -NR > /tmp/obils_n.csv
   ```

   n：実行した回数

5. 出力したファイル（/tmp/obils_*n*.csv）は，アプリケーション開発者へ提供してください。このファイルは，Excel など CSV 形式のファイルを扱えるツールを使用して参照してください。

［注意事項］

SQL オブジェクトバッファサイズは本番環境と同じサイズにしてください。SQL オブジェクトから追い出された SQL については，SQL 統計情報を出力しません。確認したい SQL 文が統計情報にない場合は，次のどちらかの対策をしてください。

- SQL オブジェクト用バッファサイズを拡張する
- 統合テスト中に定期的に pdobils コマンドを実行する

5.3　SQL オブジェクト用バッファの統計情報の確認方法

5.3.1　ここは必ず確認しよう

(1)　初期調査で確認する項目の一覧

SQL オブジェクト用バッファの統計情報を確認して，性能上問題のある SQL 文を特定します。初期調査で確認する項目について，次に示します。

表 5-1　初期調査で確認する情報

項目名	説明
UAP NAME	アプリケーションを特定する情報 （クライアント環境定義 PDCLTAPNAME の指定値）
SQL	SQL 文
EXECUTE COUNT	実行回数
EXECUTE TIME AVG	実行時間（平均）[μ 秒]
EXECUTE TIME MAX	実行時間（最大）[μ 秒]
DB REFERENCE GET COUNT	ページ参照回数
DB READ COUNT	データベースに対する実 READ 回数
WKFILE READ COUNT	作業表用ファイルの READ 回数
ACCESS TYPE	アクセス表とアクセス方法

(2)　調査の進め方

確認するポイントを次に説明します。対策が必要な SQL 文が複数ある場合は，EXECUTE COUNT の大きいものから優先的に対策してください。実行回数の多い SQL 文は全体性能への影響が大きいためです。

(a)　EXECUTE TIME MAX と DB REFERENCE GET COUNT の値が大きい場合

この場合は，次に示すどれかの問題があると考えられます。

- アクセスパスに問題がある

  HiRDB SQL Executer を使って，調査する SQL を個別に実行し，問題点を調査してください。調査方法は，「表にデータを格納したら」を参照してください。

- グローバルバッファのサイズが小さい

  グローバルバッファのサイズはシステム定義の pdbuffer オペランドに指定します。オペランドの詳細は，マニュアル「HiRDB システム定義」を参照してください。また，グローバルバッファのチューニ

ング方法については，マニュアル「HiRDB システム運用ガイド」の「グローバルバッファプールのチューニング」を参照してください。

(b) EXECUTE TIME MAX と WKFILE READ COUNT の値が大きい場合

この場合は，次に示すどれかの問題があると考えられます。

- アクセスパスに問題がある

  HiRDB SQL Executer を使って，調査する SQL を個別に実行し，問題点を調査してください。調査方法は，「表にデータを格納したら」を参照してください。

- 作業表用バッファのサイズが小さい

  作業表用バッファのサイズはシステム定義の pd_work_buff_size オペランドに指定します。オペランドの詳細は，マニュアル「HiRDB システム定義」を参照してください。

(c) ACCESS TYPE の値に「TABLE SCAN」と表示されている場合

この場合は，アクセスパスに問題があると考えられます。HiRDB SQL Executer を使って，調査する SQL を個別に実行し，問題点を調査してください。調査方法は，「表にデータを格納したら」を参照してください。

6 チューニング例

この章では，チューニング例について説明します。

6.1　チューニング例の一覧

この章では，具体的なデータを使用した SQL 文例のチューニング方法について，間違いやすいポイントとともに説明します。この章で説明するチューニング例の一覧を次の表に示します。

表 6-1　チューニング例の一覧

タイトル	概要	ポイント
効率の悪いアクセスパス（FULL SCAN）のチューニング例	インデクスのサーチ範囲を絞り込んだアクセスパスに変更します。	• インデクス構成列の順序を意識してください。 • 主キー以外のインデクスも検討してください。
効率の悪いアクセスパス（MERGE JOIN）のチューニング例	結合方法を NESTED LOOPS JOIN に変更します。	• NESTED LOOPS JOIN の結合方法は，内表の結合列にインデクスを定義してください。 • 外部キーを結合条件に指定する場合は外部キーを指定した列にインデクスを定義してください。
中間結果情報（SCAN）の件数が多い場合のチューニング例	データを絞り込めるように，使用するインデクスを変更します。	• フラグのような種類の少ない値を格納する列にはインデクスを定義しないでください。

6.2　効率の悪いアクセスパス（FULL SCAN）のチューニング例

6.2.1　効率の悪いアクセスパス（FULL SCAN）のチューニング例の説明

インデクスの全範囲をサーチする FULL SCAN のアクセスパスを，インデクスのサーチ範囲を絞り込んだアクセスパスに変更するチューニング例を説明します。

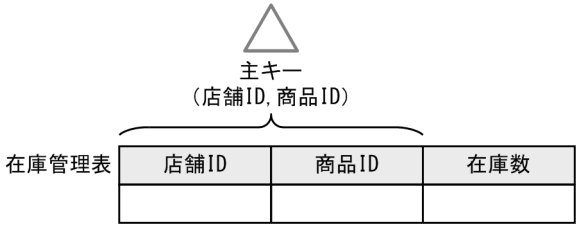
(1)　例題の概要

在庫管理表から，ある商品の店舗ごとの在庫数を検索します。

(a)　表およびインデクス定義

在庫管理表の構成と，この表に定義されたインデクスについて，次に示します。

図 6-1　表およびインデクス定義



(b)　SQL 文

実行する SQL 文を次に示します。

```
SELECT 店舗ID,在庫数 FROM 在庫管理表 WHERE 商品ID = ?
```

(c)　データ件数とヒット件数

データ件数と SQL 文のヒット件数について，次に示します。

表 6-2　データ件数とヒット件数

内容	件数
在庫管理表のデータ件数	約 1,000,000 件
SQL 文のヒット件数	約 100 件

(d)　アクセスパス

この SQL 文のアクセスパスを出力すると，アクセスパス情報のサーチ条件の行に「FULL SCAN」と表示されます。HiRDB SQL Tuning Advisor と UAP 統計レポートのアクセスパス出力結果を次に示します。

図 6-2　HiRDB SQL Tuning Advisor の出力結果（チューニング前）

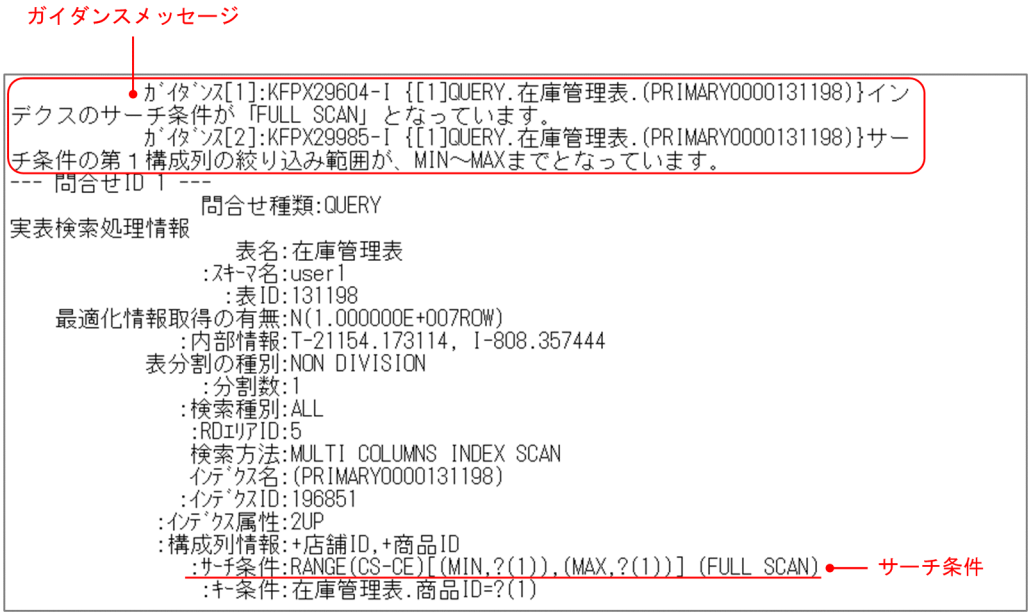
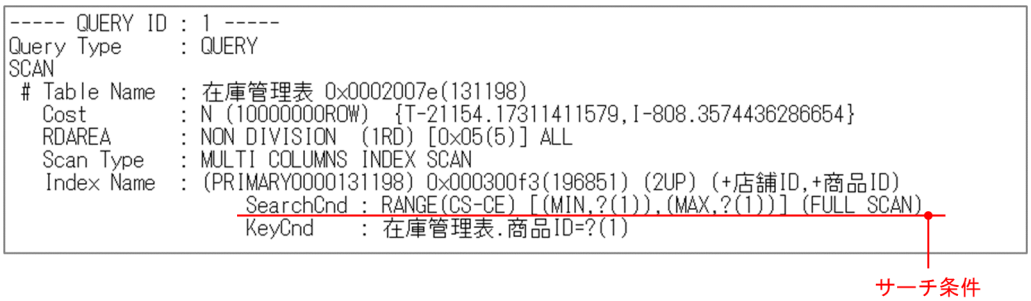


図 6-3　UAP 統計レポートの出力結果（チューニング前）



(2)　チューニングの考え方

FULL SCAN は効率の悪いアクセスパスのため，対策が必要です。FULL SCAN の対策方法は，「FULL SCAN の対策」で説明しています。この内容に沿って対策してください。

(a)　要因の特定

まず，FULL SCAN になる要因を特定してください。例題の SQL 文は，次に示す理由から「インデクスの第 1 構成列に対する探索条件が指定されていない」ことが要因であることがわかります。

- HiRDB SQL Tuning Advisor でアクセスパスを出力した場合
  次のガイダンスメッセージが出力されています。
  - KFPX29604-I
  - KFPX29985-I
- UAP 統計レポートでアクセスパスを出力した場合
  「SearchCnd」が「RANGE」で行の末尾に「(FULL SCAN)」と表示されています。

［主キーとインデクスについて］

在庫管理表には，主キーとして店舗 ID 列，商品 ID 列が指定されています。主キーを指定した列には，インデクスが定義されます。この例では複数の列を主キーに指定しているため，店舗 ID 列が第 1 構成列，商品 ID 列が第 2 構成列である複数列インデクスが定義されます。この例の SQL 文では，インデクスの第 1 構成列である店舗 ID 列に探索条件が指定されていません。このため，インデクスの全範囲をサーチする「FULL SCAN」のアクセスパスになっています。

(b) 対策方法

第 1 構成列に対する探索条件が指定されていない場合の対策方法について，どの対策方法が採用できるか検討してください。この例での検討結果を次に示します。

表 6-3 第 1 構成列に対する探索条件が指定されていない場合の対策方法

対策方法	説明	検討結果
SQL 文の変更	第 1 構成列に対する探索条件の指定が漏れている場合は，追加してください。	【不採用】 例題の SQL 文は，第 1 構成列である店舗 ID 列の探索条件が漏れている訳ではないため，SQL 文の変更はできません。
インデクス構成列順序の変更	=条件など最も絞り込める条件を指定している列を第 1 構成列にできないか，インデクスの構成列の順序を見直してください。	【不採用】 例題の SQL 文以外に，店舗 ID 列だけに探索条件を指定した SQL 文を実行するため，インデクス構成列の順序を変更することはできません。
新しいインデクスの追加	=条件など最も絞り込める条件を指定している列が第 1 構成列である新しいインデクスの追加を検討してください。	【採用】 在庫管理表には主キー以外にインデクスは定義されていないため，新しいインデクスを追加しても，インデクスの数は問題ありません。このため，商品 ID 列に単一列インデクスを定義することで対策できます。

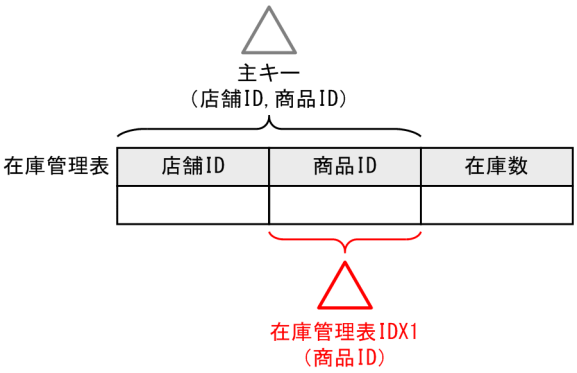
この例では，新しいインデクスを追加して対策します。

(3) チューニング結果

(a) 表およびインデクス定義

商品 ID を第 1 構成列とするインデクス「在庫管理表 IDX1」を追加してください。変更後のインデクス定義について，次に示します。

図 6-4　表およびインデクス定義



(b)　SQL 文

変更しません。

(c)　アクセスパス

アクセスパスを出力して，新しいインデクス「在庫管理表 IDX1」を使用した INDEX SCAN に変更されたことを確認してください。

図 6-5　HiRDB SQL Tuning Advisor の出力結果（チューニング後）

```
                検索方法:INDEX SCAN
                ｲﾝﾃﾞｸｽ名:在庫管理表IDX1 ●── 使用するインデクス名
              :ｲﾝﾃﾞｸｽID:196854
            :ｲﾝﾃﾞｸｽ属性:1
            :構成列情報:+商品ID
              :ｻｰﾁ条件:AT[?(1)] ●── サーチ条件
```

図 6-6　UAP 統計レポートの出力結果（チューニング後）

```
Scan Type    : INDEX SCAN
Index Name   : 在庫管理表IDX1 0x000300f6(196854) (1) (+商品ID)
               SearchCnd : AT [?(1)]
```

使用するインデクス名　　サーチ条件

(4)　まとめ

このチューニング例のポイントを次に示します。

[ポイント]

- SQL 文の探索条件の内容を考慮して，主キー以外のインデクスも検討してください。
- 複数列インデクスでは，構成列の順序を意識してください。詳細は，マニュアル「HiRDB システム導入・設計ガイド」の「インデクス構成列の検討」を参照してください。

6.3　効率の悪いアクセスパス（MERGE JOIN）のチューニング例

6.3.1　効率の悪いアクセスパス（MERGE JOIN）のチューニング例の説明

効率の悪い結合方法の MERGE JOIN を，NESTED LOOPS JOIN に変更するチューニング例を説明します。

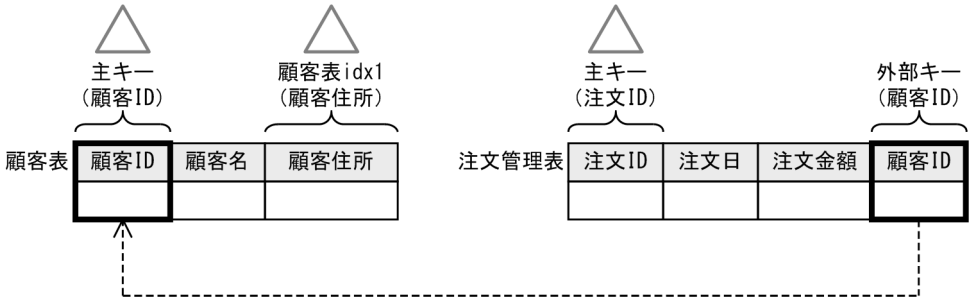
(1)　例題の概要

顧客表および注文管理表から，特定の市区町村に住む顧客が注文した情報を顧客名とともに検索します。

(a)　表およびインデクス定義

顧客表および注文管理表の構成と，それぞれの表に定義されたインデクスについて，次に示します。

図 6-7　表およびインデクス定義



(b)　SQL 文

実行する SQL 文を次に示します。LIKE 述語の後の?パラメタには，市区町村名を前方一致で検索するパターン文字列を指定します。

```
SELECT 顧客表.顧客名，注文管理表.注文日，注文管理表.注文金額
 FROM 顧客表,注文管理表
 WHERE 顧客表.顧客ID = 注文管理表.顧客ID
 AND 顧客表.顧客住所 LIKE ?
```

(c)　データ件数とヒット件数

データ件数と SQL 文のヒット件数について，次に示します。

表 6-4　データ件数とヒット件数

内容	件数
顧客表のデータ件数	約 1,000,000 件
顧客表から取り出す件数	約 5 件

内容	件数
注文管理表のデータ件数	約 12,000,000 件
SQL 文のヒット件数	約 25 件

(d)　アクセスパス

この SQL 文のアクセスパスを出力すると，アクセスパス情報の結合方法の行に「MERGE JOIN」と表示されます。HiRDB SQL Tuning Advisor と UAP 統計レポートのアクセスパス出力結果を次に示します。

図 6-8　HiRDB SQL Tuning Advisor の出力結果（チューニング前）

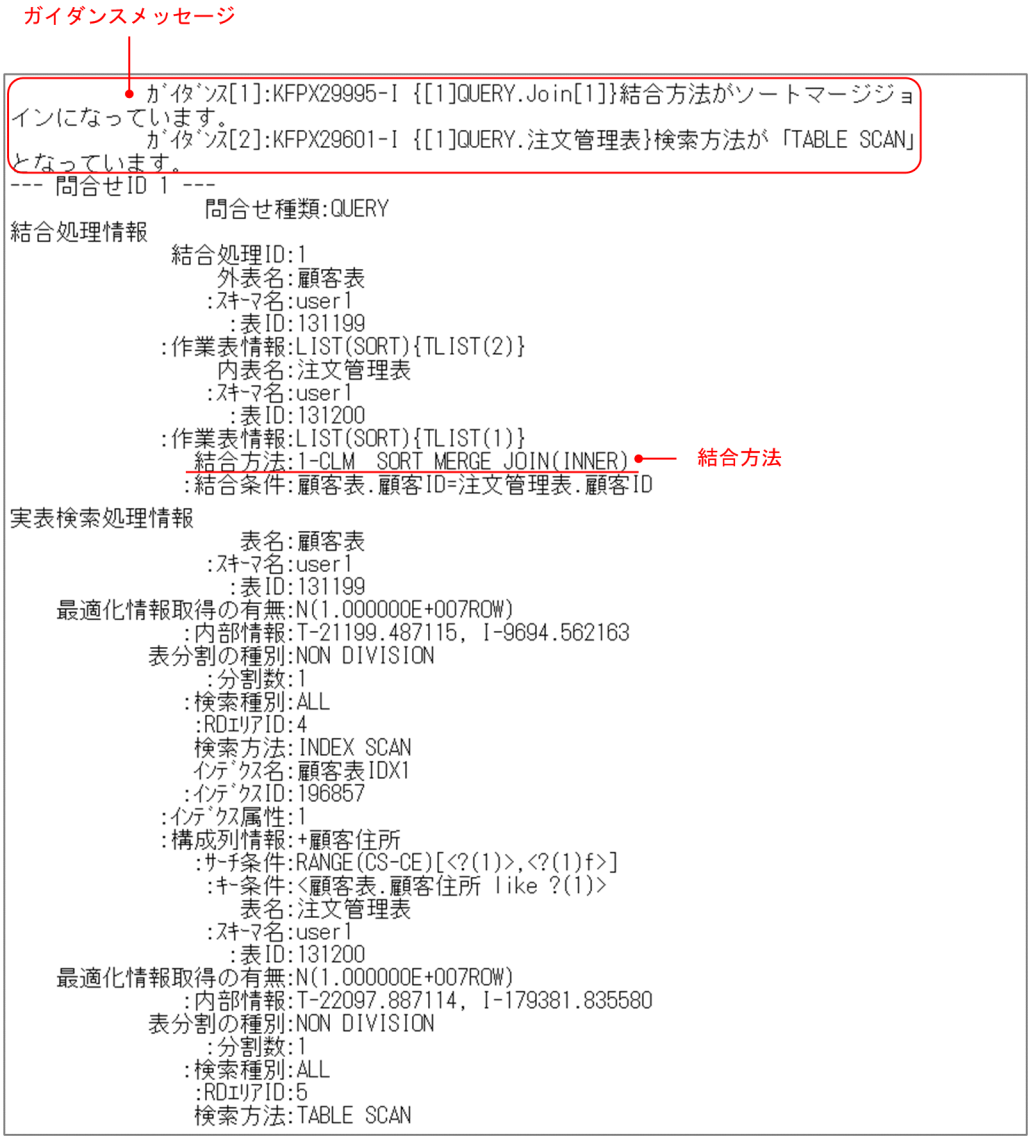


図 6-9　UAP 統計レポートの出力結果（チューニング前）

```
----- QUERY ID : 1 -----
Query Type      : QUERY
JOIN
 # Join  ID     : 1
   L Table      :  顧客表 0x0002007f(131199)  LIST(SORT){TLIST(2)}
   R Table      :  注文管理表 0x00020080(131200)  LIST(SORT){TLIST(1)}
   JoinCnd      : {顧客表.顧客ID=注文管理表.顧客ID}
   Join Type    : 1-CLM SORT MERGE JOIN(INNER)
SCAN
 # Table Name   : 顧客表 0x0002007f(131199)
   Cost         : N (10000000ROW)  {T-21199.4871147817,I-9694.562163099263}
   RDAREA       : NON DIVISION  (1RD) [0x04(4)] ALL
   Scan Type    : INDEX SCAN
   Index Name   : 顧客表IDX1 0x000300f9(196857) (1) (+顧客住所)
                      SearchCnd : RANGE(CS-CE) [<?(1)>,<?(1)f>]
                      KeyCnd    : <顧客表.顧客住所 like ?(1)>
 # Table Name   : 注文管理表 0x00020080(131200)
   Cost         : N (10000000ROW)  {T-22097.88711410221,I-179381.8355795134}
   RDAREA       : NON DIVISION  (1RD) [0x05(5)] ALL
   Scan Type    : TABLE SCAN
```

結合方法

(2)　チューニングの考え方

MERGE JOIN は効率の悪いアクセスパスのため，対策が必要です。MERGE JOIN の対策方法は，「MERGE JOIN の対策」で説明しています。この内容に沿って対策してください。

(a)　結合方法の検討

「MERGE JOIN の対策」の「対策方法」を参照して，どの結合方法に変更するか検討してください。この例の SQL 文はヒット件数が少ないため，NESTED LOOPS JOIN に変更します。

(b)　対策方法

NESTED LOOPS JOIN にする手順は，「MERGE JOIN の対策」の「[NESTED LOOPS JOIN にするには]」を参照してください。

この例では，次の対策を実施します。

1. 注文管理表の顧客 ID 列にインデクスを追加する
2. SQL 文に結合方法を明示的に指定する

[外部キーとインデクスについて]

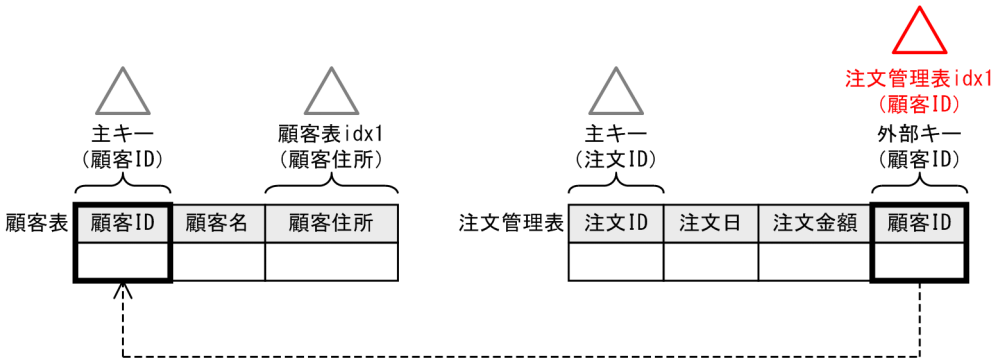
注文管理表の顧客 ID 列には，顧客表の顧客 ID 列を参照する外部キーが定義されています。外部キーを定義しても，インデクスは定義されません。外部キーを結合条件に使用する場合は，定義系 SQL の CREATE INDEX でインデクスを定義してください。

(3)　チューニング結果

(a)　表およびインデクス定義

注文管理表の顧客 ID 列にインデクス「注文管理表 IDX1」を追加してください。変更後のインデクス定義について，次に示します。

図 6-10　表およびインデクス定義



(b)　SQL 文

結合方法を明示的に指定する SQL 文に変更してください。下線部分が変更した個所です。

```
SELECT 顧客表.顧客名, 注文管理表.注文日, 注文管理表.注文金額
 FROM 顧客表 INNER JOIN BY NEST 注文管理表
 ON 顧客表.顧客ID = 注文管理表.顧客ID
 WHERE 顧客表.顧客住所 LIKE ?
```

(c)　アクセスパス

アクセスパスを出力して，注文管理表が内表である NESTED LOOPS JOIN に変更されたことを確認してください。

図 6-11　HiRDB SQL Tuning Advisor の出力結果（チューニング後）

```
--- 問合せID 1 ---
                問合せ種類:QUERY
結合処理情報
              結合処理ID:1
                  外表名:顧客表
                 :ｽｷｰﾏ名:user1
                   :表ID:131199
                  内表名:注文管理表
                 :ｽｷｰﾏ名:user1
                   :表ID:131200
                結合方法:1-CLM  NESTED LOOPS JOIN(INNER)  ←― 確認個所
 SQL最適化指定の有効無効:AS SPECIFIED
実表検索処理情報
                    表名:顧客表
                 :ｽｷｰﾏ名:user1
                   :表ID:131199
    最適化情報取得の有無:N(1.000000E+007ROW)
               :内部情報:T-21211.087115, I-9704.562163
          表分割の種別:NON DIVISION
                 :分割数:1
              :検索種別:ALL
               :RDｴﾘｱID:4
               検索方法:INDEX SCAN
              ｲﾝﾃﾞｸｽ名:顧客表IDX1
             :ｲﾝﾃﾞｸｽID:196857
            :ｲﾝﾃﾞｸｽ属性:1
            :構成列情報:+顧客住所
               :ｻｰﾁ条件:RANGE(CS-CE)[<?(1)>,<?(1)f>]
                :ｷｰ条件:<顧客表.顧客住所 like ?(1)>
                    表名:注文管理表
                 :ｽｷｰﾏ名:user1
                   :表ID:131200
    最適化情報取得の有無:N(1.000000E+007ROW)
               :内部情報:T-22329.887114, I-179581.835580
          表分割の種別:NON DIVISION
                 :分割数:1
              :検索種別:ALL
               :RDｴﾘｱID:5
               検索方法:INDEX SCAN
              ｲﾝﾃﾞｸｽ名:注文管理表IDX1
             :ｲﾝﾃﾞｸｽID:196860
            :ｲﾝﾃﾞｸｽ属性:1
            :構成列情報:+顧客ID
               :ｻｰﾁ条件:AT[顧客表.顧客ID]
```

図 6-12　UAP 統計レポートの出力結果（チューニング後）

```
----- QUERY ID : 1 -----
Query Type     : QUERY
JOIN
 # Join  ID    : 1
   L Table     :  顧客表 0x0002007f(131199)
   R Table     :  注文管理表 0x00020080(131200)
   Join Type   : 1-CLM NESTED LOOPS JOIN(INNER)  [AS SPECIFIED]  ←― 確認個所
SCAN
 # Table Name  : 顧客表 0x0002007f(131199)
   Cost        : N (10000000ROW)  {T-21211.08711495456,I-9704.562163248274}
   RDAREA      : NON DIVISION  (1RD) [0x04(4)] ALL
   Scan Type   : INDEX SCAN
   Index Name  : 顧客表IDX1 0x000300f9(196857) (1) (+顧客住所)
                     SearchCnd : RANGE(CS-CE) [<?(1)>,<?(1)f>]
                     KeyCnd    : <顧客表.顧客住所 like ?(1)>
 # Table Name  : 注文管理表 0x00020080(131200)
   Cost        : N (10000000ROW)  {T-22329.88711410221,I-179581.8355795134}
   RDAREA      : NON DIVISION  (1RD) [0x05(5)] ALL
   Scan Type   : INDEX SCAN
   Index Name  : 注文管理表IDX1 0x000300fc(196860) (1) (+顧客ID)
                     SearchCnd : AT [顧客表.顧客ID]
```

(4)　まとめ

このチューニング例のポイントを次に示します。

［ポイント］

外部キーを指定した列を結合条件に使用する場合は，インデクスを追加してください。

6.4　中間結果情報（SCAN）の件数が多い場合のチューニング例

6.4.1　中間結果情報（SCAN）の件数が多い場合のチューニング例の説明

データを絞り込めるように，使用するインデクスを変更するチューニング例を説明します。

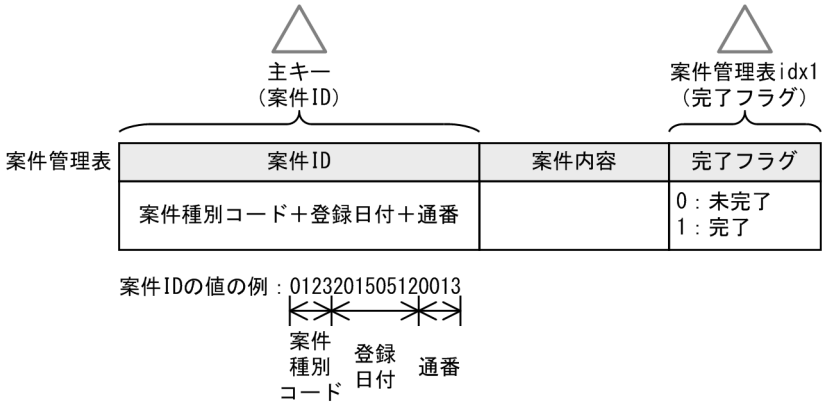
(1)　例題の概要

案件管理表から，ある月に登録された特定の種別の案件のうち，完了している案件を検索します。

(a)　表およびインデクス定義

案件管理表の構成と，この表に定義されたインデクス，各列の値について，次に示します。

図 6-13　表およびインデクス定義と各列の値



[各列の値の説明]

- 完了フラグ列は，0 と 1 の 2 種類の値しか格納されません。
- 案件 ID 列は，案件種別コード，案件日付，通番の値を組み合わせた値です。

(b)　SQL 文

実行する SQL 文を次に示します。BETWEEN 述語の後の?パラメタには，検索したい月の案件 ID の最小値と最大値を指定します。例えば，案件種別コード 0123 で 2015 年 5 月に登録された案件を検索する場合は，'0123201505010001'と'0123201505319999'になります。

```
SELECT * FROM 案件管理表
 WHERE 案件ID BETWEEN ? AND ?
 AND 完了フラグ = 1
```

(c)　データ件数とヒット件数

データ件数と SQL 文のヒット件数について，次に示します。

表 6-5　データ件数とヒット件数

内容	件数
案件管理表のデータ件数	約 1,000,000 件
完了フラグの値が 1 である件数	約 700,000 件
SQL 文のヒット件数	約 50 件

(d)　アクセスパスと中間結果情報

この SQL 文のアクセスパスを出力すると，アクセスパス情報の検索方法は「INDEX SCAN」となり，問題ありません。しかし，中間結果情報の実表検索処理情報を確認すると，インデクスで絞り込んだ行数（Search）と取り出した行数（Row Count）に開きがあります。UAP 統計レポートのアクセスパス情報と実表検索処理情報の出力結果を次に示します。

図 6-14　UAP 統計レポートの出力結果（チューニング前）

アクセスパス

```
----- QUERY ID : 1 -----
Query Type     : QUERY
SCAN
 # Table Name  : 案件管理表 0x00020082(131202)
   Cost        : N (10000000ROW)  {T-21353.88900610229,I-44.73635602198853}
   RDAREA      : NON DIVISION  (1RD) [0x05(5)] ALL
   Scan Type   : INDEX SCAN
   Index Name  : 案件管理表IDX1 0x000300fe(196862) (1) (+完了フラグ)
                     SearchCnd : AT [1]
   RowCnd      : {案件管理表.案件ID between ?(1) and ?(2)}
```

実表検索処理情報

```
SCAN
 # Table Name  : 案件管理表 0x00020082(131202)
   Row Count   : 50 ROWS
   Index Name  : 案件管理表IDX1 0x000300fe(196862)
                     Search    : 700000 ROWS
```

取り出した行数　　インデクスで絞り込んだ行数

(2)　チューニングの考え方

検索に使用しているインデクス（案件管理表 IDX1）では，取り出すデータを絞り込めていないため，対策が必要です。インデクスで絞り込めていない場合の対策方法は，「実表検索処理情報（SCAN）の Row Count と Search」で説明しています。この内容に沿って対策してください。

この例では，アクセスパスで「RowCnd」に表示されている案件 ID 列に主キーが定義されています。したがって，使用するインデクスを変更します。SQL 文に SQL 最適化指定（WITH INDEX）で主キーを指定します。

［インデクスを定義する列の値の種類について］

フラグのような種類の少ない値を格納する列に定義したインデクスは，データを絞り込むことができません。このような列にはインデクスを定義しないで，別の絞り込める列に定義したインデクスを使用するようにしてください。

(3)　チューニング結果

(a)　表およびインデクス定義

使用するインデクスを変更するチューニングのため，表およびインデクス定義の変更はありません。ただし，完了フラグ列に定義したインデクスは，絞り込めないため，削除してください。

(b)　SQL 文

SQL 最適化指定（WITH INDEX）で使用するインデクスに主キーを指定する SQL 文に変更してください。下線部分が変更した個所です。

```
SELECT * FROM 案件管理表 WITH INDEX (PRIMARY KEY)
 WHERE 案件ID BETWEEN ? AND ?
 AND 完了フラグ = 1;
```

(c)　アクセスパスと中間結果情報

アクセスパスを出力して，使用するインデクスが変更になり，インデクスで絞り込めていることを確認してください。

図 6-15　UAP 統計レポートの出力結果（チューニング後）

アクセスパス

```
----- QUERY ID : 1 -----
Query Type      : QUERY
SCAN
 # Table Name   : 案件管理表 0x00020082(131202)
   Cost         : N (10000000ROW)  {T-21353.88900610229,I-982.9206748303258}
   RDAREA       : NON DIVISION  (1RD) [0x05(5)] ALL
   Scan Type    : INDEX SCAN  [AS SPECIFIED]
   Index Name   : (PRIMARY0000131202) 0x000300fd(196861) (1UP) (+案件ID)
                      SearchCnd : RANGE(CS-CE) [?(1),?(2)]
   RowCnd       : {案件管理表.完了フラグ=1}
```

実表検索処理情報

```
SCAN
 # Table Name   : 案件管理表 0x00020082(131202)
   Row Count    : 50 ROWS
   Index Name   : (PRIMARY0000131202) 0x000300fd(196861)
                      Search     : 50 ROWS
```

取り出した行数　　インデクスで絞り込んだ行数

(4)　まとめ

このチューニング例のポイントを次に示します。

［ポイント］

フラグのような種類の少ない値を格納する列にはインデクスを定義しないでください。詳細は，マニュアル「HiRDB システム導入・設計ガイド」の「インデクス作成に適さない列」を参照してください。

7 データベース設計編の概要

この章では，データベース設計編の概要について説明します。

7.1　データベース設計編の読み方

この編では，データベース設計工程での，性能を向上させるためのポイントについて説明します。表，インデクスなど，設計する対象ごとに説明しています。また，章ごとに設計が必須であるポイントは「ここは必ず確認しよう」で説明しています。

- 表の設計
- インデクスの設計
- RD エリアの設計
- グローバルバッファの設計

なお，データベースの設計は，性能面と同時に運用面についても考慮が必要です。運用面でのポイントについては，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。

7.2 表の特性

データベース設計では，表の特性に応じて設計することが重要です。この編では，次の特性を持つ表について，個別の設計のポイントを説明しています。

表 7-1 表の特性

特性	説明	特長
トランザクション表	在庫管理表やワークフロー管理表などのオンライントランザクションで更新される表	• SELECT 文，INSERT 文，UPDATE 文，DELETE 文が実行される • データ量が多い
マスタ表	社員情報や商品情報などのマスタ情報を管理する表	• SELECT 文の実行が主な操作 • データ量が少ない • すべてのデータが偏りなくアクセスされる
採番表	採番データを管理する表	• SELECT 文と UPDATE 文の実行が主な操作 • データ量が少ない • すべてのデータが偏りなくアクセスされる
履歴蓄積表	ログ管理表や注文履歴表などのオンライントランザクションで INSERT 文が実行される表	• INSERT 文の実行が主な操作 • データ量が多い • データを挿入するページにアクセスが集中する

8 表の設計

この章では，性能を向上させるための表設計のポイントについて説明します。

8.1　ここは必ず設計しよう

ここでは，性能を向上させるために，設計が必須であるポイントについて説明します。

8.1.1　表の正規化

表の正規化は，よりよいパフォーマンスにするための重要なポイントです。必ず設計してください。

表を正規化すると，データを取り出す際に複数の表を結合することがあります。結合する表の数が増えると，SQL の処理時間も長くなります。このため，性能を優先するデータの場合は，あえて正規化を崩して，結合する表の数を少なくすることも考慮してください。

設計方法の詳細は，マニュアル「HiRDB システム導入・設計ガイド」の「表の正規化」を参照してください。

8.2　さらに性能を向上させるポイント

ここでは，性能をさらに向上させるための設計ポイントについて説明します。

8.2.1　レスポンスを向上させるポイント

処理のレスポンスを向上させるポイントについて説明します。

(1)　トランザクション表

トランザクション表に対する処理のレスポンスを向上させるには，次のポイントがあります。

(a)　断片化の抑止

トランザクション表は，データ更新の頻度が高いですが，データを更新すると，データやインデクスのページには断片化した空き領域が発生し，性能が劣化することがあります。性能を安定させるために，定期的にデータベースを再編成する運用を設計してください。

データベース再編成を実行する間隔が長い場合は，断片化の発生を抑えるデータベース設計も考慮してください。UPDATE 文の実行頻度が高い場合に，ページの断片化の発生を抑える設計方法について，次に説明します。

- ページ内の未使用領域の比率（PCTFREE）の拡大
- 固定長データ型の適用

それぞれの方法の詳細について，次に説明します。

- ページ内の未使用領域の比率（PCTFREE）の拡大

  可変長データ型の列に対する UPDATE 文の実行で行長が長くなるとき，行を格納するページに，更新後の行長が格納できる連続した空き領域がある場合は，行を格納するページは変わらず，断片化も発生しません。行長が変わる UPDATE 文の実行頻度が高い場合は，連続した空き領域を通常よりも大きく確保できるよう，ページ内の未使用領域の比率の値を大きくしてください。ただし，ページ内の未使用領域の比率を大きくすると，格納効率に影響があるので，トレードオフを考慮して設計してください。ページ内の未使用領域の比率の詳細および値の求め方は，マニュアル「HiRDB システム導入・設計ガイド」の「ページ内の未使用領域の比率の設定」を参照してください。
- 固定長データ型の適用

  UPDATE 文の実行前後で行長が変わらない場合は，行の格納ページは変わらず，断片化も発生しません。よって，UPDATE 文を頻繁に実行する列で，業務上固定長でも問題ない場合は，固定長のデータ型にしてください。

(b)　表の横分割の適用

トランザクション表は，横分割するとレスポンスを向上できることがあります。表の横分割の詳細は，「表の横分割」を参照してください。

(2)　その他のポイント

(a)　列のデータ型

列のデータ型を検討するときに，考慮する点を次に示します。

- DECIMAL 型に関する注意事項

  DECIMAL 型の列に対して四則演算などの演算をする場合，10 進パック形式のため，ほかの数値データ型の列の演算に比べて，処理の効率が悪くなります。格納するデータが整数の場合は，列のデータ型を INTEGER 型または SMALLINT 型にしてください。

- MCHAR および MVARCHAR に関する注意事項

  MCHAR 型および MVARCHAR 型（混在文字データ）の列の場合は，条件判定する際に半角文字と全角文字を区別するオーバーヘッドがあります。このため，半角文字と全角文字が混在する場合だけ使用してください。列のデータ型は，格納するデータが半角文字だけの場合は CHAR 型および VARCHAR 型，全角文字だけの場合は NCHAR 型および NVARCHAR 型にしてください。

- BLOB 型と BINARY 型の使い分け

  バイナリデータを格納できる BLOB 型と BINARY 型の使い分け方については，マニュアル「HiRDB システム導入・設計ガイド」の「BLOB 型と BINARY 型の使い分け」を参照してください。

(b)　空き領域の再利用機能と性能のトレードオフ

空き領域の再利用機能とは，表の使用中セグメントがユーザの指定したセグメント数に達し，そのセグメントが満杯になるとページサーチモードを空きページ再利用モードに切り替えて，使用中ページの空き領域を使用する機能です。機能の詳細は，マニュアル「HiRDB システム導入・設計ガイド」の「空き領域の再利用機能」を参照してください。

空き領域の再利用機能を適用すると，格納効率は良くなりますが，データの追加および更新時にデータの格納場所を使用中のページからサーチするため，オーバーヘッドがあります。空き領域再利用機能は，データの追加と削除を繰り返す表の場合だけ適用を検討してください。

また，データの追加および更新の性能を優先する表の場合は，空き領域再利用機能は適用しないで，空き領域をデータベース再編成や空きページ解放などで解放して，格納効率を上げることを推奨します。

8.2.2　スループットを向上させるポイント

スループットを向上させるポイントを，表の特性別に説明します。

(1)　トランザクション表の場合

トランザクション表は，横分割するとスループットを向上できることがあります。表の横分割の詳細は，「表の横分割」を参照してください。

(2)　履歴蓄積表の場合

履歴蓄積表は，横分割するとスループットを向上できることがあります。表の横分割の詳細は，「表の横分割」を参照してください。

(3)　採番表の場合

(a)　性能を優先する採番の場合

性能を優先する採番の設計ポイントを次に説明します。

- 性能を優先する採番は，1 つの採番表で 1 つの採番を管理してください。また，この表の格納先 RD エリアには，その 1 表だけを格納し，個別にグローバルバッファを割り当ててください。これによって，グローバルバッファや RD エリアに関する競合を回避できます。
- 1 表で 1 つの採番を管理する場合は，インデクスは定義しないでください。格納している行が 1 行だけの場合は，インデクスを定義しない方が，アクセスするページ数を少なくできます。

(b)　性能よりも運用性を優先する場合

性能よりも運用性を優先する採番は，1 つの採番表で複数の採番を管理してください。この場合の設計ポイントを次に説明します。

- 採番の列以外の採番種別を示す列に，インデクスを定義してください。インデクスを定義することで，効率良く行にアクセスできます。
- 1 ページに格納する行数を 1 行だけにしてください。これによって，RD エリアに関する競合を回避できます。1 ページに格納する行数を 1 行にする方法を，次に説明します。

  1. ページ内の未使用領域の比率（PCTFREE）を 99%にする
  2. 予備の列を作成するなどして，行長を RD エリアのページサイズの 1%以上の長さにする
  3. データベース作成ユティリティ（pdload）でデータを挿入する

8.2.3　表の横分割

ここでは，性能をさらに向上させるために，表を横分割する場合の設計方法を説明します。横分割の効果，使用する HiRDB サーバの種類，および表特性ごとの設計方法について説明します。

(1)　表の横分割の効果

次の条件に該当する場合は，表を横分割すると性能が向上できることがあります。性能をより速くしたい場合は，横分割の適用を検討してください。

- 表に対するアクセスの多重度が高い場合
  表を横分割することで，負荷を分散し，スループットが向上できることがあります。
- 表のデータ量が多く，かつ大部分のデータにアクセスする場合
  表を横分割することで，アクセスを並列実行し，レスポンスが向上できることがあります。
- 表のデータ量が多く，かつ一部分のデータにアクセスする場合
  表を横分割することで，アクセスするデータの範囲を限定し，レスポンスが向上できることがあります。

表の横分割の効果の概要を，次に示します。

図 8-1　負荷分散によるスループットの向上

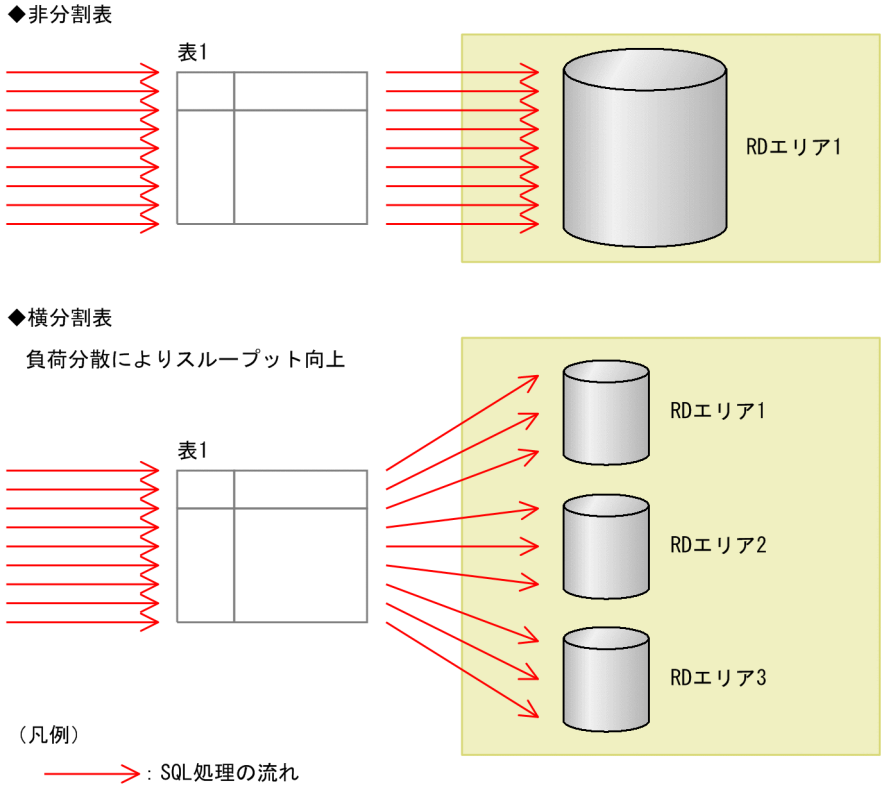


図 8-2　並列実行によるレスポンスの向上

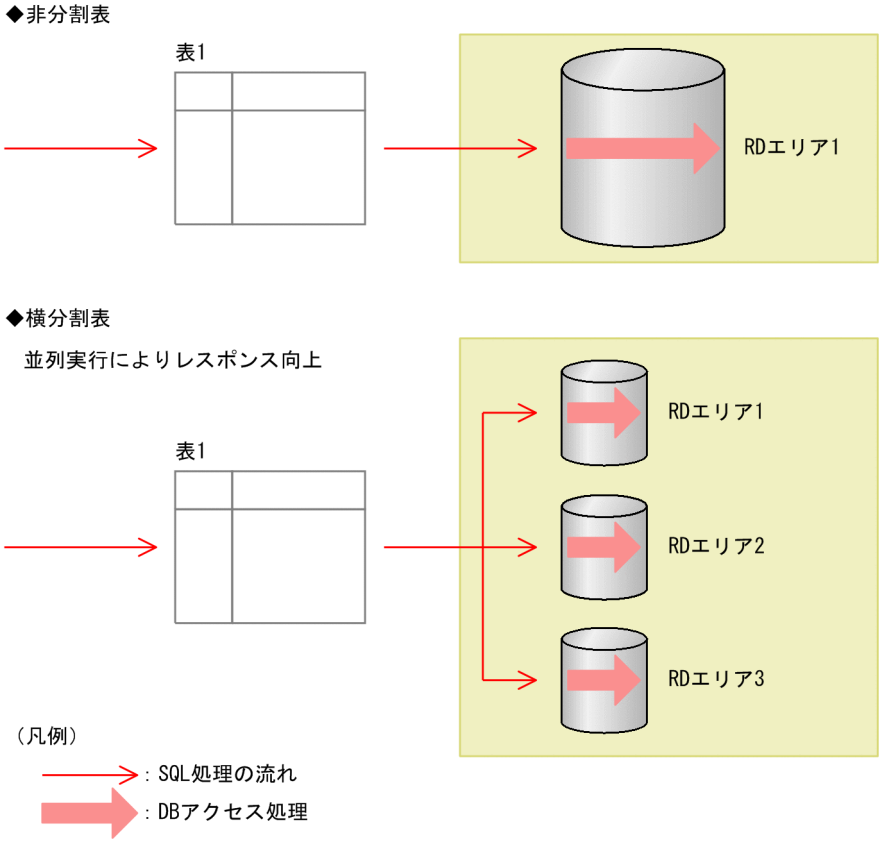
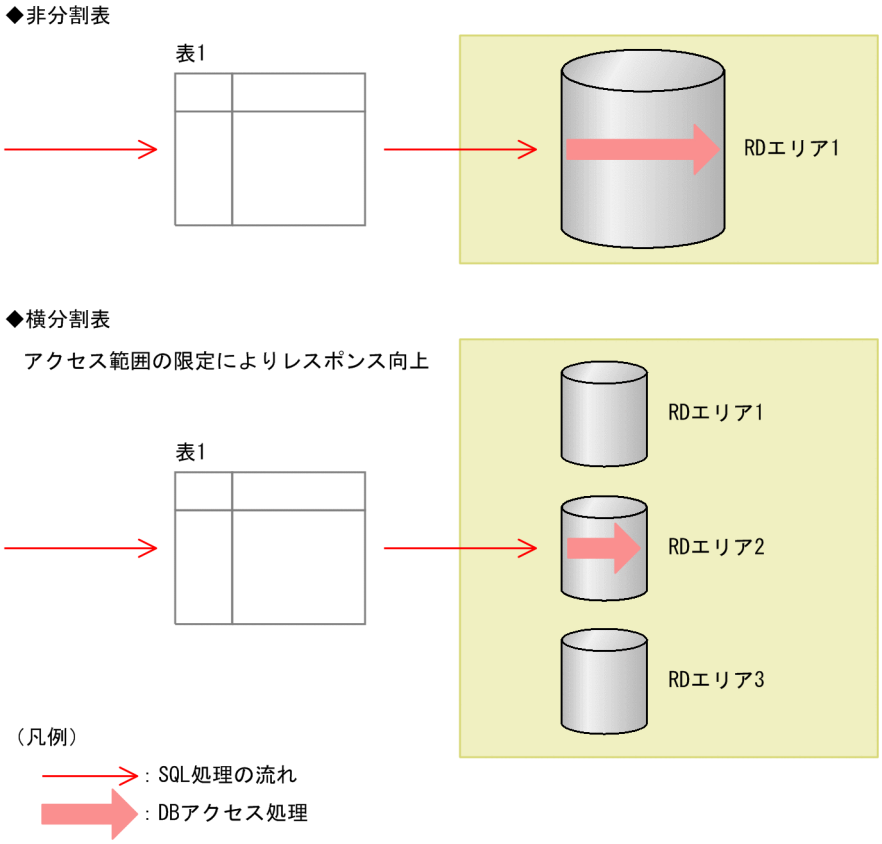


図 8-3　アクセス範囲の限定によるレスポンスの向上



(2) 使用する HiRDB サーバの種類

表を横分割する目的によっては，HiRDB/パラレルサーバの場合だけ実現できることがあります。目的に応じて，使用する HiRDB サーバを決定してください。使用する HiRDB サーバの種類を，次の表に示します。

表 8-1 使用する HiRDB サーバの種類

表を横分割する目的		使用する HiRDB サーバ
負荷を分散し，スループットを向上したい場合	RD エリアやグローバルバッファに関する競合を分散したい場合	HiRDB/シングルサーバおよび HiRDB/パラレルサーバ
	ディスクの入出力処理の負荷を分散したい場合	HiRDB/シングルサーバおよび HiRDB/パラレルサーバ※1
	CPU の負荷を分散したい場合	HiRDB/パラレルサーバ
並列実行し，レスポンスを向上したい場合		HiRDB/パラレルサーバ※2
アクセスするデータの範囲を限定し，レスポンスを向上したい場合		HiRDB/シングルサーバおよび HiRDB/パラレルサーバ

注※1

格納先 RD エリアに対する入出力処理が分散できるハードディスク構成にしてください。

注※2

この目的の場合は，HiRDB/パラレルサーバを使用して，表はサーバ間横分割してください。HiRDB/パラレルサーバでサーバ内横分割した表や，HiRDB/シングルサーバで横分割した表の場合，同じサーバ内にある RD エリアへのアクセスは並列実行になりません。サーバ間横分割とサーバ内横分割については，マニュアル「HiRDB システム導入・設計ガイド」の「表の横分割の形態」を参照してください。

(3) トランザクション表

トランザクション表は，表を横分割すると，性能向上できることがあります。設計方法について次に説明します。

(a) 分割キーの設計

次の条件を満たす列を分割キーにしてください。複数の列の組み合わせによって条件を満たす場合は，複数列の分割キーにしてください。

- 表を横分割する目的に適した列
- 結合キーの列（HiRDB/パラレルサーバで表をサーバ間横分割する場合）

条件の詳細を，次に説明します。

- 表を横分割する目的に適した列

  表を横分割する目的に応じて，分割キーを設計してください。

  - 表の横分割の目的がスループット向上の場合

アクセスが均一に分散できる列を分割キーにしてください。

- 表の横分割の目的がレスポンス向上の場合

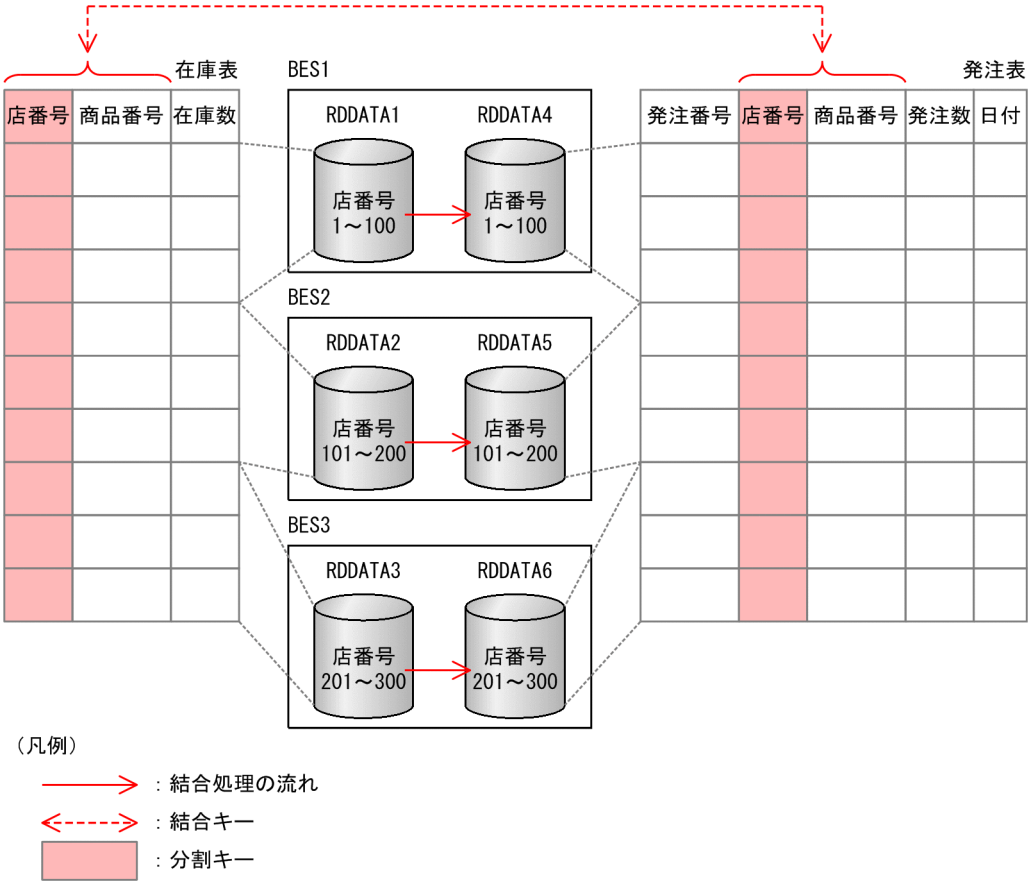
  データを均一に分散できる列を分割キーにしてください。

- 結合キーの列（HiRDB/パラレルサーバで表をサーバ間横分割する場合）

  HiRDB/パラレルサーバで表をサーバ間横分割する場合は，結合キーである列を分割キーにしてください。これによって，表を結合する処理をサーバごとに独立できるため，効率良く処理できます。結合キーではない列を分割キーにすると，1 件のデータの結合処理で複数のサーバを使うため，処理効率が悪くなります。

  結合キーを分割キーにする場合と，結合キーではない列を分割キーにする場合の例を次に示します。

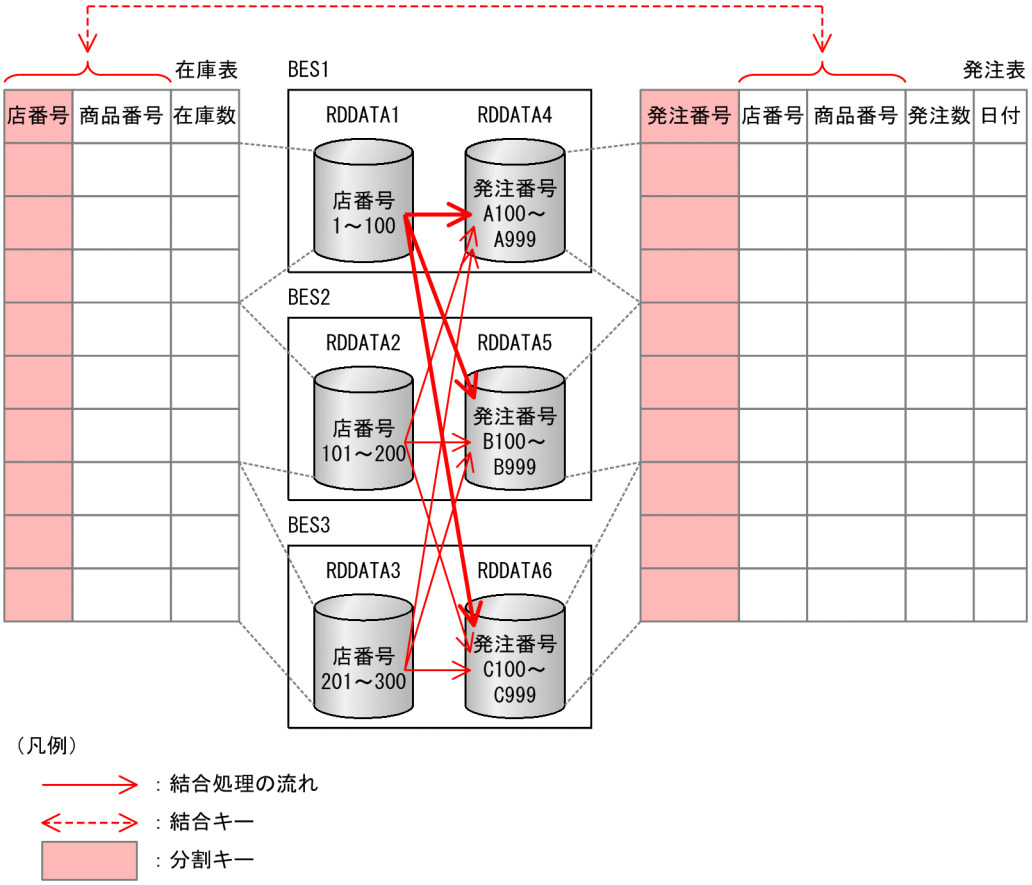
図 8-4　結合キーである列を分割キーにする例



[説明]

- RDDATA1，RDDATA2，RDDATA3：在庫表の格納先 RD エリア
- RDDATA4，RDDATA5，RDDATA6：発注表の格納先 RD エリア
- BES1：RDDATA1，RDDATA4 を管理するバックエンドサーバ
- BES2：RDDATA2，RDDATA5 を管理するバックエンドサーバ
- BES3：RDDATA3，RDDATA6 を管理するバックエンドサーバ

図 8-5　結合キーではない列を分割キーにする例



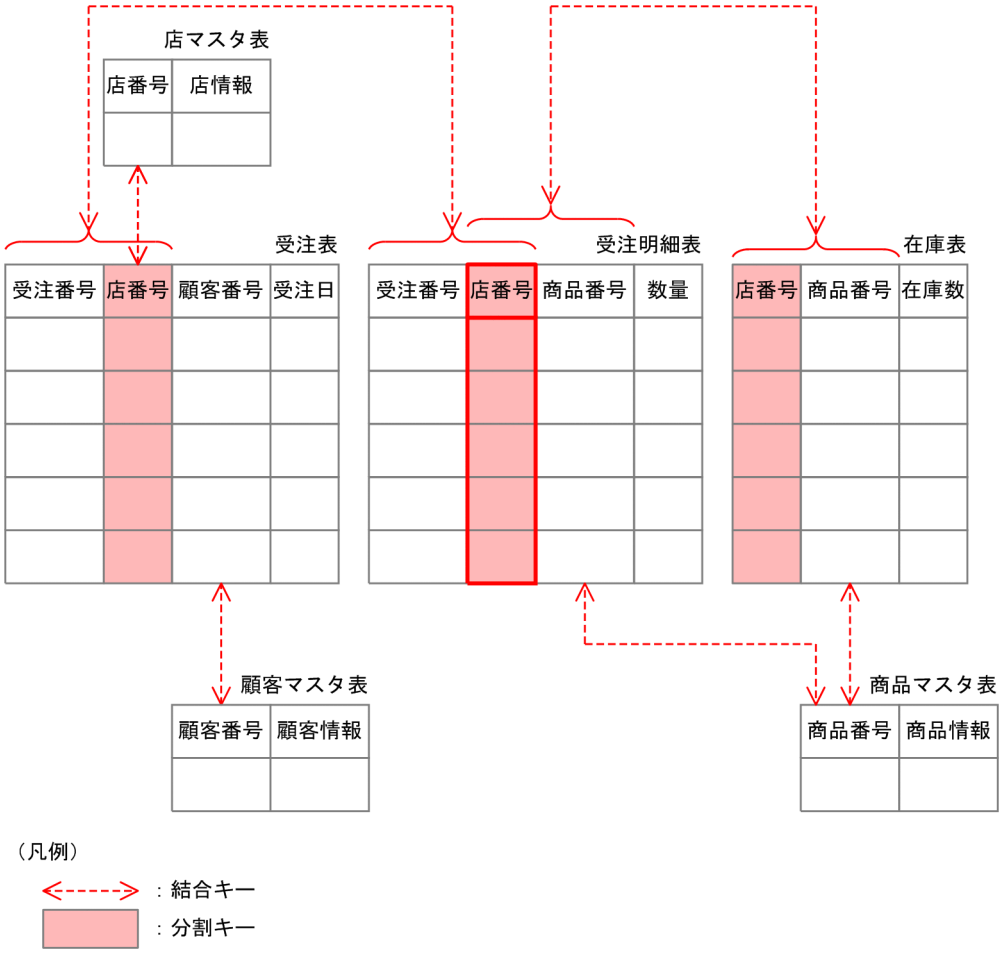
[説明]

- RDDATA1，RDDATA2，RDDATA3：在庫表の格納先 RD エリア
- RDDATA4，RDDATA5，RDDATA6：発注表の格納先 RD エリア
- BES1：RDDATA1，RDDATA4 を管理するバックエンドサーバ
- BES2：RDDATA2，RDDATA5 を管理するバックエンドサーバ
- BES3：RDDATA3，RDDATA6 を管理するバックエンドサーバ

[ここがポイント]

結合キーが複数ある場合は，結合キーの中から，表を横分割する目的に適した結合キーを分割キーにしてください。このとき，業務上はその列が不要であっても，結合するすべての表に分割キーの列を作成し，同じ分割キーで横分割してください。結合キーが複数ある場合の例を次に示します。

図 8-6　結合キーが複数ある例



[説明]

この例では，表を横分割する目的に適した結合キーが，店番号列です。業務上，受注明細表に店番号列は不要ですが，分割キーであるため作成し，結合キーに入れています。

(b)　分割方法の設計

表を横分割する方法には，次に示す 2 種類があります。

- キーレンジ分割
- ハッシュ分割

横分割方法の詳細は，マニュアル「HiRDB システム導入・設計ガイド」の「表の横分割の種類」を参照してください。

横分割方法は，次の考え方で設計してください。

1. 分割キーのデータの種類や分布が把握できる場合

   キーレンジ分割にすることをお勧めします。キーレンジ分割の場合，データの配置を細かく調整できます。

2. 分割キーのデータの種類や分布が把握できない場合

ハッシュ分割にしてください。

(4) マスタ表

マスタは，データ量が少ないため，横分割しても性能向上の効果は低いです。ただし，HiRDB/パラレルサーバで，結合するトランザクション表がサーバ間横分割する場合は，結合の処理をサーバごとに独立できるように設計してください。次に，設計の考え方を説明します。

1. トランザクション表との結合キーが，トランザクション表の分割キーを含む場合

   トランザクション表の分割キーと同じ列を分割キーにして，横分割してください。この対応方法が，最も効率が良いです。

2. トランザクション表との結合キーが，トランザクション表の分割キーを含まない場合

   非分割表にしてください。横分割の目的が次の場合は，共用表にすると，結合処理をサーバごとに独立できます。

   - 負荷分散によるスループット向上
   - アクセス範囲の限定によるレスポンス向上

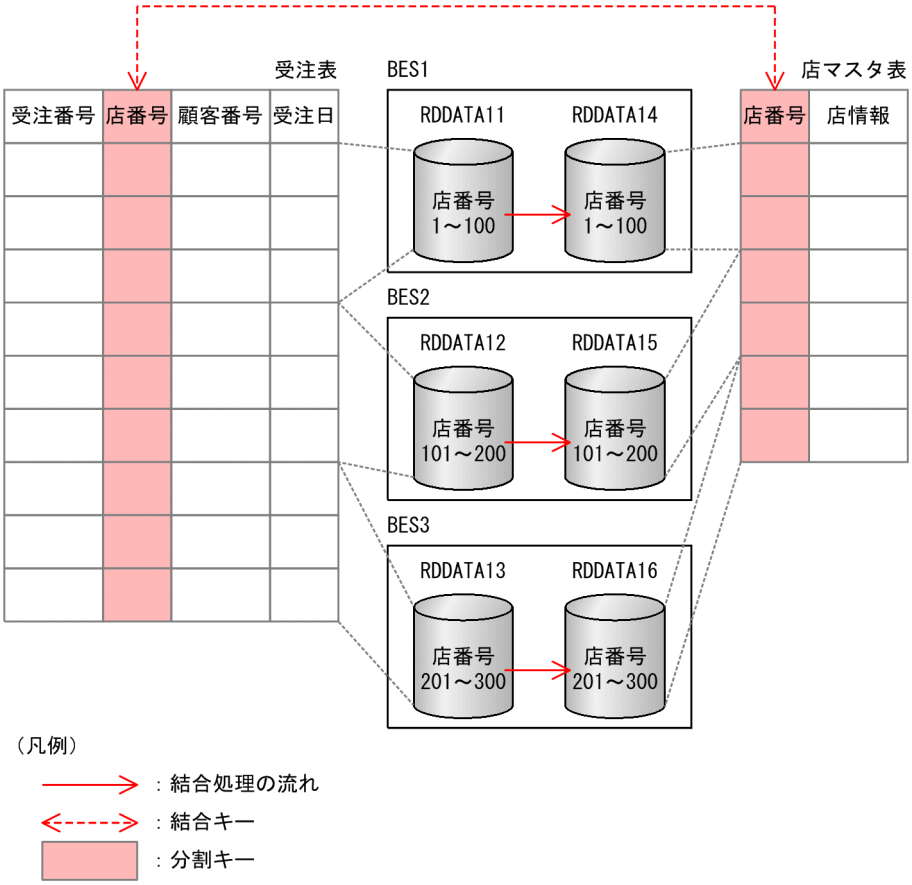
   ただし，共用表にするためには，条件があります。共用表の詳細は，マニュアル「HiRDB システム導入・設計ガイド」の「共用表」を参照してください。

   共用表にしない場合は，マスタ表を管理するバックエンドサーバに処理が集中しないように，非分割のマスタ表は，別のバックエンドサーバに配置してください。

   ただし，1 つのマスタ表へのアクセス頻度が非常に高く負荷が集中する場合は，そのマスタ表は主キーを分割キーにして横分割してください。

マスタ表を横分割する対応の例を次に示します。

図 8-7　マスタ表の対応の例（横分割する場合）



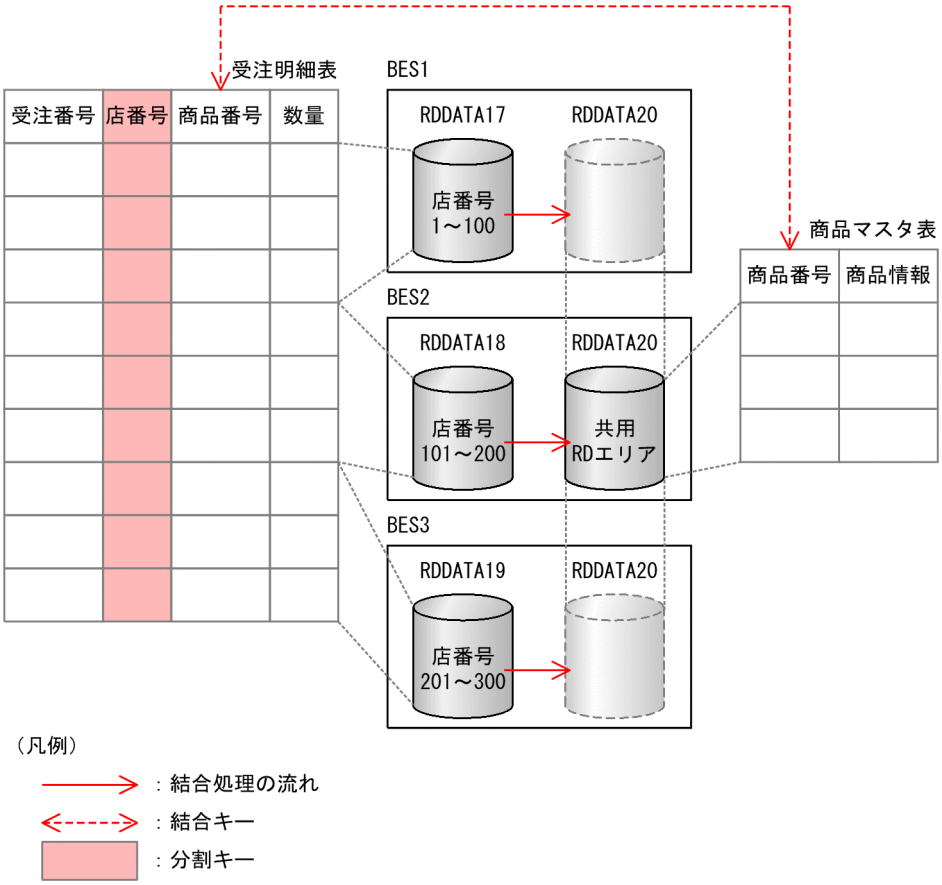
[説明]

店マスタ表には，結合するトランザクション表の分割キーである店番号列があるため，店番号列を分割キーとして横分割します。

- RDDATA11，RDDATA12，RDDATA13：受注表の格納先 RD エリア
- RDDATA14，RDDATA15，RDDATA16：店マスタ表の格納先 RD エリア
- BES1：RDDATA11，RDDATA14 を管理するバックエンドサーバ
- BES2：RDDATA12，RDDATA15 を管理するバックエンドサーバ
- BES3：RDDATA13，RDDATA16 を管理するバックエンドサーバ

マスタ表を共用表にする対応の例を次に示します。

図 8-8　マスタ表の対応の例（共用表にする場合）

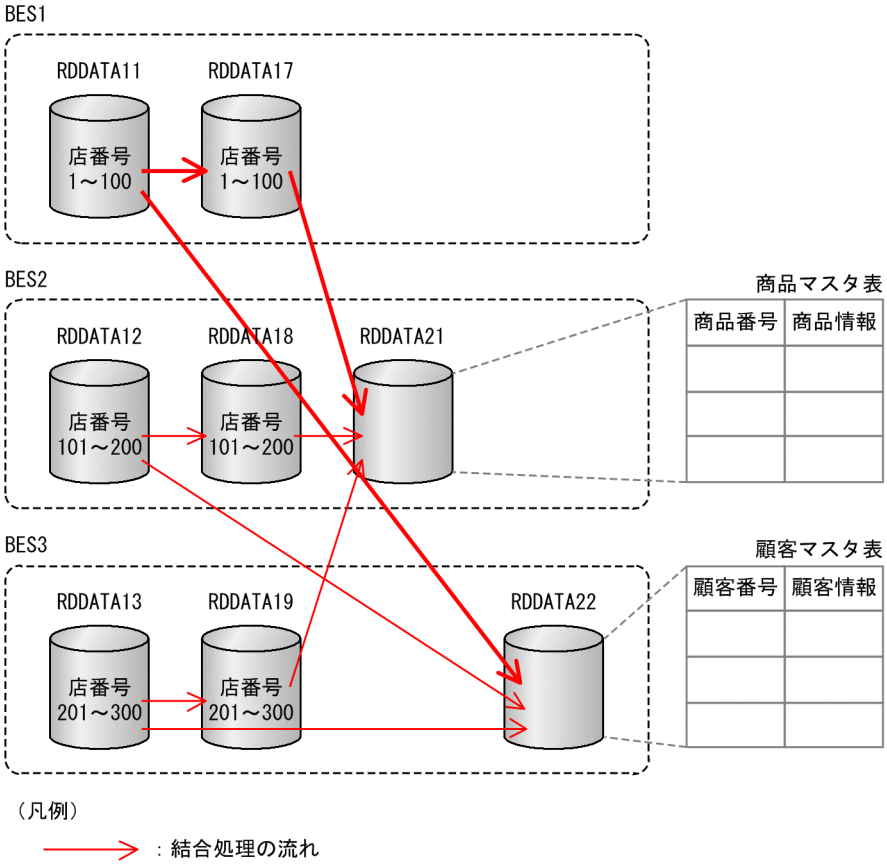


[説明]

商品マスタ表には，結合するトランザクション表の分割キーである店番号列がないため，横分割はしません。この例では，共用表にしています。

- RDDATA17，RDDATA18，RDDATA19：受注明細表の格納先 RD エリア
- RDDATA20：商品マスタ表の格納先 RD エリア（共用 RD エリア）
- BES1：RDDATA17 を管理し，RDDATA20 が参照できるバックエンドサーバ
- BES2：RDDATA18，RDDATA20 を管理するバックエンドサーバ
- BES3：RDDATA19 を管理し，RDDATA20 が参照できるバックエンドサーバ

図 8-9　マスタ表の対応の例（非分割表にする場合）



[説明]

商品マスタ表には，結合するトランザクション表の分割キーである店番号列がないため，横分割はしません。この例では，商品マスタ表は共用表にする条件を満たさないため，非分割表にしています。同様に，非分割表にした顧客マスタ表は，負荷を分散するため，商品マスタ表とは別のバックエンドサーバに配置します。

- RDDATA11，RDDATA12，RDDATA13：受注表の格納先 RD エリア
- RDDATA17，RDDATA18，RDDATA19：受注明細表の格納先 RD エリア
- RDDATA21：商品マスタ表の格納先 RD エリア
- RDDATA22：顧客マスタ表の格納先 RD エリア
- BES1：RDDATA11，RDDATA17 を管理するバックエンドサーバ
- BES2：RDDATA12，RDDATA18，RDDATA21 を管理するバックエンドサーバ
- BES3：RDDATA13，RDDATA19，RDDATA22 を管理するバックエンドサーバ

(5)　採番表

採番表は，トランザクション表と比べてアクセス頻度が少ないため，横分割しても性能向上の効果は低いです。ただし，HiRDB/パラレルサーバで，採番データを格納するトランザクション表がサーバ間横分割する場合は，採番データがサーバ内で受け渡しできると効率が良いです。サーバごとの採番とし，各サーバに採番表を作成することをお勧めします。

(6) 履歴蓄積表

表の横分割の考え方は，トランザクション表と同じです。「トランザクション表」を参照してください。

9 インデクスの設計

この章では，性能を向上させるためのインデクス設計のポイントについて説明します。

9.1 ここは必ず設計しよう

インデクスの設計は，よりよいパフォーマンスにするための重要なポイントです。必ず設計してください。インデクスの設計に関する基本的なポイントは，次の説明を参照してください。

- インデクスの効果
  インデクスの効果については，マニュアル「HiRDB 解説」の「インデクスの基本構造」を参照してください。
- インデクスの設計ポイント
  インデクスの設計ポイントについては，マニュアル「HiRDB システム導入・設計ガイド」の「インデクスの設計」を参照してください。

［参照先］

- インデクス定義に適している列，インデクス定義に適さない列を知りたい
  マニュアル「HiRDB システム導入・設計ガイド」の「インデクスの作成」を参照してください。
- インデクス構成列の組み合わせや構成列の順序の検討方法を知りたい
  マニュアル「HiRDB システム導入・設計ガイド」の「インデクス構成列の検討」を参照してください。
- 1 つの表に複数のインデクスを定義した場合，使用するインデクスの優先順位を知りたい
  マニュアル「HiRDB システム導入・設計ガイド」の「インデクスの優先順位」を参照してください。

10 RD エリアの設計

この章では，性能を向上させるための RD エリア設計のポイントについて説明します。

10.1　ここは必ず設計しよう

ここでは，性能を向上させるために，設計が必須であるポイントについて説明します。

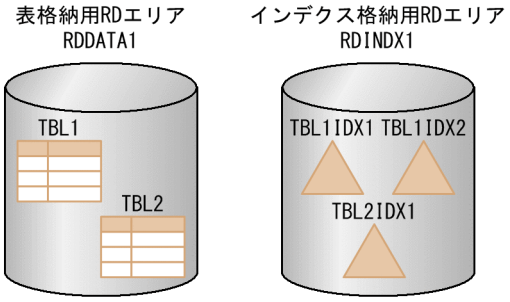
10.1.1　表およびインデクスの格納先 RD エリアの構成

1 つのユーザ用 RD エリアには，1 つまたは複数の表を格納できます。インデクスを格納する場合も同様です。ここでは，性能を考慮した格納先 RD エリアの構成について，説明します。

(1)　表とインデクスの区別

表とインデクスでは，RD エリアのページサイズやグローバルバッファのヒット率の最適な値が異なります。したがって，表とインデクスを格納する RD エリアは区別してください。表とインデクスの格納先 RD エリアを区別する構成例を，次に示します。

図 10-1　表とインデクスの格納先 RD エリアを区別する構成例



[説明]

- RDDATA1：表格納用 RD エリア
- RDINDX1：インデクス格納用 RD エリア
- TBL1，TBL2：表
- TBL1IDX1，TBL1IDX2：表 TBL1 のインデクス
- TBL2IDX1：表 TBL2 のインデクス

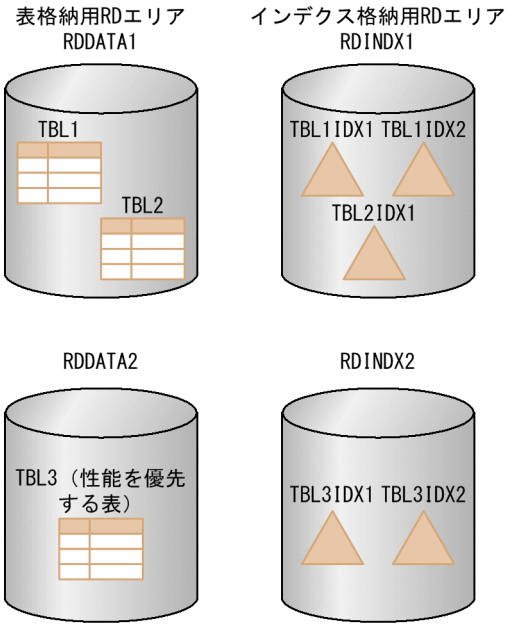
(2)　性能を優先する表の場合

グローバルバッファは RD エリア単位に割り当てるため，1 つの RD エリアに複数の表を格納すると，それらの表でグローバルバッファを共用します。したがって，性能を優先する表の格納先 RD エリアには，グローバルバッファを占有できるように，その 1 表だけを格納してください。これによって，性能を優先する表とほかの表で，グローバルバッファの設計およびチューニングを区別できます。

性能を優先する表に定義したインデクスの格納先 RD エリアも同様に，1 つの RD エリアにその 1 表のインデクスだけを格納してください。

性能を優先する表の格納先 RD エリアの構成例を，次に示します。

図 10-2　性能を優先する表の格納先 RD エリアの構成例



[説明]

- RDDATA1，RDDATA2：表格納用 RD エリア
- RDINDX1，RDINDX2：インデクス格納用 RD エリア
- TBL1，TBL2：表
- TBL3：性能を優先する表
- TBL1IDX1，TBL1IDX2：表 TBL1 のインデクス
- TBL2IDX1：表 TBL2 のインデクス
- TBL3IDX1，TBL3IDX2：性能を優先する表 TBL3 のインデクス

[注意事項]

すべての表ごとに格納先 RD エリアを分けてしまうと，RD エリアの数が多くなり，運用が複雑になります。性能面と運用面のトレードオフを考慮の上，格納先 RD エリアの構成を決定してください。

10.1.2　ページサイズとセグメントサイズ

ページサイズとセグメントサイズの設計方法については，次の説明を参照してください。

- ページサイズの設計方法を知りたい

  マニュアル「HiRDB システム導入・設計ガイド」の「RD エリアの設計」の「ページ長の決定」を参照してください。

- セグメントサイズの設計方法を知りたい

  マニュアル「HiRDB システム導入・設計ガイド」の「RD エリアの設計」の「セグメントサイズの決定」を参照してください。

［表特性別のポイント］

トランザクション表および履歴蓄積表のように，INSERT 文のレスポンスが求められる場合は，セグメントサイズは大きくすることをお勧めします。セグメントサイズを大きくすると，新しいセグメントへデータを格納する際のオーバーヘッドが低減できます。

11 グローバルバッファの設計

この章では，性能を向上させるためのグローバルバッファ設計のポイントについて説明します。

11.1　ここは必ず設計しよう

ここでは，グローバルバッファの初期設計に関する基本的なポイントについて説明します。なお，ここで説明する内容は初期値の考え方ですので，性能評価の工程で，必ずグローバルバッファのチューニングを実施してください。

11.1.1　グローバルバッファの構成

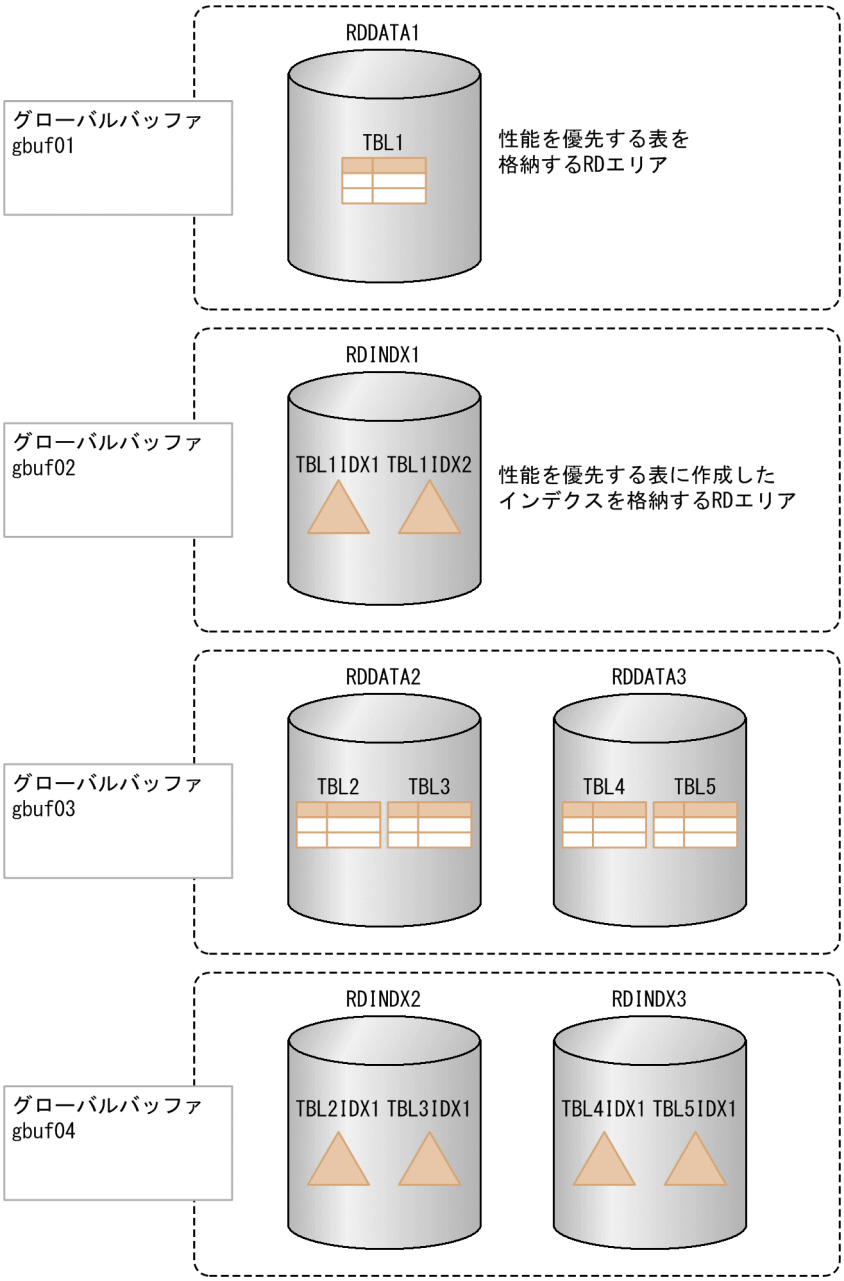
グローバルバッファを効率良く使用するために，次に示す点を考慮して構成を設計してください。

1. 次に示す RD エリアごとに，グローバルバッファは分けてください。
   - インデクス格納用 RD エリア
   - 表格納用 RD エリア
   - LOB 用 RD エリア

   例えば，表格納用 RD エリアとインデクス格納用 RD エリアを同じグローバルバッファに割り当てた場合，表のページをバッファに読み込むことで，アクセス頻度の高いインデクスのページがバッファから追い出されることがあります。表，インデクス，および BLOB データは，それぞれバッファに常駐したいページの割合が異なりますので，グローバルバッファは分けてください。
2. 性能を優先する RD エリアは，個別にグローバルバッファを割り当ててください。次に理由を説明します。
   - 性能を優先する RD エリアは，その他の RD エリアとはバッファに常駐したいページの割合が異なります。ほかの RD エリアのページをバッファに読み込むことで，性能を優先する RD エリアのページがバッファから追い出されることがありますので，グローバルバッファは分けてください。
   - 1 つのグローバルバッファに複数の RD エリアを割り当てると，トランザクション間のバッファ競合，およびそれに伴う排他待ちが発生する確率が高くなります。
3. 1 つのグローバルバッファに複数の RD エリアを割り当てる場合は，同じページサイズの RD エリアにしてください。異なるページサイズの RD エリアを同じグローバルバッファに割り当てると，バッファ 1 面のサイズは，グローバルバッファに割り当てた RD エリアの最大ページ長になります。このため，メモリの使用効率が悪くなります。1 つのグローバルバッファに割り当てる RD エリアは，必ず同じページサイズの RD エリアにしてください。

グローバルバッファの構成例を次に示します。

図 11-1　グローバルバッファの構成例



11.1.2　グローバルバッファ面数の設計

ここでは，グローバルバッファのバッファ面数の初期設計について説明します。バッファ面数は，システム共通定義の pdbuffer オペランドの-n オプションで指定します。指定方法の詳細は，マニュアル「HiRDB システム定義」の「グローバルバッファに関するオペランド」を参照してください。

グローバルバッファの面数は，アクセス頻度の高いページをバッファ上に常駐できるように設計します。一般的に，アクセス頻度の高いページは，次に示す順序です。

1. インデクスのページ

2. LOB 用 RD エリアのディレクトリページ
3. 表のページ
4. LOB 用 RD エリアのデータページ

グローバルバッファの面数は，この順序で設計することをお勧めします。なお，LOB 用 RD エリアを使用していない場合，LOB 用 RD エリアのバッファ面数の設計は不要です。

(1) インデクスのバッファ面数

条件に一致するデータを取り出す場合，インデクスのページを参照して条件に一致するデータを絞り込んでから，表のページにアクセスします。よって，インデクスのページの方がアクセスする頻度が高いです。このため，インデクス用のグローバルバッファは優先して割り当ててください。

(a) メモリに余裕がある場合

インデクス格納用 RD エリアの全ページをグローバルバッファに常駐できる面数にしてください。

(b) メモリに余裕がない場合

メモリに余裕がない場合は，性能の優先度によって，バッファに常駐する面数を設計してください。次に考え方を説明します。

- 性能を優先する表に定義したインデクスを格納している RD エリア
  性能を優先するインデクス格納用 RD エリアは，全ページをグローバルバッファに常駐できる面数にしてください。
- その他の RD エリア
  その他のインデクス格納用 RD エリアは，全ページの 60%以上が常駐できる面数にすることをお勧めします。この割合でもメモリに余裕がない場合は，リーフページ以外のページが常駐できるように，全ページの 15%以上の面数にすることをお勧めします。

［表特性別のポイント］

マスタ表および採番表は，データ量が少なく，すべてのデータが偏りなくアクセスされるため，インデクスは全ページを常駐してください。

［こんなときは］

RD エリアに複数のインデクスが格納されている場合，特定のインデクスについて優先的にグローバルバッファを割り当てたいときは，インデクス用グローバルバッファを指定してください。インデクス用グローバルバッファは，システム共通定義の pdbuffer オペランドの-i オプションで指定します。詳細は，マニュアル「HiRDB システム定義」の「グローバルバッファに関するオペランド」を参照してください。

ただし，サーバ内の横分割表に定義した分割インデクスの場合は，インデクス用グローバルバッファを使用しないで，格納先 RD エリアをほかのインデクスとは分けてください。インデクス用グローバル

バッファにすると，サーバ内にある複数の格納先 RD エリアのページが，1 つのグローバルバッファに割り当てられるため，バッファ競合，およびそれに伴う排他待ちが発生する確率が高くなります。

(2) 表のバッファ面数

(a) メモリに余裕がある場合

表格納用 RD エリアの全ページをグローバルバッファに常駐できる面数にしてください。

(b) メモリに余裕がない場合

メモリに余裕がない場合は，性能の優先度によって，バッファに常駐する面数を設計してください。次に考え方を説明します。

- 性能を優先する表を格納した RD エリアの場合

  表格納用 RD エリアの全ページをグローバルバッファに常駐できる面数にしてください。
- その他の RD エリアの場合

  その他の表格納用 RD エリアは，全ページの 50%以上が常駐できる面数にすることをお勧めします。この割合でもメモリに余裕がない場合は，全ページの 5%以上の面数にすることをお勧めします。

[特性別のポイント]

マスタ表および採番表は，データ量が少なく，すべてのデータが偏りなくアクセスされるため，表は全ページを常駐してください。

(3) LOB 用 RD エリア（ディレクトリページ）のバッファ面数

LOB 用 RD エリアのディレクトリページに割り当てるグローバルバッファは，システム共通定義の pdbuffer オペランドの-r オプションで指定します。LOB 用 RD エリアのすべてのディレクトリページが常駐できる面数にしてください。

ディレクトリページは RD エリアの管理情報を格納するページであるため，データページよりもアクセスする頻度が高いです。LOB 用 RD エリアのデータページよりも，LOB 用 RD エリアのディレクトリページの方を優先的にグローバルバッファへ割り当ててください。ディレクトリページ数は，全ページ数の 0.2%を目安にしてください。詳細なページ数の計算式は，マニュアル「HiRDB システム導入・設計ガイド」の「ユーザ LOB 用 RD エリアの容量の見積もり」を参照してください。

(4) LOB 用 RD エリア（データページ）のバッファ面数

LOB 用 RD エリアのデータページに割り当てるグローバルバッファは，システム共通定義の pdbuffer オペランドの-b オプションで指定します。

次の場合に，このバッファを割り当てると性能が良くなります。

- 同じ BLOB データに複数回アクセスする業務の場合

同じ BLOB データへアクセスする業務が継続している間は，その BLOB データをバッファに常駐させると効率が良くなります。例えば，登録したデータを登録日のうちに処理する場合は，1 日分の登録データが常駐できるサイズにすることをお勧めします。メモリに余裕がない場合は，メモリが使用できる範囲で面数を決定してください。

- LOB 用 RD エリアにプラグインインデクスを格納している場合

  LOB 用 RD エリアにプラグインインデクスを格納している場合は，同じページを複数回アクセスすることが多いため，バッファを割り当てることをお勧めします。メモリが使用できる範囲で面数を決定してください。

索引

さ

し

ち

て

と

は

ひ

ま

よ

り