SQLコーディング規約（Oracle）

Future Enterprise Coding Standards

Table of Contents

本コーディング規約は、世の中のシステム開発プロジェクトのために無償で提供致します。  
ただし、掲載内容および利用に際して発生した問題、それに伴う損害については、フューチャー株式会社は一切の責務を負わないものとします。  
また、掲載している情報は予告なく変更することがございますので、あらかじめご了承下さい。

# はじめに

## 前提条件

本書は、SQLコーディング規約についてまとめたものである。  
今回RDBMSとして採用するOracleでのSQLの使用を前提に記述している。

# SQLコーティング規約（可読性・管理性）

本章では可読性・管理性を高めることを目的としたコーディング規約について記載する。

## 書式全般

書式全般についてのコーディング規約を下記に示す。

* 1行につき、1文のみを記述する。
* SQL中のインデントは、Javaコーディング規約にあわせて半角スペースではなくタブ文字とする。
* ヘボン式ローマ字を使用する。
* 外来語に関しては、原語の綴りを使用する。
* 横は80文字を目安に改行する。
* 定数を条件に用いる場合やインライン・ビューで取得したいデータなど開発者の意図はコメントにて記載する。

## 予約語

予約語に対しては、大文字を使用する。(例 ： SELECT、INSERT、UPDATE、DELETE 等)

## 予約語以外

予約語以外に対しても、予約語と同様に大文字を使用する。(例 ： オブジェクト名、カラム名 等)

## 短縮名称

SQL中に記述するエイリアス名など単語の短縮について示す。

1. 外来語に関しては、原語の短縮形を使用する。短縮形が存在しない場合には、母音を抜かして表記する。  
   例) corporation → corp / computer → cmptr
2. ローマ字の短縮は、単語の区切れの頭文字、または母音を抜かした子音字等を利用する。  
   例) nichijo → nchj

* カラムには必ずテーブルエイリアスを付与する
* テーブルのエイリアスは必ず付与すること。  
  必要ない場合（単一テーブルへのSELECT等）も必ず付与すること  
  また、テーブルのエイリアス名は同SQL文の中で重複しないように命名すること。  
  （副問い合わせで利用したエイリアス名をメインのSQL中のエイリアス名に利用しない。など）

## 文字コード

SQLファイルの文字コード（エンコーディング）はJavaソースファイルと同じく『 Unicode UTF-8 』で保存する。

## 不要な空白文字（スペース）は除去する

不要な空白文字（スペース）は除去すること。

## SQL文の整形

DML文の節に対する予約語は左揃えにする。  
項目ごとに改行を入れ、項目の前にはインデントを挿入する。カンマは項目の前へ記入する。  
Javaソースファイルのようにファイルの先頭にコメント行を入れるとDB分析作業に支障があるため禁止とする。  
よってSQLファイルの先頭は必ずSELECT、UPDATE、INSERT、DELETE、MERGEの何れかになる。

物理カラム名、テーブル名に対応する論理名を入れる場合、その後ろに単数行コメント（--）にて記述する。  
SQL内に挿入する単数行コメントは、/\*（半角スペース）コメント本文（半角スペース）\*/ で行う。  
,（カンマ）とANDについては各行の先頭に記述する。（以下の例を参照のこと）  
SQLフレームワークで実行するSQLの場合、SQLステートメントの終わりを示す;（セミコロン）は記述しない。

良い例：

SELECT  
/\*  
 処理名  
\*/  
 TBL1.COLUMN1 AS COLUMN1 -- カラム1  
, TBL1.COLUMN2 AS COLUMN2 -- カラム2  
FROM  
 TABLE1 TBL1 -- テーブル1

WHERE句の=、!=、IS（IS NULLのIS）の縦位置は揃える。

良い例：

WHERE  
 A.TEN\_NO = B.TEN\_NO  
AND A.KAMOKU\_CD != '1'  
AND A.ANKEN\_NO IS NULL

### SELECT文

良い例：

SELECT  
 TBL1.COLUMN1 AS COLUMN1 -- カラム1  
, TBL1.COLUMN2 AS COLUMN2 -- カラム2  
FROM  
 TABLE1 TBL1 -- テーブル1  
WHERE  
 TBL1.COLUMN3 = 1  
ORDER BY  
 TBL1.COLUMN1  
, TBL1.COLUMN2

### INSERT文

良い例：

INSERT  
INTO  
 TABLE1 -- テーブル1  
(  
 COLUMN1 -- カラム1  
, COLUMN2 -- カラム2  
, COLUMN3 -- カラム3  
) VALUES (  
 VALUE1  
, VALUE2  
, VALUE3  
)

### UPDATE文

良い例：

UPDATE  
 TABLE1 TBL1 -- テーブル1  
SET  
 TBL1.COLUMN2 = 100 -- カラム2  
, TBL1.COLUMN3 = 100 -- カラム3  
WHERE  
 TBL1.COLUMN1 = 10

### DELETE文

良い例：

DELETE  
FROM  
 TABLE1 TBL1 -- テーブル1  
WHERE  
 TBL1.VALUE = 1

### MERGE文

DB結合条件が一致した場合と不一致になった場合の、それぞれのSQL文を記述する。

良い例：

MERGE  
INTO  
 PROFILE P -- PROFILEテーブル  
USING  
 NAME N -- NAMEテーブル  
ON  
 (  
 P.NAME = N.NAME -- 条件  
 )  
WHEN MATCHED THEN -- 条件が一致  
 /\*UPDATE文\*/  
WHEN NOT MATCHED THEN -- 条件が不一致  
 /\*INSERT文\*/

### AND（副問い合わせ）

良い例：

AND XXX = (  
 SELECT  
 TBL1.COLUMN1  
 , TBL1.COLUMN2  
 FROM  
 TABLE1 TBL1  
 )

### CASE式

CASE式は下記のように記載すること。

良い例：

CASE  
 WHEN  
 XXX.HOGE = YYY.FUGA  
 AND XXX.FUGA = YYY.FUGA  
 THEN  
 1  
 ELSE  
 0  
END

CASE、WHEN、THEN、ELSEの後に改行を挿入すること。  
CASEの後、ENDの前までは1インデント挿入すること。

### IN句

カンマの後にスペースを１文字入れる。

### 比較演算子

比較演算子の前後にタブ（またはスペース）を１文字入れる。

### 改行位置

SELECT句、ORDER BY句、GROUP BY句等は最初に出現するカラムとカラムの区切りのカンマ前に改行を入れること。  
SELECTのFROM句の最初に出現するテーブルと結合テーブルの区切りのカンマ前に改行を入れること。  
WHERE句、MERGEのON句の各条件文の（ANDやORの）前に改行を入れること。  
命令句の後は、ヒント句が挿入できるように改行すること。

良い例：

SELECT  
 T1.COLUMN1 AS COLUMN1  
, T2.COLUMN2 AS COLUMN2  
FROM  
 TABLE1 T1  
, TABLE2 T2  
WHERE  
 T1.COLUMN3 = 1  
AND T1.COLUMN4 = T2.COLUMN4  
ORDER BY  
 T1.COLUMN1  
, T2.COLUMN2

### WITH句

WITHの前後に改行を挿入すること  
また、インデントは下記のように記述すること

良い例：

-- カラムエイリアスあり   
WITH  
 NAME1 (  
 COL\_ALIAS1 -- カラム１  
 , COL\_ALIAS2 -- カラム２  
 ) AS (  
 SELECT  
 ・・・  
 )  
, NAME2 (  
 COL\_ALIAS1 -- カラム１  
 , COL\_ALIAS2 -- カラム２  
 ) AS (  
 SELECT  
 ・・・  
 )  
   
-- カラムエイリアスなし   
WITH  
 NAME1 AS (  
 SELECT  
 ・・・  
 )  
, NAME2 AS (  
 SELECT  
 ・・・  
 )

### OFFSET句 FETCH句

OFFSET、FETCHの前に改行を挿入すること

良い例：

SELECT  
 TBL1.COLUMN1 AS COLUMN1  
FROM  
 TABLE1 TBL1  
ORDER BY  
 TBL1.COLUMN2 DESC  
OFFSET 5 ROWS  
FETCH NEXT 5 ROWS ONLY

### HINT句

HINT句は独立した行で記載すること  
HINT内容にはインデントを付けること

良い例：

SELECT  
/\*+  
 ここにHINTを記載  
\*/  
 TBL1.COLUMN1 AS COLUMN1  
FROM  
 TABLE1 TBL1  
WHERE  
 TBL1.COLUMN2 = 1

## コメント

* 修正コメント
* （修正コメントが必要な場合、）  
  処理追加の際、追加行の1行目の前と最終行の次の行にコメントを入れる。単一行の場合は、同一行の最後にコメントをつける。
* 良い例：
* -- 2004/04/23 仕様変更管理番号 ADD(または、MOD、DEL) 変更者名 S ←　修正開始点コメント   
  (追加処理)   
  -- 2004/04/23仕様変更管理番号 ADD(または、MOD、DEL) 変更者名 E ←　修正終了点コメント
* 単数行コメント
* SQL内で使用する単数行コメント（カラムコメントなど）には、「 -- 」を使用する。
* 複数行コメント
* 「 /\* \*/ 」を使用する。下記に例を示す。  
  なお、前述で触れたとおり、SQLファイルの先頭にコメントを記述することは禁止とする。
* 良い例：
* /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
  /\*  
   \* コメントを始めるスラッシュとアスタリスクは、それだけを１行に置く。  
   \* それから、コメント・ブロック内の各行は縦にアスタリスクを置き  
   \* コメントがあることを強調する。  
   \* 最後に、アスタリスクとスラッシュは、それだけを１行に置く。  
   \*/  
  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
* 複数行コメントアウト
* 複数行をコメントアウトする場合は、各行を「--」でコメントアウトする。  
  「 /\* \*/ 」を使用すると、その中に「 /\* \*/ 」が存在した場合、コメントアウトが途中で切れてしまう恐れがあるため、  
  使用しない。
* 論理名の記載
* SELECT・INSERT・UPDATE・MERGEのカラム名記述箇所には単数行コメントでカラムの論理名を記載する。  
  SELECT・INSERT・UPDATE・DELETE・MERGEのテーブル名記述箇所には単数行コメントでテーブルの論理名を記載する。  
  論理名はERD等で定義された論理名と必ず一致させること。

## 外部結合

結合方法はANSI形式（～outer join ～）ではなくOracle形式(+)を使用する。  
原則として(+)は条件文の右にくるカラムに付与する。

良い例：

T1.COL1 = T2.COL2(+)

## 型変換

代入やWHERE句の条件設定などでデータ・タイプが異なる場合、明示的にタイプ変換を行う。（暗黙の型変換は使用しない）

## 比較演算子

「等しくない」を示す演算子は「!=」を利用する。「<>」「^=」も動作するが統一の観点から利用しない。

## ORDER BY句

ORDER BY 数字の構文は使用せずに、カラム名を記載する。

## GROUP BY句

GROUP BY 数字の構文は使用せずに、カラム名を記載する。

集約関数を利用する場合は必ず記載すること。（省略可能であっても省略しない）

## EXISTS句

EXISTS句を記載する際、サブクエリになるSELECT句の指定は定数「1」とする。  
「\*」（ワイルドカード）や「'X'」は統一の観点から利用しない。  
また「\*」（ワイルドカード）についてはパフォーマンスの観点からも禁止とする。

良い例：

WHERE  
 EXISTS(  
 SELECT  
 1  
 FROM  
 FOO F  
 WHERE  
 F.COL1 = M.KEY  
 )

## AS句

トップレベルのSELECT句には必ずAS句を記載し別名を付ける。  
同一の名前であってもAS句を付与する。  
また、「AS」は省略可能であるが、省略はしないこと。

## WHERE句

* 論理名の記載  
  WHERE句でカラムと式を比較する際は左辺がカラムになるように記載すること。
* 良い例：
* WHERE  
   TBL.COLUMN1 = 1  
  AND TBL.AMOUNT2 > TBL.AMOUNT3 + TBL.AMOUNT4
* 条件式の順序  
  原則として、WHERE句で条件式を列挙する際、下記の順序を守ること。
  1. テーブル単位にまとめて順番に記述する  
     この際、テーブルの順序はFROM句に記述した順序に準ずること。
  2. 1.のテーブル単位の中で絞り込み条件をまず記述し、その後結合条件を記述する。
* 良い例：
* FROM  
   A\_TABLE A -- A\_TABLE  
  , B\_TABLE B -- B\_TABLE  
  , C\_TABLE C -- C\_TABLE  
  WHERE  
  -- A\_TABLEの絞り込み  
   A.KEY1 = ?  
  AND A.KEY2 = ?  
  -- B\_TABLEの絞り込み  
  AND B.KEY1 = ?  
  AND B.KEY2 = ?  
  -- B\_TABLEの結合条件  
  AND B.COL1 = A.COL1  
  -- C\_TABLEの絞り込み  
  AND C.KEY1 = ?  
  AND C.KEY2 = ?  
  -- C\_TABLEの結合条件  
  AND C.COL1 = A.COL1  
  AND C.COL2 = A.COL2  
  AND C.COL3 = B.COL3

## COUNT文

レコード数を数えるCOUNT文の記述はCOUNT(\*)と記述する。  
COUNT(1)・COUNT('X') ・COUNT(KEY1)という記載はNG。

# SQLコーディング規約（パフォーマンス性）

本章ではパフォーマンス性を高めることを目的としたコーディング規約について記載する。

## 検索

検索処理におけるコーディング規約を下記に示す。

* 中間一致、後方一致検索はインデックスを利用できないため避ける
* 検索条件で=（等号）を使用できる場合は必ず使用する
* A=1 or A=2とする方がA>0 and A<3などと記述するのよりパフォーマンス上優位な場合が多い。  
  これはAにインデックスがある場合、=であれば、インデックスが有効に使われるためである。
* 可能な限り検索条件にパーティションキーの値を指定する
* 全列ワイルドカード「\*」の使用はせず、カラム名を明記する
* インデックスによる検索を指定したい場合、下記の記載を行わない
  + インデックスカラムを含む演算に対して条件指定
  + 悪い例：
  + TBL1.COL1 + 1 > 100 /\* NGパターン 右辺で演算してください \*/
  + 良い例：
  + TBL1.COL1 > 100 - 1
  + インデックスカラムに関数を通した値に対して条件指定
  + 悪い例：
  + /\* NGパターン 右辺に関数を通してください \*/   
    TO\_CHAR(TBL1.COL1, 'YYYYMMDD') > '20151231'
  + 良い例：
  + TBL1.COL1 >= TO\_DATE('20160101', 'YYYYMMDD')
  + インデックスカラムをORで条件指定（禁止ではないが原則行わない）
  + 悪い例：
  + (  
     /\* NGパターンINDEXが利用されない場合があります。他の方法を検討してください \*/  
     TBL1.COL1 = 1  
     OR TBL1.COL1 = 2  
     )

## 挿入

挿入処理におけるコーディング規約を下記に示す。

* 全列ワイルドカード「\*」の使用はせず、カラム名を明記する。

## 更新

更新処理におけるコーディング規約を下記に示す。

* 主キーの値のUPDATEは原則行わない。外部キーがあればエラーになる。  
  外部キーが無い場合でも、事実上、主キーの値を利用して、検索、更新する場合は、リンクが切れてしまう。
* パーティションキーのUPDATEは原則行わない。
* VIEWを使用するデータ更新は禁止。更新は実表に対して行う。

## 削除

削除処理におけるコーディング規約を下記に示す。

* 大量件数（数百万件レベル）のdelete文発行は避ける。

## WITH句

WITH句の誤った使い方はパフォーマンスの劣化を招くため、WITH句を利用する場合はDBAに相談する。

## DISTINCT句

DISTINCTは、暗黙のソート処理が行われる可能性があるため性能劣化につながる。  
EXISTS句の使用・代替を検討すること。

悪い例：

SELECT  
 DISTINCT  
 D.NO AS DEPT\_NO  
, D.NAME AS DEPT\_NAME  
FROM  
 DEPARTMENT D  
, EMPLOYEE E  
WHERE  
 D.NO = E.DEPT\_NO -- 社員が一人以上属している部門を取得

良い例：

SELECT  
 D.NO AS DEPT\_NO  
, D.NAME AS DEPT\_NAME  
FROM  
 DEPARTMENT D  
WHERE  
 EXISTS(  
 SELECT  
 1  
 FROM  
 EMPLOYEE E  
 WHERE  
 D.NO = E.DEPT\_NO -- 社員が一人以上属している部門を取得  
 )

## IN句

IN句は最大1000個まで指定できるが、200個程度でもORAエラーが発生するケースがある。  
またIN句の少しだけ異なるSQLが大量に発行されるとCPU高騰やメモリ枯渇を招く。  
従って100を超えるような長いIN句は使用せず、一時表を利用して IN (SELECT ・・・ FROM 一時表)のように書き換える。

## NOT IN句

原則NOT IN（SELECT～）は使用せずに、NOT EXISTS (SELECT～)を使用する。  
NOT IN句は、内部的にソートマージの結合をすることでテーブルをフルスキャンする場合があるため、性能が悪化する可能性がある。

## UNION句

UNIONはDISTINCT処理が含まれパフォーマンス上問題があるため、UNION ALLを使用する。

## パラレルヒント句

パラレルヒント句はDB負荷が高いため、原則使用禁止とする。付与したい場合はDBAに相談する。

## SELECT FOR UPDATE

* SELECT FOR UPDATEはNO WAITまたは「WAIT秒数指定」を必ず付ける。  
  WAIT指定なしの場合はロックが解除されてもプログラムに制御が返らないことがある。  
  ※WAIT秒数指定を行う際の秒数は各開発者で決めるのではなくプロジェクトで決定した方針に従うこと。  
  また、SQLライブラリを利用していて定数が記述できる場合は定数で記述すること。
* SELECT FOR UPDATEで複数行にロックをかける場合、同時実行されるとデットロックを起こす可能性があるため、１件のロックでない場合はORDER BYを指定する。

## 分析関数

分析関数の使用は可能だが、性能悪化を招く恐れがあるため、特性を知らない方はDBAに相談する。

## インデックス

インデックスの必要性についてはDBAで最終判断するため、必要とする場合はDBAへ相談する。

## 外部結合

外部結合する際、内部表（駆動表）はなるべく想定件数の少ない表にする。