オラクルデータベース SQL 文基本知識

Oracle 基本 SQL 文

第１章　リレーショナルデータベースの概要

１－１．リレーショナルデータベース

１－１－１．リレーショナルデータベースとは

行と列によって構成された「表形式のテーブル」と呼ばれるデータの集合を、互いに関連付けて関係モデルを使ったデータベースのこと

- データベースは、データの格納の仕組みの違いから、大きく分けて４つのタイプがあります。
  1. 階層型
  2. ネットワーク型
  3. リレーショナル型
  4. オブジェクトリレーショナル型
- リレーショナル型の特徴が以下あります。
  1. データは表として格納される
  2. 表の構成要素は「行」と「列」である
  3. 複数の表を値で関連付けることができる

１－１－２．リレーショナルデータベースの用語

用語	説明
表	データを格納するもの。
行	表に格納されているデータ。社員表であれば１行が１人の社員データを表す
列	表の属性。列名とデータ型を持つ。社員表であれば社員番号や社員名の項目を表す。
フィールド	行と列の交差する部分。フィールドに値は１つのみ格納できる
NULL	フィールドに値がない状態を表している。スペースや数字の０とは異なる。

- 複数の表の関連付け

  リレーショナルデータベースの特徴として、複数の表を関連付けることができます。関連付けは、表の主キーと外部キーによって行われます。

  1. 主キー
     主キーは、表の行を一意に識別する列です。
  2. 外部キー
     外部キーは表を関連付けるときに使用する列です。

- よく覚えておきましょう。
  1. 表内の各行は、主キーによって一意に識別できる。
  2. 主キーは１つの列または列の組合せからなり、表に１つだけ存在する。
  3. 主キーの値は一意でなければならない（値の重複は許されない）、かつ NULL は禁止される
  4. 一般的に主キーの値は変更しない
  5. 外部キーは表を関連付けるときに使用する列であり、主キーの値を参照する。
  6. 外部キーは参照する主キーの値または NULL を格納できる。

１－２．SQL の概要

１－２－１．SQL

SQL はリレーショナルデータベースを操作するための言語です。

- SQL コマンドのタイプ

タイプ	SQL コマンド	説明
データ操作言語 DML（Data Manipulation Language）	Select Insert Update Delete Merge	表の行を操作する
データ定義言語 DDL（Data Definition Language）	Create Alter Drop Rename Truncate Comment	表などのオブジェクトを作成、変更、削除する。
データ制御言語 DCL（Data Control Language）	Grant Revoke	データベースに対するアクセス権の付与や取消を行う
トランザクション制御	Commit Rollback Savepoint	トランザクションの確定や取消などの制御を行う

１－２－２．E-R モデリング

ER モデルとは、情報システムの扱う対象を、実体（entity）、関連（relationship）、属性（attribute）の三要素でモデル化したもの。リレーショナルデータベースのデータ設計などでよく用いられる。

第２章　SELECT 文の使用によるデータ取得

２－１．SELECT 文の基本構文

２－１－１．SELECT 文の機能

SELECT 文は表に格納されている行を取得します。SELECT 文には、行を取得するための三つの機能があります。

- SELECT 文の機能

機能	説明
投影／射影	表から取得した行の中から、参照したい列を選択できる機能
選択	表から条件に合った行を取得する機能
結合	複数の表を値のリンクによって 1 つのにする操作

- 本書でのデータベース（Autonomous Database）

  独自の表を作成せずにサービスの使用を開始するユーザー向けに、Autonomous Database では、読取り専用の Sales History および Star Schema Benchmark データ・セットを提供しています。これらのデータ・セットは、それぞれ Oracle Database スキーマの SH および SSB として提供されます。 ユーザーは誰でも、手動構成なしでこれらのデータ・セットを問い合せることができます。

  URL：https://docs.oracle.com/cd/E83857_01/paas/autonomous-database/adbsa/autonomous-sample-data.html#GUID-4BB2B49B-0C20-4E38-BCC7-A61D3F45390B

1. 以下 SQL でスキーマ（SH と SSB）のテーブルを確認できます。

```
SELECT OWNER, TABLE_NAME, READ_ONLY
 FROM DBA_TABLES
 WHERE OWNER IN('SH','SSB')
 ORDER BY OWNER, TABLE_NAME;
```

２－１－２．SELECT 文の基本構文

SELECT 文は、ＳＱＬの中でも使用頻度（ひんど）が高く代表的な文といえるでしょう。SELECT 文の基本構文は、SELECT 句と FROM 句のみで構成されます。

- 特徴

1. SELECT 句と FROM 句で SELECT 文は成り立つ
2. アスタリスク(*)は特殊な記号で、表のすべての列を意味する
3. アスタリスク(*)を使わない場合は、列名を具体的に指定する
4. 戻される列の表示順については、列名を直接指定する場合は、指定した順番で表示され、アスタリスク(*)を指定した場合は、表の定義の順番に表示される。
5. 表示され列見出しは、デフォルトでは列名が大文字で表示される
6. 列別名を使用することで、列見出しは変更することができる。

２－１－３．SQL 文作成時の注意点とガイドライン

ＳＱＬを作成する際には、いくつかの守るべき注意点があります。また、ＳＱＬ文実行時に発生するエラーに対するデバッグのしやすさ、ＳＱＬ文の読みやすさのためのガイドラインがあります。

- 注意点

１．キーワード、表名、列名のアルファベットの大文字や小文字の区別はない

２．キーワードの一部を省略することや、1 つのキーワードの途中で改行はできない。
３．SQL*Plus など環境によっては、文の終了を意味するセミコロン（;）を末尾につける必要がある。

- ガイドライン

１．句ごとに改行する（select 句で１行、改行して FROM 句で 1 行）
２．大文字や小文字を使い分ける（SQL キーワードを大文字で、列名や表名を小文字で記述する）
３．必要に応じてインデントを使用する

２－２．算術式

ここでは、SELECT 句における算術式の使用方法について紹介します。

２－２－１．算術式の使用

SELECT 句には四則演算を行う算術式を指定することもできます。算術式を使うと、表に存在しない列を仮想的（かそうてき）作り出すことができます。

- 算術演算子

演算子	説明
＋	加算
-	減算
*	乗算（じょうざん）
/	除算

- 算術演算子の優先順位

1. select 句には、算術演算子を使用した式を指定できる。
2. 算術演算子の優先順位は、乗算（＊）および除算（／）が加算（＋）および減算（－）よりも優先して平価される。
3. 任意の優先順位を指定するにはかっこを使用する。

２－２－２．NULL を含む算術式

NULL を含む列で式を指定する場合は注意が必要です。NULL とは、値が不明であることを表しているため、いかなる算術演算子による計算結果も NULL（不明）になります。

- 重要

1. NULL は、割り当てられていない値、不明な値、適用できない値
2. NULL が含まれている式は、いかなる算術演算子による計算結果も NULL になる（エラーにはならない）

２－３．列見出し

ここでは、SELECT 文の実行結果に表示される。列見出しの変更方法を紹介します。

２－３－１．列見出しの変更

デフォルトの列見出しがどのように表示されるか、その特徴を整理してみると、次のようになります。

１．Select 句に指定した列名を大文字で表示する。
２．指定した式をそのまま表示する。

- 列別名の指定方法

列別名の定義は、列名と列別名の間をスペースで区切る方法と、スペースの代わりに AS キーワードで列名と列別名を区切る方法があります。

✓ スペースで区切る方法

```
SELECT 列名 列別名
FROM 表名
```

✓ AS キーワードで区切る方法

```
SELECT 列名 AS 列別名
FROM 表名
```

- 重要

1. 列別名は、スペースで区切る方法と、AS キーワードで区切る方法がある
2. 列別名に次のものを含める場合は、二重引用符（にじゅういんようふ）で囲むことが必要
   ・スペースを含め
   ・大文字・小文字を区別する
   ・数字や記号で開始する

２－４．連結演算子

ここでは、列と列を連結して表示する方法を紹介します。

２－４－１．連結演算子とは

連結演算子を使うと、列を、他の列または定数（リテラル）と連結させた文字式を作ることができ、1 つの列として表示することができます。

(一) 連結演算子の使用方法

連結したい要素と要素の間に連結演算子（||）を配置します。

✓ 列と列を連結する場合

```
SELECT 列名 || 列名
FROM 表名
```

✓ 列とリテラルを連結する場合

```
SELECT 列名 || リテラル
FROM 表名
```

(二) エスケープシーケンスの使用

```
SELECT 'I'm' || 'OCI' from dual;
```

SELECT 文を実行すると、エラーが発生してしまいました。原因は「I'm」のところで指定した。単一引用符（'）がリテラルとして認識されていないためです。

エスケープシーケンスとして単一引用符(')を追加することで、エスケープシーケンスの後ろにある単一引用符(')は、リテラルとなります。

```
SELECT 'I''m' || ' OCI' AS NAME from dual;
```

(三) 代替引用符（q）演算子

代替（だいたい）引用符(q)演算子を使用すると、q の後ろの[]の間にあるものすべてをリテラルとして扱います。リテラルの範囲を示す記号として[]以外に、{}、()、<>など対の記号やシングルバイト文字を使用することもできます。

```
SELECT q'[I'm]' || ' OCI' AS NAME FROM DUAL;
SELECT q'{I'm}' || ' OCI' AS NAME FROM DUAL;
```

- 重要
  1. 連結演算子は、列を、他の列または定数（リテラル）と連結させた文字式を作る
  2. リテラルとして、単一引用符（'）を扱う場合は代替引用符(q)演算子を使用するか、または単一引用符(')を追加する
  3. 代替引用符(q)演算子は、リテラルの範囲を示す記号として、[]以外に{},(),<>など対の記号やシングルバイト文字を使用できる。

２－５．重複行の排除

ここでは、重複した行を制御する方法を紹介します。

２－５－１．重複行

SELECT 文における表示のデフォルトは、重複した値を持つ行を除外せずにすべての行が表示されます。DISTINCT オプションを使用すると、重複した値を持つ行を一意にして表示できます。

- 試験：2-5-1_DISTINCT オプションを使用する方法.sql

- 重要
  1. DISTINCT を使用すると重複した値を持つ行を一意にして表示できる
  2. DISTINCT は SELECT キーワードの直後に配置する
  3. DISTINCT の後ろに複数の列を指定すると、列の組み合わせで一意にして表示する（DISTINCT を列ごとに指定しない）

第３章　データの制限とソート

実務では、表に何千万件もしくは何億件という行が格納されます。その中から必要とする行を取得するには、select 文が持つ行の選択機能を利用します。また、取得した行を特定の基準で並べ替えて表示することもできます。本章では、select 文の行選択機能と、取得した行を並べ替える方法について解説します。

３－１．行の選択

ここでは、WHERE 句と条件式を使って行を選択する方法を紹介します。

３－１－１．WHERE 句

ある条件を満たす行だけを取り出す、つまり行を選択するには、Where 句を使用します。

- 構文

```
SELECT
FROM
```

```
WHERE 条件式
```

- 特徴
  1. WHERE 句には、WHERE キーワードに続いて、行を選択するための条件式を指定する。
  2. 表のすべての行に対して、条件式の結果が TRUE となる行が戻される
  3. WHERE 句は FROM 句の直後に指定する

３－１－２．条件式

WHERE 句には、1 つ以上の条件式を記述する必要があり、その条件式は、左辺、比較演算子、右辺の三つの要素で構成されます。

- 構文

```
列名　比較演算子｛列名 | 定数 | 値のリスト | 式｝
```

- 条件式のガイドライン
  1. 条件に使用される値が、文字または日付リテラルの場合、値を単一引用符（'）で囲む必要がある。
  2. 条件式では、列別名は指定できない。
  3. 文字リテラルのアルファベットは大文字・小文字を区別する。
  4. 日付リテラルでは書式を区分する。
  5. デフォルトの日付書式は DD-MON-RR(日-月-年下 2 桁)となっている

３－１－３．比較演算子

条件式で使用できる演算子の種類と、ＳＱＬ特有の演算子である。BETWEEN、IN、LIKE、および IS NULL について使用方法を説明します。主な比較演算子は次のとおりです。

比較演算子	意味	備考
=	等しい	
>	より大きい	
>=	以上	
<	より小さい	
<=	以下	
<>	等しくない	!=、^=も使用可能
BETWEEN a AND b	a 以上 b 以下の範囲	
IN(a,b,c,d,…)	a,b,c,d,…のいずれかと等しい	
LIKE	文字パタン―と一致する	
IS NULL	NULL である	

３－１－３－１．BETWEEN 演算子

BETWEEN 演算子は、値の下限と上限を指定して、列値がその範囲に含まれる行を検索します。

- 構文

```
WHERE 列名 BETWEEN 下限値 AND 上限値
```

- 特徴
  1. 指定した下限値および上限値も範囲に含まれる
  2. 数値、文字および日付データの範囲を検索できる

3. BETWEEN は、>= AND <=を使って書き換えることが可能
4. BETWEEN 演算子は「下限値」AND「上限値」と指定し、「上限値」AND「下限値」と指定しない
5. 検索対象の列が文字型または日付型の場合、下限値および上限値を単一引用符(')で囲む必要がある

- 試験：3-1-3-1_BETWEEN 演算子.sql

３－１－３－２．IN 演算子

IN 演算子は、値リスト内の値のいずれかと等しい行を検索します。

- 構文

```
WHERE 列名 IN(値リスト)
```

- 特徴
  1. 値リストには、値を 1 個以上リストでき、値と値の間を（,）で区切る
  2. IN 演算子は、＝OR＝を使って書き換えることが可能
  3. 検索対象の列が文字型または日付型の場合、値リストの値を単一引用符(')で囲む必要がある
- 試験：3-1-3-2_IN 演算子.sql

３－１－３－３．LIKE 演算子

LIKE 演算子は、指定した文字パターンに一致する文字を検索します。検索したい文字が正確にわからない場合や、値の一部で検索したい場合に有効です。文字パターンは、指定したリテラルとワイルドカード文字を組み合わせて作成します。

- 構文

```
WHERE 列名 LIKE '文字パターン'
```

- 特徴
  1. ワイルドカード文字に配置する場合により、前方一致、後方（ごほう）一致、部分一致という検索パターンになる
  2. リテラル文字として「%」および「_」を検索対象にするには、ESCAPE 識別子を使用する
  3. アルファベットを文字パターンに指定した場合、大文字、小文字を区別する
- 使用例：3-1-3-3_LIKE 演算子.sql

３－１－４．論理演算子

これまでに比較演算子の種類と用法を解説しました。演算子には、論理演算子というものもあります。WHERE 句に複数の条件を指定したい場合や、条件を満たさないデータを取得したい場合に論理演算子を使います。詳細紹介は省略

論理演算子	意味	結果
NOT	条件式を否定する	条件式が偽の場合、TRUE を戻す
AND	複数の条件式の論理積となる	複数の条件式がいずれも真の場合、TRUE を戻す
OR	複数の条件式の論理和となる	複数の条件式がいずれかが真の場合、TRUE を戻す

３－１－５．演算子の優先順位

ここまでで、比較演算子により値を比較する条件式を組み、論理演算子により条件式を否定する方法や条件式を結合する方法を学習しました。では、論理演算子が複数使用されている場合に、どのように評価していければよいのでしょう。詳細紹介は省略

優先順位	演算子
1	算術演算子
2	連結演算子
3	比較条件
4	IS [NOT] NULL、LIKE、[NOT] IN
5	[NOT] BETWEEN
6	<>
7	NOT 演算子
8	AND 演算子
9	OR 演算子

- 重要
  1. 論理演算子は、デフォルトの優先順位は、NOT→AND→OR の順に評価される
  2. カッコ（）で囲むことで論理演算子の有効範囲を明確に表現することや、デフォルトの優先順位を変更することができる

３－２．行の並べ替え（ソート）

SELECT 文により戻される結果に対して、特定の基準で行を並べ替えることができる ORDER　BY 句について紹介します。

３－２－１．ORDER　BY 句

検索された行の表示順序は、明示的指定しない限り決まりはありません。検索結果ををソートして表示するには、ORDER BY 句を使用します。

- 構文

```
SELECT
FROM
[WHERE 条件式]
ORDER BY{列名 | 列別名 |N}{[ASC]|[DESC]} [,列名〔[ASC]|[DESC]〕,・・・]
```

- 特徴
  1．ORDER BY 句には、ソート基準となる列名を指定する。
  2．ORDER BY 句には、列名以外に、式、列別名、SELECT 句内の位置を表す数値できる。
  3．昇順の場合、ASC キーワードを指定する。
  4．降順の場合、DESC キーワードを指定する。
  5．昇順がデフォルトの並べ替えである。
     a)　数値は小さい値が最初に表示される
     b)　日付は前の値が最初に表示される
     c)　文字はアルファベット順に表示される

６．基準となる列は複数指定可能であり、左の列からソートされる

３－２－２．NULL のデフォルトの表示位置

ソートの基準列に NULL がある場合、デフォルトの昇順では NULL は末尾に表示され、降順では先頭に表示されます。つまり NULL は「最も大きい」という扱いで表示されます。

３－２－３．NULL の表示位置制御

NULL を先頭もしくは末尾に表示するかを制御できます。ORDER BY 句の中で NULLS FIRST または NULLS LAST を指定することで、ソート列の値の昇順、降順に関わらず、NULL の表示位置を制御できます。

オプション	説明
NULLS FIRST	常に NULL を先頭に表示する
NULLS LAST	常に NULL を末尾に表示する

３－２－４．その他のソート

- 注意
  1. ORDER BY 句に列位置を指定したソートは、表に定義された列の位置ではなく、その SELECT 句内にリストした列の位置です。
- 重要
  1. ORDER BY 句には、列名、式、列別名、SELECT 句内の列位置を指定できる
  2. SELECT 句にない列を指定することも可能
  3. ORDER BY 句に複数の列を指定した場合は、左の列からソートされ、列ごとに降順／昇順を指定する

３－２－５．問合せの SQL 行制限

問合せで戻された行に対して、「最初の N 件だけを表示する」や「最初の N 件をスキップして次の X 件を表示する」というふうに出力行を制限することができます。検索された結果行から表示する行を制限します。

- 構文

```
SELECT
FROM
[WHERE]
[ORDER BY 列名 ｛[ASC] | [DESC]｝]
OFFSET オフセット行数{ROW | ROWS}
FETCH {FIRST | NEXT} {フェッチ行数 | フェッチ行割合 PERCENT} {ROW | ROWS}
{ONLY | WITH TIES}
```

- 試験：3-2-5_問合せの SQL 行制.sql

- 行制限のオプション

オプション	説明
OFFSET オフセット行数{ROW | ROWS}	・読み飛ばす行数を指定する ・負の数を指定するとオフセット０（先頭）として扱われる ・NULL または、問合せ行数以上を指定すると０行

	が戻る ・行数が 1 の場合「ROW」、2 行以上の場合「ROWS」を指定
FETCH {FIRST | NEXT}	・行を戻す位置を指定する ・オフセットしない場合は、FIRST を使用する ・オフセットする場合は、NEXT を使用する
{フェッチ行数 | フェッチ行割合 PERCENT} {ROW | ROWS}	・指定した行数を戻す場合は、「フェッチ行数 ROWS」を指定 ・指定した行割合で戻す場合は、「フェッチ行割合 PERCENT ROWS」を指定 ・1 行だけ戻す場合は、「ROW」だけを指定
{ONLY | WITH TIES}	・厳密に指定した行数を戻す場合「ONLY」を指定 ・同値が連続し、戻す行数を超えて表示する場合「WITH TIES」を指定

３－３．置換変数

ここでは置換変数について紹介します。

３－３－１．置換変数とは

SQL*Plus では、置換変数という機能により、WHERE 句の条件値を動的に指定することができます。これにより、条件値だけが異なる類似した SQL を繰り返し使用する場合に便利です。

- 構文

```
&置換変数名
```

```
&&置換変数名
```

&記号が 1 つの置換変数は、代入された値は使用後に廃棄されます。&記号が２つの置換変数は、代入された値は保持されます。

- 試験：3-3-1_置換変数.sql

３－３－２．DEFINE コマンド

DEFINE コマンドを使用すると、SQL 文実行前にあらかじめ置換変数の値をセットしておくことができます。

- 構文

```
DEFINE　変数名 = 値
```

- 特徴
  - ・DEFINE コマンドで宣言した変数は、文字型の変数として扱われる
  - ・UNDEFINE コマンドで削除するまで、値は保持される
- 試験：3-3-2_DEFINE コマンド.sql

３－３－３．VERIFY コマンド

デフォルトでは、置換変数使用時に文を実行すると置換変数に対する代入前、代入後を現す「旧」や「新」のメッセージが表示されます。この表示は SQL＊Plus 環境変数の

VERIFY 変数によって表示／非表示を制御できます。非表示にするには「SET VERIFY OFF」とします。

コマンド	説明
Set verify on	Sql 文実行時に「旧」や「新」のメッセージを表示
Set verify off	Sql 文実行時に「旧」や「新」のメッセージを非表示
Show verify	Verify コマンドの現在の設定を表示

第４章　単一行関数による出力のカスタマイズ

SQL 関数は SQL 文の中で使用され、データ項目を操作して結果を返す関数です。「アルファベット・データを大文字・小文字を気にせずに検索したい」、「文字データの長さを数えたい」、また「金額の合計を計算したい」、「平均値を求めたい」など、ユーザーのニーズを満たすさまざまな関数が用意されています。

４－１．SQL 関数

SQL 関数には、単一行関数とグループ関数の二つのタイプがあります。

・単一行関数：行ごとに 1 つの結果を戻す。
・複数行関数（グループ関数）：行グループごとに 1 つの結果を戻す。

４－２．文字関数

文字関数は、引数として文字データを受け取り、結果として文字または数値を戻します。文字関数は「アルファベットの大文字・小文字変換関数」と「文字操作関数」の二つに分類されます。

４－２－１．大文字・小文字変数関数

・UPPER 関数
・LOWER 関数
・INITCAP 関数

４－２－２．文字操作関数

この関数は、引数として文字データを受け取り、文字、または数値を戻します。

・CONCAT 関数
・SUBSTR 関数
・LENGTH 関数
・INSTR 関数
・LPAD 関数
・RPAD 関数
・REPLACE 関数
・TRIM 関数

４－３．数値関数

数値関数は、入力される引数が数値データであり、結果として戻される値が数値である関数です。主な数値関数として ROUND、TRUNC、MOD 関数の 1 つを押さえておきましょう。

・ROUND 関数：引数で指定した数値を小数点以下 N 桁に四捨五入して戻す
・TRUNC 関数：引数で指定した数値を小数点以下 N 桁に切捨てて戻す
・MOD 関数：引数で指定した数値を除算したときの余りを戻す

４－４．日付関数

日付関数は、入力される引数が日付データであり、結果として戻される値が数値または日付である関数です。また、日付に対して、算術演算子（加算、減算）を使用することができます。

・SYSDATE 関数：データベースサーバ側の現在の日時を戻す
・ADD_MONTHS 関数：日付の数ヶ月前または数ヶ月後の日付を戻す
・MONTHS_BETWEEN 関数：２つの日付の期間を月単位で戻す
・NEXT_DAY 関数：特定曜日の日付がいつであるかを戻す
・LAST_DAY 関数：月末日付を戻す

第５章　変数関数と条件式の使用方法

この章では、入力値のデータ型を別のデータ型に変換する関数や NULL を扱う関数、そして SQL 関数で IF－THEN－ELSE（条件によって変換する値を変える）ロジックを適用する条件式について解説します。

５－１．データ型の変換

SQL の処理の中では、データ型の変換が必要な場面があります。以降で具体例を示していきますが、WHERE 句における式の評価や DML 文による値の変換、あるいは日付や金額などの表示書式を変更する場合です。データ型の変換には「暗黙的な変換」と「明示的な変換」があります。

５－１－１．暗黙的なデータ型変換

特定のデータ型のデータを想定している場合で別のデータ型を受け取ったときに想定されるデータ型のデータに自動的に変換する機能です。

変換前	変換後
VARCHAR2 または CHAR	NUMBER
VARCHAR2 または CHAR	DATE
NUMBER	VARCHAR2
DATE	VARCHAR2

1. 変換関数は、式の評価、値の代入、表示書式を変換したい場合、使用する。
2. 日付を文字に変換する関数、数値を文字に変換する関数がある
3. 日付を数値に変換する関数や、数値を日付に変換する関数はない

５－１２．明示的なデータ型変換

書式を変換しての式評価や、表示される書式をデフォルトとは異なるものにしたい場合、データ型変換関数を使用します。

関数名	説明
TO_CHAR	数値型または日付型を文字型に変換する
TO_NUMBER	文字型を数値型に変換する
TO_DATE	文字型を日付型に変換する

５－２．汎用関数

その他の単一行関数として、ここでは主に NULL を扱う関数を紹介します。

関数	説明
NVL	列に含まれる NULL だけを異なる値に変換数
NVL2	列に含まれる値を NULL の場合と、NULL でない場合に分けて変換する
NULLIF	２つの値を比較し等価であれば、NULL を戻す
COALESCE（コアレス）	指定された２つ以上の列の中で、最初に検出した非 NULL の列値を戻す

５－２－１．NVL 関数

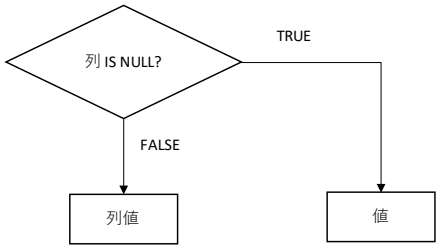
NVL 関数により、列に含まれる NULL を NULL でない値に置き換えて戻します。

- 次の特徴があります。

・NULL を NULL ではない値に置換する

・比較する列のデータ型と、置換する値のデータ型が一致している必要がある

NVL（列、値）



- NVL（列、値）の引数

引数	説明
列	列または式を指定する
値	・列が NULL 以外であれば列値を戻す ・列が NULL の場合は、指定した値を戻す

５－２－２．NVL2 関数

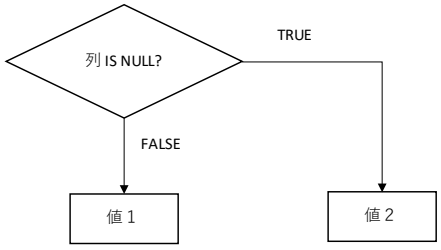
NVL2 関数により、列に含まれる値を異なる値に置き換えて戻します。列値が NULL の場合と、NULL でない場合で異なる値が戻ります。

- 次の特徴があります。

・列値が NULL の場合と、NULL ではない場合に分けて戻す値を指定できる。

・比較する列のデータ型は、戻す値のデータ型と一致している必要はない。

NVL2（列、値１、値２）



- NVL2（列、値１、値２）の引数

引数	説明

列	・列または式を指定する
値１	・列、式または定数を指定する ・列値が NULL ではない場合に戻される値
値２	・列、式または定数を指定する ・列値が NULL の場合に戻される値

５－２－３．NULLIF 関数

NULLIF 関数は、2 つの列の値を比較し、等価の場合、NULL を戻します。

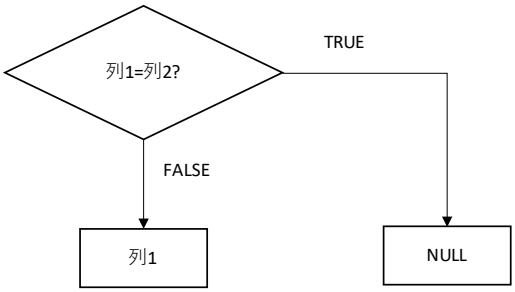
次の特徴があります。

・2 つの列値を比較し、等価であれば、NULL を戻す

・2 つの列値が非等価であれば、最初の引数として指定した列値を戻す

・引数として指定する 2 つの列のデータ型は、一致している必要がある

NULLIF（列１、列２）



NULLIF（列１、列２）の引数

引数	説明
列１	・列または式を指定する ・列２と比較する ・列２と等価の場合は NULL を戻し、非等価であれば列１の値を戻す
列２	・列１との比較のためにのみ使用される

５－２－４．COALESCE 関数

COALESCE 関数では、引数として指定した複数の列の中で、最初に検出された NULL でない値を戻します。

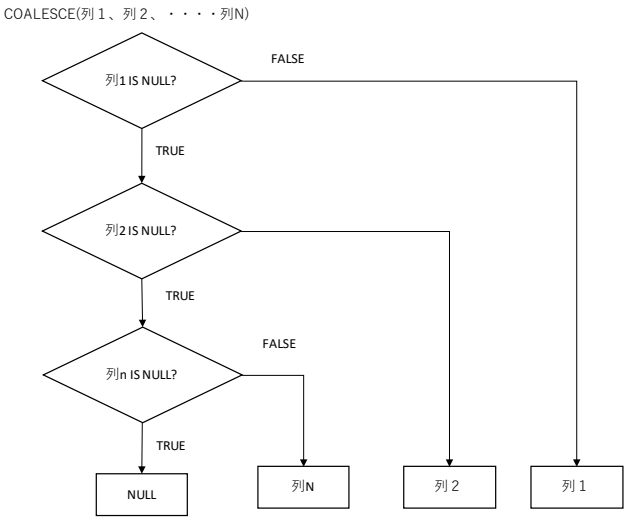
次の特徴があります。

・列１から列 n の中で検出した、最初の NULL ではない列値を戻す

・すべての列でデータ型が一致している必要がある

・引数は最低 2 つ指定する必要がある

・すべての列が NULL と評価された場合は、NULL を戻す



- COALESCE(列１、列２、・・・・列 N)の引数

引数	説明
列１	・列または、式を指定する ・この列が null ではない場合はこの列値を戻す
列２	・列または、式を指定する ・列１が null であり、この列が null でない場合はこの列値を戻す
列N	・列または、式を指定する ・先行する列が null の場合、この列値を戻す

５－３．条件式

CASE 式または DECODE 関数を使用して、SQL 関数で IF-THEN-ELSE(条件によって変換する値を変える)ロジックを適用することができます。CASE 式と DECODE 関数は同じ機能です。

５－３－１．CASE 式

- 単純 CASE 式の構文

```
CASE　列　WHEN　条件値１　THEN　値１
　　「WHEN　条件値２　THEN　値２
　　 WHEN　条件値 N　THEN　値 N
　　 ELSE　デフォルト値」
END
```

- 引数

引数	説明
列	列または、式を指定する

条件値	列＝条件値の評価が行われる
値	条件を満たした場合に戻し値
デフォルト値	いずれの条件も満たさない場合に戻す値

５－３－２．DECODE 関数

DECODE 関数は、CASE 式と同様に if-then-else の機能により、条件付きの問合せを実行できます。

- 構文

```
DECODE（列、条件値１、値１
        「,  ・・・条件値 N、値 N
          ,  デフォルト値」)
```

- 引数

引数	説明
列	列または、式を指定する
条件値	列＝条件値の評価が行われる
値	条件を満たした場合に戻し値
デフォルト値	いずれの条件も満たさない場合に戻す値

- 重要

・CASE や DECODE は、条件によって戻す値を変更する IF－THEN－ELSE の機能を持つ関数

・検索 CASE 式には副問い合わせを含めることができる

・検索 CASE 式には非等価条件（＝以外の演算子）を使用することができる。

第６章　グループ関数による集計データのレポート

この章では、グループ関数について学習します。グループ関数を使うと、クラス全体のテスト平均や、都道府県別の人口など、行のまとまり（行グループ）ごとの情報を算出できます。グループ関数の基本的な使用方法や、特定の列を基準にして行をグループ化する方法およびグループを選択する方法を理解しましょう。

６－１．グループ関数

グループ関数は、問合せの行のグループに対して作用し、行グループごとに 1 つの結果を戻します。

６－１－１．グループ関数の機能

機能	説明
基本動作	問合せによって戻される行のグループごとに、1 つの値を戻す
ネスト構造	複数の関数を組み合わせたネスト構造にできるが、階層は 2 階層まで

- 構文

```
SELECT グループ関数([DISTINCT | ALL] 列名)
FROM 表名
・・・
```

- グループ関数のオプション

オプション	説明
DISTINCT	重複を排除した値を対象とする
ALL	重複する値を含めてすべてを対象とする（デフォルト）

- 特徴
  1) DISTINCT や ALL は、列に重複した値が存在する場合の処理を指定するオプション
  2) オプションの指定を省略した場合のデフォルトは ALL となる
  3) グループ関数では NULL 以外の値を対象とする
- 重要
  1) グループ関数は、問合せによって戻される行のグループごとに 1 つの値を戻す
  2) グループ関数は SELECT 句、ORDER BY 句、HAVING 句で使用可能
  3) グループ関数は２階層までネストでき、必ず GROUP BY 句の指定が必要

６－１－２．グループ関数の種類

- グループ関数の種類

関数	説明
AVG(列名)	列の平均値
COUNT(列名)	列の値がある行数
MAX(列名)	列の最大値
MIN(列名)	列の最小値
SUM(列名)	列の合計値
LISTAGG	ORDER BY 句に指定された各グループ内でデータを順序付けメジャー列の値を結合する。

６－２．GROUP BY 句 HAVING 句

ここまでは、表全体を 1 つの行グループと捉えてきました。この節では、表を複数の行グループに分割する方法、および行グループを選択する方法について解説します。

６－２－１．GROUP BY 句

GROUP BY 句を使用して表を複数の行グループに分割することができます。

- 特徴
  1) グループ分けの基準となる列名を指定する
  2) 列別名は使用不可

６－２－２．HAVING 句

行グループの選択条件を指定するには、HAVING 句を使用します。

- 特徴
  1) グループ関数を使用した条件を指定できる
  2) SELECT 句で指定したグループ関数と異なるグループ関数を使用できる
  3) GROUPBY 句に指定した列を条件に指定できる
  4) HAVING 句と GROUP BY 句の順番を逆にすることもできる
- WHERE 句と HAVING 句の違い

句	説明
WHERE 句	・行の選択を行う ・グループ関数の前に処理を行う ・グループ関数以外なら使用できる
HAVING 句	・行グループの選択を行う

	・グループ関数の後に処理を行う ・グループ関数または GROUP BY 句で指定した列に対する条件の指定ができる

６－２－３．LISTAGG 関数

LISTAGG 関数は、1 つのフィールド内に、グループを構成している行データで値リストを作成する関数です。

LISTAGG(メジャー列名,'区切記号') WITHIN GROUP (ORDER BY 並べ替え基準列名)

- 試験：6-2-3_LISTAGG.sql

- 重要
  1) LISTAGG 関数は 1 つのフィールド内に、グループを構成している行データで値リストを作成する。
  2) メジャー列の値リストの並べ替えは可能
  3) メジャー列の値リストの表示順と行グループの表示順は、それぞれ ORDER BY 句で異なる列を指定できる。

第７章　結合を使用した複数の表のデータの表示

必要なデータが複数の異なる表に格納されている場合、表ごとに SELECT 文を作成して必要なデータを取得する方法と、表を結合する SELECT 文を 1 つ作成してデータを一度に取得する方法があります。取得したデータをプログラムの変数に保存して利用する場合などでは、表を結合する方がプログラミングをシンプルにできるなどのメリットがあります。

７－１．結合

ここまでは 1 つの表からのみ行を表示してきました。結合を使うと、複数の表の異なる行を 1 行にして表示することができます。

７－１－１．結合とは

DEPARTMENTS

No.	カラムID	型定義	NULL制約	PK
1	DEPT_NO	CHAR(4)	NOT NULL	○
2	DEPT_NAME	VARCHAR(40)	NOT NULL	

EMPLOYEES

No.	カラムID	型定義	NULL制約	PK
1	EMP_NO	NUMBER	NOT NULL	○
2	BIRTH_DATE	DATE	NOT NULL	
3	FIRST_NAME	VARCHAR(14)	NOT NULL	
4	LAST_NAME	VARCHAR(16)	NOT NULL	
5	GENDER	CHAR(1)	NOT NULL	
6	HIRE_DATE	DATE	NOT NULL	

DEPT_EMP

No.	カラムID	型定義	NULL制約	PK	
1	EMP_NO	NUMBER	NOT NULL	○	FK
2	DEPT_NO	CHAR(4)	NOT NULL	○	FK
3	FROM_DATE	DATE	NOT NULL		
4	TO_DATE	DATE			

TITLES

No.	カラムID	型定義	NULL制約	PK	
1	EMP_NO	NUMBER	NOT NULL	○	FK
2	TITLE	VARCHAR(50)	NOT NULL	○	
3	FROM_DATE	DATE	NOT NULL	○	
4	TO_DATE	DATE	NOT NULL		

SALARIES

No.	カラムID	型定義	NULL制約	PK	
1	EMP_NO	NUMBER	NOT NULL	○	FK
2	SALARY	NUMBER	NOT NULL		
3	FROM_DATE	DATE	NOT NULL	○	
4	TO_DATE	DATE	NOT NULL		

- 試験：7-2-1_結合.sql

  EMPLOYEES 表から従業員番号と DEPARTMENT 表から従業員の所属している部門名を検索する。

```
select e.EMP_NO,d.DEPT_NAME
 from EMPLOYEES e
     ,DEPT_EMP de
     ,DEPARTMENTS d
where e.EMP_NO=de.EMP_NO
   and de.DEPT_NO=d.DEPT_NO;
```

７－１－２．結合のタイプ

ANSI 規格に準拠する結合タイプとしては、内部結合、外部結合、およびクロス結合があります。

- 結合タイプ

結合タイプ	キーワード
内部結合	結合条件に一致する行だけを戻す
外部結合	結合条件に一致する行と、一致しない行も含めて戻す
クロス結合	デカルト積（行の総組合せ）を戻す

- 結合条件タイプ

結合条件	キーワード

等価結合	比較演算子にイコール（＝）を使用するもの
非等価結合	比較演算子にイコール以外を使用するもの

- そのた

その他のタイプ	キーワード
自己結合	1 つの表を使って結合するもの
自然結合	NATURAL　JOIN 構文で結合するもの
USING 句	USING キーワードで結合するもの

７－２．内部結合

複数の表の中から、結合条件に一致する行のみを取得する内部結合には、何とおりかの記述方法があります。中でも ON 句による等価結合は万能です。まずは ON 句による記述方法を習得しましょう。

７－２－１．ON 句による内部結合

内部結合は、結合条件に一致する行のみを結果として戻します。結合条件として、比較演算子にイコール（＝）を使用する等価結合を紹介していきます。ON 句による内部結合には次の特徴があります。

- ON 句に結合条件を記述する。
- 等価結合の場合としてイコール（＝）を使用する
- 「あいまいな列」は表接頭辞で修飾する
- 表名に対して表別名を割り当てることができる

- 試験：7-2-1_結合.sql

```
SELECT E.EMP_NO,D.DEPT_NAME
 FROM EMPLOYEES E JOIN DEPT_EMP DE    ON E.EMP_NO=DE.EMP_NO
                  JOIN DEPARTMENTS D ON DE.DEPT_NO=D.DEPT_NO;
```

上記の書き方を以下に変更してもよいです。↓

```
SELECT E.EMP_NO,D.DEPT_NAME
 FROM EMPLOYEES E INNER JOIN DEPT_EMP DE    ON E.EMP_NO=DE.EMP_NO
                  INNER JOIN DEPARTMENTS D ON DE.DEPT_NO=D.DEPT_NO;
```

７－２－２．NATURAL　JOIN 構文による内部結合

これまでに紹介してきた ON 句を使用した SQL 文では、結合条件を明示的に指定する必要がありました。しかし NATURAL　JOIN 構文を使用した SQL 文では結合条件を省略することができます。NATURAL　JOIN 構文には、次の特徴があります。

- NATURAL　JOIN 構文は、2 つの表で名前とデータ型が一致する列を結合列にする
- 結合条件は、結合列に基づく等価結合となる
- 自然結合と呼ばれている
- 結合列は、表名を修飾してはいけない
- 同じ名前の列が複数存在すると、それらすべてが結合条件に含まれる

・ 同じ名前の列は存在するがデータ型が異なる場合は、結合が行われずエラーが戻される

試験：7-2-2_NATURAL JOIN 構文による内部結合.sql

```
SELECT EMP_NO,DEPT_NAME
 FROM EMPLOYEES E NATURAL JOIN DEPT_EMP DE
                  NATURAL JOIN DEPARTMENTS D;
```

上記の書き方を以下に変更してもよいです。↓

```
SELECT EMP_NO,DEPT_NAME
 FROM EMPLOYEES E NATURAL INNER JOIN DEPT_EMP DE
                  NATURAL INNER JOIN DEPARTMENTS D;
```

７－２－３．USING 句による自然結合

前述のとおり NATURAL JOIN 構文は 2 つの表で同じ名前とデータ型を持つすべての列に基づく等価結合です。

名前は同じでデータ型が異なる列の場合は、USING 句を使用して等価結合に使用する列を指定できます。また、複数の列名やデータ型が一致する列があるときに、特定の列のみを結合列にすることができます。

・ 結合列の列名が同じデータ型が異なる場合に使用できる
・ 結合列を限定したい場合に使用できる

試験：7-2-3_USING 句による自然結合.sql

```
SELECT EMP_NO,DEPT_NAME
 FROM EMPLOYEES E JOIN DEPT_EMP DE USING(EMP_NO)
                  JOIN DEPARTMENTS D USING(DEPT_NO);
```

重要

1. JOIN USING は、列を指定して等価結合を行う。
2. JOIN USING は、2 つの表で結合列の名前が同じデータ型が異なる場合に使用する
3. JOIN USING は、2 つの表で列名が一致する列が複数ある場合に列を限定する場合に使用する。
4. 結合列は、文表名または表別名を SELECT 文のいかなる句でも修飾してはいけない。

７－２－４．NATURAL JOIN 構文、および USING 句における結合列に対する表接頭辞の注意点

NATURAL JOIN 構文、USING 句において結合列に表接頭辞を使用してはいけません。結合列は、双方の表に存在する列だからこそ自然結合に使われます。そのため「どちらの表の」といった表名指示を、文のどの場所でもしてはいけないのです。

試験：7-2-4_NATURAL JOIN 構文による内部結合エラー例.sql

７－２－５．ON 句による非等価結合

ON 句を使用する構文では、等価条件以外の非等価条件も指定できます。BETWEEN 演算子を使用した場合の構文は以下のようになります。

1. 結合条件としてイコール以外の比較演算子を使用できる。
2. 外部キー、および主キー以外の列で結合できる

- 試験：7-2-5_ON 句による非等価結合.sql

７－２－６．ON 句による三つ以上の表結合

結合は三つ以上も可能です。結合する表の順番は、SQL で指定した順番というわけではありません。データベースによって結合順序が自動的に判断されます。

1. 結合する表の数に制限はない
2. 結合する順番は、データベースにより決定される
3. 最低でも、結合する（表の数 - 1）分の結合条件が必要となる。

- 試験：7-2-6_ON 句による三つ以上の表結合.sql

７－２－７．ON 句による自己結合

結合は 1 つの表だけで行うことも可能です。これを同一表の結合、または自己結合といいます。

- 自己結合には、次の特徴があります。

1. 結合列である外部キー、および主キーがすべて同一の表内にある
2. 表には異なる表別名を付け、区別できるようにする

- 試験：7-2-7_ON 句による自己結合.sql

７－３．外部結合

結合条件に一致しない行を取得する必要がある場合があります。たとえば、内部結合では、部門に配属されている従業員を検索することができますが、配属先の決まっていない従業員は検索できません。配属先の有無に関わらず従業員名を確認したい場合に、外部結合が必要になります。

７－３－１．外部結合のタイプ

外部結合を明示的に指定しない限り、結合は内部結合です。内部結合とは結合条件を満たす行のみ戻される結合をいいます。外部結合は、内部結合の結果と、一方の表の結合条件を満たさない行を含めて戻します。

- 外部結合には次の特徴とタイプがあります。

1. 内部結合の結果と、一方の表もしくは、両方の表の結合条件を満たさない行が戻る
2. 結合条件を満たさない行の列値として、NULL が戻される

- 外部結合の三つのタイプ

タイプ	キーワード	説明
左側外部	LEFT OUTER JOIN	キーワードの左側に指定された表のすべての行を取得する
右側外部	RIGHT OUTER JOIN	キーワードの右側に指定された表のすべての行を取得する
完全外部	FULL OUTER JOIN	キーワードの両側に指定された表のすべての行を取得する

７－３－２．左側外部結合

左側外部結合は、キーワード（LEFT OUTER JOIN）の左側、つまり下記の構文では「表１」のすべての行を表示します。

- 構文

```
SELECT [表１.]列名, [表 2.]列名
FROM 表 1 LEFT OUTER JOIN 表２
ON 表１．外部キー＝表２．主キー
[WHERE 検索条件]
```

- 試験：7-3-2_左側外部結合.sql

７－３－３．右側外部結合

右側外部結合は、キーワード（RIGHT OUTER JOIN）の右側、つまり下記の構文では「表２」のすべての行を表示します。

- 構文

```
SELECT [表１.]列名, [表 2.]列名
FROM 表 1 RIGHT OUTER JOIN 表２
ON 表１．外部キー＝表２．主キー
[WHERE 検索条件]
```

- 試験：7-3-3_右側外部結合.sql

７－３－４．完全外部結合

完全外部結合は、キーワード（FULL OUTER JOIN）の両側、つまり下記の構文で「表１」と「表２」のすべての行を表示します。

- 構文

```
SELECT [表１.]列名, [表 2.]列名
FROM 表 1 FULL OUTER JOIN 表２
ON 表１．外部キー＝表２．主キー
[WHERE 検索条件]
```

７－３－５．クロス結合

CROSS JOIN 構文は 2 つの表のクロス積を作成します。クロス積はデカルト積ともよばれます。

- 特徴

・ 行の総組合せが戻される
・ 結合条件は不要

- 構文

```
SELECT [表１.]列名, [表 2.]列名
FROM 表 1 CROSS JOIN 表 2
[WHERE 検索条件]
```

- 試験：7-3-5_クロス結合.sql

- 結合タイプと結合条件の組み合わせ

結合タイプ	結合条件	結合構文	自己結合
内部結合	等価結合	ON 句	○
		NATURAL JOIN 構文	×
		USING 句	×
	非等価	ON 句	○
外部結合	等価結合	ON 句	○
		NATURAL JOIN 構文	×
		USING 句	×
	非等価	ON 句	○
クロス結合	-	CROSS JOIN 構文	○

- Oracle データベースでは、ANSI 規格の結合文のほか、独自の結合構文もサポートしています。Oracle 独自の結合文の特徴が次の通りです。
  1. FROM 句に結合する表をカンマ区切りで記述する（JOIN キーワードは使わない）
  2. WHERE 句に結合条件を記述する（ON 句は使わない）
  3. 結合条件を省略した場合もしくは記述が無効の場合はクロス結合（デカルト積）が行われる。

第８章　副問合せの使用方法

ある条件で表を問い合わせるとき、条件となる値が、「給与列が１００００ドル以上」などのように既知の場合もあれば、「平均給与以上」などのように未知の場合もあります。後者のような場合、副問合せを使用することで問題を解決できます。

８－１．副問合せとは

副問合せとは、SQL 文の中に埋め込まれた別の SELECT 文のことです。主問合せの検索の条件となる値を現在のデータベースから取得する場合に使用します。

８－１－１．副問合せの使い方

「平均給与より高額な給与を得ている従業員は誰なのか？」

この問題を解決するには次のような 2 つの問合せが必要です。

1. 平均給与額を求める
2. 上記「1.」の結果より高額な給与を得ている従業員を求める

- 試験：8-1-1_副問合せの使い方.sql
- 副問合せには次の特徴があります。
  1. 実行順序は、副問合せから主問合せの順序である
  2. 副問合せの結果が主問合せの検索条件に渡され、行の検索が行われます。

- ガイドライン
  1. 副問合せはカッコで囲み、主問合せの比較条件の右側に副問合せを配置する
  2. 単一副問合せでは単一行比較演算子、複数行副問合せでは複数行比較演算子を使用する
  3. 主問合せの WHERE 句の比較する列のデータ型と、副問合せの戻す値のデータ型を一致させる
  4. 主問合せの WHERE 句に複数の副問合せを指定できる
  5. 主問合せと、副問合せでそれぞれ異なる表を検索できる
  6. SELECT 文の WHRE 句に限らず SQL 文のさまざまな箇所で使用できる

- 重要
  1．副問合せは、SELECT 句、FROM 句、WHERE 句、HAVING 句、ORDER BY 句、UPDATE 文の SET 句で使用できる
  2．WHERE 句において、副問合せは比較演算子の左右どちらに記述してもよい
  3．副問合せ内に ORDER BY 句は使用できない
  4．副問合せ内にグループ関数、GROUP BY、HAVING は使用できる
  5．最大２５５レベルの副問合せをネストできる

８－２－１．単一行副問合せと複数行副問合せ

副問合せは、戻す行数や列数によって次のように分類されます。副問合せのタイプにより主問合せで使用する比較演算子が異なります。

- 副問合せの種類

タイプ	説明
単一行副問合せ	・副問合せが１行戻す問合せ ・単一行比較演算子（=、>、>=、<=、<>）が使用できる
複数行問合せ	・副問合せが１行以上戻す問合せ ・複数行比較演算子（IN、ANY、ALL）を使用できる
複数列副問合せ	・副問合せが複数列戻す問合せ ・複数の列をカッコで囲みペアにして比較する ・例）WHERE（列名１、列名２）比較演算子（select(列名１、列名２)FROM 表名...）

- 単一行比較演算子

演算子	意味
＝	等しい
＞	より大きい
＞＝	以上
＜	より小さい
＜＝	以下
＜＞、！＝、＾＝	等しくない

- 主キーに対してイコール（＝）で問合せしている場合
  試験：8-2-1_単一行副問合せと複数行副問合せ.sql

- GROUP　BY 句のないグループ関数を使用した問合せの場合
  試験：8-2-1_単一行副問合せと複数行副問合せ.sql

８－２－２．複数行複数問合せ

複数行副問合せは、副問合せが１行以上を戻す場合を想定した問合せです。複数行副問合せを含む主問合せでは、副問合せが戻す値との比較には、次に示す複数行比較演算子を使用します。

- 複数行比較演算子の種類

複数行比較演算子	説明
IN	・副問合せの結果のいずれかと等しい ・＝ANY と IN は同じ意味として動作する
ANY	・直前に単一行比較演算子（=, >, >=, <, <=, <>）が必要 ・副問合せの結果のいずれかが比較演算子を満たすもの
ALL	・直前に単一行比較演算子（=, >, >=, <, <=, <>）が必要 ・副問合せの結果のすべてが比較演算子を満たすもの

- IN を使った複数行副問合せ
  試験：8-2-2_複数行複数問合せ.sql

- ANY を使った複数行副問合せ
  - ✓ ANY を使った比較条件

比較条件	説明
列名<ANY(X,Y,Z)	・最大値より小さいものを意味する。 ・（列名<X）OR（列名<Y）OR（列名<Z）と同等
列名>ANY(X,Y,Z)	・最小値より大きいものを意味する。 ・（列名>X）OR（列名>Y）OR（列名>Z）と同等
列名=ANY(X,Y,Z)	・IN と同等 ・列名 IN（X,Y,Z）または（列名=X）OR（列名=Y）OR（列名=Z）と同等

- ALL を使った複数行副問合せ
  - ✓ ALL を使った比較条件

比較条件	説明
列名<ALL(X,Y,Z)	・最小値より小さいものを意味する。 ・（列名<X）AND（列名<Y）AND（列名<Z）と同等
列名>ALL(X,Y,Z)	・最大値より大きいものを意味する。 ・（列名>X）and（列名>Y）and（列名>Z）と同等
列名=ALL(X,Y,Z)	・副問合せが戻す値がすべて同一の場合のみ有効 ・（列名=X）and（列名=Y）and（列名=Z）と同等

- 試験：8-2-2_複数行複数問合せ.sql

８－２－３．複数列副問合せ

複数列副問合せは、副問合せが複数列を戻す問合せです。主問合せの条件に使用する複数の列をカッコで囲んで、副問合せから戻されるそれぞれの列が返す値のペアと一致する行を求めます。

- 試験：8-2-3_複数列副問合せ.sql

８－２－４．副問合せにおいて考慮すべきこと

副問合せを使った SELECT 文において、次のことを考慮する必要があります。

1. 副問合せが 1 行も戻さない場合
2. 副問合せが戻す結果に NULL が含まれている場合

- 試験：8-2-4_副問合せにおいて考慮すべきこと.sql

８－２－５．EXISTS／NOT EXISTS 演算子

前述のとおり、副問合せに NULL が含まれるときに NOT IN を使用すると、問合せ全体が行を戻しません。これは SQL 文を作成する上で知っておかなければいけないことです。しかし、副問合せの結果に NULL が存在するかどうかをすべて把握するのは困難です。

- EXISTS 演算子

  EXISTS 演算子は、表に特定の行が存在するかどうかによって結果が変化する問合せで使用します。主問合せで問合せた 1 行が、副問合せの条件を満たしている場合、TRUE として評価され、結果として戻されます。
- NOT EXISTS 演算子

  NOT EXISTS 演算子では、主問合せで問合せた 1 行が、副問合せの条件を満たしていない場合、TRUE として評価され、結果として戻されます。

- 重要
  1. 1 行だけ副問合せを単一行副問合せという。単一行副問合せは副問合せを使用できる場合なら、どこでも使用できる
  2. 複数行戻す副問合せを複数行副問合せという。比較演算子は、IN、NOT IN、>ANY、>ALL などを使用する
  3. 複数列を戻す副問合せを複数列副問合せという。比較する複数の列はカッコで囲む
  4. NOT IN 演算子で比較している複数行副問合せが NULL を戻すと、副問合せが NULL 以外の値を戻していても、結果は 0 件となる
  5. NOT EXISTS を使用すれば、複数行副問合せが NULL を戻しても大丈夫

第９章　集合演算子の使用方法

この章では、集合演算子を使用した複合問合せについて説明します。たとえば、ある表を検索して取得したコンビニエンスストアの「A 店で売れた商品の種類と数量」と、「B 店で売れた商品の種類と数量」情報から、共通して売れた商品は何か、または、A 店でのみ売れた商品は何か、ということを調べるにはどんな方法を使えばよいでしょうか。集合演算子を使用するとより単純な方法で調べることができます。

９－１．集合演算子とは

1 つの問合せによって戻される行を 1 つの集合として扱い、集合演算子を使用して、複数の集合間で論理和や論理積などの演算ができます。表の結合とは異なることを理解しましょう。結合は、2 つの表で関連する値を持つ列、いわゆる外部キーと主キーを結合条件で結合して行を戻しました。集合演算子は、任意の問合せの結果行である集合を操作した行を戻します。集合を操作するのに外部キーや主キーを必要としません。集合演算子を使用した問合せを複合問合せといいます。

９－１－１．集合演算子の種類

集合演算子は、次の４種類があります。

集合演算子	戻り値
UNION ALL	・各問合せ集合のすべての行を戻す ・重複する行は排除しない ・論理和を求める ・結果をソートしない
UNION	・各問合せ集合のすべての行を戻す ・重複する行を排除する ・論理和を求める ・結果をソートする
INTERSECT	・各問合せ集合内の共通する行だけを戻す ・重複する行を排除する ・論理積を求める ・結果をソートする
MINUS	・最初の問合せ集合と、２番目の問合せ集合を比較して最初の問合せにしか含まれていない行を戻す ・重複する行を排除する ・部分集合を求める ・結果をソートする。

例)

A 表	B 表
ID 項目	ID 項目
1	2
2	3

- 試験：9-1-1_集合演算子の書き方.sql

９－２．集合演算子の使用ガイドライン

複合問合せを記述する場合のガイドラインを示します。以下、1 つずつ説明していきます。

1. 各問合せの select 句内の列数とデータ型を一致させる必要がある
2. カッコを使用して実行順序を明示的に指定できる
3. ORDER BY 句は、文の最後にのみ指定できる
4. 集合演算子は副問合せ内で使用できる。

９－２－１．各問合せの SELECT 句内の列数とデータ型を一致させる

各問合せの SELECT 句内の列数、または対応する列のデータ型が異なる場合はエラーに

なります。

- 試験：9-2-1_各問合せの SELECT 句内の列数とデータ型を一致させる.sql

９－２－２．カッコを使用して実行順序を明示的に指定できる

集合演算子は 2 つ以上することが可能です。実行は上から下の順ですが、カッコを使用して実行順序を変更できます。

- 試験：9-2-2_カッコを使用して実行順序を明示的に指定できる.sql

９－２－３．ORDER BY 句は文の最後にのみ指定できる

ORDER BY 句は問合せ全体に対して指定するため、文あたり一度しか使用できません。

- 試験：9-2-3_ORDER BY 句は文の最後にのみ指定できる.sql

- 集合演算子では最初の問合せの列名または列別名が、結果の列名に使用されます。よって、ORDER BY に指定できるのは、先頭の問合せに記述した列名、式、列別名および列の位置を表す数値

９－２－４．集合演算子は副問合せでも使用できる

集合演算子は、FROM 句、WHERE 句など副問合せが使えるところで使用できます。

- 試験：9-2-4_集合演算子は副問合せでも使用できる.sql

第１０章　データ操作文を使用した表の管理

ここまでに学習した SQL はすべて SELECT（データ操作文）でした。本章では表に行を追加する INSERT、列の値を更新する UPDATE、行を削除する DELETE について解説します。また COMMIT や ROLLBACK コマンドを使ったトランザクションの制御方法についても解説します。

１０－１．データ操作文

表に新しい行を追加したり、既存の行を更新したり、行を削除する SQL 文をデータ操作文（DML）と呼びます。

１０－１－１．INSERT 文の基本構文

- 構文

```
INSERT INTO 表名（列名１,列名２,・・・）
VALUES（値１、値２、・・・）；
```

- NULL の追加

INSERT 時に列に追加する値が未定の場合はどうすればよいでしょう。列の値が未定の場合は NULL として追加できます。（NULL を登録できるのは、列に NOT NULL が設定されていないことが前提です）

- NULL を指定する方法が 2 つあります。
  1. 暗黙的方法：列リストから列を省略

2. 明示的方法：NULL キーワードもしくは空の文字列「''」(単一引用符 2 つ)を指定する。

- 試験：10-1-1_NULL の追加.sql

- INSERT 文実行時のエラー

INSERT 文でよく起こるエラーについて以下あります。

1. 列の数と値の数が不一致のためエラー
2. 列のデータ型と値のデータ型が不一致のためエラー
3. NOT NULL 列に NULL を指定したためエラー
4. 主キー制約に違反する値を指定したためエラー

- その他の INSERT

1. VALUES 句における関数の使用
2. 置換変数を使った対話式の INSERT 文
3. 別の表からの行のコピー

INSERT INTO・・・VALUES 句の構文では、一度に 1 行しか追加できません。副問合せ付きの INSERT 文を使用すると、一度に複数行を追加することができます。INSERT 文に副問合せを記述すると、表から取得した行がそのまま、INSERT 文により追記できます。

- 試験：10-1-1_その他の INSERT.sql

１０－１－２．UPDATE 文の基本構文

表の既存行の値を変更するには、UPDATE 文を使用します。使用に際して、次の注意点があります。

1. WHERE 句の条件を満たす行のみ値が変更される
2. 条件を満たす行がない場合、「0 行が更新されました。」というメッセージが表示される
3. WHERE 句を省略すると、表内のすべての行の値が変更される

- 構文

```
UPDATE 表名
SET 列名１＝値[,列名２＝値、・・・]
[WHERE 条件式];
```

- 副問合せによる UPDATE 文

Set 句の右辺に副問合せを指定することができます。使用できる副問合せは、単一副問合せのみです。（必ずイコール「=」を使用するためです）

- 試験：10-1-2_副問合せによる UPDATE 文.sql

１０－１－３．DELETE 文の基本構文

表の行を削除するには、DELETE 文を使用します。

- 構文

```
DELETE [FROM ] 表名
[WHERE 条件式];
```

- 副問合せによる DELETE 文

WHERE 句に副問合せを使用することもできます。

- 試験：10-1-3_副問合せによる DELETE 文.sql
- 重要

・WHERE 句に副問合せ（単一行、複数行、単一列、複数列すべて可能）を使用することができる

・WHERE 句を記述しないと表内のすべての行が削除される

１０－１－４．TRUNCATE 文の基本構文

TRUNCATE 文は表を切捨てます。TRUNCATE 文の特徴と構文は次のとおりです。

・行がすべて削除される

・ロールバック情報を生成しないため高速に行を削除できる

・データ定義文（DDL）であるためロールバックできない。

- 構文

```
TRUNCATE TABLE 表名;
```

１０－２．トランザクション

データベースシステムは多くのユーザに同時使用されることを想定して設計されています。そのため、1 つのデータが複数のセッション（ユーザ）から変更される場合、データに不整合が起こらないようにする機能が必要です。リレーショナルデータベースでは、データの不整合が起こらないように「トランザクション」という単位でデータの変更を行っています。

１０－２－１．トランザクションとは

トランザクションとは、データベースにおいてデータを変更する 1 つの単位です。

１０－２－２．トランザクション制御文

トランザクション制御文を使用すると、トランザクションの終了が宣言されます。

- トランザクション制御文のタイプ

トランザクション制御文	説明
COMMIT	現在のトランザクション内の変更（dml）をすべて確定し、トランザクションを終了する
ROLLBACK	現在のトランザクション内の変更（dml）をすべて取消し、トランザクションを終了する
SAVEPOINT セーブポイント名	現在のトランザクションにセーブポイントのマーク付けをする
ROLLBACK TO セーブポイント名	現在のトランザクション内の変更（dml）から、マーク付けより後方の dml を取り消す

- 試験：10-2-2_トランザクション制御文.sql

１０－２－３．トランザクションを構成する単位

トランザクションという単位を構成するのは、DML だけではありません。DCL や DDL は 1 文でトランザクションが終了します。

- トランザクションの構成単位

SQL 文カテゴリ	説明
データ操作文（DML）	SELECT、UPDATE、INSERT、DELETE 文など任意の数で１トランザクションを構成できる
データ定義文（DDL）	CREATE、ALTER、DROP、TRUNCATE 文など 1 つの DDL で１トランザクションが構成される
データ制御文（DCL）	GRANT、REVOKE 文など 1 つの DCL で１トランザクションが構成される

- 試験：10-2-3_トランザクションを構成する単位.sql

１０－２－４．トランザクションの開始と終了

DDL や DCL は、文それ自体にトランザクションの開始と終了を含んでいますが、DML によって構成されるトランザクションは何によって開始し、何によって終了するのでしょうか。

- DML のトランザクションの開始と終了

トランザクション	操作
開始	・最初の DML により暗黙で開始される
終了	次のいずれかによって終了 ・COMMIT 文または ROLLBACK 文の発行（トランザクション制御文の発行） ・DDL または DCL の発行（自動コミット） ・ユーザによる SQL*Plus、SQL*Developer の終了 ・マシン障害やシステム・クラッシュの発生

１０－２－５．トランザクションの同時実行性

省略

１０－２－６．セーブポイント

省略

１０－２－７．読取り一貫性

「トランザクション終了前の状態」の項で、あるセッションで変更中の値は、コミットされるまでは別のユーザからは参照できず、変更前の値が戻されることを説明しました。この機能を読み取り一貫性といいます。読み取り一貫性はデータベースにより自動的に実現されます。

読み取り一貫性の目的は、SELECT 文を実行する際に、SELECT 文開始時点の、確定している値が各ユーザに表示されるように保証することです。

読み取り一貫性を確保するために、データの変更時には、変更開始前のデータのコピーが UNDO セグメントに書き込まれます。読み取り一貫性は、UNDO セグメントに格納されている変更開始前のデータにより行われます。

１０－２－８．SELECT 文の for UPDATE 句

Oracle は通常 SELECT 文で選択された行に対してロックはかけません。しかしプログラム内で行を変更する前にロックを必要とする場合、SELECT 文に FOR UPDATE 句を指定すると、選択した行に対してロックをかけることができます。

- 構文

```
SELECT 列名１、列名２・・・
FROM 表名
[WHERE 条件式]
FOR UPDATE [OF 列名 [NOWAIT | WAIT ｎ] ]
```

- 特徴
  1. デフォルトでは、既に他のセッションによってロックがかけられている場合、トランザクション終了まで待ちになる
  2. NOWAIT を指定すると、既に他のセッションによってロックがかけられている場合はロック待ちをせずにエラーで返す
  3. WAIT を指定するとエラーで返すまでに待機する秒数を指定できる
  4. デフォルトでは、結合の場合、問合せ対象のすべての表の行にロックがかかる
  5. FOR UPDATE OF 列名を指定した場合は、指定した列名を保有する表の行のみがロックされる
  6. ロックは COMMIT または ROLLBACK を発行するまで保持される
- 試験：10-2-8_SELECT 文の for UPDATE 句.sql

第１１章　データ定義言語の紹介

システム開発の中では、データベース設計を行い、開発対象のシステムにどのような表が必要なのかを明らかにします。さらに、非機能要件を満たすために、索引やビューの設計など表以外のものも設計します。この机上で設計された表をデータベースに実装するためには、データ定義文（DDL）を使用します。DDL としての構文は各ベンダーで共通ですが、表名および列名の命名規則や、データ型などには違いがあります。

１１－１．データベースオブジェクト

表やビュー、索引などデータベースの中で扱われるものを「データベースオブジェクト」と呼びます。1 つの表は 1 つのデータベースオブジェクトです。主要なオブジェクトを次にまとめます。

- 主要なオブジェクト

オブジェクト	説明
表	・行（データ）を格納する ・データベースの基本となるオブジェクト

ビュー	・SELECT 文に名前を付けて保存したものしたもの ・データアクセスを制限する
順序	・一意な数値を生成する
索引	・問合せのパフォーマンスを向上させる
シノニム	・オブジェクトの別名を指定する

１１－１－１．命名規則

- 命名規則は以下のとおりです。
  1. 英数字（A から Z、a から z、0 から９）、記号（$、_、#）が使用可能
  2. 文字で開始する必要がある
  3. 長さは３０バイト以下
  4. Oracle の予約語は使用できない
  5. 同じユーザが所有する別のオブジェクトと重複する名前は使用できない
  6. 大文字／小文字は区別されない

１１－１－２．データ型

表を作成するときには、それぞれの列に対してデータ型を指定します。Oracle では、文字型、数値型、日付型、期間型、バイナリ型などのデータ型があります。主要なデータ型を次にまとめます。

- 主要なデータ型

分類	データ型	説明
文字型	CHAR[(size)]	固定長文字データ
	VARCHAR2(size)	可変長文字データ
	LONG	可変長文字データ（最大２GB）
	CLOB	最大サイズは（４GB-1）*（DB_BLOCK_SIZE）
数値型	NUMBER[(p,s)]	可変長数値データ
日付型	DATE	日付と時刻の値
バイナリ型	RAW(size)	RAW バイナリデータ
	LONG RAW	RAW バイナリデータ
	BLOB	最大サイズは（４GB-1）*（DB_BLOCK_SIZE） ８TB から１２８TB
	BFILE	外部ファイルに格納されるバイナリデータ（最大４GB）

1. CHAR(size)：固定長の文字データ型です。Size は１バイト～2000 バイトまで指定可能です。サイズを省略した場合は１バイトです。CHAR 型は固定長の文字データとして取り扱われ、挿入された値の桁数が、指定した桁数に満たない部分は空白が充填（じゅうてん）されます。
2. VARCHAR2(size)：可変長の文字データ型です。size は１バイト～4000 バイトまで指定可能です。サイズを省略することはできません。CHAR との違いは入力した文字列データのみが格納される点です。このため、データベースの容量を節約できます。
3. LONG：可変長の文字データです。最大２GB まで格納可能です。サイズを指定することはできません。また、LONG 列には次のような制限があることを理解しましょう。
   - １つの表あたり１列しか定義できない（LONG 型と LONG RAW 型のいずれか１つ）

・ 制約を指定できない
・ GROUP BY 句および ORDER BY 句で使用できない
・ 副問合せを指定して表を作成した場合、コピーされない

4. NUMBER(p,s)：数値データ型。P は、最大精度で３８桁まで使用可能です。S は、位取り（くらいどり）（少数点以下の桁数）を指定します。精度と位取りを省略すると、最大３８桁の浮動小数点として使用可能です。S を省略すると、ｐの精度の整数として使用可能です。例えば：NUMBER(7)と指定すると整数７桁として扱われ、NUMBER(7,2)と指定すると整数部５桁、小数部２桁として扱われます。

１１－２．表の作成

表を作成する CREATE TABLE 文について説明します。表を作成するには前提としてユーザが CREATE TABLE システム権限を持っていることと、表を格納するための記憶域が必要です。

１１－２－１．CREATE TABLE 文

表の作成には、DDL の 1 つである、CREATE TABLE 文を使用します。

```
CREATE TABLE [スキーマ.]表名
(
列名１　データ型(サイズ)　[DEFAULT 値]
[,列名 2　データ型(サイズ)　[DEFAULT 値],・・・]
)
```

- CREATE TABLE 文の構文の説明

構文	説明
CREATE TABLE	・表を作成するときのキーワード
スキーマ	・表を所有するスキーマ名を指定する
表名	・命名規則に反しない名前を指定する
列名	・表内で一意の列名を指定する ・1 つ以上の列を指定する必要がある
データ型(サイズ)	・列に格納するデータの形式およびサイズをデータ型名として指定する
DEFAULT オプション	・列のデフォルト値を指定する ・デフォルト値は、列のデータ型と一致している必要がある ・デフォルト値は、リテラル値、式または sql 関数を指定できる ・デフォルト値は、他の列名や擬似（ぎじ）列などは使用できない

- 重要

1. 表の作成時には、表名、列定義（列名、データ型、サイズ）を設定する
2. DEFAULT オプションでは、列のデフォルト値として次のものが指定できる
   ① SYSDATE、CURRVAL、NEXTVAL
3. 表の作成（CREATE TABLE）は自動的にコミットされる
4. 表が作成されたことを確認するには DESCRIBE(DESC)コマンドを使用する。

5. DESCRIBE コマンドでは、表の列名、データ型、サイズおよび null を許可するかどうかを確認できる。

１１－３．制約の設定

表は、列のデータ型およびサイズに反しなければ、どのような値でも格納できます。制約を設定すると、「列には一意の値が格納されるようにする」、「列には値を必ず登録する」などのビジネスルールを実現することができます。

１１－３－１．制約とは

制約は、表に格納可能な値を制限するルールです。制約は表に行を挿入、更新、削除するときに実施され、制約に違反した値を使用した文をエラーにします。

- 制約のタイプ

制約のタイプ	説明
NOT NULL	列に NULL 値を入力できないことを指定する
UNIQUE	列（または列の組み合わせ）の値が重複しないことを指定する
PRIMARY KEY	表の主キーを指定する。主キーにより表の各行を一意に識別する
FOREIGN KEY	表の外部キーを指定する。外部キーにより列の値が、別の表（または同じ法の一意の値と一致することを指定する。
CHECK	列の値が満たす必要がある式を指定する

１１－３－２．制約の定義

制約は表の作成と同時に設定できます。あるいは既存の表に制約のみ設定することもできます。

制約の定義方法には「列レベル」と「表レベル」の２とおりの構文がありますが、違いは主に構文であり制約の機能に変わりはありません。

- 重要
  1. 制約のタイプは、NOT NULL、UNIQUE、PRIMARY KEY、FOREIGN KEY、CHECK の五つ
  2. 制約構文には「列レベル」と「表レベル」がある。設定された制約の機能は同じ
  3. NOT NULL 制約は列レベルでのみ定義可能
  4. 1 つの CREATE TABLE 文の中で、列レベルの構文と表レベルの構文の両方を使用できる

- 列レベル構文

  列レベル構文の特徴は、列の定義の一部として制約を設定する点です。列定義の直後に制約を設定します。

```
CREATE TABLE 表名
(列名 1　データ型(サイズ)　[CONSTRAINT 制約名] 制約タイプ,
列名 2　データ型(サイズ)　)
```

例）

```
CREATE TABLE A
```

```
(ID NUMBER(10) CONSTRAINT A_PK PRIMARY KEY,
 NAME CHAR(10)
);
```

- 表レベル構文

表レベル構文の特徴は、表の定義の一部として制約を設定する点です。すべての列定義の後に制約を設定します。列定義とは、別に制約を定義するため、制約タイプの直後に制約列名を指定します。複数制約や既存の表に制約のみ設定する場合、表レベル構文でしか定義できません。

- 構文

```
CREATE TABLE 表名
(列名 1　データ型(サイズ),
列名 2　データ型(サイズ),
[CONSTRAINT 制約名] 制約タイプ(制約列名 1 [,制約列名 2,・・・]),・・・）;
```

例)

```
CREATE TABLE A
(ID NUMBER(10),
 NAME CHAR(10),
 CONSTRAINT A_PK PRIMARY KEY(id)
);
```

- 制約の定義のガイドライン

1. 制約は表の作成と同時、または表の作成後に設定できる
2. 構文上制約名の定義は省略可能であり、省略した場合は SYS_Cn 形式による一意の制約名が生成される（小文字 n の部分に一意な数値が割り当てられる）
3. 1 つの CREATE TABLE 文の中で、列レベルの構文と表レベルの構文の両方を使用できる。

１１－３－３．NOT NULL 制約

NOT NULL 制約を定義すると、列に NULL が含まれないことが保証されます。

- 試験：11-3-3_NOT NULL 制約.sql

１１－３－４．UNIQUE 制約

UNIQUE 制約を定義すると、列または、列の組み合わせの値が一意であることが保証されます。但し、NULL はこの制約の対象外であるため、NULL が複数行に設定されても重複とはみなされません。

- 試験：11-3-4_UNIQUE 制約.sql

１１－３－５．複数制約

複合制約は、列の組み合わせに対して定義された制約です。複合制約は UNIQUE 制約、PRIMARY KEY 制約、および FOREIGN KEY 制約に対して定義することができます。
それぞれ複合 PRIMARY KEY 制約、複合 UNIQUE 制約、複合 FOREIGN KEY 制約と呼びます。
複合制約は単一の属性では行を一意に識別できない場合に使用します。列の組み合わせで値の一意を保証しますので、単一の列だけで見た場合は重複した値を格納できます。

- 試験：11-3-5_複合制約.sql

１１－３－６．PRIMARY KEY 制約

PRIMARY KEY 制約を定義すると、その列または列の組み合わせが一意であること、かつ NULL を設定することができないことが保証されます。また PRIMARY KEY 制約を定義した列に対して自動的に一意索引が作成され、索引により列の値の一意性が効率良く施行されます。PRIMARY KEY は表に 1 つだけ定義できます。

- 重要

1. PRIMARY KEY 制約により表の主キーが定義される
2. 主キーを定義した列または、列の組み合わせに NULL や重複値を設定できない。
3. 主キーは表に 1 つのみ作成できる。
4. PRIMARY KEY 制約は表レベル、列レベルの両方の構文を使用できる。
5. 複合主キーを定義する場合は、表レベルの構文を使用する必要がある。
6. PRIMARY KEY 制約は 1 つの表に 1 つのみ作成できる
7. 複合 PRIMARY KEY 制約は、構成する列の一部にでも NULL があってはならない

- 試験：11-3-6_PRIMARY KEY 制約.sql

１１－３－７．FOREIGN KEY 制約

FOREIGN KEY 制約は外部キーとなる列または列の組み合わせに定義します。外部キーは、主キー（PRIMARY KEY 制約を定義した列）を参照します。リレーショナルデータベースでは、外部キーと主キーを関連づけることにより表を結合することができます。

- 列レベルの構文

```
CREATE TABLE 表名
(列名１　データ型（サイズ）[CONSTRAINT 制約名] REFERENCES 参照表[(参照列)],
列名２　データ型（サイズ）
)
```

- 表レベルの構文

```
CREATE TABLE 表名
(列名１　データ型（サイズ）,
列名２　データ型（サイズ）,
[CONSTRAINT 制約名] FOREIGN KEY(外部キー列) REFERENCES 参照表[(参照列)]
);
```

- 試験：11-3-7_FOREIGN KEY 制約.sql

１１－３－８．FOREIGN KEY によるデータ整合性の維持

FOREIGN KEY 制約を定義した影響として、次のような値のチェックが行われるようになります。

1. 子表に格納できる値は、親表の既存の値に一致するか、NULL である
2. 子表で使用されている値を親表で変更（UPDATE）および削除（DELETE）できない

- 試験：11-3-8_FOREIGN KEY によるデータ整合性の維持.sql

１１－３－９．自己参照型の FOREIGN KEY 制約

FOREIGN KEY 制約の参照表としてその表自体を設定することができます（自己参照型)。つまり、子表と親表が同じ表という関係です。このような関係は、上司や部下または、会社組織（そしき）の部や課などの親子関係に見られます。

- 試験：11-3-9_自己参照型の FOREIGN KEY 制約.sql

１１－３－１０．ON DELETE CASCADE と ON DELETE SET NULL

FOREIGN KEY 制約と組み合わせて使用できるオプションも押さえておきましょう。参照先のデータ（主キー列）に対する操作の結果として、依存するデータ（外部キー列）がどのような影響を受けるかを指定します。

- FOREIGN KEY 制約のオプション

キーワード	受ける影響
ON DELETE CASCADE	親表の行が削除されると、依存する子表の行もすべて削除される
ON DELETE SET NULL	親表の行が削除されると、依存する子表の外部キーが NULL に設定される
オプションなし	依存するデータがある場合、親表の行の更新または削除を禁止する

- 試験：11-3-10_FOREIGN KEY 制約のオプション.sql

- 重要
  1. 外部キーが参照できるのは、主キー（PRIMARY KEY を定義した列）または一意キー（UNIQUE 制約を定義した列）
  2. FOREIGN KEY 制約（外部キー）は 1 つの表に複数作成できる
  3. 外部キーを定義する列と参照先の主キー列は、名前とデータ型は異なっていても構わない
  4. 1 つの列に主キーと外部キーを同時に定義できる
  5. 外部キー制約に ON DELETE CASCADE を付けた場合、親表の行が削除されると、依存する子表の行もすべて削除する
  6. 外部キー制約に ON DELETE SET NULL を付けた場合、親表の行が削除されると、依存する子表の外部キーの値を NULL に更新する

１１－３－１１．CHECK 制約

CHECK 制約は、列の値が満たす必要がある任意の条件式を指定します。CHECK 制約では、他の制約のように一意であるとか、NULL を禁止するなどの特定のチェックをするのではなく、制約定義時に指定した任意の条件でチェックすることができます。また、1 つの列に対して CHECK 制約を複数定義することができます。その数に制限はありません。1 つの列に定義した CHECK 制約はすべて満たす必要があります。

- 試験：11-3-11_CHECK 制約.sql

- 重要
  1. CHECK 制約には BETWEEN 演算子を使用できる

2. 1 つの列に複数の CHECK 制約を定義できる
3. CHECK 制約を定義した列に同時に NOT NULL 制約を定義していなければ、NULL の登録は CHECK 制約違反にならない
4. CHECK 制約には、次の条件式は使用できない
   (ア) CURRVAL、NEXTVAL 擬似列および SYSDATE 関数
   (イ) 他の行の値を参照する問合せ

１１－４．副問合せを使用した表の作成

副問合せの結果に基づいて表を作成することができます。この方法では、表を作成することだけでなく、検索した行も格納することができます。表定義および行データをコピーする方法ともいえます。

１１－４－１．CREATE TABLE AS SELECT 文

副問合せに基づく表作成では、CREATE TABLE AS SELECT 文を使用します。AS 以降の副問合せの内容に基づき列の定義や行の挿入が行われます。

- 構文

```
CREATE TABLE [スキーマ.]表名
[(
列名１　データ型（サイズ）[DEFAULT 値]
[,列名２　データ型（サイズ）[DEFAULT 値],・・・]
)]
AS
SELECT 文;
```

- ガイドライン
  1. 列定義には、列名、デフォルト値、制約を含めることができる
  2. 列定義の列数と副問合せの列数は同数にする必要がある
  3. 列名を指定しない場合、副問合せ内の列名で定義される
  4. 副問合せ内の SELECT 句に式を含む場合、列定義で列名を指定するか、または列別名を指定する。
- 重要

  列のデータ型、サイズ、および明示的に指定された NOT NULL 制約が新しい表に複製される。
- 試験：11-4-1_CREATE TABLE AS SELECT 文.sql

１１－５．ALTER TABLE 文による表の変更

表作成後、何らかの理由で表の定義を変更する場合があります。その時に使用する DDL 文が ALTER TABLE 文です。

１１－５－１．ALTER TABLE 文

ALTER TABLE 文では、次のことができます。

1. 列の追加
2. 既存列のデータ型、サイズの変更
3. 列のデフォルト値の定義
4. 列の削除
5. 列名の変更

6. 表を読み取り専用に変更

１１－５－２．列の追加

ADD 句により列を追加できます。

- 構文

```
ALTER TABLE [スキーマ.]表名
ADD
(
列名１ [DEFAULT デフォルト値][CONSTRAINT 制約名 制約タイプ][,]
[列名 n [DEFAULT デフォルト値][CONSTRAINT 制約名 制約タイプ][,]]
)
```

- ALTER TABLE 文の構文の説明

構文	説明
ALTER TABLE	・表を変更するときのキーワード
ADD	・表に列を追加するときのキーワード
列定義	・列名、データ型およびサイズは必須 ・デフォルト値、制約を含めることができる ・列定義全体をカッコで囲む必要がある

- ガイドライン
  1. 新しい列は表の一番後ろに追加される（SELECT * とした場合に最も右側に表示される）
  2. 行を含む表にデフォルト値を指定しない列を追加する場合、NULL が格納される
  3. 行を含む表に NOT NULL 制約を定義した列を追加する場合、デフォルト値を定義する必要がある

- 試験：11-5-2_列の追加. sql

１１－５－３．列の変更

MODIFY 句により列のデータ型、サイズおよびデフォルト値を変更できます。

- 構文

```
ALTER TABLE [スキーマ.]表名
MODIFY
(列名１ [データ型(サイズ)][DEFAULT 値][NULL | NOT NULL]);
```

- MODIFY 句を使用した構文の説明

構文	説明
ALTER TABLE	・表を変更するときのキーワード
MODIFY	・表に列を変更するときのキーワード
列定義	・データ型の変更ができる ・サイズの拡大または縮小（しゅくしょう）ができる ・デフォルト値を定義できる ・NOT NULL 制約の追加または削除ができる

- ガイドライン
  1. 数値型についてサイズおよび精度の拡大は常に可能である

2. 文字型についてサイズ拡大は常に可能である
3. 数値型のサイズ縮小は、列に NULL だけが格納されているとき、または表が空の場合可能である
4. 文字型のサイズ縮小は、現在格納している値未満でなければ可能である
5. 列に NULL だけが格納されている場合、データ型を変更できる
6. 列に NULL だけを含み、またはサイズを変更しない場合、CHAR と VARCHAR2 との間で相互（そうご）に変更ができる
7. デフォルト値の変更は、それ以後の INSERT 時に影響する

- 試験：11-5-3_列の変更.sql

１１－５－４．列の削除

DROP 句により既存の列を削除できます。

- 構文（単一列の削除）

```
ALTER TABLE [スキーマ.]表名
DROP COLUMN 列名[CASCADE CONSTRAINT];
```

- 構文（複数列の削除）

```
ALTER TABLE [スキーマ.]表名
DROP (列名１[,列名２]・・・[,列名 N]) [CASCADE CONSTRAINT];
```

- DROP 句を使用した構文の説明

構文	説明
ALTER TABLE	・表を変更するときのキーワード
DROP	・列を削除するときのキーワード ・単一の列を削除する場合 COLUMN キーワードを指定し、カッコは使用不可 ・複数列削除の場合 COLUMN キーワードを指定せず、カッコで囲む必要がある

- ガイドライン
  1. 列に値が含まれるか、含まれないかに関わらず削除できる
  2. COLUMN キーワードを使用すると１列のみ削除できる
  3. 一度に複数列を削除する場合、COLUMN キーワードを指定せず削除対象列をカッコで囲む
  4. 外部キーにより参照されている列を削除する場合、「CASCADE CONSTRAINT」オプションを指定する
  5. 列を削除した後は元に戻すことはできない。
  6. 最低１列は残す必要がある
  7. 列の削除中は表にロックがかかる
- 試験：11-5-4_列の削除.sql

１１－５－５．SET UNUSED オプション

SET UNUSED 句を使い既存の列を UNUSED(未使用)にできます。

- 単一列を未使用にする構文

```
ALTER TABLE [スキーマ.]表名
SET UNUSED COLUMN 列名 [CASCADE CONSTRAINT][ONLINE];
```

- 複数列を未使用にする構文

```
ALTER TABLE [スキーマ.]表名
SET UNUSED (列名１[,列名２]・・・[,列名 N])[CASCADE CONSTRAINT];
```

- ディスク領域削除の場合

```
ALTER TABLE [スキーマ.]表名
DROP UNUSED COLUMNS;
```

- SET UNUSED 句を使用した構文の説明

構文	説明
ALTER TABLE	・表を変更するときのキーワード
SET UNUSED	・列を UNUSED にするときのキーワード ・単一の列を UNUSED にする場合 COLUMN キーワードを指定し、カッコは使用不可 ・複数列を UNUSED にする場合 COLUMN キーワードを指定せず、カッコで囲む必要がある
CASCADE CONSTRAINT	・参照整合性制約を削除するときのキーワード ・外部キーにより参照されている主キーまたは一意キー列を削除または未使用にする場合に必要
ONLINE	・表への DML 操作を許可する

- ガイドライン
  1. SET UNUSED オプションでは、列は論理的に削除された状態になっている
  2. ディスク領域は削除されないため、表にかかるロック期間は短い
  3. UNUSED 列にはアクセスできない。SELECT * 問合せでも、UNUSED 列からデータを取り出すことはできない
  4. UNUSED 列の名前および型は DESCRIBE コマンドでは表示されず、未使用列と同じ名前の新しい列を表に追加できる。
  5. UNUSED 列を元に戻すことはできない。
  6. 表に存在する UNUSED 列の数は「USER_UNUSED_COL_TABS」データディクショナリで確認できる
  7. UNUSED 列のディスク領域削除（物理削除）のために DROP UNUSED COLUMNS オプションを使用する。
  8. ONLINE オプションにより、DML 実行中に SET UNUSED オプションを使用できる

- 試験：11-5-5_SET UNUSED オプションで列の論理削除.sql

１１－５－６．読み取り専用モードの表

READ ONLY 句を使い、表を読み取り専用モードにできます。

- 読み取り専用モードへの変更

```
ALTER TABLE [スキーマ.]表名
READ ONLY
```

- 読書き可能モードへの変更

```
ALTER TABLE [スキーマ.]表名
READ WRITE
```

- READ ONLY 句、READ WRITE 句を使用した構文の説明

構文	説明
ALTER TABLE	・表を変更するときのキーワード
READ ONLY	・読み取り専用モードにする ・DML によるデータの変更は禁止される ・DDL による表の定義変更が禁止される
READ WRITE	・読書き可能モードにする ・DML によるデータの変更は可能になる ・DDL による表の定義変更は可能になる

- ガイドライン
  1. 読み取り専用モードのとき、ＤＭＬによる表の変更および、SELECT FOR UPDATE の発行は禁止される
  2. ＤＤＬによる制約追加、DEFAULT オプションの追加など表データの変更を伴う定義変更は禁止される
  3. ALTER TABLE MOVE オプションなどの DDL 文は実行できる
  4. 読み取り専用モードにある表を DROP TABLE 文で削除可能である

- 試験：11-5-6_読み取り専用モードの表.sql

１１－５－７．表の削除

DROP TABLE 文で表を削除します。

- 構文

```
DROP TABLE [スキーマ.]表名
[CASCADE CONSTRAINTS][PURGE]
```

- DROP TABLE 文の構文の説明

構文	説明
DROP TABLE	・表を削除するときのキーワード
CASCADE CONSTRAINTS	・参照整合性制約を削除するときのキーワード ・外部キーにより参照されている主キーまたは一意キーを含む表を削除する場合に必要
PURGE	・このオプションを指定すると表は完全に削除される ・このオプションを省略すると表はごみ箱へ入る

- ガイドライン
  1. 表構成と行データが削除される

2. DROP TABLE 文実行前のトランザクションはコミットされ、その後表が削除される
3. 索引や制約などの依存するオブジェクトは表の削除と同時に削除される
4. ビュー、シノニムなどのオブジェクトは表が削除されても残るが無効となる
5. 表を削除できるユーザは、表の所有者または DROP ANY TABLE システム権限の保有者である
6. PURGE オプションの指定がない場合、削除した表はごみ箱に入る
7. ごみ箱にある表は、FLASHBACK TABLE 文により削除を取り消し元に戻すことができる
8. PURGE オプションを指定すると削除した表はごみ箱に入らず完全に削除される