データの検索機構

授業計画

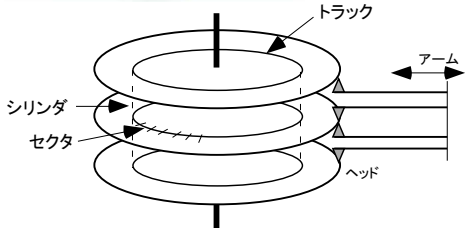
第1回　ガイダンス・データベースの基本概念
第2回　データモデル
第3回　関係代数
第4回　データベース設計
第5回　リレーションの正規化
第6回　中間まとめ
第7回　関係データベース言語（SQL1）
第8回　関係データベース言語（SQL2）
第9回　計算機実習
第10回　データの検索機構　MySQL実習
第11回　トランザクション管理　MySQL実習
第12回　障害回復　MySQL実習
第13回　分散データベース　MySQL実習
第14回　期末まとめ
第15回　応用技術と将来動向　MySQL実習

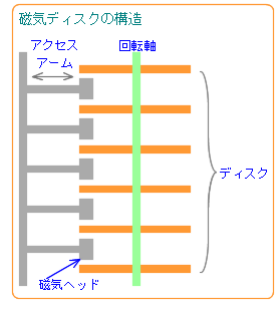
講義内容

- 磁気ディスク装置
- インデックス
  - Ｂ＋木インデックス
  - ハッシュインデックス
  - ビットマップインデックス
- テーブルのアクセス方法
- テーブルの結合方法
  - ネストループ結合
  - マージ結合
  - ハッシュ結合

１．磁気ディスク装置

■磁気ディスク装置





８．データの検索機構

１．磁気ディスク装置

■磁気ディスク装置

トラック：

・円盤上の記憶域は，同心円状のトラック（track）に分割され，その上にデータが記憶される。

・トラックには，外周のトラック0から内側に向かって順番に番号がふられている。

・1インチ当たりのトラック数を**トラック密度**とよぶ。

シリンダ：

・複数のアクセスアームは同時に移動し，常に同じ半径のトラックにアクセスする。

・1回のアクセスで記憶面の数だけのトラックを同時にアクセスすることになるため，このトラックの集合を**シリンダ**とよぶ。

８．データの検索機構

１．磁気ディスク装置

■磁気ディスク装置

セクタ：

・トラックを放射状に等分した**記憶単位**を**セクタ**とよぶ。

・セクタ容量は記憶装置内で一定であるので，円盤の中心部の記憶密度は高密度に，外周部は低密度になる。

・最近では，トラックを幾つかのゾーンに分けて，外側のゾーン程セクタ数を多くするゾーン記憶方式が採用されている。

8．データの検索機構

1．磁気ディスク装置

■磁気ディスク装置の容量計算

$$\text{容量} = (\text{セクタ長}) \times \begin{pmatrix}\text{1トラック当たり}\\ \text{のセクタ数}\end{pmatrix} \times \begin{pmatrix}\text{1シリンダ当たり}\\ \text{のトラック数}\end{pmatrix} \times (\text{シリンダ数})$$

（例1）シリンダ数=800，1シリンダ当たりのトラック数=19 トラック，1トラック当たりの記憶容量=20,000バイトの磁気ディスク装置の容量を求めよ。

$20{,}000 \times 19 \times 800 = 304{,}000{,}00\ \text{Byte} = 304\ \text{MByte}$

8．データの検索機構

1．磁気ディスク装置

■磁気ディスク装置の容量計算

$$\text{容量} = (\text{セクタ長}) \times \begin{pmatrix}\text{1トラック当たり}\\ \text{のセクタ数}\end{pmatrix} \times \begin{pmatrix}\text{1シリンダ当たり}\\ \text{のトラック数}\end{pmatrix} \times (\text{シリンダ数})$$

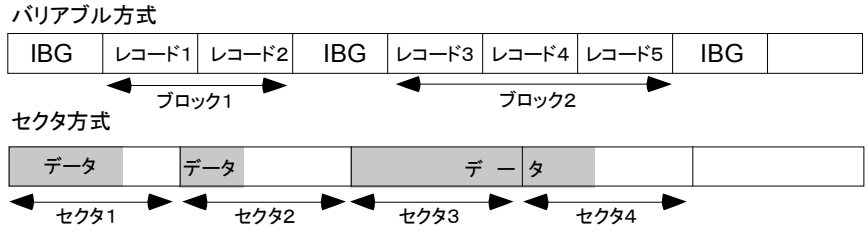
（例2）シリンダ数=2,000，1シリンダ当たりのトラック数=15トラック，1トラック当たりのセクタ数=32セクタ，1セク当たりの記憶容量=500バイトの磁気ディスク装置の容量を求めよ。

$500 \times 32 \times 15 \times 2{,}000 = 480{,}000{,}000\ \text{Byte} = 480\ \text{MByte}$

8．データの検索機構

1．磁気ディスク装置

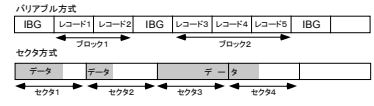
■磁気ディスク装置の記憶方式



8．データの検索機構

1．磁気ディスク装置

■磁気ディスク装置の記憶方式



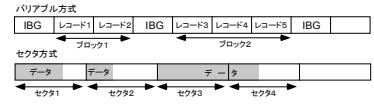
バリアブル方式：

- バリアブル方式は，データの読み書きを**ブロック単位**で行う。
- ブロックは複数の項目を1つにまとめたレコードの集合で，1ブロックに含まれるレコード数をブロック化因数とよぶ。
- 一般に，1トラックに複数のブロックを記憶し，ブロックとブロックの間には，**IBG（ブロック間ギャップ）**がある。

8．データの検索機構

1．磁気ディスク装置

■磁気ディスク装置の記憶方式



バリアブル方式：

・セクタ方式は，データの読み書きを**セクタ単位**で行う。

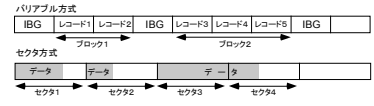
・1セクタに収まらないデータは複数のセクタにまたがるが，1セクタに複数のデータを記憶することはできない。

8．データの検索機構

1．磁気ディスク装置

■磁気ディスク装置の記憶方式

保存可能レコード数の計算



（例）1シリンダ当たりのトラック数=19トラック，1トラック当たりの記憶容量=20,000バイト，ブロック化因数=6，IBG＝250バイト，1レコード=600バイト，レコード数=15,000件の磁気ディスク装置に必要なシリンダ数を求めよ。

1ブロックの容量：600×6×250 = 3,850 Byte
1トラックに保存できるブロック数：20,000÷3,850 = 5.194…=5ブロック，
1トラックに保存できるレコード数：5×6 = 30 レコード，
1シリンダに保存できるレコード：19×30 = 570レコード
必要なシリンダ数：15,000÷570 = 26.31…　27シリンダ

８．データの検索機構

１．磁気ディスク装置

■磁気ディスク装置のアクセク時間

アクセス時間 = シーク時間 + サーチ時間 + データ転送時間

シーク時間： 磁気ヘッドを目的のトラック上まで移動させる時間を**シーク時間**または**位置決め時間**という。

サーチ時間： 磁気ヘッドが目的のトラック上に来た時，回転するトラック上のデータの先頭位置が磁気ヘッドの真下にあれば，待ち時間0でアクセスできるが，通過直後であれば1回転するのを待つ必要がある。

したがって，両者の平均をとり1/2回転を**サーチ時間**または**回転待ち時間**という。サーチ時間は，ディスクの回転速度から求めることができる。

８．データの検索機構

１．磁気ディスク装置

■磁気ディスク装置のアクセク時間

アクセス時間 = シーク時間 + サーチ時間 + データ転送時間

データ転送時間： **データ転送時間**は，回転速度とデータ容量をもとに算出される。ディスクの1回転時間が1トラックのデータ転送時間に相当する。

１．磁気ディスク装置

■磁気ディスク装置のアクセク時間

（例）平均位置決め時間=20 msec，1トラック当たりの記憶容量=20,000バイト，1分間当たりの回転数=3,000 rpmの磁気ディスクから，5,000バイトのデータのアクセス時間を求めよ。

（解答） **アクセス時間 = シーク時間 + サーチ時間 + データ転送時間**

シーク時間： 20 ms
サーチ時間 = (1/2)回転に要する時間
1 min =60 s = 60,000 msで3,000回転
→ 1回転に要する時間=60,000/3,000=20 ms
(1/2)回転 → 10 ms
データ転送時間 = 1トラックのデータを1回転で転送できるので，
20,000/20 = 1,000 ﾊﾞｲﾄ/ms
5,000ﾊﾞｲﾄを転送するには，5,000/1,000=5 ms

アクセス時間： 20 + 10 + 5 = 35 ミリ秒

２．インデックス

■**インデックス**(index)：

- **物理ファイル**として定義されたデータを効率的に検索したり，結合演算させたりする仕組みのこと。
- インデックスを設定しない場合には，検索処理や結合処理や結合演算をデータの物理順序などで逐次アクセスすることになり，データ量が多くなると検索効率が低下する。
- インデックスを設定すると，一般に**検索効率が向上**する。ただし，更新アクセスを行う場合は，データだけでなくインデックスも更新する必要がある。

8．データの検索機構

2．インデックス

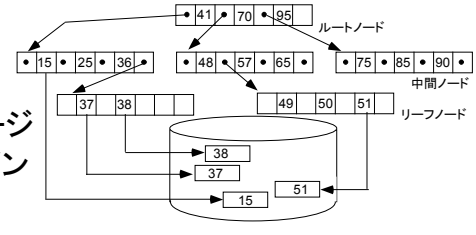
■**インデックス**(index):

- インデックスを設定しても更新効率が向上するとは限らない。
- データ量が少ないときは，インデックスを設定しないほうが良い場合がある。
- インデックスは，キー属性だけでなく非キー属性にも設定できる。
- 検索条件やデータ量，処理条件などを考慮し，インデックスを設定すべきかどうかを検討する必要がある。

8．データの検索機構

2．インデックス

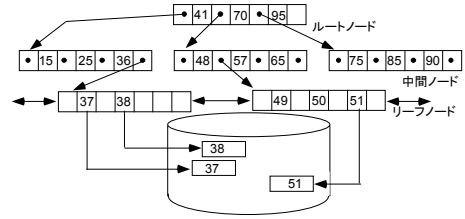
■**B木インデックス**



- 多分岐の木構造である。
- 節(ノード)に相当する各ページが枝(リンク)に相当するポインタで接続される。
- 最上位のページが**ルートノード**で，下位の複数のノードへのポイントを持つ。
- **中間ノード**は，さらに下位の複数のノードへのポインタを持ち，最下位のノードは**リーフノード**である。
- B木インデックスの木構造は**バランス木**とよばれ，行の追加，変更，削除の場合，すべての**リーフページの深さが同じ**になるように行われる。

2．インデックス

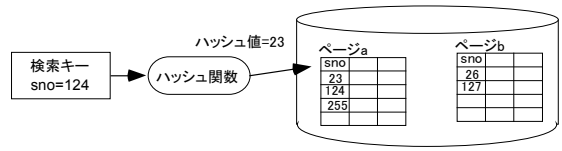
■B⁺木インデックス



- DBMSで最も広く一般的に提供されているインデックスがB⁺木インデックスである。
- B木インデックスを改良したものである。
- B木インデックスでは，ルートノードからリーフノードまでの全てのノード中にインデックスエントリを保持しているのに対して，B⁺木インデックスでは全てのインデックスエントリをリーフノード中に保持し，リーフノード以外のルートノードと中間ノードは，下位のノードへのポインタのみを保持している。

2．インデックス

■ハッシュインデックス



- 1回のアクセスでレコードを取得できるようにしようというのが**ハッシュインデックス**である。
- ハッシュインデックスは，キー値に対してある関数を適用して，その結果得られた関数値を基にして行の格納位置（ROWID）を求めるインデックスである。
- キー値から格納位置を求めるための関数を**ハッシュ関数**といい，ハッシュ関数を用いてキー値を格納位置に変換することを**ハッシング**という。

２．インデックス

■ハッシュインデックス

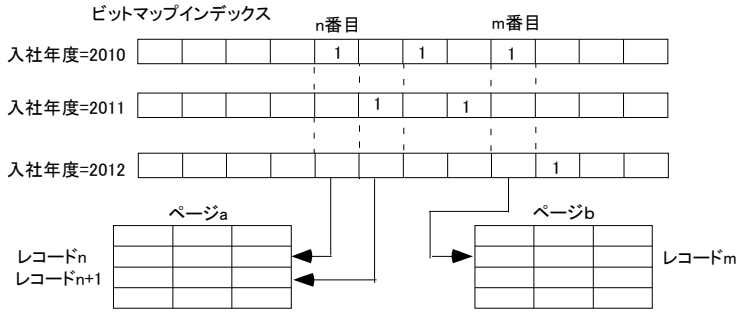
- ハッシュインデックスは，ハッシングを行うのでキーを構成する全ての列に対して等号条件（=）が指定された場合にしか利用できない。
- 範囲検索（連続したキー値を持つ行の集合の検索や，キー値の順序での検索）には利用できないという短所がある。
- ハッシングによって異なるキー値に対して，同一の関数値が求まることを衝突（コンフリクト）が発生するといい，それらのキー値を同義語（シノニム）という。
- 検索効率がデータベースの大きさに左右されない。
- シノニムの発生が増えるとアクセス効率が低下する。

８．データの検索機構

２．インデックス

■ビットマップインデックス

- ビットマップインデックスは，取り得る値の数が少ないフィールドに対して複雑な検索を行う場合に適しているインデックスの手法である。



2．インデックス

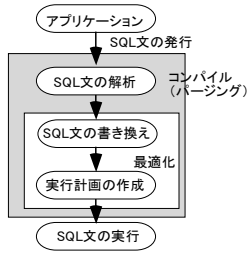
■ビットマップインデックス

- ビットマップインデックスは，リーフページ以外はB⁺木インデックスと同じ構造をもち，リーフページ内のインデックスエントリの各キー値に対して，複数の格納位置（ROWID）とその個数ではなく，各ビットが表中の各行に対応するビットマップ（ビット列）をもつ。
- ビットマップ中のONビットに対応する行がそのキー値をもつ行であることを示す。ビット演算を用いることにより，SQL文を効率的に処理できる場合がある。

3．テーブルのアクセス方法

■テーブルのアクセス方法

- RDBMSが問い合わせを実行する速度は，SQL文の書き方によって大きく異なる。
- 速いSQL文を書くためには，RDBMSがSQL文を内部でどのように処理しているのかを理解することが必要になる。
- SQLの実行過程



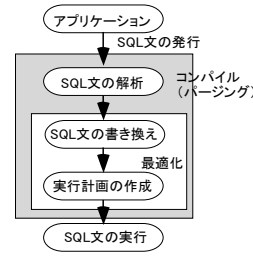
1. SQL文の解析
2. SQL文の書き換え
3. 実行計画の作成

3．テーブルのアクセス方法

■テーブルのアクセス方法

① **SQL文の解析**：SQL文が文法的に正しいかどうかをチェックし，選択，射影，結合といった処理がそれぞれどのように実施されるかという文の構造を把握する。

② **SQL文の書き換え**：SQL文をより高速に実行できるように書き換える。処理の手順や演算の種類を変更するといった書き換えを行う。

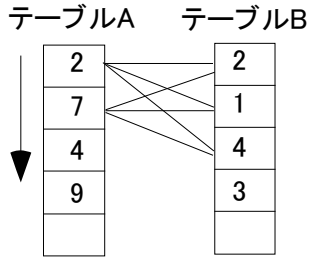


③**実行計画の作成**：処理命令を実行する手続きを何通りか作成したうえで，その中から最も効率の良いものを選択する。こうして完成したRDBMS内部の形式で表された一連の手続きを**実行計画**という。

4．テーブルの結合方法

- SQL文の高速化を実現するには結合処理の最適化を行う必要がある。
- 処理速度： ハッシュ結合＜マージ結合＜ネストループ結合

■**ネストループ結合**： 単純に2重ループでテーブルを結合する方法である。

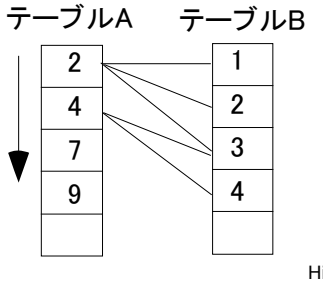


- 処理コストは2つのテーブルのレコード数の積に比例
- Bのレコードの検索にインデックスを利用することも可能
- インデックスが設定されていない小さなテーブルと，インデックスが設定されている大きなテーブルの二つを結合する場合が効果的

４．テーブルの結合方法

■**マージ結合**：ネストループ結合を改良した方法である。

- 2つのテーブルを，結合するフィールドについてあらかじめソートしておく。そして，両方のテーブルのレコードに対して持たせたポインタを，レコードの上から下へと順に走査させながらフィールドの値が一致するものを探索する方
- レコードの走査が1回で済むのが特徴

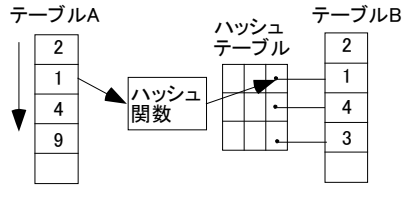


- まずA，Bのポインタの両方を先頭のレコードにおき，Aの2とBの1を比較する。Bのレコードはソート済みなので，もしBに値2を持つレコードがあるなら下方にあるはずである。
- 自分が相手よりも値が大きくなったら，相手のポインタを1つ進める。

４．テーブルの結合方法

■**ハッシュ結合**：ネストループ結合を改良した方法であると考えられる。

- ネストループ結合では，テーブルAの各レコードについて，テーブルBを全件走査する。
- この検索処理の部分にハッシュ法を使うことで高速化を図るのがハッシュ結合である。



- まず，結合するフィールドの値をキーとして，テーブルBに対するハッシュテーブルを作る。
- そしてテーブルAのレコードごとにフィールドの値が一致するものをハッシュテーブルから検索すれば，テーブルの結合が完成する。

まとめ

- 磁気ディスク装置
- インデックス
  - Ｂ＋木インデックス
  - ハッシュインデックス
  - ビットマップインデックス
- テーブルのアクセス方法
- テーブルの結合方法
  - ネストループ結合
  - マージ結合
  - ハッシュ結合