問合せ処理の効率化

データベース論Ⅰ

第8回

URL　http://homepage3.nifty.com/suetsuguf/

作成者　　末次文雄 ©

課題7の解答例→テーブル定義

```
CREATE  TABLE  学科TBL
    (学科番号     INT(7)  NOT  NULL  UNIQUE,
     年度          INT(4)  NOT  NULL,
     学科名称     NCHAR(10),
     主任          NCHAR(10)                    );
CREATE  TABLE  学生TBL
    (学籍番号     CHAR(7)  NOT  NULL  UNIQUE,
     年次          INT(1),
     入学年度     INT(4),
     氏名          NCHAR(10)                    );
CREATE  TABLE  所属TBL
    (学科コード  INT(7)      NOT  NULL,
     年度          INT(4),
     学生コード  CHAR(7),                     );
```

課題7の解答例→テーブル定義

```
CREATE  TABLE  科目TBL
    (学科番号     INT(7)  NOT  NULL,
     科目コード   INT(7)  NOT  NULL,
     年度         INT(4)  NOT  NULL,
      HS区分      CHAR(1)  NOT  NULL,
     科目名称     NCHAR(10),
     担当教師     NCHAR(10)                    );
CREATE  TABLE  受講者TBL
    (学科番号     INT(7)  NOT  NULL,
     科目コード   INT(7)  NOT  NULL,
     年度         INT(4)  NOT  NULL,
     受講者コード CHAR(7)  NOT  NULL
     評価         CHAR(1),
     粗点         INT(3)              );
```

課題7の解答例→更新文

```
INSERT   INTO   受講者TBL
            (学科番号,科目コード,年度,受講者コード)
          VALUES  (1001100,1001104,2003,'A100100' )  ;

UPDATE  学生TBL
          SET  年次  =  2,
                  氏名  =  '小泉純一郎'
           WHERE  学籍番号  =  1001110          ;

DELETE   FROM  学生TBL
          WHERE  学籍番号  =  1001200
                   AND
                   学籍番号  =  1001300          ;
```

課題7の解答例→SELECT文（1）

```
SELECT    学生TBL.学籍番号,学生TBL. 氏名,
          学生TBL.入学年度,
          受講者TBL.評価,受講者TBL.粗点
FROM      学科TBL,学生TBL,科目TBL,受講者TBL
WHERE     学科TBL.学科名称='情報'        AND
          受講者TBL.年度=2002           AND
          科目TBL.科目名称='データベース論'
    AND
        学科TBL.学科番号=科目TBL.学科番号
    AND
        (科目TBL.学科番号   AND   科目TBL.科目コード)
      =(受講者TBL.学科番号   AND   受講者TBL.科目コード)
    AND
        学生TBL.学籍番号=受講者TBL.受講者コード
ORDER   BY   学生TBL.入学年度,学生TBL.学籍番号          ;
```

4つのJOIN

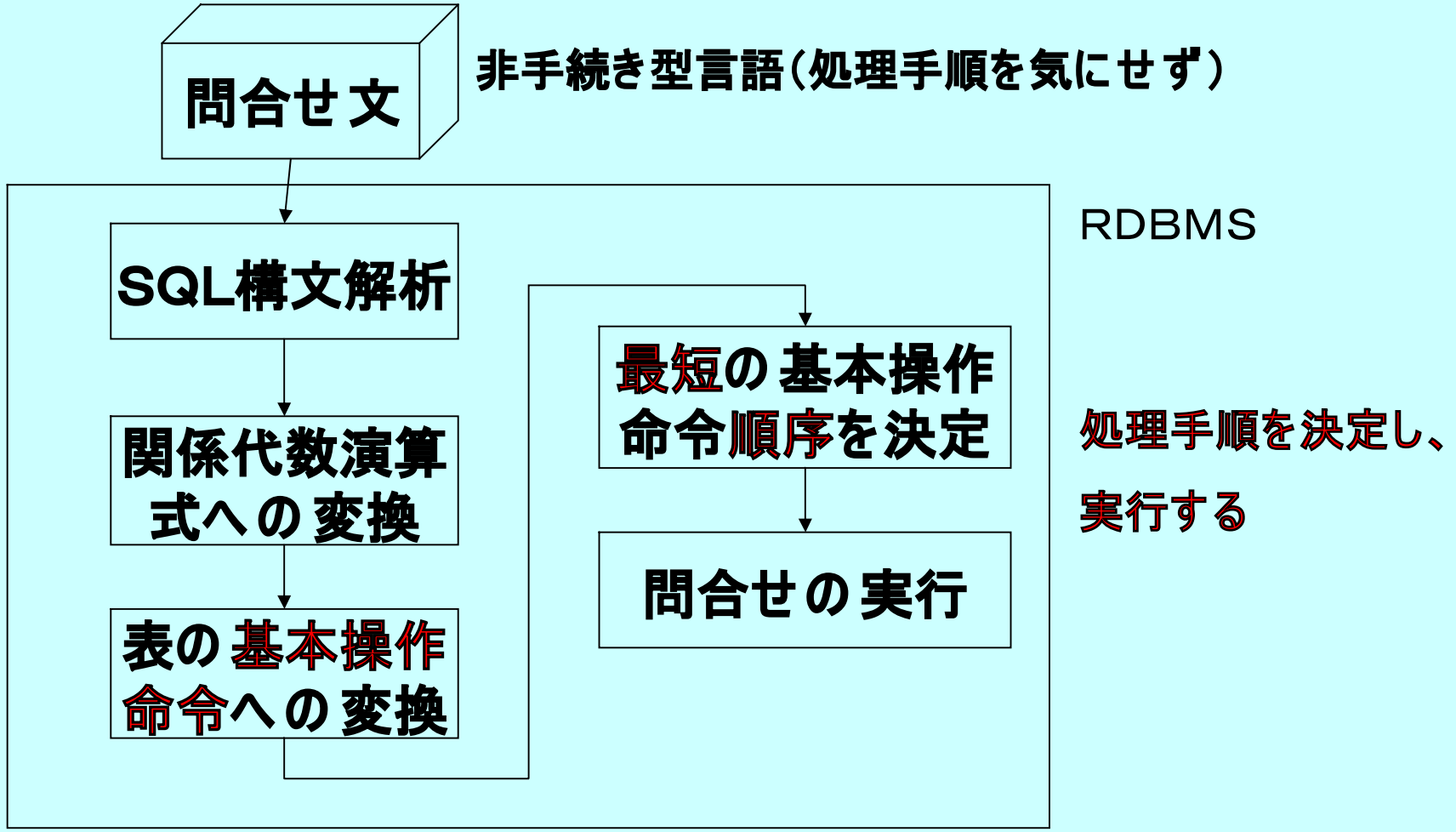
課題7の解答例→SELECT文（2）

```
SELECT     COUNT(DISTINCT  受講者TBL.受講者コード),
           AVG(受講者TBL.粗点)
 FROM      学科TBL,科目TBL,受講者TBL
 WHERE   学科TBL.学科名称='情報'    AND
           受講者TBL.年度=2002        AND
           科目TBL.科目名称='データベース論'
     AND
        学科TBL.学科番号=科目TBL.学科番号,
     AND
       (科目TBL.学科番号   AND   科目TBL.科目コード)
      =(受講者TBL.学科番号   AND   受講者TBL.科目コード)
 ;
```

目次

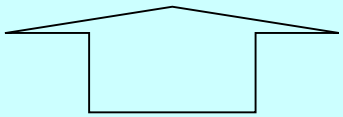
1. 問合せ式の順序付け

1.1 問合せ処理の手順



１．２　表の基本操作命令

- 表の中から、指定した列の値を持つ組の取出し。（選択）
- 表の中から、指定した列を抽出（射影）
- 表の各行を、列の値による並び替え（SORT）
- 表同士の、列の値によるマッチング（結合）
- 表同士をマージする（和演算）



上記の操作には、いずれも一時的な記憶域を必要とする。

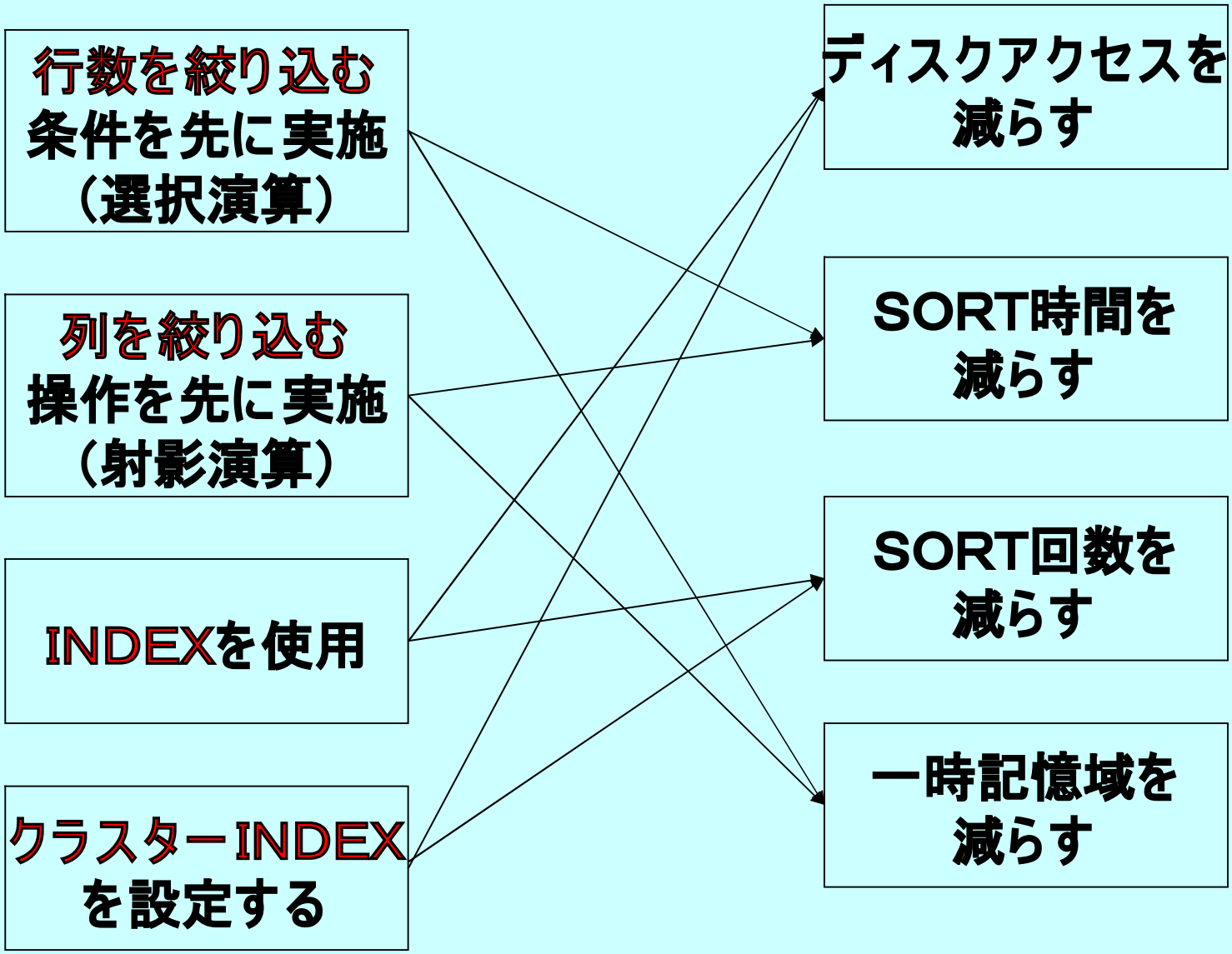
２．データ操作順序の最適化

「複数の処理順序案から
データ処理時間が最短になるように、
処理順序を決定する。」

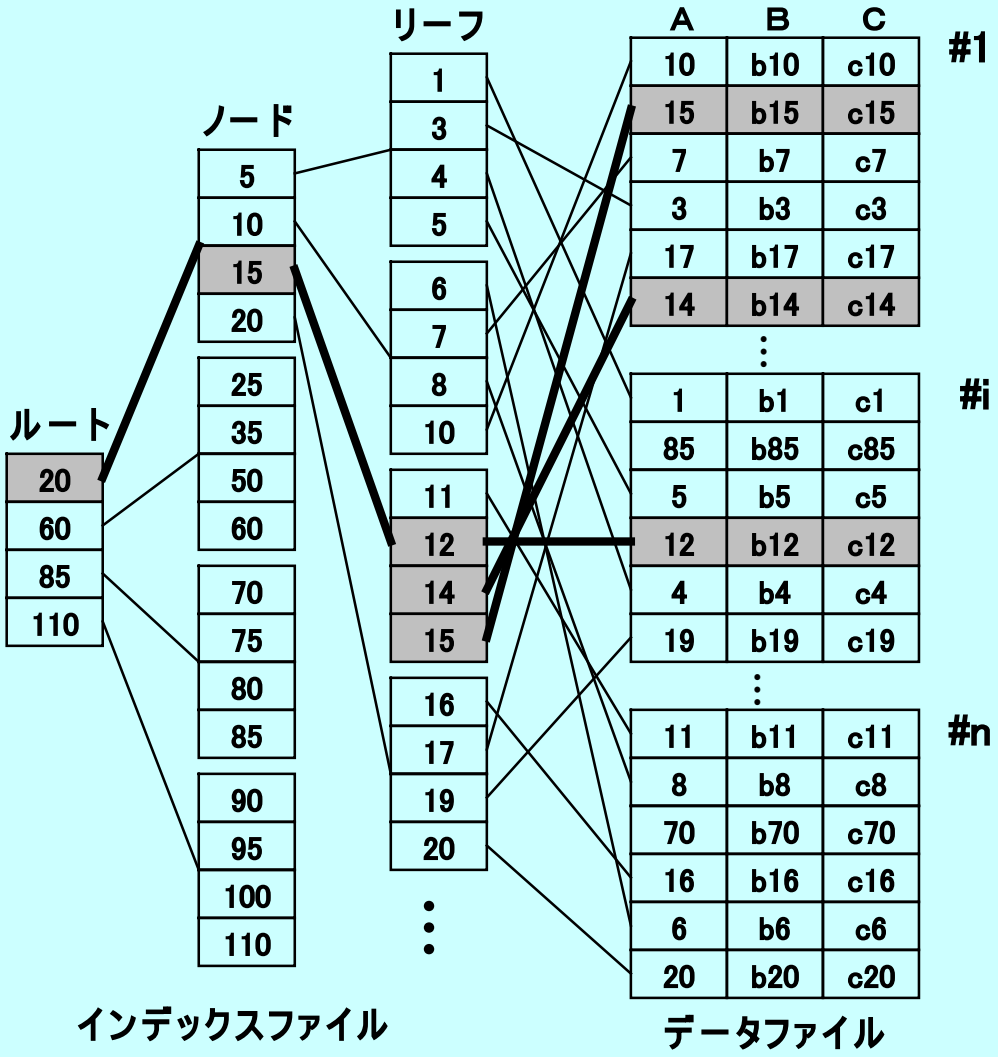
＝

① ディスク・アクセスを減らす。
② SORT回数を減らし、MERGEにする。
③ 一時的な記憶域を減らす。

２．１　最適化の方法


行数を絞り込む
条件を先に実施
（選択演算）
列を絞り込む
操作を先に実施
（射影演算）
INDEXを使用
クラスターINDEX
を設定する
ディスクアクセスを
減らす
SORT時間を
減らす
SORT回数を
減らす
一時記憶域を
減らす

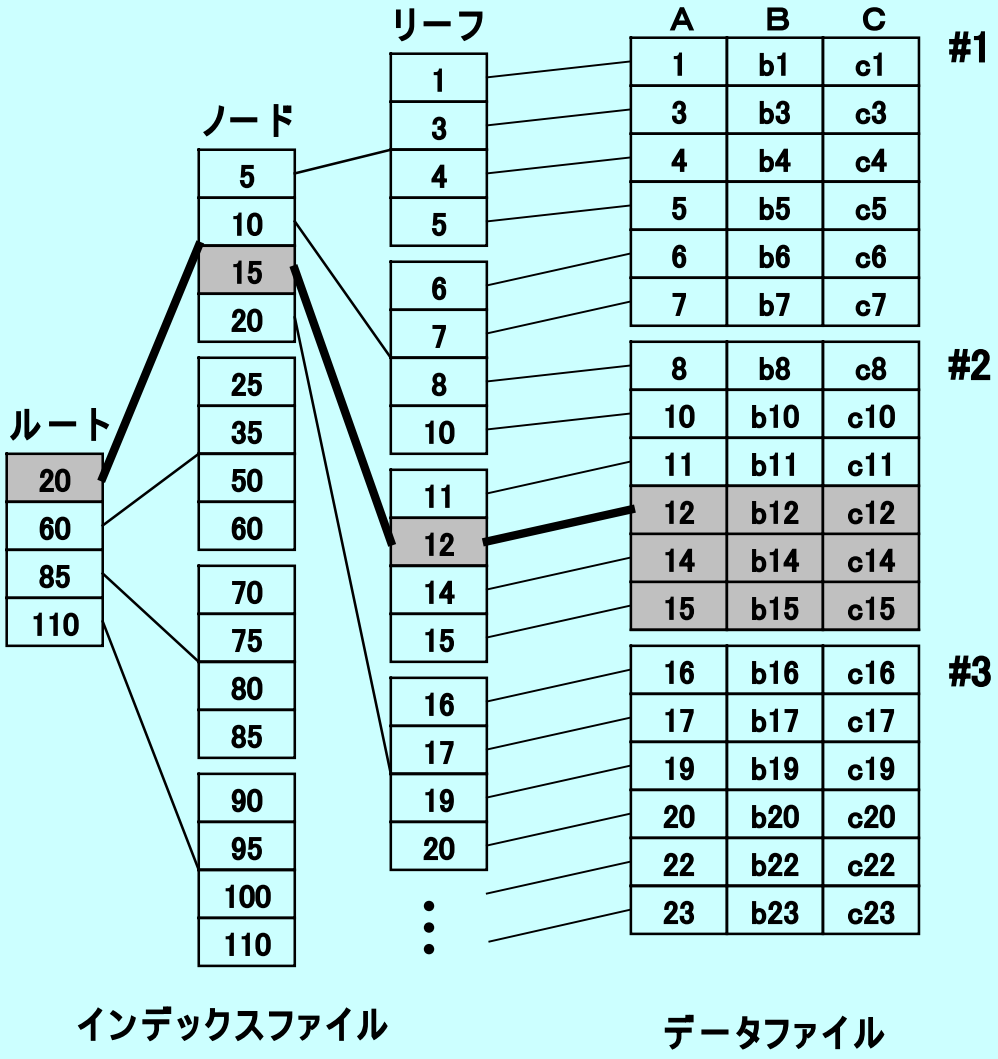
インデックスのクラスタ化（ノンクラスタード）



探索条件

A ≧ 12 AND A ≦ 15

インデックスのクラスタ化（クラスタード）



探索条件

A ≧ 12 AND A ≦ 15

２．２　最適化の例示

「学籍番号‘95510’の学生について、受講の科目番号、科目名、成績を取出す」

１）SQL文

```
SELECT  科目番号, 科目名, 成績
  FROM  科目表, 履修表
 WHERE  科目番号  ＝  科目No
   AND  学籍番号＝‘95510’  ;
```

科目表

科目番号	科目名	単位数	教官名

履修表

科目No	学籍番号	成績

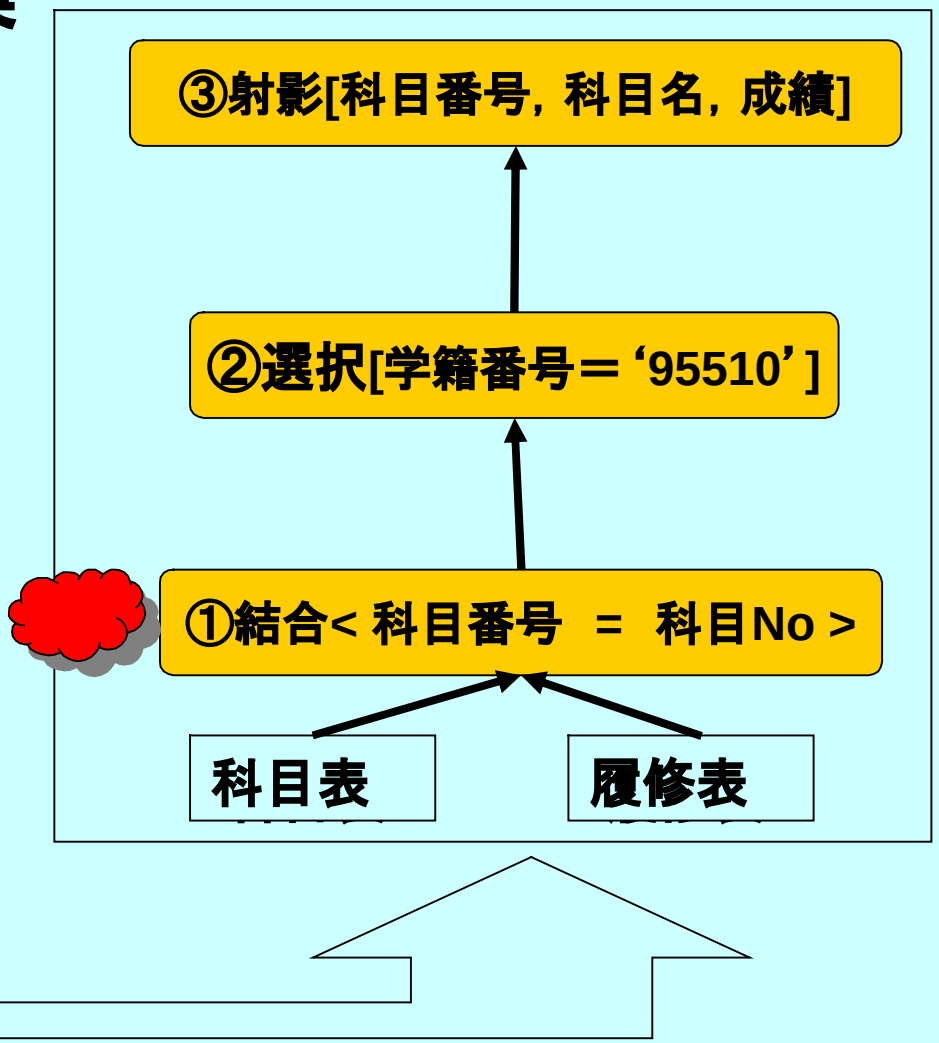
２．２（続き） 最適化の例示

2)関係代数演算式への変換

```
SELECT  科目番号, 科目名, 成績
  FROM  科目表, 履修表
 WHERE  科目番号 ＝ 科目No
   AND  学籍番号＝‘95510’ ;
```

①結合< 科目番号 ＝ 科目No >
②選択[学籍番号＝‘95510’]
③射影[科目番号, 科目名, 成績]

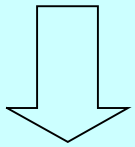
処理木(問合せ木)



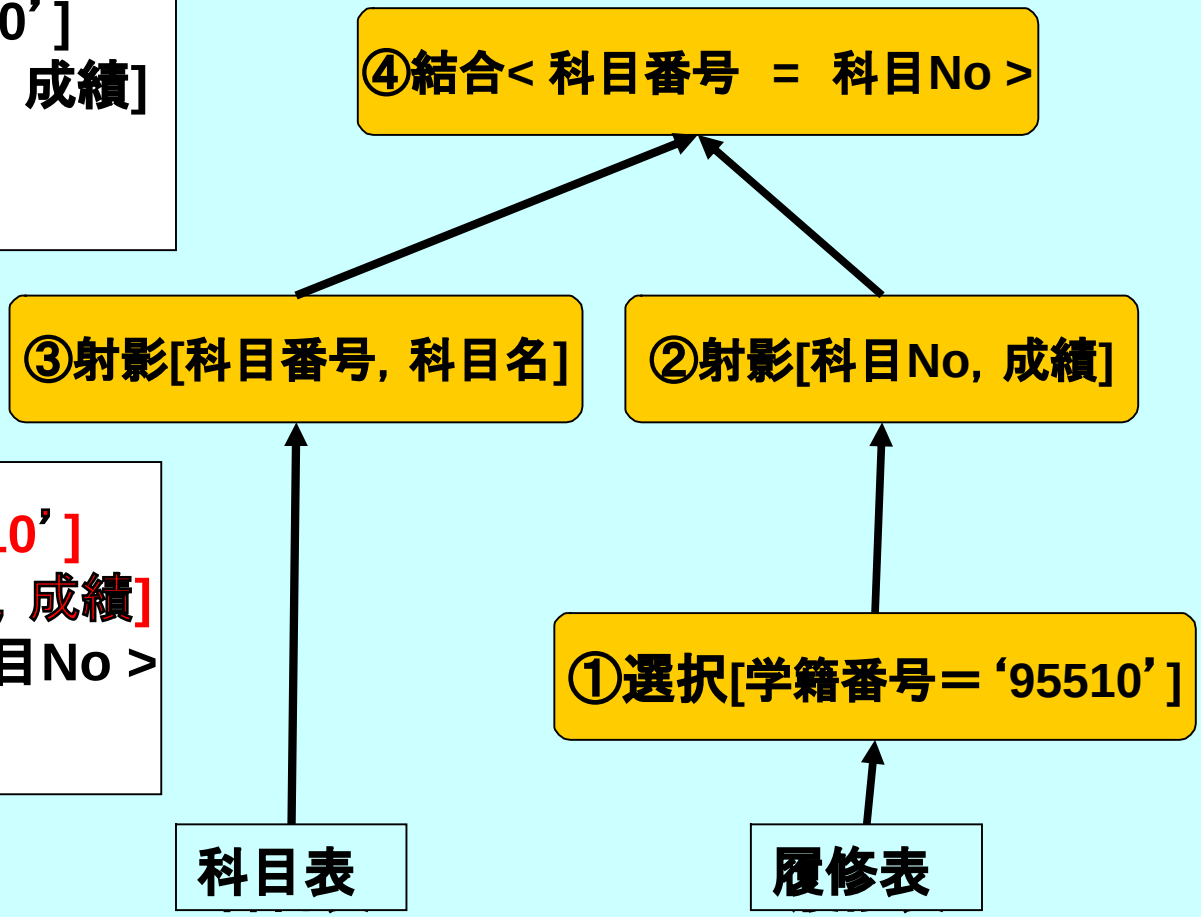
２．２（続き） 最適化の例示

３）関係代数演算の交換規則の適用

①結合< 科目番号 ＝ 科目No >
②選択[学籍番号＝‘95510’]
③射影[科目番号，科目名，成績]



①選択[学籍番号＝‘95510’]
②射影[科目番号，科目名，成績]
③結合< 科目番号 ＝ 科目No >

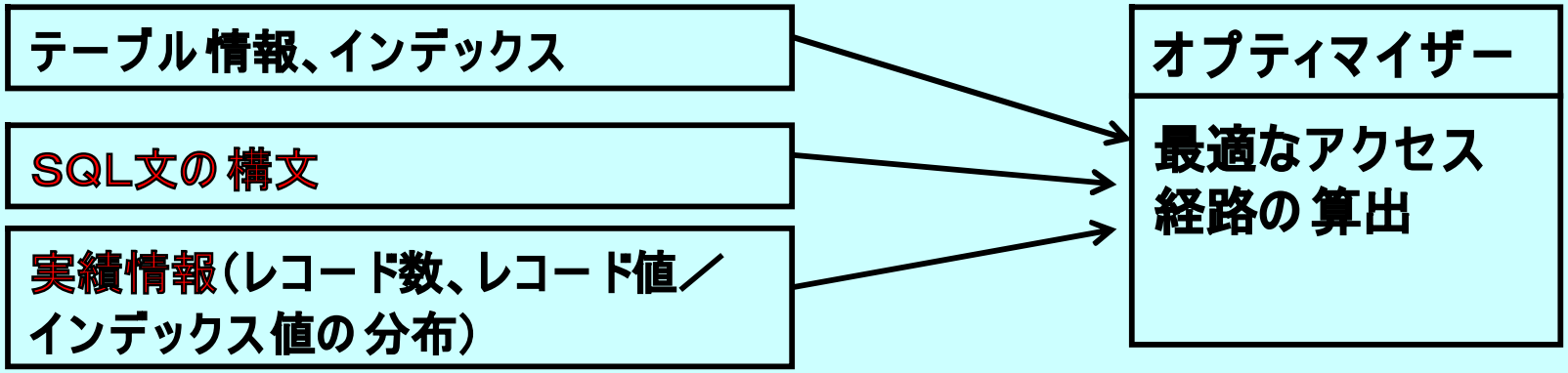


３．　オプティマイザー

３．１　オプティマイザーとは？

・RDBMSには、検索経路の自動最適化ロジックが付いている。
（最適化処理optimizer）

・SQL文の構文解析し、テーブル構造を分析することにより、SQL文の実行計画（plan）を立てて、最適なアクセス経路を検出する。



３．２　オプティマイザーの制約

- データ件数が少ない、キー値の種類が少ない場合は、インデックスを経由しない場合がある。
- SQL文が稚拙であれば、インデックスを使用せず、全件アクセスとなる場合がある。
  （例：SELECT文で、WHERE句のモレ）
- インデックスだけでなく、SQL文の作成方法も、アクセススピードに大いに関係する。
  （AND条件、結合の順序などに注意する）

３．３　オプティマイザーの種類

- 二つの手法が用意されている。
  - ルールベース
    - 一定の事前に定めた優先順位に従って、
      planを立て、アクセス経路を決定。
  - コストベース←有利
    - 予め収集した情報をもとに複数のplanを立て
      最も安価なアクセス経路を決定。
      （データ項目の長さ、木構造インデックスの
      深さ、列データの分布、過去の利用統計情報など）

（例示）ルールベースの優先順位

優先順位	アクセス経路
1	ユニークキーまたは主キーによる単一行
2	インデックス付きのクラスターキー
3	複合インデックス
4	単一カラムインデックス
5	Sort／Merge結合
6	インデックス付きカラムのMAX/MIN関数
7	インデックス付きカラムのORDER　BY
8	テーブル全体のスキャン

３．４　EXCUTION－PLAN

- 文字通り、問合せの「実行計画」である。
- どういう順序で、どのアクセス・パスを経由するかの説明文である。
  （順序、使用INDEXが分る）
- SQL文に対して、RDBMSが最適化した処理の「実行計画」を入手できる。
  （例）EXPLAIN　PLAN　命令

４．性能チューニング

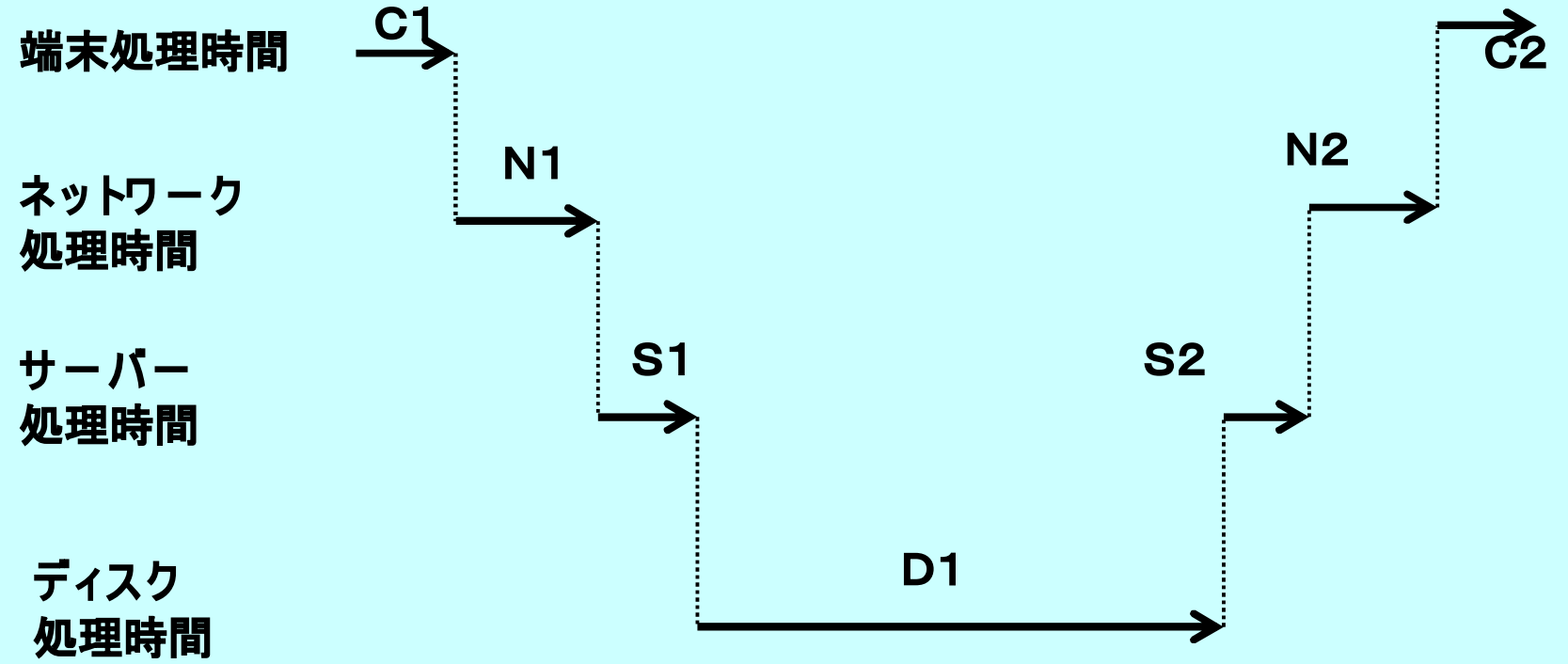
４．１　データ構造の見直し

４．２　INDEXの追加

４．３　その他

（補足） レスポンスタイムの内訳

例:オンライン処理の 場合



４．１　データ構造の見直し

①導出データの保管

- 頻繁にアクセス対象になる場合は、計算処理時間を短縮するため、導出データといえどもデータとして持つ
  - ー合計値など（小計、中計、総合計、計算値・・・）
  - ー事例：給与計算結果など
- ただし、リアルタイムで元のデータが更新される場合はその都度、導出データが変わる場合は、導出データの更新というオーバーヘッドがかかり、不利となる。

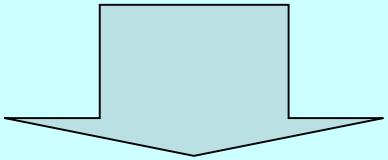
４．１（続き）データ構造の見直し

②逆正規化

- 頻繁に複数のテーブル、行を参照する場合は、ディスクアクセス回数を減らすために、あえて正規化をしない場合がある。
  （ただし、データ更新は重複する。）

例１：　繰り返しデータを持つ（時系列データなど）

営業店	四半期区分	四半期売上



営業店	1Q売上	2Q売上	3Q売上	4Q売上

②（続き） 逆正規化

例2： データを重複させる

社員番号	社員氏名	入社年月	住所

社員番号	支給年月	社員氏名	支給金額

組織コード	組織名	社員番号	社員氏名

Projコード	Proj名称	社員番号	社員氏名

（ただし、データ更新は重複する。）

４．２　INDEXの作成

- 表（テーブル）の結合、WHERE句などの条件句に頻繁に出現する列（データ項目、属性）に関しては、INDEXを作成しておくことが、非常に有効である。

- ただし、本来のデータ以外にも、INDEXファイルの更新がなされており、オーバーヘッドがかかり、ディスク容量も増大する。

４．３　その他

①プログラムの改良

・選択条件句の漏れを無くす。

・AND条件では、適合件数の少ない列から指定する。

・結合の順序の見直し。

②オプティマイザーの調整

・ルールベースから、コストベースへ変更

・優先項目を変更（RDBMSにより異なる）

４．３（続き）その他

③RDBMSのパラメータ変更

・バッファーサイズを大きくする。

④HWの増強

・サーバーの高性能化
・並列プロセッサー
・メモリー増設
・ディスクの高性能化
・連結ディスク（ディスク・アレイ）

６．レポート課題

①別途用意したテスト問題に答えなさい。

　問１、問２、問３、問４　があります。

②テスト用紙の末尾に、MySQLのインストール状況を追記しておいてください。

（インストール済み、稼動可能、①の問題を実行した）

次回の授業開始時に、提出して下さい 。

（ただし、それ以前に提出する場合は、

メールで願います。

アドレス：　fwhy6454@mb.infoweb.ne.jp　）

7. 参考書ほか

- 大木幹雄「ﾃﾞｰﾀﾍﾞｰｽ設計の 基礎」（日本理工出版会）
- 小野哲ほか「まるごと図解、SQLがわかる」（技術評論社）
- 宮坂雅輝「SQLハンドブック」（ソフトバンク社）
- ライアンほか「SQLﾌﾟﾛｸﾞﾗﾐﾝｸﾞ入門」（ｿﾌﾄﾊﾞﾝｸ）
- 村上毅ほか「MySQL活用ガイド」（秀和システム）
- 松崎為豁「データベースの 基礎の 基礎」（ソフトバンク）
- 山田精一「Oracleのデータベース」（翔泳社）
- http://www.ann.hi-ho.ne.jp/hirok/sql/index.html
- http://www.rfs.jp/sitebuilder/sql/