データベース管理システム
(Database Management System, DBMS)

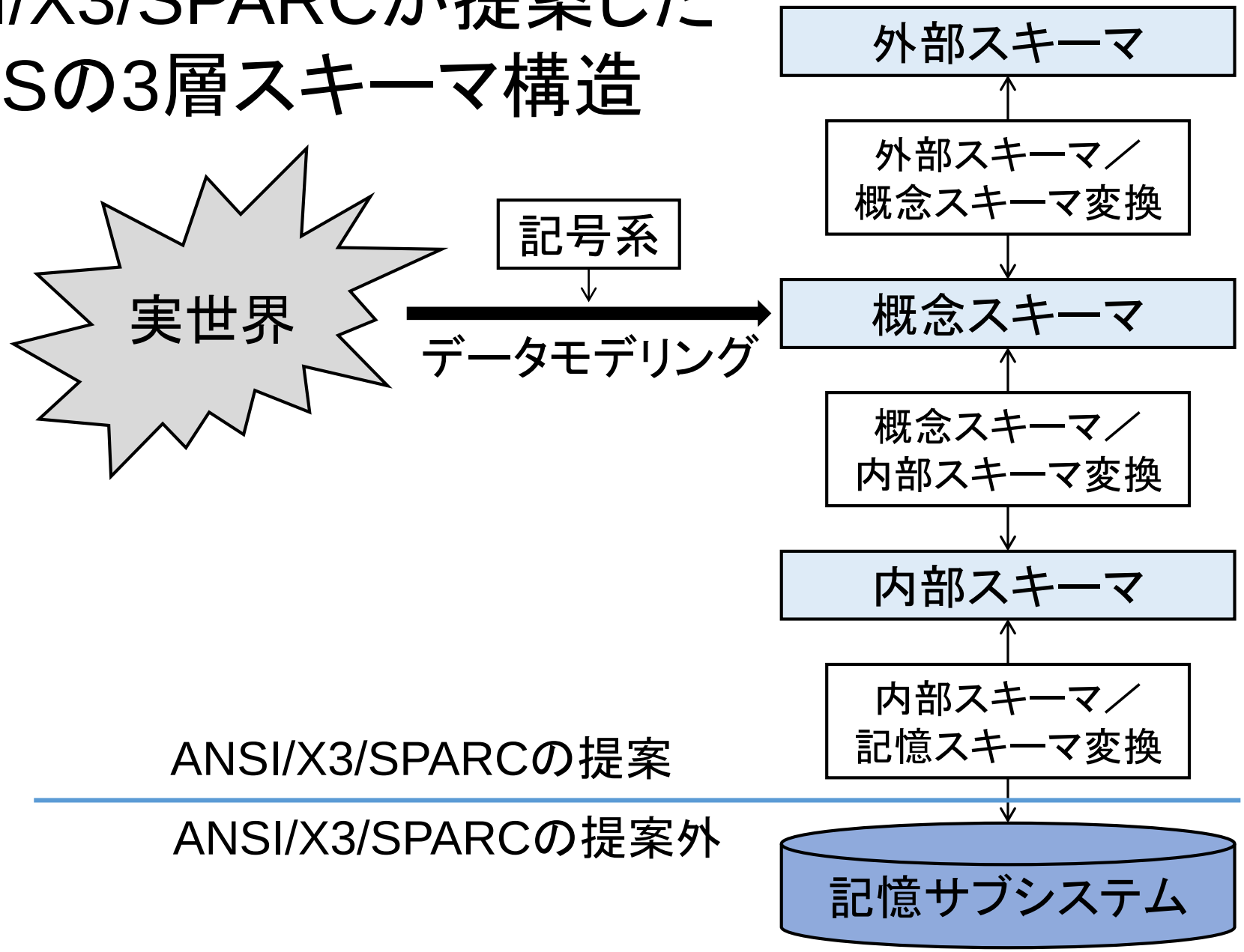
内容

- 3層スキーマ構造
- 標準アーキテクチャ
- 外部スキーマ
- 内部スキーマ

3層スキーマ

- 概念スキーマ
  - 構文的，意味的な構造
  - 実世界をモデリングした結果
- 内部スキーマ
  - コンピュータ上でのデータベースの見え方
  - 概念スキーマで定義された項目の，実際のコンピュータ上での実装
- 外部スキーマ
  - 外部からのデータベースの見え方
  - 概念スキーマ上に構築される，目的に応じたデータベース空間

ANSI/X3/SPARCが提案したDBMSの3層スキーマ構造



データ独立性

- 物理的データ独立性
  - 内部スキーマの変更があっても概念スキーマに影響しないときに物理的データ独立性がある
  - 実際にデータを格納しているファイルの構造を変えても，データベースの論理的構造の変更を必要としない
- 論理的データ独立性
  - 概念スキーマの変更があっても外部スキーマに影響しないときに論理的データ独立性がある
  - データベースの論理的構造を変えても，外部からの見え方は変わらない

標準アーキテクチャ（図9.1参照）

- ANSI/X3/SPARKの提案
  - 機能間のインタフェースの規定
- 構成要素
  - 管理者
  - 処理機能
  - データ辞書

管理者とは

- 組織体管理者
  - データベースの概念スキーマを定義する責任を負う
- データベース管理者
  - 概念スキーマをデータベースの内部スキーマに変換する責任を負う
- アプリケーションシステム管理者
  - 概念スキーマを基にしたデータベースのさまざまな外部スキーマを定義する責任を負う

処理機能

- 概念スキーマプロセッサ
  - 概念スキーマを構文的かつ意味的にチェックする
  - コンピュータが理解できる形式にコード化してできた概念スキーマ記述をデータ辞書に格納する
  - 他のスキーマに対するプロセッサも同様に規定される
- 外部／概念データベース変換
  - 外部スキーマと概念スキーマ間の対応関係を規定し，外部データベースをユーザに提供する
- 概念／内部データベース変換
  - 概念的なデータベースを (ファイルレベルの) 内部データベースとして実現する
- 内部データベース／内部記憶変換
  - 内部データベースと内部記憶間の対応関係を司る

データ辞書

- データベースのメタデータ
  - データベースに関する情報を保存する
  - 外部スキーマ，概念スキーマ，内部スキーマ，およびそれらの間の変換定義が入っている

外部スキーマの実現

- ビュー (view) を用いて定義する
- 概念スキーマとして格納されているリレーションを実リレーションと呼ぶ

ビューの例

(1) 選択ビュー

```
CREATE VIEW 低賃金社員 AS
  SELECT *
  FROM 社員
  WHERE 給与 < 20
```

低賃金社員，取引がリレーションのように見える

(2) 結合ビュー

```
CREATE VIEW 取引 AS
  SELECT X.仕入先,Y.納入先
  FROM 供給 X,需要 Y
  WHERE X.部品=Y.部品
```

前回の射影で分解したリレーションを，
自然結合するビューを用いることで，
分解前のリレーションとして見せている

ビューの更新可能性

ビューを指定して実リレーションを更新できるか

(1) 低賃金社員の給料を2倍にすることを考える

UPDATE 低賃金社員 SET 給与=給与*2　　は

UPDATE 社員 SET 給与=給与*2 WHERE 給与<20

に変換でき更新できる ⇒ 更新可能

(2) 取引からタップルを削除することを考える

この場合，　供給の表からのみ削除 ← 製造しなくなった

需要の表からのみ削除 ← 納入をしなくなった

両方から削除 ← 両方が発生した

の3つのケースが考えられ，それぞれで意味が異なるため，

削除の原因を特定できない　⇒ 更新可能でない

ビュー：取引

供給.仕入れ先	需要.納入先
S1	D1
S1	D2
S2	D1

(S1, D2) の削除を考える

射影の結果

射影

供給[部品=部品]需要

供給.仕入れ先	供給.部品	需要.部品	需要.納入先
S1	P1	P1	D1
S1	P2	P2	D2
S2	P1	P1	D1

等結合の結果

等結合

供給

仕入れ先	部品
S1	P1
S1	P2
S2	P1

需要

部品	納入先
P1	D1
P2	D2

実リレーション

ビュー：取引

供給.仕入れ先	需要.納入先
S1	D1
S1	D2
S2	D1

射影の結果

射影

供給[部品=部品]需要

供給.仕入れ先	供給.部品	需要.部品	需要.納入先
S1	P1	P1	D1
S1	P2	P2	D2
S2	P1	P1	D1

等結合の結果

等結合

供給

仕入れ先	部品
S1	P1
S1	P2
S2	P1

←削除

需要

部品	納入先
P1	D1
P2	D2

実リレーション

ビュー：取引

供給.仕入れ先	需要.納入先
S1	D1
S1	D2
S2	D1

射影の結果

射影

供給[部品=部品]需要

供給.仕入れ先	供給.部品	需要.部品	需要.納入先
S1	P1	P1	D1
S1	P2	P2	D2
S2	P1	P1	D1

等結合の結果

等結合

供給

仕入れ先	部品
S1	P1
S1	P2
S2	P1

需要

部品	納入先
P1	D1
P2	D2

削除→

実リレーション

ビュー：取引

供給.仕入れ先	需要.納入先
S1	D1
S1	D2
S2	D1

射影の結果

射影

供給[部品=部品]需要

供給.仕入れ先	供給.部品	需要.部品	需要.納入先
S1	P1	P1	D1
S1	P2	P2	D2
S2	P1	P1	D1

等結合の結果

等結合

供給

仕入れ先	部品
S1	P1
S1	P2
S2	P1

←削除→

需要

部品	納入先
P1	D1
P2	D2

実リレーション

内部スキーマの実現

- 内部スキーマはリレーションの実装モデルを表す

リレーションをファイルに対応させるためには

↓

リレーションの操作を
ファイルの操作への変換が必要

⇩

所望のタプルがファイルの
どこに位置するかを同定する技術が必要

用語説明

- ページ
  - タップルを格納する領域の外部記憶での単位
- レコード
  - 外部記憶でのタップルのこと
  - 属性値を持つフィールドと以下に示すような情報から構成される
    - ✓ レコード形式，大きさ
    - ✓ 可変長フィールドを持つ場合はフィールドの大きさ
    - ✓ 削除フラグ，空きフラグ
    - ✓ …

代表的な格納の方法

- ヒープファイル
- ハッシュファイル
- 索引付ファイル

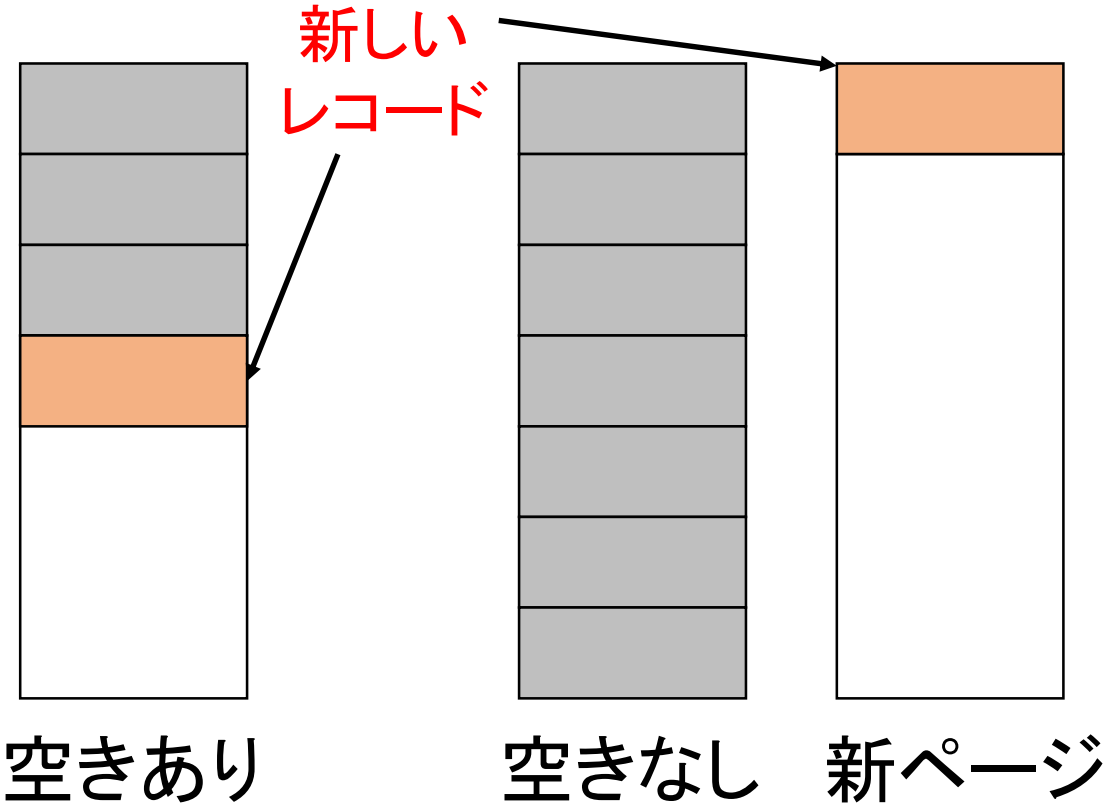
詳細は参考図書で

- B木
- B^+木

教科書で紹介されている

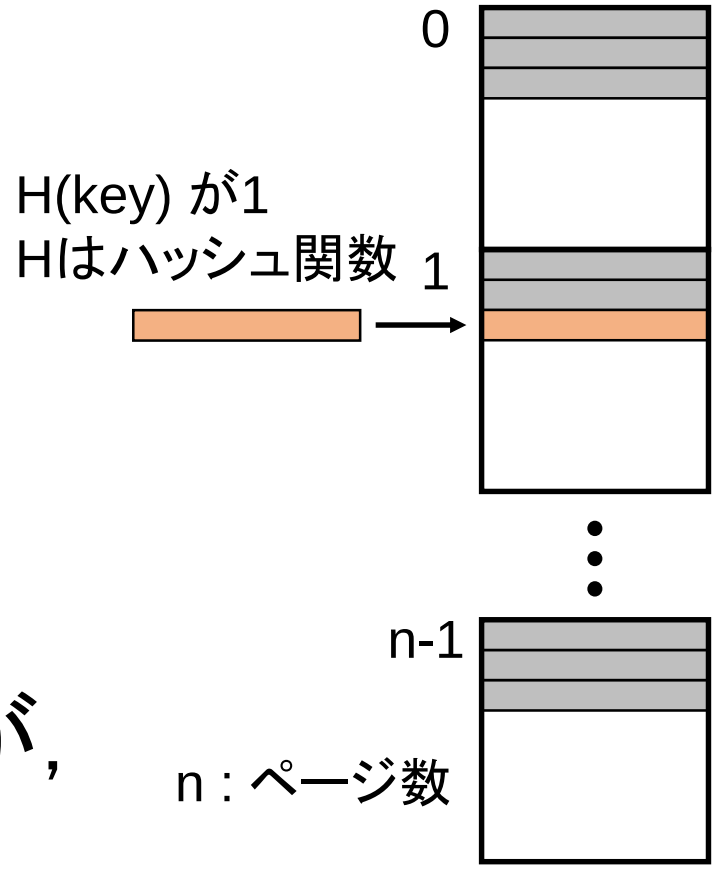
ヒープファイル

- ファイルを構成するページにレコードを順次格納する
  - 空きのあるページがあればそこに追加し，なければ新しいページを確保する
  - 格納効率は高くなる
- キーの値を用いたレコード検索に向かない
  - 検索効率が悪い



ハッシュファイル

- キーのハッシュ値を用いてレコードを格納する場所(ページ) を決める
  - ランダムアクセスを行う
- 該当するページに空きがなければ オーバフロー処理が必要である
  - 再ハッシュ，オーバフローページを置く方法などがある
- 一致するキーの検索には適しているが，範囲を指定する検索には向かない



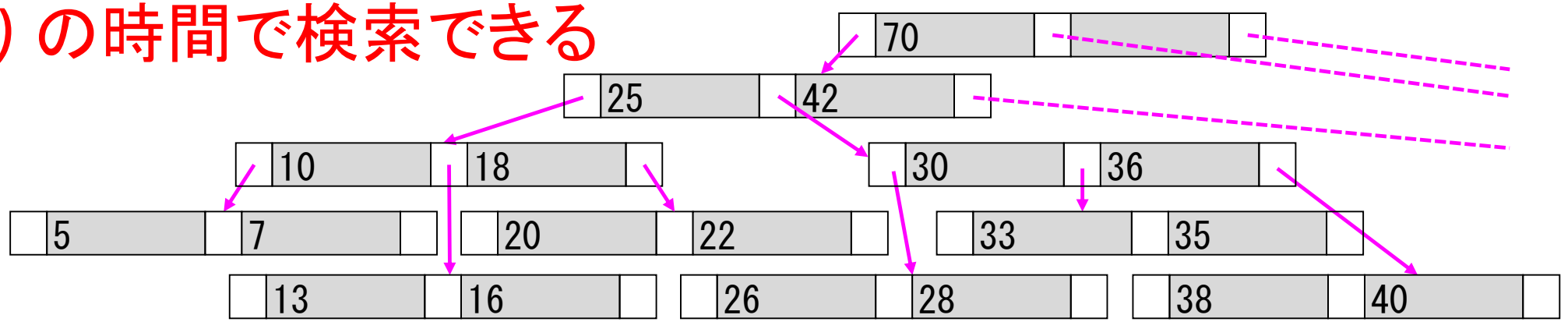
索引付ファイル

- データ部と索引部に分ける
  - データ部：キーの値順に格納 (キーの値でソート済み)
    キーの値の範囲などでの検索に適している
  - 索引部：データページの先頭レコードのキーの値と
    そのデータページへのポインタを組にして格納
- 新たなレコードを挿入する時に問題が生じる
  - 挿入すべきレコードのキーの値より大きなキーの値を持つレコード全てを後ろに移動する必要がある
  - 対策としては
    - ✓ 各ページに空き領域を残すことで，レコードの移動をページ内に抑える
    - ✓ オーバフローページを用いる

    などがある

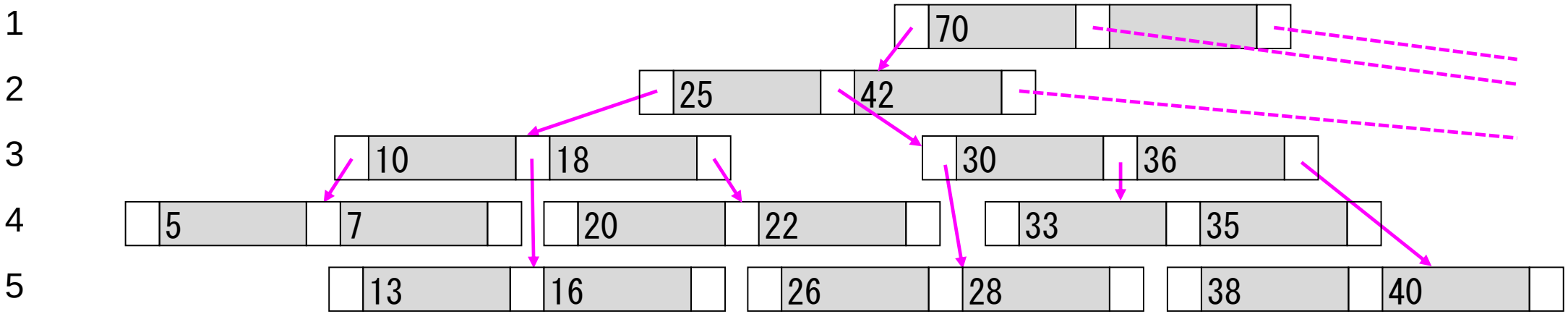
B木 (B-Tree)

- 木構造で管理する
  - 1つのノード (ページ) はレコードと部分木へのポインタが交互に配置されている
  - あるレコードのキーの値は，左側のポインターで指される部分木中のレコードのキーの値よりも**大きく**，右側のポインターで指される部分木中のレコードのキーの値より小さい.
  - ルートからリーフまでの長さが全て同じ
- $O(\log n)$ の時間で検索できる

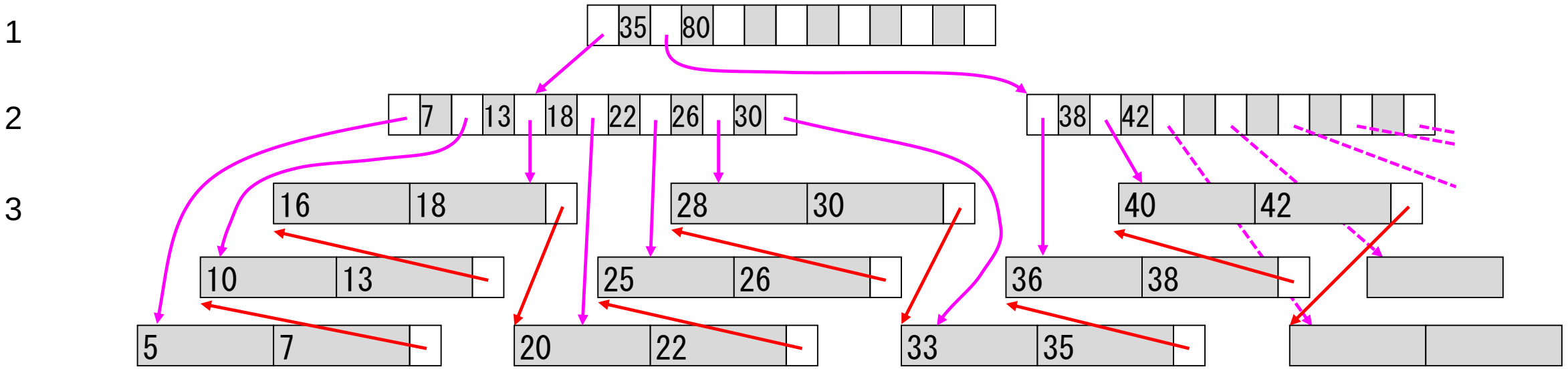


B^+木 (B^+−Tree)

- リーフ以外のノードにはレコードを持たない
  - レコードの代わりにキーの値を保持 (レコードより遙かに小さい)
    あるレコードのキーの値は，左側のポインターで指される部分木中のレコードのキーの値以上であり，右側のポインターで指される部分木中のレコードのキーの値より小さい.
  - 1つのノードの部分木へのポインタ数をB木よりも多くでき，木の高さを低く (検索回数を減らす) できる
- リーフノード間をリンクすることにより (次図の赤い矢印)，順アクセスを高速に行うことができる


1
2
3
4
5
70
25
42
10
18
30
36
5
7
20
22
33
35
13
16
26
28
38
40

B-Tree


1
2
3
35
80
7
13
18
22
26
30
38
42
16
18
28
30
40
42
10
13
25
26
36
38
5
7
20
22
33
35

B+-Tree

二次索引

- 索引付ファイル
  - キーの値とレコードの格納場所を直接結び付ける索引構造を持っている
  - 索引に用いられたキーでの検索は高速である
    - ✓ このようなキーを主索引 (primary index) と呼ぶ
- 主索引以外での検索も必要な場合がある
  - 主索引と異なるフィールド (二次索引, secondary index) の値とレコードの格納場所の組み合わせからなるファイル (インデックスファイル) を作成する
  - 指定したフィールド値を持つレコードが複数存在することがある