

データ操作

資料 p.227~

集合演算

- データベースでは次の演算によりデータ集合を表現
- 集合演算
 - 和, 差, 積, 直積
- 関係演算
 - 選択, 射影, 結合, 商

直積

- 2つのリレーションのすべての組み合わせ

R		S		R×S			
X	Y	X	Z	R.X	R.Y	S.X	S.Z
1	あ	1	か	1	あ	1	か
2	い	2	き	1	あ	2	き
3	う	3	く	1	あ	3	く
				2	い	1	か
				2	い	2	き
				2	い	3	く
				3	う	1	か
				3	う	2	き
				3	う	3	く

直積

選択, 射影, 結合

行を取り出す

選択

R×S			
R.X	R.Y	S.X	S.Z
1	あ	1	か
1	あ	2	き
1	あ	3	く
2	い	1	か
2	い	2	き
2	い	3	く
3	う	1	か
3	う	2	き
3	う	3	く

射影

列を取り出す

自然結合

結合

直積, 選択, 射影の組み合わせ

RとSの自然結合

R.X	R.Y	S.Z
1	あ	か
2	い	き
3	う	く

SQL

- データ定義言語
 - SQL-DDL : SQL Data Definition language
- データ操作言語
 - SQL-DML : Sql Data Manipulation Language
- プログラム言語から呼び出し
- カーソル
- ストアドプロシージャ

データ定義言語

- 新しくデータベースやテーブル, ビュー, インデックス, ストアドプロシージャなどを作成(CREATE)
- 削除(DROP)変更(ALTER)

```
CREATE TABLE 顧客 (  
  顧客番号 INTEGER PRIMARY KEY, 主キー制約  
  氏名 CHAR(20) NOT NULL, 非ナル制約  
  住所 CHAR(100), 参照制約  
  郵便番号 CHAR(8) REFERENCES 担当支店 (担当支店コード),  
  電話番号 CHAR(12),  
  電子メールアドレス CHAR(30) UNIQUE ) 一意性制約
```

テーブルの作成

問題

販売管理システムに関する次の図を見て設問に答えよ。

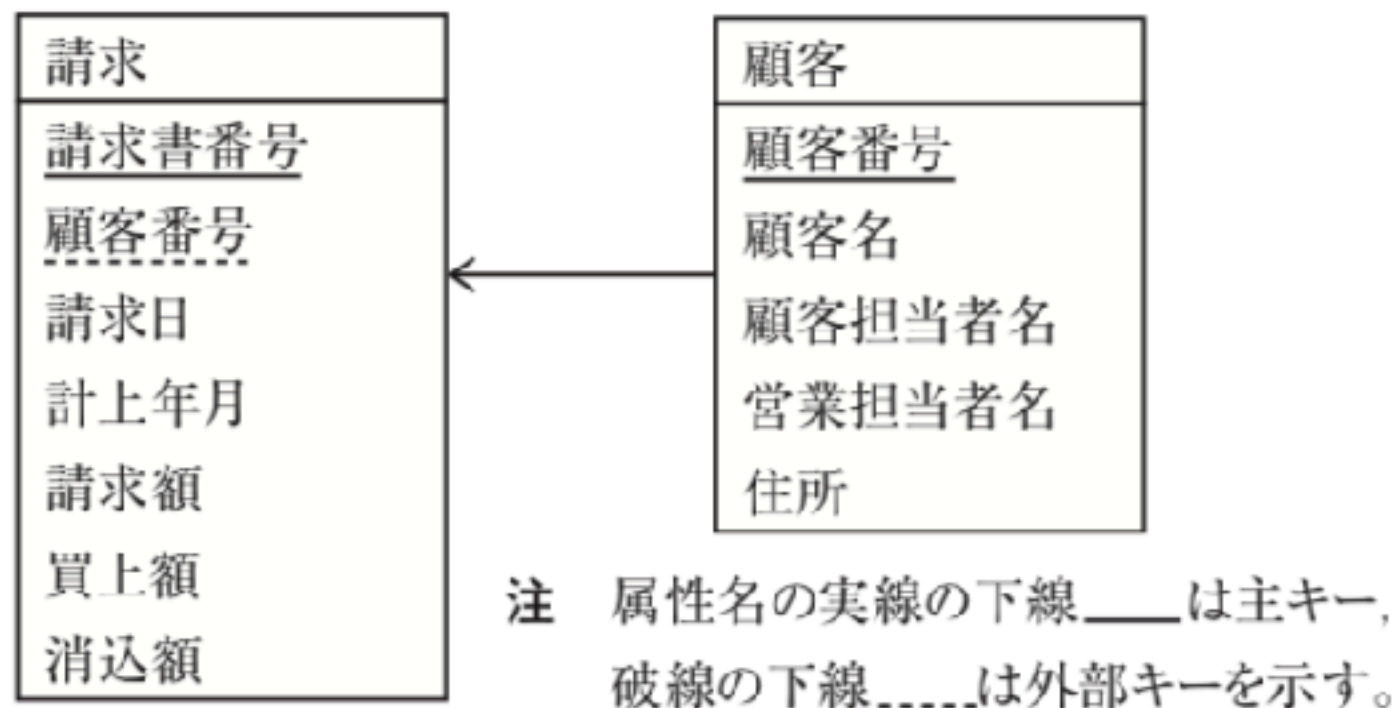
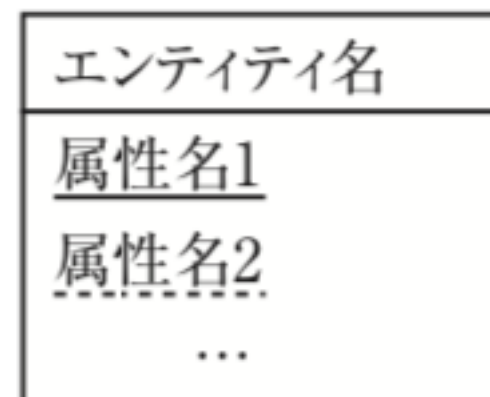


図1 販売管理システムのE-R図(抜粋)

```
CREATE TABLE 請求
(  CHAR(5),
  顧客番号 CHAR(5), 請求日 CHAR(8), 計上年月 CHAR(6), 請求額 NUMERIC(10),
  買上額 NUMERIC(10), 消込額 NUMERIC(10),
   (  ),
  FOREIGN KEY ( 顧客番号 ) REFERENCES 顧客 (  ))
```



E-R図は、エンティティのみを書く場合と、この問題のような形式で書く場合があります。



このとき、実線は主キー、点線は外部キーを示します。

図1を見ると、エンティティ“請求”には七つの属性があります。図2を見ると、請求書番号以外の六つの属性はすでに記述されているので、空欄aが請求書番号だと分かります。空欄aはもう1か所ありますが、図1を見ると請求書番号となっているので、この属性が主キーだということが分かります。したがって、空欄bは主キーを示す**PRIMARY KEY**となります。PRIMARY KEYはこのように、列の後にまとめて書く場合もあります。主キーが複数の列で構成される場合には、こちらが使われます。

顧客番号は外部キーです。外部キーの表記方法では、主キーなどは省略できますが、すべて書くと構文は次のようになります。

FOREIGN KEY（外部キー名） **REFERENCES** 表名（主キー名）

したがって、空欄cには顧客テーブルの主キーである顧客番号が入ります。

《 解答 》 a : 請求書番号, b : PRIMARY KEY, c : 顧客番号

データ操作言語

- データを検索して表示(SELECT)

```
SELECT * | [ALL | DISTINCT] <列名1> [, <列名2>, ... ]  
FROM <表名1> [, <表名2>, ..., ]  
    [JOIN <表名2> ON <結合条件>]  
[WHERE <検索条件> (AND <検索条件2> ...)]  
[GROUP BY グループ化する列の位置 (または列名) ]  
[HAVING グループ化した後の行を抽出する条件]  
[ORDER BY 整列の元となる列 [ ASC | DESC] ]
```

- []内はオプション
- | はORを示し、いずれか一つを選択する

SELECT文の構文

データ操作言語

- DISTINCT 重複を許さない

```
SELECT * | [ALL | DISTINCT] <列名1> [, <列名2>, ... ]  
FROM <表名1> [, <表名2>, ..., ]  
    [JOIN <表名2> ON <結合条件>]  
[WHERE <検索条件> (AND <検索条件2> ...)]  
[GROUP BY グループ化する列の位置 (または列名) ]  
[HAVING グループ化した後の行を抽出する条件]  
[ORDER BY 整列の元となる列 [ ASC | DESC] ]
```

- []内はオプション
- | はORを示し、いずれか一つを選択する

SELECT文の構文

データ操作言語

- FROM 使用する表名

```
SELECT * | [ALL | DISTINCT] <列名1> [, <列名2>, ... ]  
    FROM <表名1> [, <表名2>, ..., ]  
        [JOIN <表名2> ON <結合条件>]  
        [WHERE <検索条件> (AND <検索条件2> ...)]  
        [GROUP BY グループ化する列の位置 (または列名) ]  
        [HAVING グループ化した後の行を抽出する条件]  
        [ORDER BY 整列の元となる列 [ ASC | DESC] ]
```

- []内はオプション
- | はORを示し、いずれか一つを選択する

SELECT文の構文

データ操作言語

- WHERE 行の選択条件

```
SELECT * | [ALL | DISTINCT] <列名1> [, <列名2>, ... ]  
FROM <表名1> [, <表名2>, ..., ]  
      [JOIN <表名2> ON <結合条件>]  
      [WHERE <検索条件> (AND <検索条件2> ...)]  
      [GROUP BY グループ化する列の位置 (または列名)]  
      [HAVING グループ化した後の行を抽出する条件]  
      [ORDER BY 整列の元となる列 [ASC | DESC]]
```

- ・ []内はオプション
- ・ | はORを示し、いずれか一つを選択する

SELECT文の構文

データ操作言語

- GROUP BY グループ化

(指定列の値が同じ複数行をまとめる)

```
SELECT * | [ALL | DISTINCT] <列名1> [, <列名2>, ... ]  
FROM <表名1> [, <表名2>, ..., ]  
    [JOIN <表名2> ON <結合条件>]  
[WHERE <検索条件> (AND <検索条件2> ...)]  
[GROUP BY グループ化する列の位置 (または列名) ]  
[HAVING グループ化した後の行を抽出する条件]  
[ORDER BY 整列の元となる列 [ ASC | DESC] ]
```

- ・ []内はオプション
- ・ | はORを示し、いずれか一つを選択する

SELECT文の構文

データ操作言語

- HAVING グループ化後の選択条件

```
SELECT * | [ALL | DISTINCT] <列名1> [, <列名2>, ... ]  
FROM <表名1> [, <表名2>, ..., ]  
    [JOIN <表名2> ON <結合条件>]  
[WHERE <検索条件> (AND <検索条件2> ...)]  
[GROUP BY グループ化する列の位置 (または列名)]  
[HAVING グループ化した後の行を抽出する条件]  
[ORDER BY 整列の元となる列 [ASC | DESC]]
```

- ・ []内はオプション
- ・ | はORを示し、いずれか一つを選択する

SELECT文の構文

データ操作言語

- ORDER BY 指定列を使って整列

```
SELECT * | [ALL | DISTINCT] <列名1> [, <列名2>, ... ]  
FROM <表名1> [, <表名2>, ..., ]  
    [JOIN <表名2> ON <結合条件>]  
[WHERE <検索条件> (AND <検索条件2> ...)]  
[GROUP BY グループ化する列の位置 (または列名) ]  
[HAVING グループ化した後の行を抽出する条件]  
[ORDER BY 整列の元となる列 [ ASC | DESC] ]
```

- ・ []内はオプション
- ・ | はORを示し、いずれか一つを選択する

SELECT文の構文

問題

A

社員コード	名前	部署コード	給料
10010	伊藤幸子	101	200,000
10020	斉藤栄一	201	300,000
10030	鈴木裕一	101	250,000
10040	本田一弘	102	350,000
10050	山田五郎	102	300,000
10060	若山まり	201	250,000

A→Bを得るためのSQLは？

実行結果B

部署コード	社員コード	名前
101	10010	伊藤幸子
101	10030	鈴木裕一
102	10040	本田一弘
102	10050	山田五郎
201	10020	斉藤栄一
201	10060	若山まり

ア SELECT 部署コード, 社員コード, 名前 FROM A
GROUP BY 社員コード

イ SELECT 部署コード, 社員コード, 名前 FROM A
GROUP BY 部署コード

ウ SELECT 部署コード, 社員コード, 名前 FROM A
ORDER BY 社員コード, 部署コード

エ SELECT 部署コード, 社員コード, 名前 FROM A
ORDER BY 部署コード, 社員コード

表Aと実行結果Bでは、行数が変わっていないことと、特にグループでまとめられていないことから、GROUP BY 句は使われていないことが分かります。また、実行結果Bを見ると、データは部署コード順に整列されていて、さらに同じ部署コードなら社員コード順に並んでいることが読み取れるので、**ORDER BY 部署コード, 社員コード**というかたちで整列していることが分かります。したがって、エが正解です。

その他データ操作言語

- SELECT以外のSQL-DMLは更新系SQLと呼ばれる

- INSERT
- UPDATE
- DELETE

```
INSERT INTO <表名> (<列名1>, <列名2>, ...)  
VALUES (<値1>, <値2>, ...)  
または  
INSERT INTO <表名> (<列名1>, <列名2>, ...)  
SELECT ~ (以降, 通常のSELECT文)
```

```
UPDATE <表名> SET <列名1> = <値1>  
[, <列名2> = <値2> ...] WHERE <検索条件>
```

```
DELETE FROM <表名> WHERE <検索条件>
```

副問合わせ

- SQL文の中でSQLを呼び出すこと
- カッコを使ってもう一つ記述
 - カッコ内のSQLを 副問合わせ と呼ぶ
- カッコ内外で共通テーブルを用い、外と中のSQL文を結びつけた問い合わせを 相関副問合わせ と呼ぶ

問題

社員（社員番号，社員名，性別，生年月日）

```
SELECT 社員番号, 社員名 FROM 社員 AS S1
      WHERE 生年月日 > ( a )
```

“社員”表から、男女それぞれの最年長社員を除くすべての社員を取り出すSQL文とするために、aに入る副問合せはどれか。ここで、“社員”表は次の構造とし、下線部は主キーを表す。

ア SELECT MIN(生年月日) FROM 社員 AS S2
 GROUP BY S2.性別

イ SELECT MIN(生年月日) FROM 社員 AS S2
 WHERE S1.生年月日 > S2.生年月日
 OR S1.性別 = S2.性別

ウ SELECT MIN(生年月日) FROM 社員 AS S2
 WHERE S1.性別 = S2.性別

エ SELECT MIN(生年月日) FROM 社員
 GROUP BY S2.性別

解説

“社員”表の例に、次のような表を考えてみます。

社員番号	社員名	性別	生年月日
1001	吉井 九兵衛	男	1961-01-01
1002	姫路 円	女	1971-03-03
1003	木下 理樹	男	1981-05-05
1004	坂本 恭介	男	1986-07-07
1005	島田 真実	女	1991-09-09

このとき、外側の問合せでは、「男女それぞれの最年長社員を除くすべての社員を取り出す」必要があるので、空欄aには、**自分と同じ性別の最年長社員を求めるSQL文が入ります**。したがって、**S1.性別=S2.性別**という条件で最年長社員の生年月日**MIN**（生年月日）を検索するウが正解です。

ウを前提に上記の例を考えてみると、最初の社員番号1001の吉井さんは性別が男です。**'男'=S2.性別**という条件で**MIN(生年月日)**を求めると'1961-01-01'となり、生年月日> a という条件にはあてはまらないので、表示されません。

同様に、社員番号1002の姫路さんは性別が女です。ここで、**'女'=S2.性別**という条件で**MIN**（生年月日）を求めると'1971-03-03'となり、生年月日> a という条件にはあてはまらず、表示されません。あとの3行では、男でも女でも生年月日は最年長の社員より新しいため、表示されます。

その他SQL構文

- 集計関数
 - SUM, AVG, MAX, MIN, COUNT(列名)
- 比較演算子
- 評価を行う演算子
 - EXISTS, NOT EXISTS(行が存在する/しない)
- 曖昧な条件を比較する演算子
 - LIKE LIKE '吉井'なら吉井で始まる氏名を検索
- SELECTを合わせるUNION句
 - 2つのSELECTの結果を合わせる演算集合

カーソル

- 一連のデータに順にアクセスするための仕組み
- 1行ずつ読み出して処理を行うのに適している

DECLARE カーソル名 CURSOR FOR SELECT ~

- 上記構文でカーソル名を定義
- OPEN：カーソルをオープン
- FETCH：1行ずつ取り込み
- CLOSE：カーソルを閉じる

ビュー

- 見せるための表(導出表)
- 元の表から必要なデータを選択, 表示
- 条件として
 - 複数行をまとめていない
 - 集計関数, DISTINCT, BY句など✖
 - 複数の表を結合していない
 - ※元の行を特定できれば変更可能

ストアドプロシージャ

- データベースへの問い合わせを一連の処理としてまとめ
DBMSに保存したもの

