【例2】

表5-2 (p.178) の表“部署”中の列“部署ID”と“部署業務”を射影するSQL文の例を示します。

```
SELECT 部署ID, 部署業務 FROM 部署
```

このSQLの実行結果は、次のとおりです。

▼**表5-4** 例2の実行結果

部署ID	部署業務
0001	設計
0004	設計
0002	開発
0003	総務

5.1.4 選択（表中の特定の行の検索）

```
SELECT * FROM 表名 WHERE 探索条件
```

特定の行を検索するためには、WHERE句中に探索条件として、行を選択するための条件を指定します（**図5-3**）。WHERE句で指定した探索条件の結果が真となる行だけが選択されます。ただし、WHERE句を省略した場合には、すべての行が選択されることに留意してください。

▲**図5-3** 選択（表中の特定の行の検索）

【例3】

表“部署”から“部署業務”が'設計'である行を検索するSQL文の例を示します。

```
SELECT * FROM 部署 WHERE 部署業務 = '設計'
```

このSQLの実行結果は、次のとおりです。

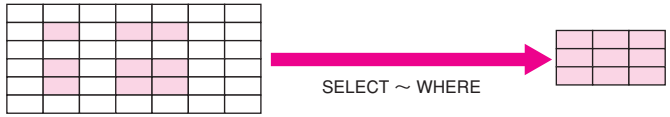
▼**表5-5** 例3の実行結果

部署ID	部署業務	業務区分
0001	設計	直接業務
0004	設計	直接業務

5.1.5 選択＆射影（表中の一部分の検索）

```
SELECT 列名1, 列名2, ... 列名n   FROM 表名
  WHERE 探索条件
```

SELECT句で取り出す列の並びを指定し、WHERE句で行を選択するための条件を指定します（**図5-4**）。



▲**図5-4** 選択＆射影（表中の一部分の検索）

【例4】

表5-2 (p.178) の表“部署”中の列“部署ID”と“部署業務”を射影し、“部署業務”が'設計'である行を検索するSQL文の例を示します。

```
SELECT 部署ID, 部署業務 FROM 部署 WHERE 部署業務 = '設計'
```

このSQLの実行結果は、次のとおりです。

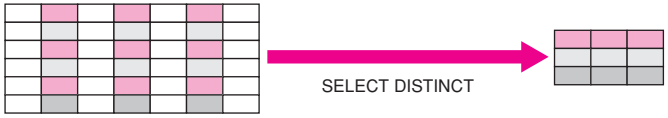
▼**表5-6** 例4の実行結果

部署ID	部署業務
0001	設計
0004	設計

5.1.6 検索結果からの冗長な重複行の排除

```
SELECT DISTINCT 列名1, 列名2, ... 列名n  FROM 表名
    ...
```

表中の特定の列だけを検索すると、結果中に重複した行が存在する場合があります。このような重複行を排除するためには、SELECT句の列名の並びの前、つまり、キーワードSELECTの直後にキーワードDISTINCTを指定します（**図5-5**）。



▲**図5-5**　検索結果からの冗長な重複行の排除（DISTINCT）

【例5】

表5-2（p.178）の表“部署”中の列“部署業務”と“業務区分”を射影し、選択した行の重複を排除するSQL文の例を示します。

```
SELECT DISTINCT 部署業務, 業務区分 FROM 部署
```

このSQLの実行結果は、次のとおりです。

▼**表5-7**　例5の実行結果

部署業務	業務区分
設計	直接業務
開発	直接業務
総務	間接業務

なお、DISTINCTを省略するか、またはDISTINCTの代わりにキーワードALLを指定すると、結果中に重複行が存在してもそれらの重複行を排除しないでそのまま検索されます。

5.1.7 SELECT句の形式

```
SELECT [ ALL | DISTINCT ] { * | 選択式の並び }
```

選択式の並びは、複数の選択式を指定する場合、コンマで区切って並べます。選択式には、列のほかに、演算を含む値式などを指定できます。値式を指定した後にAS句を指定して、選択項目に対して列名を付加することができます。この列名は、後で説明するORDER BY句中で用いることができます。

AS句を省略した場合、値式として列を指定した場合には、その選択項目の列名としては、指定した列の列名が仮定されますが、演算を含む値式の場合には、その選択項目は、列名なしになります。

```
選択式::= 値式 [ [AS] 列名 ]  |  {表名 | 相関名}.*
```

SELECT句に * を修飾なしで指定した場合には、FROM句で指定したすべての表の列を意味しますが、選択式として、表名または相関名でピリオドを挿んで修飾した * を指定した場合には、指定した表のすべての列を意味します。なお、相関名については、後で説明します。

5.2 探索条件で指定する述語

5.2.1 検索条件として指定できる述語の種類

探索条件として指定できる述語には、次の述語があります。

- 比較述語
- IN述語
- NULL述語
- 限定比較述語
- BETWEEN述語
- LIKE述語
- EXISTS述語

また、探索条件中では、ブール演算（AND、OR、NOT）を指定することもできます。また、ブール演算の演算項には、述語やブール演算を指定することができます。

5.2.2 ブール演算

ブール演算子には、次の演算子があります。

▼**表5-8**　ブール演算

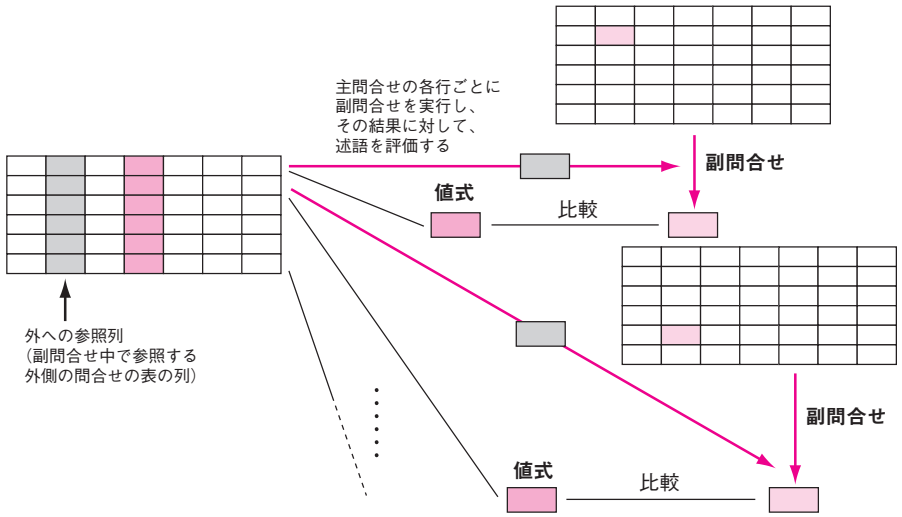
ブール演算子	意味
AND	両方の演算項を満たすかどうかを判定
OR	いずれかの演算項を満たすかどうかを判定
NOT	演算子に続く演算項を満たさないかどうかを判定

▼表5-18　例7の実行結果

科目名	得点
データベース	70
SQL	95
Java	50

この問合せでは、副問合せが実行され、学生表から“学生名 = '佐藤春雄'”を満たす行が選択され、その行中の学生IDが求められます。その後、履修表から学生IDが副問合せの結果得られた値と等しい行が選択され、それらの行中の科目名と得点が出力されます。

外への参照がある副問合せは、主問合せの各行ごとに、外への参照列の値を参照して副問合せを評価し、その結果を用いて主問合せの各行に対して述語を評価することになります（図5-10）。



▲図5-10　外への参照がある副問合せ

【例8】

表5-17（p.189）の2つの表で、学生名が“佐藤春雄”である学生が履修している科目名と得点を、外への参照がある副問合せを用いて求めるSQLの例を示します。

```
SELECT 科目名, 得点 FROM 履修
  WHERE EXISTS ( SELECT * FROM 学生
                 WHERE 学生名 = '佐藤春雄' AND 学生ID = 履修.学生ID)
```


この問合せの結果として、次の表が得られます。

▼表5-19　例8実行結果

科目名	得点
データベース	70
SQL	95
Java	50

この問合せの副問合せ中で指定している“履修.学生ID”は、副問合せのFROM句で指定した学生表ではなく、副問合せの外側の主問合せのFROM句で指定した履修表の列を参照しています。この問合せでは、履修表が検索され、履修表の各行に対して、その行中の学生IDを用いて副問合せが実行されます。副問合せでは、学生表から“学生名 = '佐藤春雄' ”を満たし、学生IDが履修表の学生IDと等しい行が選択されます。ここで、EXISTSは、副問合せの結果が1行以上の場合に真になる述語です。したがって、副問合せの結果が1行以上になる履修表の行が選択され、それらの行中の科目名と得点が最終的に出力されます。

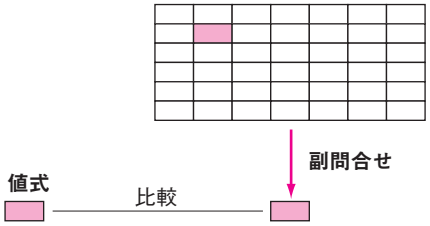
このように、副問合せが外への参照を含むかどうかによって、その意味が大きく異なることに注意する必要があります。

5.2.8　副問合せを含む比較述語

副問合せを含む比較述語の形式は、次のとおりです。

値式　比較演算子　副問合せ

比較述語で比較する値は、左辺がスカラ値（1行1列からなる単純な値）ならば、右辺もスカラ値でなければなりません。つまり、比較述語の右辺に副問合せを指定した場合、その副問合せの結果は、スカラ値でなければなりません。したがって、比較演算子の右辺に指定した副問合せは、1列を射影し、結果の行数が1行以下でなければなりません（図5-11）。副問合せの結果が2行以上になる場合は、基数違反のエラーとなります。副問合せの結果が1行の場合に、副問合せで得られたスカラ値と左辺に指定した値とを指定した比較演算子に従って比較します。副問合せの結果が0行の場合には、副問合せで得られるスカラ値をナル値として評価し、比較述語の結果は、不定になります。



副問合せで取り出す列は1つ

副問合せの結果	比較述語の結果
0行	不定(unknown)
1行	比較演算の結果
2行以上	エラー(基数違反)

▲**図5-11**　副問合せを含む比較述語

【例9】

次の例でも、**表5-17** (p.189) の学生表と履修表を用います。

科目名が“言語工学”の科目を履修している学生の学生名を、副問合せを含む比較述語を用いて求めるSQLの例を示します。

```
SELECT 学生名 FROM 学生
  WHERE 学生ID = ( SELECT 学生ID FROM 履修 WHERE 科目名 = '言語工学' )
```

この問合せの結果として、次の表が得られます。

▼**表5-20**　例9の実行結果

学生名
田中夏美

言語工学を履修している学生が学生ID'S06002'の学生1人しかいないので、副問合せの結果が1行で、'S06002'となり、学生表から学生IDが'S06002'の行が出力されます。

【例10】

科目名が“XML”の科目を履修している学生の学生名を、副問合せを含む比較述語を用いて求めるSQLは、次のとおりです。

```
SELECT 学生名 FROM 学生
  WHERE 学生ID = ( SELECT 学生ID FROM 履修 WHERE 科目名 = 'XML' )
```

この問合せの結果としては、どの行も出力されません。XMLを履修している学生がいないので、副問合せの結果が0行で、ナル値になり、学生表の学生IDとの比較の結果が不定になるからです。

【例11】

科目名が“データベース”の科目を履修している学生の学生名を、副問合せを含む比較述語を用いて求めるSQLは、次のとおりです。

```
SELECT 学生名 FROM 学生
  WHERE 学生ID = ( SELECT 学生ID FROM 履修 WHERE 科目名 = 'データベース' )
```

この問合せは、基数違反のエラーを引き起こします。データベースを履修している学生が4人なので、副問合せの結果が複数行になり、学生表の学生IDと比較できないからです。

比較述語で副問合せを用いる場合は、一般的に、結果が必ず1行以下になるような問合せを副問合せに指定します。

5.2.9　限定比較述語

限定比較述語の形式は、次のとおりです。

```
値式　比較演算子　限定子　副問合せ
```

限定子には、ANYもしくはSOME、またはALLが指定できます。

限定子のANYとSOMEとは、同じ意味で、“いずれか”を意味します。限定子のALLは、“すべて”を意味します。

▼**表5-21**　限定述語の限定子の意味

限定子	副問合せの結果に対する限定子の意味
ANY | SOME	いずれかの行に対して比較演算を満たすかどうか
ALL	すべての行に対して比較演算を満たすかどうか

限定比較述語は、左辺に指定した値が比較演算子で指定した条件を、右辺の副問合せの結果として得られる複数行からの(ANYもしくはSOME指定の場合)“いずれか”の値を満たすか、または(ALL指定の場合)“すべて”の値を満たすかどうかを検査します。

なお、限定比較述語の場合も、左辺がスカラ値ならば、右辺の副問合せで射影する列は、1列でなければなりません(**図5-12**)。

【例23】

表5-39(p.206)の2つの表を用いて例を示します。2006年度の**販売中止商品は除くが、新規商品を含めて**、2005年度と2006年度の商品別売上を比較するために、2005年度商品別売上表と2006年度商品別売上表とを**右外結合**するSQLの例を示します。

```
SELECT Y.商品ID, Y.商品名,
       X.売上金額AS 2005年度売上金額, Y.売上金額 AS 2006年度売上金額
    FROM 2005年度商品別売上 AS X RIGHT OUTER JOIN 2006年度商品別売上 AS Y
            ON X.商品ID = Y.商品ID
```

この問合せの結果として、次の表が得られます。

▼**表5-42** 例23の実行結果

商品ID	商品名	2005年度売上金額	2006年度売上金額
PT0003	液晶ハイビジョンテレビ	95,000,000	135,000,000
PT0004	プラズマハイビジョンテレビ	105,000,000	145,000,000
PT0005	プラズマハイビジョン録画テレビ	(NULL)	84,000,000
PV0002	ハイビジョンビデオレコーダ	85,000,000	77,000,000
PV0003	W録ハイビジョンビデオレコーダ	(NULL)	99,000,000
PV0004	W録ハイビジョンビデオレコーダ1T	(NULL)	43,000,000
PA0003	CDプレーヤ	22,200,000	21,100,000
PA0004	メモリオーディオプレーヤ	(NULL)	55,500,000
PA0005	HDDオーディオプレーヤ	(NULL)	33,300,000

【例24】

2006年度の**販売中止商品も、新規商品も含めて**、2005年度と2006年度の商品別売上を比較するために、2005年度商品別売上表と2006年度商品別売上表とを**完全外結合**するSQLの例を示します。

```
SELECT COALESCE(X.商品ID, Y.商品ID), COALESCE(X.商品名, Y.商品名),
       X.売上金額AS 2005年度売上金額, Y.売上金額 AS 2006年度売上金額
    FROM 2005年度商品別売上 AS X FULL OUTER JOIN 2006年度商品別売上 AS Y
            ON X.商品ID = Y.商品ID
```

この問合せの結果として、次の表が得られます。

▼**表5-43** 例24の実行結果

商品ID	商品名	2005年度売上金額	2006年度売上金額
PT0001	テレビ	15,000,000	(NULL)
PT0002	液晶ワイドテレビ	45,000,000	(NULL)
PT0003	液晶ハイビジョンテレビ	95,000,000	135,000,000
PT0004	プラズマハイビジョンテレビ	105,000,000	145,000,000
PT0005	プラズマハイビジョン録画テレビ	(NULL)	84,000,000
PV0001	ビデオレコーダ	32,000,000	(NULL)
PV0002	ハイビジョンビデオレコーダ	85,000,000	77,000,000
PV0003	W録ハイビジョンビデオレコーダ	(NULL)	99,000,000
PV0004	W録ハイビジョンビデオレコーダ1T	(NULL)	43,000,000
PA0001	カセットテープレコーダ	9,900,000	(NULL)
PA0002	MDレコーダ	11,100,000	(NULL)
PA0003	CDプレーヤ	22,200,000	21,100,000
PA0004	メモリオーディオプレーヤ	(NULL)	55,500,000
PA0005	HDDオーディオプレーヤ	(NULL)	33,300,000

なお、COALESCE(X.商品ID, Y.商品ID)は、X.商品IDがナル値でなければ、X.商品IDを、ナル値ならば、Y.商品IDを求めています。

5.3.5 自然結合

自然結合を行うSQLの形式は、次のとおりです。

```
表参照1 NATURAL [結合型] JOIN 表参照2
```

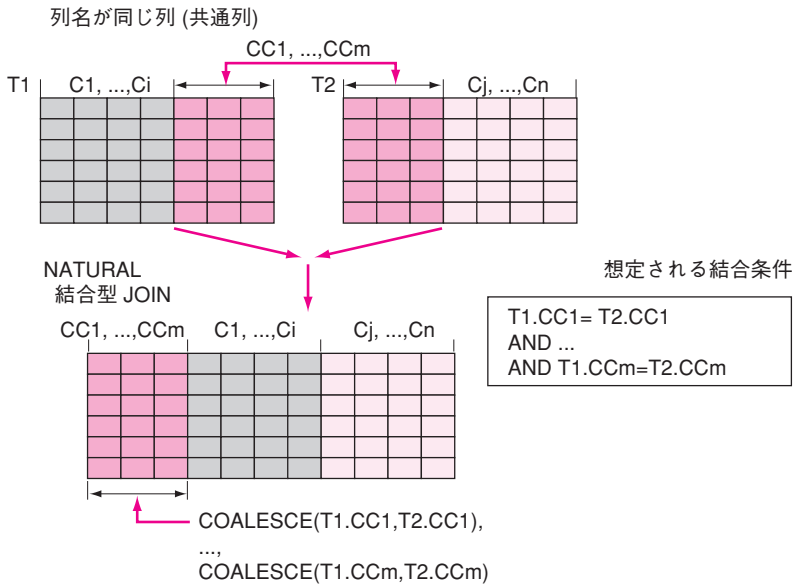
結合型の形式は、外結合と内結合での結合種別の指定と同様であり、次のとおりです。

```
{LEFT | FULL | RIGHT} [OUTER] | INNER
```

つまり、自然結合にも、内結合ならびに外結合(左外結合、完全外結合および右外結合)があります。

また、自然結合の形式も、結合表という構文要素に含まれます。

自然結合は、結合する2つの表中で共通の名前をもつ列を結合列として等結合を行う結合演算です。したがって、結合条件として、共通列名をもつ列間の = 演算が仮定されます。もちろん、共通列名をもつ列が複数存在する場合には、それらの = 結合条件のAND演算が結合条件として仮定されます(**図5-19**)。



▲**図5-19** 自然結合

【例25】

表5-39 (p.206) の2005年度商品別売上表と2006年度商品別売上表とを**自然結合**を用いて**内結合**するSQLの例を示します。

```
SELECT  商品ID, 商品名, 2005年度売上金額, 2006年度売上金額
    FROM  2005年度商品別売上 AS X ( 商品ID, 商品名, 2005年度売上金額 )
                NATURAL INNER JOIN
                    2006年度商品別売上 AS Y ( 商品ID, 商品名, 2006年度売上金額 )
```

自然結合を指定すると、結合する2つの表中で共通の名前をもつ列間の等結合が仮定されるので、このSQLでは、結合条件を指定しません。また、これらの結合列をSQL中で指定するとき、2つの表の列から導出された列を参照することになるので、相関名や表名での修飾を行なうことができません。

この問合せの結果として、次の表が得られます。

▼**表5-44** 例25の実行結果

商品ID	商品名	2005年度売上金額	2006年度売上金額
PT0003	液晶ハイビジョンテレビ	95,000,000	135,000,000
PT0004	プラズマハイビジョンテレビ	105,000,000	145,000,000
PV0002	ハイビジョンビデオレコーダ	85,000,000	77,000,000
PA0003	CDプレーヤ	22,200,000	21,100,000

【例26】

表5-39 (p.206) の2005年度商品別売上表と2006年度商品別売上表とを**自然結合**を用いて**完全外結合**するSQLの例を示します。

```
SELECT  商品ID, 商品名, 2005年度売上金額, 2006年度売上金額
    FROM  2005年度商品別売上 AS X ( 商品ID, 商品名, 2005年度売上金額 )
                NATURAL FULL OUTER JOIN
                    2006年度商品別売上 AS Y ( 商品ID, 商品名, 2006年度売上金額 )
```

この問合せの結果として、次の表が得られます。

▼**表5-45** 例26の実行結果

商品ID	商品名	2005年度売上金額	2006年度売上金額
PT0001	テレビ	15,000,000	(NULL)
PT0002	液晶ワイドテレビ	45,000,000	(NULL)
PT0003	液晶ハイビジョンテレビ	95,000,000	135,000,000
PT0004	プラズマハイビジョンテレビ	105,000,000	145,000,000
PT0005	プラズマハイビジョン録画テレビ	(NULL)	84,000,000
PV0001	ビデオレコーダ	32,000,000	(NULL)
PV0002	ハイビジョンビデオレコーダ	85,000,000	77,000,000
PV0003	W録ハイビジョンビデオレコーダ	(NULL)	99,000,000
PV0004	W録ハイビジョンビデオレコーダ1T	(NULL)	43,000,000
PA0001	カセットテープレコーダ	9,900,000	(NULL)
PA0002	MDレコーダ	11,100,000	(NULL)
PA0003	CDプレーヤ	22,200,000	21,100,000
PA0004	メモリオーディオプレーヤ	(NULL)	55,500,000
PA0005	HDDオーディオプレーヤ	(NULL)	33,300,000

なお、自然結合を用いると、共通の列の名前で参照する結果が、自動的に2つ表の列から導出される値になるので、COALESCE式を指定する必要がありません。

自然結合で注意すべき点は、自然結合によって得られる列の並びが通常の結合と異なる点です。自然結合の結果の列の並びは、共通列名をもつ列、左の表の列、右の表の列となり、さらにそれらの表中の列の順序で構成されます。共通列名をもつ列の順序が2つの表で異なる場合には、左の表の列の順序になります。また、共通列名をもつ列は、自然結合を行う2つの表から導出される列であり、自然結合は、等結合を行うので、自然結合の結果の行中において、内結合の場合、2つの表の列の値は必ず等しい値となりますが、外結合の場合には、いずれかがナル値の行が存在することになります。したがって、自然結合の結果中の共通列名をもつ列の値は、左の表の列の値がナル値ならば、右側の列の値になります(**図5-19**中では、COALESCE式を用いてこのことを表現しています)。また、**共通列名をもつ列**は、2つの表から導出される列となるので、その列を含む問合せ中で参照するときには、**表名または相関名で修飾できない**ことにも、留意してください。

5.4.2 グループ化（グループ分け）

グループ化は、**GROUP BY句**で指定します。GROUP BY句の形式を次に示します。

```
GROUP BY 列名 [ , 列名 ] ...
```

GROUP BY句は、WHERE句を指定した場合には、WHERE句の後に、WHERE句を省略した場合には、FROM句の後に指定します。

GROUP BY句に指定した列を**グループ化列**といいます。

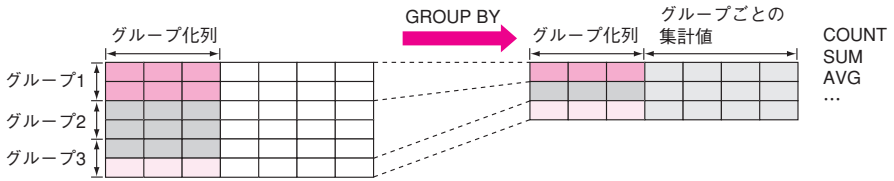
グループ化は、先行するWHERE句またはFROM句の結果である行のマルチ集合を入力として、GROUP BY句で指定したすべてのグループ化列の値が同じ行の集合を同じグループとして、グループ分けします。グループ化を行った場合、各グループごとに1つの行が生成されます。SELECT句で射影できる列は、グループ化列か、または集合関数の値になります。つまり、SELECT句に指定する列は、グループ化列かまたは集合関数の引数中で指定しなければなりません。

GROUP BY句を指定するSQLの形式は、次のようになります。

```
SELECT 列名1, [ , 列名2 ] ... , 集合関数1, 集合関数2, ...
    FROM句 [WHERE句]  GROUP BY 列名1 [ , 列名2 ] ...
```

ただし、SELECT句に指定する射影項目の指定順序は、任意です。

また、グループ化を行った場合の集合関数の引数の入力は、各グループごとのそのグループ内の行の集合になります（**図5-22**）。



▲**図5-22**　グループ化

【例30】

表5-46（p.214）の履修表を用いて例を示します。

履修表中の各科目ごとの科目名、履修人数、合計得点、平均得点、最大得点および最小得点を求めるSQLの例を示します。


```
SELECT 科目名, COUNT(*) AS 履修人数, SUM(得点) AS 合計得点, AVG(得点) AS 平均得点,
          MAX(得点) AS 最大得点, MIN(得点) AS 最小得点 FROM 履修
    GROUP BY 科目名
```

この問合せでは、科目ごとの集計値を求めるので、科目名をグループ化列として指定し、グループ化を行っています。科目名が同じものを同じグループにし、各グループ内の行を集合関数の対象にしています。

この問合せの結果は、次のとおりです。

▼**表5-50**　例28の実行結果

科目名	履修人数	合計得点	平均得点	最大得点	最小得点
データベース	4	325	81.25	95	70
SQL	4	315	78.75	100	55
Java	3	225	75.00	90	50
ネットワーク	2	115	57.50	60	55
言語工学	1	70	70	70	70

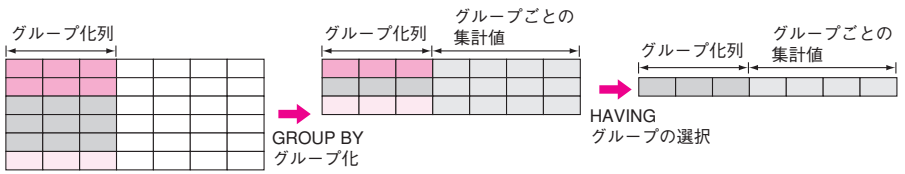
5.4.3 グループの選択

グループ化によって得られた各グループを選択するかどうかを**HAVING句**を用いることによって指定することができます。その場合、HAVING句は、GROUP BY句の後に指定します。

HAVING句の形式を次に示します。

```
HAVING 探索条件
```

HAVING句の探索条件には、グループ化によって得られた各グループを選択するための探索条件を指定します。したがって、グループ化を指定した場合のSELECT句と同様に、HAVING句の探索条件に指定する列は、グループ化列かまたは集合関数中の引数で指定しなければなりません（**図5-23**）。



▲**図5-23**　グループ化の選択

また、GROUP BY句を指定しないで、HAVING句を指定することもできます。その場合には、先行するWHERE句またはFROM句の結果得られる行のマルチ集合全体を1つのグループとみなし、その結果を選択するかどうかを指定することになります。この場合のSELECT句とHAVING句に指定できる列は、グループ化列がないので、集合関数中の引数にだけ指定することができます。

【例31】

表5-46 (p.214) の履修表を用いて例を示します。

履修表中の各科目ごとに、履修人数が3人以上の科目の科目名, 履修人数、合計得点、平均得点、最大得点および最小得点を求めるSQLの例を示します。

```
SELECT 科目名, COUNT(*) AS 履修人数, SUM(得点) AS 合計得点, AVG(得点) AS 平均得点,
       MAX(得点) AS 最大得点, MIN(得点) AS 最小得点 FROM 履修
  GROUP BY 科目名
  HAVING COUNT(*) >= 3
```

この問合せでは、科目名でグループ化を行い、その結果から履修人数が3人以上のグループだけを選択しています。

この問合せの結果は、次のとおりです。

▼**表5-51** 例31の実行結果

科目名	履修人数	合計得点	平均得点	最大得点	最小得点
データベース	4	325	81.25	95	70
SQL	4	315	78.75	100	55
Java	3	225	75.00	90	50

5.5 集合演算

5.5.1 和集合演算

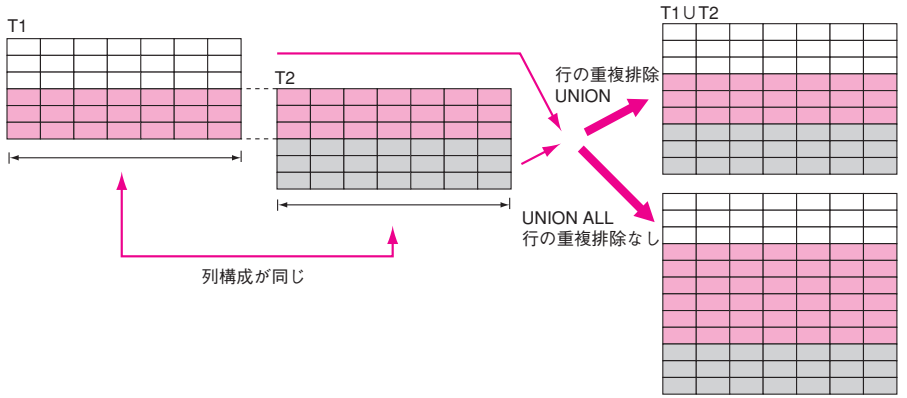
和集合演算は、問合せの結果として導出される表である行の集合どうしの和集合を求めます。和集合演算の形式を次に示します。

```
問合せ式  UNION [ALL] { 問合せ指定 | ( 問合せ式 ) }
```

ここで、問合せ式は、問合せ指定かまたは集合演算を含む問合せ式であり、問合せ指定の形式は、次のとおりです。

```
SELECT句  FROM句
    [WHERE句] [GROUP BY句] [HAVING句]
```

和集合演算は、左辺の問合せの結果導出される表と右辺の問合せの結果導出される表との和集合を求める演算です。ただし、導出された表を行の集合として集合演算を行うためには、集合演算を行う表の列構成が同じでなければなりません。つまり、集合演算の演算項の2つの表の列数が同じでかつ対応する列のデータ型が互いに変換可能なデータ型でなければなりません。対応する列の名前は、同じでなくてもかまいません。また、ALLを省略してUNIONだけを指定した場合には、重複する行が存在するならば、重複が排除されますが、ALLを指定した場合には、重複する行が存在しても、重複を排除しないでそのまま出力されます (**図5-24**)。



▲**図5-24** 和集合演算

▲**図5-25**　検索行の順序付け

【例33】

表5-52（p.220）の演劇部員表とコーラス部員表を用いて例を示します。

演劇部とコーラス部のいずれかに所属する学生を求めて、学生ID順に出力するSQLの例を示します。

```
SELECT * FROM 演劇部員 UNION SELECT * FROM コーラス部員
ORDER BY 学生ID
```

この問合せの結果は、次のとおりです。

▼**表5-55**　例33の実行結果

学生ID	学生名	年齢	性別	出身地
S000002	中山美樹	22	女	東京都
S000004	佐藤恵子	21	女	京都府
S000011	山下純一	22	男	長崎県
S000013	田中慎治	23	男	宮崎県
S000024	池田美穂	21	女	大阪府
S000025	川上香織	21	女	奈良県
S000033	大江智子	20	女	広島県

5.7 表中のデータの更新

5.7.1 行の挿入（INSERT文）

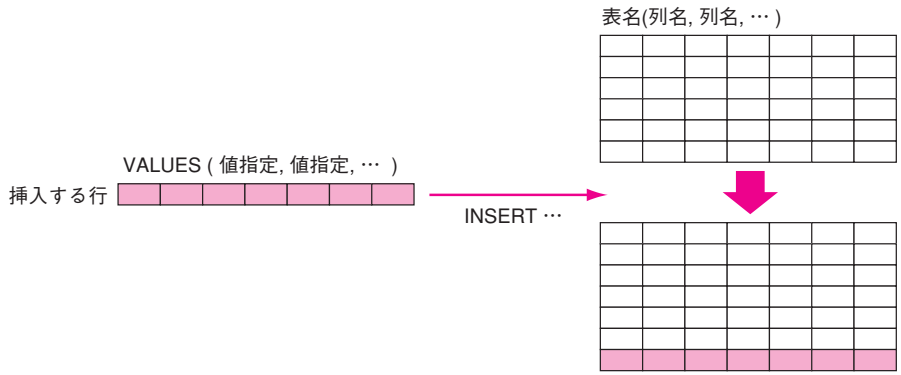
表に行を挿入するINSERT文の形式を次に示します。

```
INSERT INTO  表名 [ ( 列名 [, 列名 ] ... ) ]
    { VALUES( 値指定 [, 値指定 ] ... ) | 問合せ式 }
```

INSERT文のINTO句中で、行を挿入する表を指定します。その表名の後に括弧で囲んだ列名の並びを指定することもできます。この列名の並びを省略すると、指定した表のすべての列がその表を構成する列の順序で仮定されます。列名の並びを指定した場合、その表中のその他の列には、ナル値が挿入されます。ただし、その列の定義時にDEFAULT句を指定して、列の既定値が定義されている場合には、列の既定値が挿入されます。

表に挿入する行の値は、その後のVALUES句または問合せ式で指定します（**図5-26**）。

VALUES句には、列名の並びの各列に対する挿入値をその順序で指定します。指定した値から構成される行が挿入されます。



▲**図5-26**　表への行の挿入（INSERT ～ VALUES）

【例34】

表5-52（p.220）の演劇部員表を用いて例を示します。

演劇部に新入部員（'S000045', '松山美幸', '18', '女', '兵庫県'）が入ったので、新しい行を演劇部員表に挿入するSQLの例を示します。

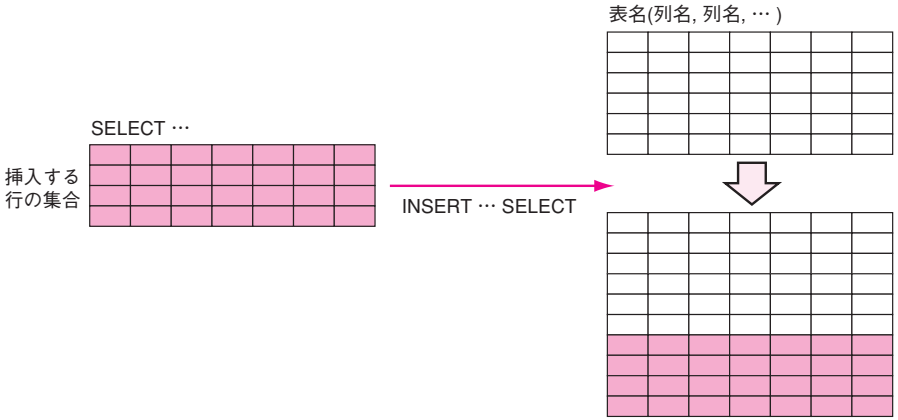
```
INSERT INTO 演劇部員（学生ID, 学生名, 年齢, 性別, 出身地）
  VALUES('S000045', '松山美幸', '18', '女', '兵庫県')
```

このINSERT文を実行した後の演劇部員表は、次のようになります。

▼表5-56 例34の実行結果

学生ID	学生名	年齢	性別	出身地
S000004	佐藤恵子	21	女	京都府
S000011	山下純一	22	男	長崎県
S000024	池田美穂	21	女	大阪府
S000025	川上香織	21	女	奈良県
S000033	大江智子	20	女	広島県
S000045	松山美幸	18	女	兵庫県

問合せ式を指定する場合には、問合せ式の選択項目を列名の並びの各列に対応させて指定します（**図5-27**）。問合せの結果得られる行の集合がINTO句で指定した表に挿入されます。



▲図5-27 表への行の挿入（INSERT ～ SELECT）

【例35】

表5-52（p.220）の演劇部員表とコーラス部員表を用いて例を示します。

コーラス部員は全員演劇部員も兼任することになったので、演劇部に所属していないコーラス部員を演劇部員表に追加するSQLの例を示します。

```
INSERT INTO 演劇部員
  SELECT * FROM コーラス部員
  WHERE NOT EXISTS( SELECT * FROM 演劇部員 WHERE 学生ID = コーラス部員.学生ID )
```

このINSERT文を実行した後の演劇部員表は、次のようになります。

▼表5-57 例33の実行結果

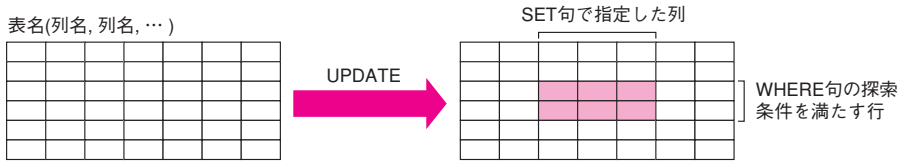
学生ID	学生名	年齢	性別	出身地
S000004	佐藤恵子	21	女	京都府
S000011	山下純一	22	男	長崎県
S000024	池田美穂	21	女	大阪府
S000025	川上香織	21	女	奈良県
S000033	大江智子	20	女	広島県
S000002	中山美樹	22	女	東京都
S000013	田中慎治	23	男	宮崎県

5.7.2 行の更新（UPDATE文）

表中の行を更新するUPDATE文の形式を次に示します。

```
UPDATE  表名
    SET  列名 = 値式 [ , 列名 = 値式 ] ...
  [ WHERE  探索条件 ]
```

UPDATEの後に行を更新する表を指定します。SET句中で、更新する列ごとに、更新する列、=、および更新値（その列の更新後の値）を組にして、コンマで区切って並べたものを指定します。最後に更新する行を選択するための探索条件をWHERE句で指定します（**図5-28**）。WHERE句を省略すると、指定した表中のすべての行が更新されます。



▲図5-28 表中の行の更新（UPDATE）

【例36】

表5-52（p.220）の演劇部員表を用いて例を示します。

演劇部の各部員の年齢に1を加えるために、演劇部員表のすべての行の年齢の列の値を更新するSQLの例を示します。

```
UPDATE 演劇部員
  SET 年齢 = 年齢 + 1
```

このUPDATE文を実行した後の演劇部員表は、次のようになります。

▼**表5-58** 例36の実行結果

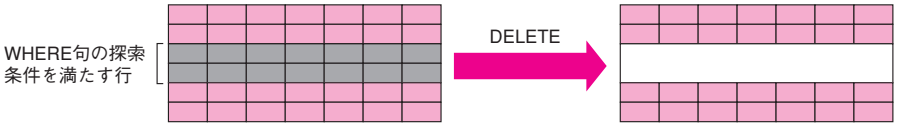
学生ID	学生名	年齢	性別	出身地
S000004	佐藤恵子	22	女	京都府
S000011	山下純一	23	男	長崎県
S000024	池田美穂	22	女	大阪府
S000025	川上香織	22	女	奈良県
S000033	大江智子	21	女	広島県

5.7.3 行の削除（DELETE文）

表中の行を削除するDELETE文の形式を次に示します。

```
DELETE  FROM  表名
  [ WHERE  探索条件 ]
```

DELETE文のFROM句で行を削除する表を指定します。削除する行を選択するための探索条件をWHERE句で指定します（**図5-29**）。WHERE句を省略すると、指定した表中のすべての行が削除されます。



▲**図5-29** 表中の行の削除（DELETE）

【例37】

表5-52（p.220）の演劇部員表を用いて例を示します。

山下純一が演劇部を退部したので、演劇部員表から該当する行を削除するSQLの例を示します。

```
DELETE FROM 演劇部員 WHERE 学生名 = '山下純一'
```

このDELETE文を実行した後の演劇部員表は、次のようになります。

▼**表5-59** 例35の実行結果

学生ID	学生名	年齢	性別	出身地
S000004	佐藤恵子	21	女	京都府
S000024	池田美穂	21	女	大阪府
S000025	川上香織	21	女	奈良県
S000033	大江智子	20	女	広島県

5.8 カーソルを用いたデータ操作

5.8.1 カーソルと埋込みSQL

SQLを直接起動し、対話的に実行するアプリケーションに対しては、すでに説明したSQLを入力して、実行結果を出力することができます。しかし、アプリケーションプログラムにSQLを埋め込んでデータベースにアクセスし、そのデータに対する何らかの処理を行う場合には、そのプログラムは、問合せの結果集合を取り出すために**カーソル**を用いる必要があります。

これは、SQLが一般に非手続き的な言語であり、複数行からなる行の集合を処理するのに対して、一般のプログラム言語は、手続き的な言語であり、一度に複数のデータをまとめて処理しないからです。

ただし、問合せの結果が必ず1行以下になることがあらかじめわかっている場合には、FROM句の前に結果を受け取るためのINTO句をもつ問合せを**1行SELECT文**として指定することができます。1行SELECT文を実行した結果が2行以上であった場合には、基数違反のエラーになります。

一般のプログラム言語にSQLを埋め込んだプログラムをSQL埋込みプログラムといい、そのSQL文またはこの機能を**埋込みSQL**、そのプログラム言語を**ホスト言語（親言語）**といいます。

埋込みSQLでは、ホスト言語中にSQL文を埋め込むときにSQL先頭子（一部のホスト言語では異なりますが、通常EXEC SQL）とホスト言語によって異なるSQL終了子で囲みます。また、ホスト言語の変数（ホスト変数）をSQL文中でも用いて、ホスト言語とSQLの間でデータを受け渡すためには、埋込みSQL宣言節中で宣言します。埋込みSQLには、この埋込みSQL宣言節を指定するための2つのSQL文があり、BEGIN DECLARE SECTION文で開始し、END DECLARE SECTION文で終了します。埋込みSQL宣言節中で宣言したホスト変数を**埋込み変数**として、SQL文中で指定するためには、ホスト識別子（ホスト変数の名前）の前にコロン（:）を付加する必要があります。そのほかに、**SQL手続文**（宣言文ではなく、実行文であるSQL文）を実行したときに例外条件（エラー、NOT FOUNDなど）が発生したときに実行する処理（GO TOによる分岐、CONTINUEによる処理続行など）を宣言するための**埋込み例外宣言（WHENEVER文）**などがあります。

なお、SQL言語でも、ストアードプロシジャなどのSQL呼出しルーチンのSQLルーチンを記述するためのSQL文（SQL/PSM）は、処理の流れを制御するための制御文