目次

[データベース資料 2](#_Toc395006559)

[１．リレーショナルデータベース 2](#_Toc395006560)

[－１．テーブル 2](#_Toc395006561)

[－２．データの不整合 2](#_Toc395006562)

[－３．制約（ Constraint ） 2](#_Toc395006563)

[２．トランザクション 3](#_Toc395006564)

[３．ACID（アシッド）特性 3](#_Toc395006565)

[－１．Atomicity(原子性) 3](#_Toc395006566)

[－２．Consistency (一貫性) 3](#_Toc395006567)

[－３．Isolation(独立性) 3](#_Toc395006568)

[－４．Durability(永続性) 3](#_Toc395006569)

[４．分離（Isolation）レベル 4](#_Toc395006570)

[５．テーブルの正規化 5](#_Toc395006571)

[－１．第１正規化 5](#_Toc395006572)

[－２．第２正規化 6](#_Toc395006573)

[－３．第３正規化 7](#_Toc395006574)

[６．SQL 8](#_Toc395006575)

[－１．CREATE　TABLE 9](#_Toc395006576)

[－２．SELECT 11](#_Toc395006577)

[－３．CREATE　VIEW 14](#_Toc395006578)

[－４．INSERT　INTO 14](#_Toc395006579)

[－５．DELETE 15](#_Toc395006580)

[－６．UPDATE 15](#_Toc395006581)

[－７．ALTER　TABLE 15](#_Toc395006582)

# データベース資料

## １．リレーショナルデータベース

リレーショナルデータベースは、実際に最も多く使用されています。データを行（タプル）、と列（アトリビュート、属性）の二次元の表（テーブル）として表現します。1行に1件のデータを、1列には１つだけの値を格納します。

特定の行を取り出す操作（演算）を「選択（Select）」、特定の列を取り出す操作（演算）は「射影（Projection）」、二つの表を一つにする操作（演算）を「結合（Join）」と言います。この３つはリレーショナルデータベースで最も重要なデータ操作（演算）と言えます。

### －１．テーブル

ANSI　SPARC　3層スキーマの「概念スキーマ」に該当するのが、テーブルです。データを行（タプル）と列（アトリビュート、属性）の二次元の表として実装します。

### －２．データの不整合

データ更新不整合には修正不整合、挿入不整合、削除不整合があります。データ更新不整合の発生を防ぐには、正規化されたテーブル設計をし、テーブルに適切な制約を設定する必要があります。

### －３．制約（ Constraint ）

テーブルに制約を設定することで、矛盾した挿入、削除、更新などのデータ操作を禁止することができます。基本情報では主キー（Primary　Key）制約、外部参照（Foreign　Key）制約、ユニーク制約（Unique）、チェック制約（Check）などが出題されます。それぞれがどのような特性を持つか理解できるようにしましょう。特に主キー、外部参照制約はSQL文での文法・記法も含めてしっかり理解しておきましょう。

|  |  |
| --- | --- |
| 制約 | 機能・役割 |
| PRIMARY KEY  主キー制約 | テーブルのタプル（行）を一意に特定できる列、または列の組み合わせ（複合キー）に設定する。同時に「UNIQUE」「NOT NULL」制約も設定される。 |
| FOREIGN KEY  外部参照制約 | 他テーブルの行との関係付けをするために設定する。他テーブルの列を参照する側のテーブルを「子表」　参照する列を外部キーと言い、他テーブルから参照される側のテーブルを「親表」　参照される列を親キーと言う。外部キーを定義するには「親表」が定義済みで、親キーにはPRIMARY KEY又はUNIQUEが設定されていなければならない。  一度設定すると親表の親キーに存在しない値を子表の外部キーの列が持つことはできない（NULL値は除く）。また、子表から参照されている親表の行は削除できない。 |
| NOT NULL | 指定した列のNULL値の入力を禁止する。指定された列には必ず値を入力しなければならない。 |
| UNIQUE  一意キー制約 | 指定した列（又は列の組み合わせ）に対して、同値の入力を禁止する。一般的（製品により差異あり）にはNULLは適用外。NULL値の行が複数あってもよい。 |
| CHECK | 明示的に条件式を指定して、条件式を満たす（真）値のみ入力を許す。 |

## ２．トランザクション

コンピュータシステムにおいては、多くのテーブルを読み込んだり、更新することで1件のデータ処理が完結します。1件のデータ処理内で行われるデータ操作は不可分単位になります。データベースではこれをトランザクション処理と呼んでいます。

トランザクションはCOMMIT（コミット）か、ROLLBACK(ロールバック)のいずれかで終了させます。COMMITで終わった一連のデータ操作は確定となり、全てデータベースに反映され、永続性が保障されます。ROLLBACKで終わった一連のデータ操作は全て無効となり、データ操作前の状態の戻ります。

トランザクションが問題なく完結した時は、COMMITで、何らかのエラーなどが発生して不完全なデータ操作となってしまう時はROLLBACKで、トランザクションを終了させます。

## ３．ACID（アシッド）特性

トランザクション処理においては、４つの特性Atomicity(原子性)、Consistency (一貫性)、Isolation(独立性)、Durability(永続性)が満たされている必要があります。4つの特性の頭文字からACID特性と呼ばれています。

### －１．Atomicity(原子性)

トランザクションが終了したとき、すべて処理が終了している。もしくは、処理が行われていない状態のどちらかでなくてはいけない。途中で処理が止まっているなど中途半端な状態であってはいけない。

### －２．Consistency (一貫性)

トランザクションの状態に関係なく、データベースに矛盾がない状態が継続されていなくてはならない。

### －３．Isolation(独立性)

複数のトランザクションを実行した場合でも、それぞれの処理に影響を与えないで処理されなくてはならない。

### －４．Durability(永続性)

処理されたトランザクションは、データベース障害が発生しても、データの内容が失われてはならない。

## ４．分離（Isolation）レベル

データベースでは複数のトランザクション処理が同時に同じデータへの更新や読み取りなどのデータ操作をおこないます。

データ不整合を防ぐためには排他制御が必要となります。排他制御では分離（Isolation）レベルを適切に設定しなければいけません。トランザクション同士の影響を少なくする（分離レベルを上げる）とパフォーマンスは低下し、トランザクション同士の影響を大きくする（分離レベルを下げる）とパフォーマンスは向上します。

分離レベルと、データ一貫性において発生する事象は以下の表のとおりです。



・ダーティリード

コミット前の更新データが参照されること

・ノンリピータブルリード

トランザクション内で一度読み込んだデータを再読み込みした場合、他のトランザクションで更新されたデータに変わってしまうこと

・ファントムリード

トランザクション内で条件検索を行った後に再度同じ条件検索を行った場合、他のトランザクションの挿入操作により異なる結果になること

## ５．テーブルの正規化

データの不整合を防ぐためには、データベースの正規化を行わなければいけません。基本情報では第１正規化、第２正規化、第３正規化までを勉強しておく必要があります。

### －１．第１正規化

繰り返しや集合を排除し、1列に入る値は１つ、一事実一箇所（1 fact in 1 place）にすることです。

以下は学生が科目を履修するデータを非正規形で実装したものです。

学生科目履修（学番、氏名、科目コード１、科目名１、単位１、科目コード２、科目名２、単位２、．．．）

このテーブル（スキーマ）は｛科目コード、科目名、単位｝が集合体で繰り返し構造になっているので、非正規化です。

第1正規形にするには集合や繰り返し項目を排除して、１列（１か所）には１つの値（１事実）しか格納できない形式にする必要があります。

これを第１正規化にするとどうなるか考えてみましょう。主キーとなる列には実線の下線を引きます。

**学生科目履修（　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　）**

主キーの列に対する関数従属の関係を書いてみましょう。

**｛　　　　　　　　　　　　　｝　→　｛　　　　　　　　　　　　　　｝**

以下の受注伝票は非正規形のデータです。

受注伝票 （非正規形）



|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 伝票  番号 | 受注日 | 顧客ID | 顧客名 | 商品情報１ | | | | | 商品情報２ | | | | |
| 商品ID | 商品名 | 数量 | 単価 | 金額 | 商品ID | 商品名 | 数量 | 単価 | 金額 |
| D0001 | 2013/8/5 | K0001 | ABC商会 | S001 | テレビA | 1 | 40000 | 40000 | S002 | ラジオA | 2 | 1500 | 3000 |

以下第1正規形にする演習です。

受注伝票　→　以下の空欄に列名を記入し、主キー列には実線の下線を引きましょう。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

一事実一箇所（1 fact in 1 place）を守るように記入しましょう。

そうしなければ第1正規形にはなりませんよ　(^^♪

第1正規化した「受注伝票」は以下のようになりました。

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 伝票番号 | 受注日 | 顧客ID | 顧客名 | 商品ID | 商品名 | 数量 | 単価 | 金額 |
| D0001 | 2013/8/5 | K0001 | ABC商会 | S001 | テレビA | １ | 40000 | 40000 |
| D0001 | 2013/8/5 | K0001 | ABC商会 | S002 | ラジオA | ２ | 1500 | 3000 |

### －２．第２正規化

タプル（行）を特定できる列の極小の組み合わせを候補キーと言います。複数の候補キーから一つだけ主キーを選びます。主キーとなった列（あるいは列の組み合わせ）には必ず値を設定する必要があり（NOT NULL制約）、更に重複も許可されません（UNIQUE制約）。

非キーの列が主キーに完全関数従属するのが、第2正規形です。複数列から構成される主キーは複合キーと言いますが、第2正規形では複合キーの一部だけに関数従属（部分関数従属）する非キー属性が存在してはいけません。

第1正規形に分割した以下の概念スキーマは主キー｛学番、科目コード｝の一部である学番と科目コードに

学番→氏名、科目コード→｛科目名、単位｝の部分関数従属があり、第2正規形ではありません。

学生科目履修（学番、氏名、科目コード、科目名、単位）

このままではどんな問題が発生すると思いますか？　→　追加、削除、更新

第2正規形にするとどうなるか考えてみましょう。

**学生（　　　　　　　　　　　　　　　　）**

**科目（　　　　　　　　　　　　　　　　）**

**履修（　　　　　　　　　　　　　　　　）**

この正規化を行うと推移的関数従属（第3正規化で説明します）がないので、第3正規形にもなっています。

第1正規形にした「受注伝票」は｛伝票番号、商品ID｝が主キーとなりますが、

伝票番号→｛受注日、顧客ID、顧客名｝、商品ID→｛商品名、単価｝の部分関数従属が存在しています。

「受注伝票」を第2正規形にし、「受注伝票」と「受注明細」に分割しましょう。

主キー列には実線の下線を引き、関数従属も記述してみましょう。

受注伝票

｛　　　　　　｝　→　｛　　　　　　　　　　　　　　　　　　　　　　　　　　　　　｝

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

受注明細

｛　　　　　　　　　　　　　｝→｛　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　｝

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 伝票番号 | 伝票明細番号 |  |  |  |  |  |

第2正規形に分割したテーブルは以下のようになり、主キーの一部の列に部分関数従属の関係は排除されています。

受注伝票　　伝票番号→｛受注日、顧客ID、顧客名｝

|  |  |  |  |
| --- | --- | --- | --- |
| 伝票番号 | 受注日 | 顧客ID | 顧客名 |
| D0001 | 2013/8/5 | K0001 | ABC商会 |

受注明細　　｛伝票番号、伝票明細番号｝　→　｛商品ID、商品名、単価、数量、金額｝

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 伝票番号 | 伝票明細番号 | 商品ID | 商品名 | 数量 | 単価 | 金額 |
| D0001 | 01 | S001 | テレビA | １ | 40000 | 40000 |
| D0001 | 02 | S002 | ラジオA | ２ | 1500 | 3000 |

### －３．第３正規化

第3正規形では、推移的関数従属が存在してはいけません。A→B　、B→C　かつB ≠> Aが成り立つ時、これを推移的関数従属と言います。　第3正規化では推移的関数従属を排除します。

上述の第2正規形の「受注伝票」には伝票番号→顧客ID、顧客ID→顧客名かつ顧客名 ≠> 顧客IDが、

「受注明細」には｛伝票番号、伝票明細｝→商品ID、商品ID→｛商品名、単価｝かつ｛商品名、単価｝≠>商品IDの推移的関数従属が存在するので、これを「顧客」、「商品」の別テーブルに分割します。

主キー列には実線の下線、外部参照キー列には破線の下線を引き、関数従属も記述してみましょう。

受注伝票　　｛　　　　　　　　　　｝　→　｛　　　　　　　　　　　　　　　　｝

|  |  |  |
| --- | --- | --- |
|  |  |  |

受注明細　　｛　　　　　　　　　　　　　　　　　｝　→　｛　　　　　　　　　　　　　　　　　｝

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |

顧客　　｛　　　　　　　　　　｝　→　｛　　　　　　　　　　　　｝

|  |  |
| --- | --- |
|  |  |

商品　　｛　　　　　　　　　　｝　→　｛　　　　　　　　　　　　　　｝

|  |  |  |
| --- | --- | --- |
|  |  |  |

第3正規形に分割したテーブルは以下のようになります。全てのテーブルで主キーに主キー以外の列は完全関数従属しています。推移的関数従属もありません。

受注伝票　　伝票番号　→｛受注日、顧客ID｝

|  |  |  |
| --- | --- | --- |
| 伝票番号 | 受注日 | 顧客ID |
| D0001 | 2013/8/5 | K0001 |

受注明細　　｛伝票番号、伝票明細番号｝　→　｛商品ID、数量、金額｝

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 伝票番号 | 伝票明細番号 | 商品ID | 数量 | 金額 |
| D0001 | 01 | S001 | 1 | 40000 |
| D0001 | 02 | S002 | 2 | 3000 |

顧客　　顧客ID　→　顧客名

|  |  |
| --- | --- |
| 顧客ID | 顧客名 |
| K0001 | ABC商会 |

商品　　商品ID　→｛商品名、単価｝

|  |  |  |
| --- | --- | --- |
| 商品ID | 商品名 | 単価 |
| S001 | テレビA | 40000 |
| S002 | ラジオA | 1500 |

## ６．SQL

SQLはリレーショナルデータベースで使用される業界標準のデータベース言語です。次の３つの機能に大別されます。データ定義言語　DDL（Data Definition Language）、データ操作言語　DML (Data Manipulation Language）、データ制御言語　DCL (Data Control Language)。

|  |  |  |
| --- | --- | --- |
| 分類 | SQL文 | 機能 |
| DDL | CREATE文 | 新規にテーブルやVIEWなどを作成する。 |
| DROP文 | 既存のテーブルやVIEWなどを削除する。 |
| ALTER文 | 既存のテーブルやVIEWなどの定義を変更する。 |
| DML | SELECT文 | テーブルからデータを抽出する。 |
| INSERT　INTO文 | テーブルに行を挿入する。 |
| DELETE文 | テーブルの行を削除する。 |
| UPDATE文 | テーブルのデータを変更する。 |
| DCL | GRANT文 | セキュリティに関わる権限やロールの付与をします。 |
| REVOKE | 権限やロールの剥奪をします。 |
| COMMIT、ROLLBACK | トランザクションの制御をします。 |

リレーショナルデータベースの代表的な演算は、選択（selection）、射影（projection）、結合（join）の３つです。選択は特定行の抽出。射影は特定列の抽出、結合はリレーションシップがあるテーブルを列方向に繋げる演算です。

### －１．CREATE　TABLE

以下の構文で新規テーブルを作成できます。代表的なデータ型は数字：DECIMAL、NUMBER、文字列：CHAR(文字数)などがあります。

CREATE　TABLE　テーブル名（列名１　データ型 ，列名２　データ型 ，…）

CREATE　TABLE　テーブル名（列名１　データ型 ，列名２　データ型 ，… ，制約定義）

**制約定義**

PRIMARY　KEY（主キー列名リスト）

列名　CHAR(4)　 PRIMARY　KEY　→　列定義と一緒に記述。複合キーの指定はできない。

FOREIGN　KEY（外部参照キー列名リスト） REFERENCES　参照されるテーブル名（参照される列名）

列名　CHAR(4) REFERENCES　参照されるテーブル名（参照される列名）→列定義と一緒に記述

演習１　以下のテーブル「顧客」

|  |  |
| --- | --- |
| 顧客ID CHAR(4) | 顧客名 CHAR(20) |

CREATE TABLE 顧客 (

)

演習２　以下のテーブル「商品」を定義するCREATE　TABLE文を考えましょう。

|  |  |  |
| --- | --- | --- |
| 商品ID CHAR(4) | 単価 　DECIMAL(6) | 商品名 　CHAR(20) |

CREATE TABLE 商品 (

)

演習３　以下のテーブル「受注伝票」

※列「顧客ID」は演習１で定義したテーブル「顧客」を外部参照しています。

|  |  |  |
| --- | --- | --- |
| 伝票番号　CHAR(4) | 受注日 DATE | 顧客ID CHAR(4) |

CREATE TABLE 受注伝票 (

)

演習４　以下のテーブル「受注明細」を定義するCREATE　TABLE文を考えましょう。

※　主キーが2列から構成されています。

※　列「商品ID」は演習２で定義したテーブル「商品」の列「商品ID」を、列「伝票番号」は演習３で定義したテーブル「受注伝票」の列「伝票番号」をそれぞれ外部参照しています。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 伝票番号　CHAR(4) | 伝票明細番号 CHAR(4) | 商品ID CHAR(4) | 数量 DECIMAL(6) | 金額 DECIMAL(6) |

CREATE TABLE 受注明細 (

）

### －２．SELECT

データ抽出するSELECT文は以下の句の順番が決まっています。句の順番を覚えましょう。

SELECT　[ DISTINCT | ALL ]｛ 列名リスト ｜ ＊ ｝　FROM　テーブル名リスト

[ JOIN　結合するテーブル名　ON　結合条件 ]

[ WHERE　選択条件 ]

[ GROUP　BY 　グループ化する列名リスト ]

[ HAVING　条件式 ]

[ ORDER　BY　{ 列名リスト ｜ 列番号 } ]

GROUP BY句が使用された場合は列リストに集合関数（集計関数）が使用されます。集合関数は以下の通りです。

平均値を求める AVG( 列名 )

合計値を求める SUM( 列名 )

最大値を求める MAX( 列名 )

最小値を求める MIN( 列名 )

行数（件数）を求める COUNT( \*　｜　[DISTINCT] 列名 )

重要な演算子

LIKE ワイルドカード（　％　と　＿　）

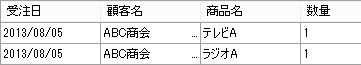
IN

EXISTS

SOME( ANY )

IS NULL

演習１－１　以下を出力するSELECT文を考えましょう。テーブル「受注伝票」「受注明細」「顧客」「商品」４つのテーブルを結合（JOIN）しなければいけません。まずは、直積（WHERE句にテーブル名リストを記述する）タイプで出力してみましょう。



受注伝票（伝票番号、受注日、顧客ID ）

受注明細　（伝票番号、伝票明細番号、商品ID、数量、金額 ）

顧客 （顧客ID、顧客名 ）

商品 （商品ID、商品名、単価 ）

SELECT a　　　　 　 , b　　　　　　 , c　　　　　 , d

FROM e　　　　　 , f　　　　　 , g　　　　　 , h

WHERE 受注明細**．** i　　　　　＝受注伝票**．** j　　　　　 AND

受注明細**．** k　　　　　＝商品**．** l　　　　　 AND

受注伝票**．** m　　　　　＝ n　　　　　**．** o

演習１－２　上記演習１－１と同じ結果をJOIN句を使用して出力してみましょう。

SELECT a　　　　 　 , b　　　　　　 , c　　　　　 , d

FROM 受注伝票　A

e　　　　　　受注明細　　B　　ON　　B． f　　　　　　＝　A． g

e　 　　　　顧客　C　　ON　　C． h　　　　 　＝　 i　　　　 ． j

e 　　　　商品　D　　ON　　D． k　　　　 　＝　 l　　　　 ． m

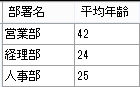
解答群

ア．受注伝票　　イ．受注明細　　ウ．顧客　　エ．商品　　オ．伝票番号

カ．受注日　　キ．顧客ID　　ク．伝票明細番号　　ケ．数量　　コ．金額

サ．顧客名　　シ．商品ID　　ス．商品名　　セ．単価

ソ．A　　タ．B　　チ．C　　ト．D　　ナ．JOIN

演習２－１　以下を出力するSELECT文を考えましょう。テーブル「社員」「所属」２つテーブルを使用します。

社員( 社員番号 ,氏名 ,年齢 ,性別コード ,所属コード )

所属( 所属コード ,部署名 )

SELECT a　　　　, b　　　　（年齢）　FROM　 社員　A

c　　　　　所属　B　　 d　　　　　A． e　　　　　＝　B． e

f　　　　　B．所属コード , B． g

解答群

ア．AVG　　イ．GROUP BY　　ウ．JOIN　　エ．ON　　オ．所属コード　　カ．部署名

演習２－２　以下を出力するSELECT文を考えましょう。テーブル「社員」「所属」「性別」３つテーブルを使用します。

社員( 社員番号 ,氏名 ,年齢 ,性別コード ,所属コード )

所属( 所属コード ,部署名 )

性別(性別コード ,性別 )

SELECT a　　　　, b　　　　 , c　　　　（年齢）　FROM　 社員　A

d　　　　　所属　B e　　 　　A． f　　　 ＝ B． f

d　　　　　性別　C e　　 　　A． g　　　 ＝ C． g

h　　　　　B． B． i　　　 ,C. j

解答群

ア．AVG　　イ．GROUP BY　　ウ．JOIN　　エ．ON　　オ．所属コード

カ．性別　　キ．性別コード　　ク．部署名

更に平均年齢が25歳以上のデータだけ出力するにはどうすればいいか考えてみましょう。

演習３　以下のテーブル「在庫」「製品」からいずれの倉庫にも３０未満の在庫しかない製品を出力するSELECT文を

考えてみましょう。

在庫( 倉庫コード ,製品番号，在庫数 ) 製品( 製品番号 ,製品名，単価 )



演習４　以下を出力するSELECT文を考えましょう。テーブル「資格」「社員資格」を使用します。

資格( 資格コード ,資格名 )

社員資格( 社員番号 , 資格コード )

SELECT 資格名,

(SELECT COUNT(\*) FROM 社員資格 WHERE 資格コード=資格.資格コード)

FROM 資格

### －３．CREATE　VIEW

以下の構文でVIEWを作成できます。AS句以降で記述したSELECT文をVIEW名で実行できるようになります。一度作成したVIEWはDROP　VIEWで削除するまで使用できます。

CREATE　VIEW　VIEW名　AS　SELECT文

演習１　テーブル「社員」、「所属」から以下の出力をするVIEW「社員名簿VIEW」を定義します。

社員( 社員番号 ,氏名 ,年齢 ,性別コード ,所属コード )

所属( 所属コード ,部署名 )

a　　　　 　　社員名簿VIEW

b

SELECT 　氏名　,　部署名

FROM 　社員　 A

c　　　　 　 　所属　Ｂ

d　　　　 　　A. e　　　　 　=B. e

解答群

ア．AS　　イ．CREATE VIEW　　ウ．JOIN　　エ．ON　　オ．所属コード

### －４．INSERT　INTO

INSERT文はテーブルにデータ（行、タプル）を挿入します。

INSERT　INTO　テーブル名　VALUES(値リスト) →　値リストには全列のデータを記述

INSERT　INTO　テーブル名（列名リスト）　VALUES(値リスト)

演習１　テーブル「社員」に社員番号「1234」、氏名「広島○△」、年齢「２０」、性別コード「０」を追加します。

社員( 社員番号 ,氏名 ,年齢 ,性別コード ,所属コード )

a　　　　 　　 b

社員（　 c　　　　 　　、 d　　　　 　　、 e　　 　　　　、 f　　 　　　　）

g　　 　　　 （‘広島○△’、20、‘0’、‘1234’）

解答群

ア．INSERT　　イ．INTO　　ウ．VALUES　　エ．社員番号　　オ．氏名　　カ．性別コード　　キ．年齢

### －５．DELETE

DELETE文はテーブルのデータ（行）を削除します。WHERE句を書かない場合は全行が削除されます。

DELETE　[ FROM ] テーブル名

DELETE　[ FROM ] テーブル名　[ WHERE　削除する行の条件式 ]

演習１　テーブル「社員」から社員番号「1234」の社員を削除します。

社員( 社員番号 ,氏名 ,年齢 ,性別コード ,所属コード )

a　　　　 　 b　　　　 　社員　 c　　　　 　 d　　　　 　＝‘1234’

解答群

ア．DELETE　　イ．FROM　　ウ．WHERE　　エ．社員番号

演習２　テーブル「社員」から社員番号が「1」で始まる社員を削除します。

社員( 社員番号 ,氏名 ,年齢 ,性別コード ,所属コード )

a　　　　 　 b　　　　 社員　 c　　　　 　 d　　　　 　 e　　　　 　 f

解答群

ア．DELETE　　イ．FROM　　ウ．LIKE　　エ．WHERE　　オ．社員番号　　カ．‘1＿’　　キ．‘1%’

### －６．UPDATE

UPDATE文は既存のテーブルのデータを更新（変更）する時に使用します。

WHERE句を書かない場合は全行が更新されます。

UPDATE テーブル名　SET 列名＝値（又は式）　[ WHERE 更新する行の条件式 ]

演習１　テーブル「商品」の単価を一律５％値上げする。

商品 （商品ID、商品名、単価 ）

a　　　　 　　商品　 b　　　　 　 d　　　　 　＝ e

解答群

ア．単価　　イ．単価\*1.05　　ウ．SET　　エ．UPDATE

### －７．ALTER　TABLE

ALTER　TABLE文は既存のテーブル構造（制約や列）を変更する時に使用します。

ALTER TABLE テーブル名　MODIFY（ 列名１ データ型，列名２ データ型，･･･）

ALTER TABLE テーブル名　ADD 列名１ データ型，列名２ データ型，･･･

ALTER TABLE テーブル名　DROP COLUMN 列名１，列名２，･･･

ALTER TABLE テーブル名　ADD ［CONSTRAINT］ 制約名 制約

ALTER TABLE テーブル名　DROP ［CONSTRAINT］ 制約名

Ｐ１３演習３の答え

製品

製品番号 CHAR(6)　主キー

製品名 CHAR(30)

単価 NUMBER(10)

在庫

倉庫コード CHAR(5)

製品番号 CHAR(6)

在庫数 NUMBER(6)

主キー：倉庫コード、製品番号

外部キー：製品番号

在庫数がいずれの倉庫にも３０以下しかない製品名を表示しなさい。

CREATE TABLE 製品(

製品番号 CHAR(6),

製品名 CHAR(30),

単価 NUMBER(10),

CONSTRAINT PK\_製品 PRIMARY KEY(製品番号)

);

CREATE TABLE 在庫(

倉庫コード CHAR(5),

製品番号 CHAR(6),

在庫数 DECIMAL(6),

CONSTRAINT PK\_在庫 PRIMARY KEY (倉庫コード,製品番号),

CONSTRAINT FK\_在庫\_製品 FOREIGN KEY (製品番号) REFERENCES 製品);

SELECT 製品名 FROM 製品 WHERE 製品番号 NOT IN (

SELECT 製品番号 FROM 在庫

WHERE 在庫数 > 30)

SELECT 製品番号,製品名 FROM 製品 X

WHERE NOT EXISTS (SELECT \* FROM 在庫 WHERE 製品番号 = X.製品番号 AND 在庫数 > 30)

給与等級 社員



１．テーブル「社員」から右のように上司の社員番号と社員名を出力するSELECT文を考えてみましょう。



SELECT 部下.社員番号,部下.社員名,

上司.社員番号 AS 上司社員番号,上司.社員名 AS 上司名

FROM 社員 部下

JOIN 社員 上司 ON 部下.上司=上司.社員番号



２．テーブル「社員」と「給与等級」から右のような結果を出力するSELECT文を考えてみましょう。

SELECT 社員名,給与,等級

FROM 社員

JOIN 給与等級 ON 給与 BETWEEN 最低給与額 AND 最高給与額