演習問題の解答

■第 1 章

問題 1　図 **1.4** を示し，それをきちんと説明できればよい．

問題 2　第 1 章，2 章，4 章を理解した上での総合的な問題ではあるが，(1)，(2)，(3) については，関数従属性と多値従属性に関する設問以外は，1.5 節，1.6 節を精読することで解答できる筈(はず)である．

問題 3　実体型 社員はリレーションスキーマ $\boldsymbol{R}_{\text{社員}}$(社員番号, 社員名, 所属, 給与) へ変換する．弱実体はその所有実体と対になり初めて同定できるので，弱実体型 子はリレーションスキーマ $\boldsymbol{R}_{\text{子}}$(社員番号, 名, 生年月日, 扶養認定日) へ変換する．ID 関連型 扶養は，単に社員番号を子に渡す役割をしているだけなので，リレーションスキーマに変換する必要はない．

問題 4　**問題 5**　1.2 節を精読して答えよ．

■第 2 章

問題 1　2.7 節を精読して答えよ．

問題 2　リレーションは集合，即ち異なるタップルの集まり，として定義されるから，(最悪でも）全属性のなす集合が候補キー，従って主キーとなる．

問題 3　全属性のなす集合が初めて候補キーとなる場合を考えると，大別して 2 つあり，1 つは全属性がお互いに意味的（=概念的）に独立している場合で，もう 1 つは，属性間に意味的階層関係がある場合である．前者の例は，リレーション 5W(Who, What, When, Where, Why) が考えられるし，後者の例としては，リレーション 部署 (会社名, 支店名, 部名, 課名, 係名) を挙げられる．

問題 4　各自，試みよ．空の意味は哲学的色彩も帯びて，実に多様である．

問題 5　図 **2.16** を参考にしつつ，各自試みよ．

問題 6　(1)

```
CREATE TABLE 学生(
       ...
       成績 CHAR(1)
       CHECK(成績 IN ('S','A','B','C','D','E'))
)
```

(2)

```
CREATE ASSERTION 売上増
CHECK ((SELECT SUM(売上高) FROM 決算)
>= (SELECT SUM(売上高) FROM 予算))
```

(3)

```
CREATE TRIGGER 部員数整合
AFTER DELETE ON 社員
UPDATE 部門
SET 部員数 = 部員数 - 1
WHERE 部門.部門番号 = 社員.所属
```

問題 7 **(a) の実現**：まず，ビュー 社員2 を次のように定義する．

```
CREATE VIEW 社員2(社員番号，社員名，所属部門，職位)
AS
SELECT 社員番号，社員名，所属部門，職位
FROM 社員
```

続いて，職位が一般社員である社員の社員番号を次のSELECT文で全て求める．

```
SELECT 社員番号
FROM 社員
WHERE 職位 = N'一般社員'
```

そこで，DBSA は，上記の SELECT 文で得られた各社員，例えば 007 に対して，次に示す権限を付与する GRANT 文を発行する．ここに，U007 は社員番号 007 の社員のユーザアカウントとする．

```
GRANT REFERENCES ON 社員2 TO U007;
```

(b) の実現：まず，職位が課長である社員の社員番号と所属部門を次の SELECT 文で全て求める．

```
SELECT 社員番号，所属部門
FROM 社員
WHERE 職位 = N'課長'
```

その結果，例えば (003, DB設計課) が得られたとする．そのとき，DB設計課に対してビュー DB設計課社員を次のように定義する．

```
CREATE VIEW DB設計課社員
AS
SELECT *
FROM 社員
WHERE 所属部門 = N'DB設計課'
```

続いて，課長である社員番号 003 の社員に対して，次の GRANT 文を発行する．ここに U003 は社員番号 003 の社員のユーザアカウントとする．

```
GRANT REFERENCES ON DB設計課社員 TO U003;
```

この操作を全ての課長に対して行う．

■第 3 章

問題 1 (1) (学生 [大学名 = 東都大])[学生名, 住所]

(2) ((学生 [住所 = 池袋])[大学名 = 東都大])[学生名]

(3) ((学生1[大学名 = 東都大])[住所 = 住所](学生2[大学名 = 東都大]))[学生1.学生名, 学生2.学生名]. ここに 学生1 = 学生2 = 学生 とおいた.

(4) ((アルバイト [学生名 = 学生名] 学生)[学生.大学名 = 東都大])[アルバイト.会社名]

(5) (アルバイト [学生名, 会社名]) ÷ ((学生 [大学名 = 東都大])[学生名])

(6) ((学生 [大学名 =東都大])[学生名]) − アルバイト [学生名]

(7) ((学生 [学生名 = 学生名] アルバイト)[アルバイト.会社名 = ときめき商事])[学生.学生名, 学生.大学名]

(8) (((学生 [学生名 = 学生名] アルバイト)[アルバイト.会社名 = 会社名] 会社)[T.学生.住所 = 会社.所在地])[T.学生.学生名, T.学生.住所]. ここに, T = 学生 [学生名 = 学生名] アルバイトとおいた.

(9) ((((学生 [学生名 = 学生名] アルバイト)[アルバイト.会社名 = 会社名] 会社)[T.学生.住所 ≠ 会社.所在地])[T.学生.大学名 = 東都大])[T.学生.学生名]. ここに, T = (学生 [学生名 = 学生名] アルバイト) とおいた.

(10) (((アルバイト [会社名 = 会社名] 会社)[会社.所在地 = 新宿])[アルバイト.給与 ≧ 50])[アルバイト.学生名]

問題 2 (1) (製品 [単価 ≧ 100])[製品番号, 製品名]

(2) ((製品 [製品名 = ステレオ])[製品.製品番号 = 工場.製品番号](工場 [生産量 ≧ 10])) [工場.工場番号, 工場.所在地]

(3) ((((在庫 [所在地 = 札幌])[在庫量 = 5])[在庫.製品番号 = 製品.製品番号] 製品) [T.在庫.製品番号 = 工場.製品番号] 工場)[T.製品.製品名, 工場.工場番号]. ここに, T =((在庫 [所在地 = 札幌])[在庫量 = 5])[在庫.製品番号 = 製品.製品番号] 製品 とした.

問題 3 $R[B = C]S = \{(1, 2, 2, 4)\}$, $R[B =_\omega C]S = \{(1, 2, —, 5), (2, —, 2, 4), (2, —, —, 5), (2, —, 3, 6)\}$

■第 4 章

問題 1 $X \to Y$ ならば, $R = R[X, Y] * R[X, Z]$ が成立するが, これは $X \twoheadrightarrow Y$ を意味する.

問題 2 (a) アームストロングの公理 A1 より $Y \cup Z \to Y$ である. これと $X \to Y \cup Z$ と公理 A3 より $X \to Y$.

(b) $X \to Y$ と公理 A2 より, $X \to Y \cup X$. $X \to Z$ と 公理 A2 より $Y \cup X \to Y \cup Z$. $X \to Y \cup X$ と $Y \cup X \to Y \cup Z$ と公理 A3 より $X \to Y \cup Z$.

(c) $X \to Y$ と公理 A2 より, $W \cup X \to W \cup Y$. これと $W \cup Y \to Z$ と公理 A3 より $W \cup X \to Z$.

問題 3

X	Y	Z
x	y	z
x	y'	z

は, $X \to Z$ と $Y \to Z$ を満たすインスタンスであるが, $X \to Y$ は満たさない.

問題 4 (1) X^+ を求めるアルゴリズム（p.99）より, AB^+ を計算すると, $AB^+ = ABC \subset ABCDE$ なので, 候補キーにはならない. 一方, $ABD^+ = ABCDE$ であり, $AD^+ = AD$,

$BD^+ = BD$ なので，$\{A, B, D\}$ は候補キーである．

(2)　まず単一属性が候補キーとなり得るか，アームストロングの公理系に従い試す．$A^+ = A$, $B^+ = ABCDE$, $C^+ = CD$, $D^+ = D$, $E^+ = EA$ である．従って，唯一の候補キー B が主キーとなる．F 中，$C \to D$ と $E \to A$ は主キーが決める関数従属性ではない．そこで，R を $E \to A$ を使って $R[EA]$ と $R[BCDE]$ に情報無損失分解し，$R[BCDE]$ を $R[CD]$ と $R[BCE]$，更に，$R[BCE]$ を $R[BC]$ と $R[BE]$ に分解できる．つまり，R は $\{R[EA], R[CD], R[BC], R[BE]\}$ に分解できる．R を最初 $C \to D$ を使って分解した場合も同じ結果を得る．

問題 5　$\boldsymbol{R}(A, B)$ を 2 項リレーションスキーマとする．$\boldsymbol{R}$ の自明でない関数従属性は，あるとすれば $A \to B$ か $B \to A$ である．もし $A \to B$ が成り立っているとすれば，A は候補キーであり，スーパキーでもある．A と B は対照的であるから，定義 4.10(2) より $\boldsymbol{R}$ は BCNF である．$A \to B$ も $B \to A$ も成り立たなければ，AB が $\boldsymbol{R}$ の主キーで，この場合考えられる関数従属性は全て自明となるから，定義 4.14(1) より，$\boldsymbol{R}$ は BCNF である．

問題 6　(1)　いかなる単一の属性 A も $A^+ = \Omega_{研修}$ とならないので候補キーにはなり得ない．続けて 2 つの属性の組合せの閉包を計算すると，{社員番号, 科目番号}+ $= \Omega_{研修}$ となり候補キーであることが分かる．他の組合せは候補キーとならないので，主キーである．

(2)

1. {社員番号, 科目番号} → 得点　　（所与）
2. {社員番号, 科目番号} → {科目番号, 得点}　　（1. と添加率）
3. {科目番号, 得点} → 評価　　（所与）
4. {社員番号, 科目番号} → 評価　　（2. と 3. と推移律）

(3)　全ての属性のドメインは単純であるので，研修は第 1 正規形である．では，第 2 正規形かどうか調べる．しかし，例えば，社員番号 → 社員名 なので，非キー属性 社員名は主キーである {社員番号, 科目番号} に完全関数従属していない．従って，第 2 正規形ではない．故(ゆえ)に，研修は第 1 正規形である．

(4)　(a)　**挿入時異状の発生**：{社員番号, 科目番号} が主キーなので，例えば新しい科目の科目番号と科目名が決まっても，それを受講する社員が決まらないとデータを挿入できない．

(b)　**削除時異状の発生**：ボンドが CG の受講を取りやめると，科目番号 C002 のデータを残せない．

(c)　**修正時異状の発生**：ボンドが CG の受講をオートマトンに変更すると，CG のデータを残せない．データベースの科目名がデータベース基礎に変更になったら，タップルを 2 本修正しなければならない．

(5)　研修を情報無損失分解する．その結果，次の 4 つの射影が得られる：研修 [社員番号, 社員名]，研修 [科目番号, 科目名]，研修 [社員番号, 科目番号, 得点]，研修 [社員番号, 科目番号, 評価]

問題 7　(1) と (2) は 2.6 節を，(3) と (4) は第 4 章を精読して，各自解答を試みよ．

問題 8　(1)　4.11 節のリレーション SCT(学生名, 科目名, 教員名) を拡張して，リレーション SCTA(学生名, 科目名, 教員名, 教員住所) と，SCTA 上の関数従属性の集合 {{学生名, 科目名} → 教員名, 教員名 → 科目名, 教員名 → 教員住所} を考える．SCTA は定義 4.11 では第 2 正規形になり得ないが，緩い定義では第 2 正規形になる．なぜならば，SCTA には（SCT の場

合と同じく）候補キーが 2 つある：{学生名, 科目名} と {学生名, 教員名}．従って，SCTA の非キー属性は教員住所となる．しかし，教員住所は教員名に関数従属しているから，定義 4.11 に従えば，{学生名, 教員名} に完全関数従属していないため，SCTA は第 2 正規形になり得ない．しかし，もし {学生名，科目名} を主キーとして「緩い」定義に従えば，SCTA は第 2 正規形になる．その結果，第 2 正規形の定義に抵触することなく，SCTA に教員名に加えて教員住所を同時に格納できる．

(2)　教員名と同時に教員住所も格納を許される第 2 正規形のリレーション SCTA(学生名, 科目名, 教員名, 教員住所) ではあるが，例えば，新任教員名（T）とその教員住所（A）のデータが分かったからといって，それらを SCTA に格納できる訳ではない．なぜならば，そのために $(—, —, T, A)$ を挿入したいが，これは，{学生名, 科目名} が主キーだからキー制約に抵触して許されない．

(3)　(1)，(2) の解答を参考にして，各自考えよ．

問題 9　履修登録はあくまで登録関係を記録するためのリレーションなので，リレーションスキーマ ***履修登録*** に加えて，リレーションスキーマ ***学生*** (学籍番号) とリレーションスキーマ ***科目*** (科目番号) を作り，そこに格納させ，リレーションスキーマ ***履修登録*** の定義に，FOREING KEY 学籍番号 REFERENCES 学生 (学籍番号) と FOREING KEY 科目番号 REFERENCES 科目 (科目番号) を加える．

■第 5 章

問題 1　(1)

```
SELECT 学生名, 住所
FROM 学生
WHERE 大学名 = N'東都大'
```

(2)

```
SELECT 学生名
FROM 学生
WHERE 住所 = '池袋'
AND 大学名 = N'東都大'
```

(3)

```
SELECT X.学生名, Y.学生名
FROM 学生 X, 学生 Y
WHERE X.大学名 = N'東都大'
  AND Y.大学名 = N'東都大'
  AND X.住所 = Y.住所
```

(4)

```
SELECT X.会社名
FROM アルバイト X, 学生 Y
WHERE X.学生名 = Y.学生名
  AND X.大学名 = N'東都大'
```

(5)

```
SELECT DISTINCT X.会社名
FROM アルバイト X
WHERE NOT EXISTS
      (SELECT * FROM 学生 Y
       WHERE Y.大学名 = N'東都大'
         AND NOT EXISTS
            (SELECT *
             FROM アルバイト Z
             WHERE Z.会社名 = X.会社名
               AND Z.学生名 = Y.学生名)
```

(6)

```
SELECT X.学生名
FROM 学生 X
WHERE X.大学名 = N'東都大'
  AND NOT EXISTS
      (SELECT *
       FROM アルバイト Y
       WHERE Y.学生名 = X.学生名)
```

(7)

```
SELECT X.学生名, X.大学名
FROM 学生 X, アルバイト Y
WHERE X.学生名 = Y.学生名
  AND Y.会社名 = N'ときめき商事'
```

(8)

```
SELECT X.学生名, X.住所
FROM 学生 X, アルバイト Y, 会社 Z
WHERE X.学生名 = Y.学生名
  AND Y.会社名 = Z.会社名
  AND X.住所 = Z.所在地
```

(9)

```
SELECT X.学生名
FROM 学生 X, アルバイト Y, 会社 Z
WHERE X.大学名 = N'東都大'
  AND X.学生名 = Y.学生名
  AND Y.会社名 = Z.会社名
  AND X.住所 <> Z.所在地
```

(10)

```
SELECT X.学生名
FROM 学生 X, アルバイト Y, 会社 Z
WHERE X.学生名 = Y.学生名
  AND Y.会社名 = Z.会社名
  AND Z.住所 = N'新宿'
  AND Y.給与 >= 50
```

問題 2 (1)

```
SELECT 製品番号, 製品名
FROM 製品
WHERE 単価 >= 100
```

(2)

```
SELECT Y.工場番号, Y.所在地
FROM 製品 X, 工場 Y
WHERE X.製品名 = N'ステレオ'
  AND Y.生産量 >= 10
  AND X.製品番号 = Y.製品番号
```

(3)

```
SELECT X.製品名, Y.工場番号
FROM 製品 X, 工場 Y, 在庫 Z
WHERE Z.所在地 = N'札幌'
  AND Z.在庫量 = 5
  AND Z.製品番号 = X.製品番号
  AND Z.製品番号 = Y.製品番号
```

問題 3 (1)

```
SELECT DISTINCT X.ワインバー名
FROM 給仕 X
WHERE X.ワイン銘柄 IN
  (SELECT Y.ワイン銘柄
   FROM 嗜好 Y
   WHERE Y.客名 = N'小杉あゆみ')
```

(2)

```
SELECT X.客名
FROM 客 X
WHERE X.ワインバー名 IN
  (SELECT Y.ワインバー名
   FROM 給仕 Y
   WHERE Y.ワイン銘柄 IN
         (SELECT Z.ワイン銘柄
          FROM 嗜好 Z
          WHERE Z.客名 = X.客名)
```

(3) ((給仕 [ワインバー名 = ワインバー名](客 [客名 = '小杉あゆみ']))[給仕.ワインバー名, 給仕.ワイン銘柄]) ÷ (嗜好 [客名 = '小杉あゆみ'])[ワイン銘柄]

問題 4 (1)

```
SELECT X.業者名, X.部門名, X.部品名
FROM 納入 X, 業者 Y, 部門 Z, 部品 W
WHERE X.業者番号 = Y.業者番号
  AND X.部門番号 = Z.部門番号
  AND X.商品番号 = W.商品番号
  AND X.単価 = W.定価
```

(2)

```
SELECT Z.部門番号, W.部門名
FROM 納入 Z, 部門 W
WHERE Z.業者番号 = 'S1'
  AND Z.部品番号 = 'P1'
  AND Z.部門番号 = W.部門番号
  AND Z.単価 <
  (SELECT X.単価
   FROM 納入 X, 部門 Y
   WHERE X.部門番号 = Y.部門番号
     AND X.業者番号 = 'S1'
     AND X.部品番号 = 'P1'
     AND Y.部門名 = 'DB')
```

(3) ((納品 ÷ (部品 [部品番号]))[部門番号 = 部門番号] 部門)[部門.部門番号, 部門.部門名]

問題 5 (1)

```
SELECT 社員番号, 社員名
FROM 社員
WHERE 給与 >
      (SELECT AVG(給与)
       FROM社員)
```

(2)

```
SELECT X.社員番号, X.社員名
FROM 社員 X, 社員 Y
WHERE X.上司 = Y.社員番号
  AND X.給与 > Y.給与
```

(3)

```
SELECT X.社員番号, X.社員名
FROM 社員 X, 社員 Y, 社員 Z
WHERE X.上司 = Y.社員番号
  AND Y.上司 = Z.社員番号
  AND X.給与 > Z.給与
```

(4)

```
SELECT X.社員番号, X.社員名
FROM 社員 X
WHERE X.社員番号 IN
  (SELECT X.社員番号
   FROM 社員 Y
   WHERE X.上司 = Y.社員番号
     AND Y.社員番号 IN
     (SELECT Y.社員番号
      FROM 社員 Z
      WHERE Y.上司 = Z.社員番号
        AND X.給与 > Z.給与)
```

問題 6 (1)

```
SELECT 受注番号
FROM 受注
WHERE 受注日 = 20161114
```

(2)

```
SELECT X.受注番号, X.受注日
FROM 受注 X, 得意先 Y, 営業担当者 Z
WHERE X.得意先番号 = Y.得意先番号
  AND Y.営業担当者番号 = Z.営業担当者番号
  AND Z.営業担当者名 = N'鈴木一郎'
```

(3)

```
SELECT 営業担当者名
FROM 営業担当者
EXCEPT
((SELECT 営業担当者名
  FROM 営業担当者
  WHERE 性別 = 'M'
    AND 生年月日 = (SELECT MIN(生年月日)
                    FROM 営業担当者
                    WHERE 性別 = 'M')
UNION
(SELECT 営業担当者名
 FROM 営業担当者
 WHERE 性別 = 'F'
   AND 生年月日 = (SELECT MIN(生年月日)
                   FROM 営業担当者
                   WHERE 性別 = 'F'))
```

問題 7 (1)

```
CREATE TABLE 注文(
注文番号 CHAR(6),
明細番号 SMALLINT,
商品番号 CHAR(6),
商品名 NCHAR VARYING(30),
価格 DEC(8),
注文日 DATE,
PRIMARY KEY(注文番号, 明細番号)
)
```

(2)

```
INSERT INTO 注文(注文番号, 明細番号, 商品番号, 商品名, 価格, 注文日)
VALUES('001213', 1, 'TV0077', N'テレビ', 150000, '2016-12-23')
```

(3)

```
SELECT 商品名, 価格
FROM 注文
WHERE 注文番号 = '002536'
```

(4)

```
SELECT 注文番号, SUM(価格)
FROM 注文
GROUP BY 注文番号
```

(5) リレーショナル代数表現をすると，注文 [注文番号, 明細番号, 商品番号]，注文 [商品番号, 商品名, 価格]，注文 [注文番号, 日付] の 3 つの射影に情報無損失分解できる．それぞれ，順に，リレーション 注文明細，商品，注文日とする．

(6)

```
SELECT Y.商品名, Y.価格
FROM 注文明細 X, 商品 Y
WHERE X.商品番号 = Y.商品番号
  AND X.注文番号 = '002536'
```

(7)

```
SELECT X.注文番号, SUM(Y.価格)
FROM 注文明細 X, 商品 Y
WHERE X.商品番号 = Y.商品番号
GROUP BY X.注文番号
```

■第 6 章

問題 1 (1)〜(3) 6.2 節を精読して答えよ．(4) 6.3 節を精読して答えよ．

問題 2 6.4 節を精読して答えよ．

■第 7 章

問題 1 (1)

```
CREATE VIEW 貧乏社員 AS
SELECT *
FROM 社員
WHERE 給与 < 20
```

(2)

```
SELECT *
FROM 貧乏社員
WHERE 所属 = 'K55'
```

(3)

```
SELECT *
FROM 社員
WHERE 給与 < 20
  AND 所属 = 'K55'
```

問題 2 (1) 右図参照

ビュー 受発注

受注者	発注者
R1	O1
R1	O2
R2	O1

(2) (a) **タップル削除時異状**：設問で与えたリレーションは，**図 7.2** に示したビュー 取引とリレーション名や属性名が異なるだけで同じである．従って，7.4 節の記述がそのままあてはまる．例えば，(R2, O1) の削除を考えよ．

(b) **タップル挿入時異状**：ビュー 受発注にタップル (R, O) を挿入したいが，R と O を等結合する属性 物品の値が不明なので，挿入要求は却下される．勿論，適当な値は許されない．

(c) **タップル修正時異状**：リレーショナルデータベースの理論では，タップルの修正は，探索条件に合うタップル群を同定し，まずそれらを削除し，続いてそれらに修正を施したタップル群を挿入する操作と捉える．従って，まず (a) により異状が発生するから，修正要求は却下される．例えば，(R2, O1) を (R′, O′) に修正しようとしても，受け付けられない．

問題 3 例えば，リレーション 社員(社員番号,社員名,所属,給与)が定義されているとき，ビュー K55社員を次のように定義する：

```
CREATE VIEW K55社員 AS
SELECT *
FROM 社員
WHERE 所属 = 'K55'
```

このビューは，いわゆる選択ビューなので，SQL-92 で更新可能とされるビューである．

そこで，SQL ではビューに対しても権限付与ができるので，DBSA が K55 の部長（部長のユーザアカウントを U100 とする）にカラム 給与のみ更新権を与えたとする：

```
GRANT UPDATE ON K55 社員(給与) TO U100;
```

そうすると，部長は他部門の社員の給与を覗き見ることはできないが，自部門の部員の給与を更新できる（権限付与は 2.11 節参照のこと）．

■第 8 章

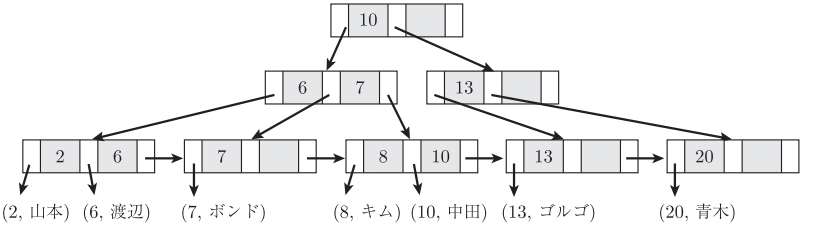
問題 1 キーと探索キー（8.5 節），順序キー（8.6 節），ソートキー（8.6 節脚注），ハッシュキー（8.8 節）を精読して，各自答えよ．

問題 2 1 次インデックスと 2 次インデックスとクラスタリングインデックス（8.6.1 項），密集インデックスと点在インデックス（8.6.1 項），多段インデックスと ISAM（8.6.2 項），B 木と B^+ 木（8.7 節），ハッシュインデックス（8.8.2 項），マルチキーインデックス（9.4.2 項），クラスタードインデックスと非クラスタードインデックス（9.4.2 項），を精読して，各自答えよ．

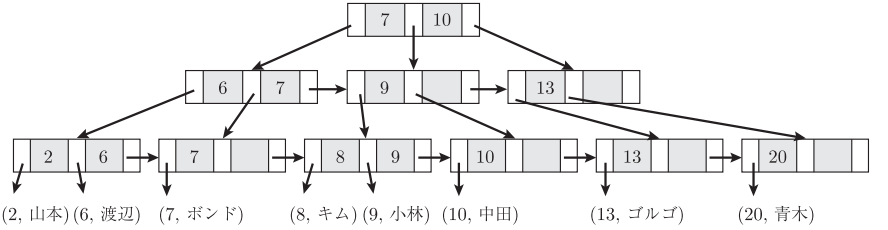
問題 3 まず，レコード (6, 渡辺) を挿入した時点での B^+ 木と，(9, 小林) を挿入し終わった時点での B^+ 木を次頁に示す．

問題 4 でき上がった B^+ 木は，8.7.2 項で示している B^+ 木でのレコードの挿入アルゴリズムで分かるように，挿入するレコードの順番が異なれば，でき上がる木構造も異なる．しかしながら，葉では，挿入したレコードは挿入順とは無関係に，探索キー値の昇順でアクセスできる．

問題 5 ハッシュ関数が $h(k) = k \pmod 2$ なので，本来バケツ B_0 に入るべきレコードは {(20, 青木),(2, 山本),(10, 中田),(8, キム)}，バケツ B_1 に入るべきレコードは {(7, ボンド),(13, ゴルゴ)} である．バケツの大きさは 2 レコード分なので，バケツ B_0 には溢れバケツ B_{overflow}



(g) レコード (6, 渡辺) を挿入し終わった時点の B^+ 木

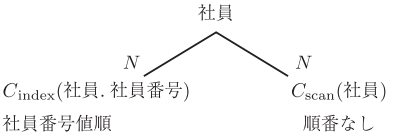


(h) レコード (9, 小林) を挿入し終わった時点の B^+ 木

を設けて，例えば，B_0 に {(20, 青木),(2, 山本)}，B_{overflow} に {(10, 中田),(8, キム)} を格納する．検索は，社員番号をランダムに与えるということなので，$((1 \times 4) + (2 \times 2)) \div 6 = 1.3$ 杯である．

■第 9 章

問題 1 最適化器はこの質問を処理するための最少の質問処理コストを推定する必要がある．リレーション 社員のアクセスパスは 2 つである：1 つはファイルスキャンで，もう 1 つは属性 社員番号上の B^+ 木である．従って，リレーション 社員に対する探索木は次のようである．



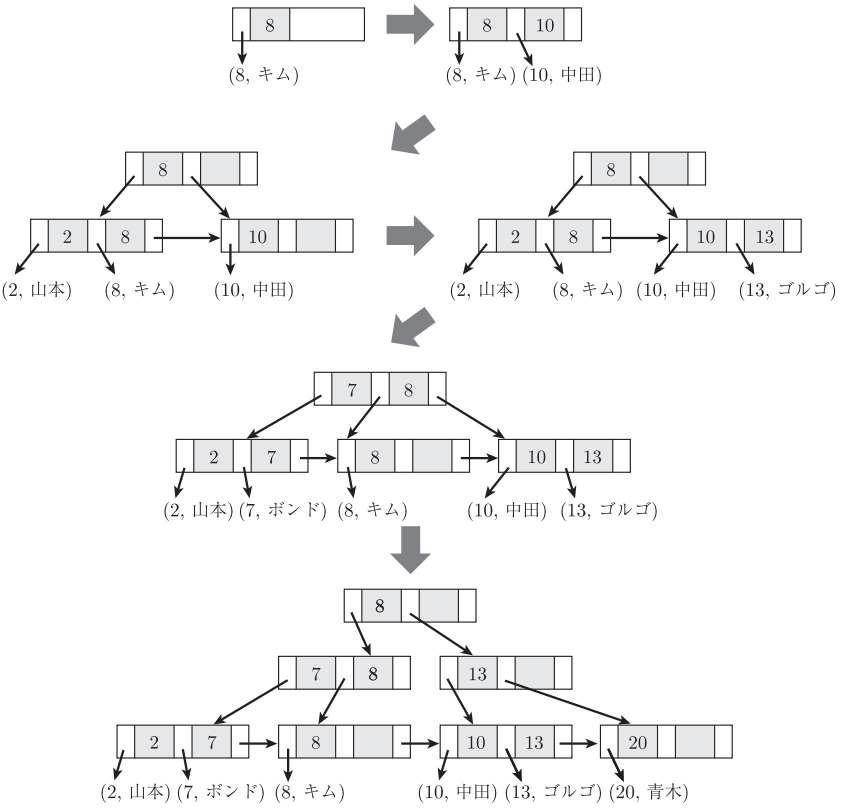
N は共に，探索条件 所属 <> 'K55' を適用し終わって得られる結果リレーションのタップル数である．従って，最適化器はコスト C_{index}(社員.社員番号) と C_{scan}(社員) を推定し，安い方を選択して，それに従い実行プランを生成する．

そこで，TCARD(社員)：社員のデータページの総数，NINDX(X社員番号)：インデックス X社員番号のインデックスページの総数，NCARD(社員)：社員の総レコード数とする．t_c は結果リレーションを生成するために社員のタップル 1 本 1 本に対して，探索条件 所属 <> 'K55' をチェックする時間を含んでいる．

$$C_{\text{scan}}(\text{社員}) = \text{TCARD}(\text{社員}) + w \times t_c$$

- インデックス X社員番号がクラスタードインデックスの場合

$$C_{\text{index}}(\text{社員.社員番号}) = \text{NINDX}(\text{社員番号}) + \text{TCARD}(\text{社員}) + w \times t_c$$



- インデックス X社員番号が非クラスタードインデックスの場合

$$C_{\text{index}}(\text{社員.社員番号}) = \text{NINDX}(X\text{社員番号}) + \text{NCARD}(\text{社員}) + w \times t_c$$

ここに，探索条件は 所属 <> 'K55' のため，社員番号上での比較ではないので，選択係数は 1 であることに注意する．以上を比較すると，インデックス X社員番号がクラスタード，非クラスタード，何れ(いず)の場合も，ファイルスキャンで実行プランを生成する方が，安い．

問題 2　リレーションの結合は入れ子型ループ結合法で行うので，次の 2 つの場合分けがある：

場合 1：R をアウタリレーションにして，(R, S) を実行

場合 2：S をアウタリレーションにして，(S, R) を実行

それぞれの場合について，更に詳しく質問処理を見ていく．

ここで，処理する SQL 文の探索条件に「$R.B = 10$」という条件が付いていることに注意する．

場合 1 のとき，考えられる実行プランは，

(1-1)　$R * S$ を実行して，その結果に，選択 $[R.B = 10]$ を施す．

(1-2)　$R[R.B = 10]$ を最初に行い，続いて $(R[R.B = 10], S)$ を行う．

場合 2 のとき，考えられる実行プランは，

(2-1)　$S * R$ を実行して，その結果に，選択 $[R.B = 10]$ を施す．

(2-2)　$R[R.B = 10]$ を最初に行い，続いて $(S, R[R.B = 10])$ を行う．

さて，(1-1) と (1-2) を比較すると，9.5 節 (1) に書いてある「発見的手法」の理由により，(1-1) は棄却される．

(2-1) と (2-2) を比較すると，同様の理由により，(2-1) は棄却される．更に，(2-2) は，選択演算の結果得られる $R[R.B = 10]$ の属性 B 上にはインデックスは定義されていないので，9.3.3 項 (1) に書いてある通り，入れ子型ループ結合法で結合をとる場合，インナリレーションの結合属性上にはインデックスが張られていることが必要であるが，その条件に適合しないので，このプランも破棄される．

従って，与えられた結合の実行プランは (1-2) のみとなる．

さて，続いて，実行プラン (1-2) の処理コスト C を計算する．

$$C = C_1 + C_2$$

である．ここに，C_1 は $R[R.B = 10]$ を得るためのコスト，C_2 は $(R[R.B = 10], S)$ を行うコストである．

$$C_1 = (100 + 1000) \div 5000 = 1100 \div 5000 = 0.22$$

$$C_2 = N_1 \times C_{\text{index}}(XS.B)$$

ここで $R_1 = R[R.B = 10]$，N_1 を R_1 の総タップル数とする．計算により，$N_1 = 10000 \div 5000 = 2$.

$$N_1 \times C_{\text{index}}(XS.B) = 2 \times (2100 \div 5000) = 0.84$$

従って，$C = 0.22 + 0.84 = 1.06$.

問題 3　(1)　与えられた条件により，索引 1 は非クラスタードで，従ってデータページの読込みはランダムであるので，商品番号=100 を満たすタップルの数だけデータページを読まねばならない．このとき，絞込み（=選択係数）は 0.1% なので，$10,000,000 \times 0.1\% = 10,000$ ページ読み込まないといけない．

(2)　索引 1 は B^+ 木なので，葉ノードには「次」のノードを指すポインタ（p_{next}）が定義されているので，レコードを探索キー——この場合，商品番号——のソート順に次々と高速に検索することができる．つまり，ファイル 注文をフィールド 商品番号でソートして順次編成ファイルとし，この場合（2%），先頭（=商品番号 1）から順に 200,000 個のレコードアクセスしたイメージであるので無駄はない．しかし，索引 1 は非クラスタードなので，アクセスするべきデータページ数も 200,000 となり，クラスタードであった場合は読み出すデータページは $100,000 \times 0.02 = 2,000$ ページあろうから，性能が落ちている．

(3)　リレーション 注文をアウタリレーションにした場合のコスト：$100,000 + 10,000,000 \times 50 = 500,100,000$，リレーション 商品をアウタリレーションにした場合：$50 + 5,000 \times 10,000,000 = 50,000,000,050$（ページ）のデータページのアクセスが必要となる．従って，注文リレーションをアウタリレーションとする．

(4)　もし，索引 1 もクラスタードだったら，リレーション 商品をアウタリレーションにした場合：$50 + 5,000 \times 100,000 = 500,000,050$（ページ）のデータページのアクセスが必要となる．注文リレーションをアウタリレーションにした場合のコストは変わらないから，この場合，商品リレーションをアウタリレーションにした方がよい．

問題 4　図 **9.5** に示したソートマージ結合法によるリレーションの自然結合のポインタの動きをもう一度確認して，プログラムを各自作成すること．結果リレーションは右図の通り．

$R * S$

A	B	C
a_2	3	c_2
a_2	3	c_3
a_3	3	c_2
a_3	3	c_3
a_5	6	c_6

問題 5　例えば，表 R が小さくて 1 枚のデータページに納まっているような場合，インデックスページをアクセスする分だけ余計にページフェッチしないといけないから，表スキャンよりコストが嵩む．

問題 6　もしセグメントスキャンならば，R のセグメントには R のデータページ以外のデータページも含まれている可能性があるから，一概にそうともいえない．

■第 10 章

問題 1　(1)　（ア）

```
EXEC SQL BEGIN TRANSACTION;
```

（イ）

```
EXEC SQL UPDATE 口座
         SET 残高 = 残高 - 金額
         WHERE 口座番号 = 依頼人;
```

（ウ）

```
EXEC SQL ROLLBACK;
```

(2)

```
DEPOSIT: PROC OPTIONS(MAIN);
         DCL依頼人 FIXED DEC (6, 0);
         DCL金額 FIXED DEC (9, 0);
         GET(依頼人, 金額);
         EXEC SQL BEGIN TRANSACTION;
         EXEC SQL UPDATE 口座
                  SET 残高 = 残高 + 金額
                  WHERE 口座番号 = 依頼人;
         EXEC SQL COMMIT;
         END DEPOSIT;
```

問題 2　10.3 節を精読せよ．

問題 3　(1) トランザクション UNDO，(2) 全局的 UNDO，(3) 局所的 REDO，(4) 全局的 REDO．説明は，10.4.3 項を精読せよ．

問題 4　(1)，(2) 10.5.1 項を見よ．(3) 10.5.2 項を見よ．(4) 10.5.3 項を見よ．

問題 5　(1) 即時更新をいう．10.5.4 項 (1) を見よ．(2) 遅延更新をいう．10.5.4 項 (2) を見よ．(3) シャドウページ法のことである．10.6 節を見よ．

問題 6　この設問では，読込みのみ（read only）のトランザクションと，読込みと書込みのあるトランザクションとを区別している．T_1, T_2, T_4, T_6, T_7 については，10.5.5 項に記載されている通りの処理をされる．

何もしなくてよいトランザクション：T_1
UNDO するトランザクション：T_6，T_7
REDO するトランザクション：T_2，T_4

T_3, T_5 の扱いは，障害時回復マネジャの設計指針による．安易な考え方は，T_3 と T_5 は UNDO される T_6 や T_7 の書込み結果を読んでいる，つまり汚読(おどく)の可能性があるので，共に，UNDO するというシナリオが考えられる．しかしながら，ログファイルを t_c まで遡(さかのぼ)って，T_3 と T_5 がどのトランザクションの書込み結果を読んでいるのかチェックして，REDO する T_2，T_4 の書出し結果のみを読み込んでいるのであれば，REDO してやればよい．どちらのシナリオをとるかはトレードオフの問題である．

■第 11 章

問題 1 (1) は例 1，(2) は例 2，(3) は例 3，(4) は例 4，(5) は例 5 を見よ．(6) については次のような例を挙げるとよい．

例えば，リレーション 社員 (社員番号, 社員名, 所属) があるとして，トランザクション T_1 を次のようであったとする．

```
SELECT *
FROM 社員
WHERE 所属 = 'K55'
```

T_1 が実行された後，別のユーザが次のようなタップル挿入トランザクション T_2 を発行して COMMIT されたとする．

```
INSERT
INTO 社員(社員番号, 社員名, 所属)
VALUES(007, N'ボンド', 'K55')
```

そうすると，この後で，T_1 を発行したユーザがもう一度 T_1 を発行すると，先程は検索されてこなかったタップル (007, ボンド, K55) が現れる．

問題 2 (1) 定理 11.2 とそれに関する記述を熟読し，解答しなさい．

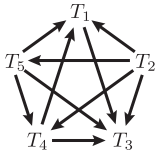
(2) 定理 11.3 とそれに関する記述を熟読し，解答しなさい．

(3) 定理 11.4 とそれに関する記述を熟読し，解答しなさい．

問題 3 (1) T_i : read(x) が T_j : write(x) に先行しているか，T_i : write(x) が T_j : read(x) に先行しているか，T_i : write(x) が T_j : write(x) に先行しているか，の何(いず)れかが成立するときである．

(2) 相反グラフ CG(S) が非巡回のときである．また，S に相反等価な直列スケジュールは CG(S) をトポロジカルソートして得られる．

(3)



(4) $T_2 \to T_5 \to T_4 \to T_1 \to T_3$

問題 4 (1) 次頁の図 (a)，(b)，(c) の 3 スケジュールが考えられる．

(2) スケジュール (c) がそうである．

(3) 相反等価の定義 11.11 より，S_1 と S_2 が相反等価とは，相反している全ての 2 つのステップの順番が，S_1 と S_2 で同一のときをいう．

(4) $T_1 \to T_2$

時刻	T_1	T_2
t_1	read(x)	—
t_2	—	read(y)
t_3	—	read(x)
t_4	—	write(y)
t_5	write(x)	—

(a)

時刻	T_1	T_2
t_1	read(x)	—
t_2	—	read(y)
t_3	—	read(x)
t_4	write(x)	—
t_5	—	write(y)

(b)

時刻	T_1	T_2
t_1	read(x)	—
t_2	—	read(y)
t_3	write(x)	—
t_4	—	read(x)
t_5	—	write(y)

(c)

問題 5　(1)，(2)，(3) 共に，11.4.4 項を精読して答えよ．

問題 6　11.5 節を精読せよ．そこに設問に該当する例が論じられている．できれば本書とは別の例も考えてみると理解が深まる．

■第 12 章

問題 1　12.2.1 項を精読して答えよ．

問題 2

長所	短所
• **可用性が向上する**．つまり，あるサイトがダウンしても，そのサイトが格納していたリレーションの複製が他のサイトにあるので，それを使えば処理を続行できる． • **並列性が向上する**．つまり，読出し専用（read-only）トランザクションが多数ある場合，複製のどれかを読めばよい．その結果，リレーションをサイト間でのデータの移動も少なくでき，ネットワークの負担（=通信コスト）も軽減される．	• **整合性維持のためにオーバヘッドが増加する**．つまり，複製の 1 つが更新された場合，複製間の整合性を保証するためにコストがかかる．即ち，その更新は他の複製に伝搬されなければならない．その結果，トランザクション処理効率（TPS）が落ちる．また当然であるが，通信コストがかかる．

問題 3　12.4.3 項を精読して答えよ．

問題 4　12.5 節を精読して答えよ．

■第 13 章

問題 1　13.2 節を精読して答えよ．

問題 2　ODBC，SQL/CLI，JDBC については 13.2 節を精読して答えよ．SQL/PSM については 13.3.2 項と 5.10 節を精読して答えよ．

問題 3　13.4 節を精読せよ．2 階層については分割問題を指摘することを忘れないように．3 階層については，TP モニタもそうであることを指摘するように．

問題 4　13.4 節を精読して答えよ．

問題 5　13.5 節を精読して答えよ．

■第 14 章

問題 1　各自，試みよ．Google トレンドの URL：https://www.google.co.jp/trends/

問題 2　14.4.2 項と 14.4.3 項を精読して答えよ．

問題 3　14.4.4 項を精読して答えよ．

問題 4　10.3 節と 14.4 節を精読して答えよ．