

SQL 基礎/応用 演習問題

【SQL 演習】

<0> 演習を始める前に

1. データの構造

この演習で用いるテーブルの構造とデータを以下に示します。

・ DEPARTMENT（部署テーブル）

表 1 部署テーブルの構造

物理カラム名	型名	論理カラム名	主キー
DEPT_NO	NUMBER (3)	部署番号	○
DEPT_NAME	VARCHAR (20)	部署名	

表 2 部署テーブルのデータ

DEPT_NO	DEPT_NAME
101	総務部
102	経理部
103	人事部
104	開発部
105	営業部

・ EMPLOYEE（社員テーブル）

表 3 社員テーブルの構造

物理カラム名	型名	論理カラム名	主キー
EMP_NO	NUMERIC (4)	社員番号	○
EMP_NAME	VARCHAR (20)	社員名	
BIRTHDAY	DATE	生年月日	
DEPT_NO	NUMERIC (3)	部署番号	

表 4 社員テーブルのデータ

EMP_NO	EMP_NAME	BIRTHDAY	DEPT_NO
1001	田中	1982-02-02	102
1002	鈴木	1985-03-03	101
1003	佐藤	1977-04-04	104
1004	井上	1986-05-05	105
1005	加藤	1975-06-06	103
1006	松本	1983-07-07	104
1007	山下	1979-08-08	105
1008	渡辺	1971-09-09	104
1009	山本	1988-01-01	

・ PARTTIME（アルバイトテーブル）

表 5 アルバイトテーブルの構造

物理カラム名	型名	論理カラム名	主キー
PART_NO	NUMERIC (4)	アルバイト番号	○
PART_NAME	VARCHAR (20)	アルバイト名	

表 6 アルバイトテーブルのデータ

PART_NO	PART_NAME
9001	佐藤
9002	森田
9003	鈴木

< 1 > 参照系 SQL の実行

演習 1-1: 社員テーブルから全てのレコード、全てのカラムを取得する SQL を実行しなさい。

演習 1-2: 社員テーブルから部署番号を取得する SQL を実行しなさい。ただし、部署番号は重複しないように取得すること。

演習 1-3: 社員テーブルから、社員番号が 1005 のレコードの社員番号、社員名、生年月日を取得する SQL を実行しなさい。

演習 1-4: 社員テーブルから、部署番号が 102 以上 104 以下のレコードの社員名、部署番号を取得する SQL を実行しなさい。

演習 1-5: 社員テーブルから、部署番号が NULL のレコードの全てのカラムを取得する SQL を実行しなさい。

演習 1-6: 社員テーブルから、部署番号が 104 以外のレコードの社員番号、社員名、部署番号を取得する SQL を実行しなさい。

演習 1-7: 社員テーブルから、部署番号が 101 または 105 のレコードの社員名、部署番号を取得する SQL を実行しなさい。

演習 1-8: 社員テーブルから、社員名が「山」で始まるレコードの社員名を取得する SQL を実行しなさい。

演習 1-9: 社員テーブルを年齢の若い順に並べ替えて、全てのレコード、全てのカラムを取得する SQL を実行しなさい。

演習 1-10: 社員テーブルを部署番号の降順、年齢の高い順に並べ替えて、全てのレコード、全てのカラムを取得する SQL を実行しなさい。

演習 1-11: 社員テーブルから全てのレコードの社員名、生年月日を取得する SQL を実行しなさい。

ただし、生年月日は YYYY 年 MM 月 DD 日の書式で取得すること。

演習 1-12: 社員テーブルから、部署番号は何種類あるかを取得する SQL を実行しなさい。

演習 1-13: 社員テーブルから、一番大きな数値の社員番号を取得する SQL を実行しなさい。
ただし、カラム名には「最大」という別名をつけること。

演習 1-14: 社員テーブルを部署ごとに分類し、各部署で年齢の一番若い社員の部署番号、生年月日を取得する SQL を実行しなさい。

演習 1-15: 社員テーブルから、部署番号が 104 のレコードの中で一番年齢が高い社員の生年月日を取得する SQL を実行しなさい。

< 2 > 更新系 SQL の実行

演習 2-1 : 社員テーブルの全社員の社員番号に 100 加算する SQL を実行しなさい。その結果を確認するための参照系の SQL も合わせて実行しなさい。

演習 2-2 : 社員テーブルの社員番号 1104 のレコードの部署番号を 101 に更新する SQL を実行しなさい。その結果を確認するための参照系の SQL も合わせて実行しなさい。

演習 2-3 : 社員テーブルに以下の表に示すレコードを追加する SQL を実行しなさい。そのとき、SQL にはカラム名の指定もつけること。その結果を確認するための参照系の SQL も合わせて実行しなさい。

表 7 追加レコード

EMP_NO	EMP_NAME	BIRTHDAY	DEPT_NO
1110	橋本	87-10-10	103

演習 2-4 : 演習 2-3 で実行した SQL を、再度実行するとどうなるか、確認しなさい。
また、なぜそうなるか理由を考えなさい。

演習 2-5 : 社員テーブルから、社員番号 1110 のレコードを削除する SQL を実行しなさい。
その結果を確認するための参照系の SQL も合わせて実行しなさい。

演習 2-6 : 社員テーブルから、全レコードを削除する SQL を実行しなさい。
その結果を確認するための参照系の SQL も合わせて実行しなさい。

※この演習を実行する前に BEGIN コマンドを実行しておくこと

※演習結果を確認したら、ROLLBACK コマンドを実行すること

【DML 演習】

<1>結合

演習 1-1：社員テーブルと部署テーブルを結合して、全てのレコードの社員番号、社員名、部署名を取得する SQL を実行しなさい。

ただし、テーブル別名を使用し、WHERE 句を使用した結合で記述すること。

演習 1-2：社員テーブルと部署テーブルを直積結合する SQL を実行しなさい。

演習 1-3：社員テーブルに対して、部署テーブルを内部結合して、全てのカラムを取得する SQL を実行しなさい。

演習 1-4：社員テーブルに対して、部署テーブルを左側外部結合して、社員番号、社員名、部署名を取得する SQL を実行しなさい。

演習 1-5：社員テーブルに対して、部署テーブルを自然結合して、全てのカラムを取得する SQL を実行しなさい。

<2>副問い合わせ

演習 2-1：社員テーブルと部署テーブルから、部署名が「営業部」の社員名を取得する SQL を実行しなさい。

ただし、IN 演算子を使用すること。

演習 2-2：演習 2-1 を EXISTS を使用した SQL に書き換えて実行しなさい。

演習 2-3：社員テーブルと部署テーブルから、どこの部署にも配属されていない社員名を取得する SQL を実行しなさい。

ただし、NOT EXISTS 演算子を使用すること。

演習 2-4：演習 2-3 を NOT IN 演算子を使用した SQL に書き換えて実行しなさい。

演習 2-5：社員テーブルと部署テーブルから、社員名と部署名を取得する SQL を実行しなさい。ただし、SELECT 句に副問い合わせを使用すること。

演習 2-6 : 社員テーブルと部署テーブルから、所属する社員が 2 人の部署の部署番号を取得する SQL を実行しなさい。

ただし、FROM 句に副問い合わせを使用すること。

演習 2-7 : 社員テーブルと部署テーブルから、社員名が「松本」のレコードの部署番号を演習 2-6 で求めた部署番号に変更する SQL を実行し、その結果を確認するための参照系の SQL も合わせて実行しなさい。

演習 2-8 : 社員テーブルに以下のレコードを追加する SQL を実行し、その結果を確認するための参照系の SQL も合わせて実行しなさい。

表 8 追加レコード

EMP_NO	EMP_NAME	BIRTHDAY	DEPT_NO
最大社員番号+1	上田	81-01-02	最小部署番号

演習 2-9 : 社員テーブルから、部署名が「人事部」のレコードを削除する SQL を実行し、その結果を確認するための参照系の SQL も合わせて実行しなさい。

※この演習を実行する前に BEGIN コマンドを実行しておくこと

※演習結果を確認したら、ROLLBACK コマンドを実行すること

＜3＞和集合

演習 3-1：社員テーブルから社員名を取得する SQL の結果とアルバイトテーブルからアルバイト名を取得する SQL の結果を結合し、すべてを取得する SQL を実行しなさい。

演習 3-2：演習 3-1 の結果から社員名とアルバイト名が同じ名前を排除した結果を取得する SQL を実行しなさい。

【DDL 演習】

＜１＞テーブル作成

演習 1-1：下記のデータ構造を持ったテーブル STUDENT（生徒テーブル）を作成する SQL を実行しなさい。

表 9 STUDENT（生徒テーブル）の構造

物理 カラム名	型名	論理 カラム名	主キー	NOT NULL
STUDENT_ID	NUMERIC (2)	生徒番号	○	○
STUDENT_NAME	VARCHAR (20)	生徒名		○
CLASSROOM	CHAR (2)	クラス		○

演習 1-2：下記の表に記載されたデータを、テーブルに追加する SQL を実行しなさい。ただし、SQL は直接タイプして実行せず、ファイルに記述した上でまとめて実行すること。実行した結果を確認するための参照系の SQL も合わせて実行しなさい。

表 10 追加レコード

STUDENT_ID	STUDENT_NAME	CLASSROOM
1	秋田	3A
2	千葉	3A
3	香川	3A
4	長崎	3B
5	山口	3B
6	福井	3B
7	石川	3B
8	福島	3C
9	宮崎	3C
10	長野	3C

演習 1-3：下記の構造をもったテーブルを作成する SQL を実行しなさい。

表 11 EXAM（試験テーブル）の構造

物理 カラム名	型名	論理 カラム名	主キー	NOT NULL
STUDENT_ID	NUMERIC(2)	生徒番号	○	○
JAPANESE	NUMERIC(3)	国語		
MATH	NUMERIC(3)	数学		
SCIENCE	NUMERIC(3)	理科		
SOCIETY	NUMERIC(3)	社会		
ENGLISH	NUMERIC(3)	英語		

演習 1-4: 下記の表に記載されたデータをテーブルに追加する SQL を実行しなさい。ただし、SQL は直接タイプして実行せず、ファイルに記述した上でまとめて実行しなさい。実行した結果を確認するための参照系の SQL も合わせて実行しなさい。

表 12 追加レコード

STUDENT_ID	JAPANESE	MATH	SCIENCE	SOCIETY	ENGLISH
1	37	49	80	90	64
2	70	69	34	58	92
3	100	97	79	83	81
4	36	47	58	69	70
5	63	100	95	32	46
6	98	36	54	100	66
7	86	48	29	77	79
8	71	97	80	88	100
9	46	58	100	66	93
10	19	79	60	44	21

<2> ビュー

演習 2-1：生徒名が「福」で始まるレコードの各教科の点数を取得する SQL を、副問い合わせを使って実行しなさい。

演習 2-2：演習 2-1 で作った副問い合わせを、V_STUDENT というビューとして登録しなさい。

演習 2-3：演習 2-2 で作ったビューの全カラムを取得する SQL を実行しなさい。

演習 2-4：演習 2-2 で作ったビューを削除する SQL を実行しなさい。また、ビューを削除してもビューの基となった生徒テーブルや試験テーブルに影響がないことを参照系 SQL を実行して確認しなさい。

<3> 制約

演習 3-1：試験テーブルの生徒番号は、生徒テーブルの生徒番号を参照させたいが、テーブル作成時には外部キー、参照制約について定義していない。SQL コマンドで試験テーブルに外部キーと参照制約の定義を追加しなさい。

演習 3-2：試験テーブルに存在している生徒番号 1 のレコードを、生徒テーブルから削除しようとするとうどうなるか確認しなさい。

演習 3-3：生徒テーブルに存在していない生徒番号 11 のレコードを試験テーブルに追加しようとするとうどうなるか確認しなさい。

【DCL 演習】

<1> トランザクション

演習 1-1 : ROLLBACK について確認する。

1. BEGIN コマンドを実行する
2. 生徒テーブルから生徒番号が 7 のレコードのクラスを「3C」に更新する
3. 参照系 SQL を実行し、2 の更新が反映されていることを確認する
4. 試験テーブルから生徒番号が 10 のレコードを削除する
5. 参照系 SQL を実行し、4 の更新が反映されていることを確認する
6. ROLLBACK コマンドを実行する
7. 生徒テーブルと試験テーブルに対して参照系 SQL を実行し、2 と 4 の更新が反映されていないことを確認する

演習 1-2 : BEGIN について確認する。

1. BEGIN コマンドを実行する
2. 生徒テーブルから生徒番号が 7 のレコードのクラスを「3C」に更新する
3. 参照系 SQL を実行し、2 の更新が反映されていることを確認する
4. 試験テーブルから生徒番号が 10 のレコードを削除する
5. 参照系 SQL を実行し、4 の更新が反映されていることを確認する
6. COMMIT コマンドを実行する
7. ROLLBACK コマンドを実行する
8. 生徒テーブルと試験テーブルに対して参照系 SQL を実行し、2 と 4 の更新が反映されていることを確認する

<2> ロック

演習 2-1：暗黙的なロックについて確認する。

準備作業：

SQL Shell を新たにもう 1 つ起動して、KENSU ユーザでログインする。

先に起動していた SQL Shell を「クライアント A」とし、後から起動したほうを「クライアント B」とする。それぞれのクライアントで順番にしたがって、コマンドを実行する。

クライアント A

1. BEGIN コマンドを実行する
2. 生徒テーブルから生徒番号が 10 のレコードのクラスを「3D」に更新する
3. 参照系 SQL を実行し、2 の更新が反映されていることを確認する

クライアント B

4. 参照系 SQL を実行し、3 の更新は反映されていないことを確認する
5. BEGIN コマンドを実行する
6. 生徒テーブルから生徒番号が 1 のレコードのクラスを「3E」に更新する
7. 参照系 SQL を実行し、6 の更新が反映されていることを確認する

クライアント A

8. 参照系 SQL を実行し、6 の更新は反映されていないことを確認する

クライアント B

9. 生徒テーブルから生徒番号が 10 のレコードのクラスを「3F」に更新する
生徒番号 10 のレコードはクライアント A によって、暗黙的なロックが該当レコードに対してかけられており、クライアント B の処理が待たされている（処理結果が表示されない）ことを確認する

クライアント A

10. COMMIT コマンドを実行する
11. 参照系 SQL を実行し、2 の更新が反映されていることを確認する

クライアント B

12. クライアント A によるロックが解除され、9 の処理が実行されたことを確認する
13. 参照系 SQL を実行し、9 の更新が反映されていることを確認する

クライアント A

14. 参照系 SQL を実行し、9 の更新が反映されていないことを確認する
15. BEGIN コマンドを実行する
16. 生徒テーブルから生徒番号が 1 のレコードのクラスを「3G」に更新する
生徒番号 1 のレコードはクライアント B によって、暗黙的なロックが該当レコードに対し

てかけられており、クライアント A の処理が待たされている（処理結果が表示されない）ことを確認する

クライアント B

17. ROLLBACK コマンドを実行する
18. 参照系 SQL を実行し、クライアント B によるトランザクション処理（6 と 9）が取り消されていることを確認する

クライアント A

19. クライアント B によるロックが解除され、16 の処理が実行されたことを確認する
20. 参照系 SQL を実行し、16 の更新が反映されていることを確認する
21. COMMIT コマンドを実行する
22. 参照系 SQL を実行し、2 と 16 の更新が反映されていることを確認する

クライアント B

23. 参照系 SQL を実行し、2 と 16 の更新が反映されていることを確認する

演習 2-2：明示的なロックについて確認する。

演習 2-1 と同様に SQL Shell を 2 つ使用する。

クライアント A

1. COMMIT コマンドを実行する
2. 生徒テーブルから生徒番号が 5 のレコードを参照する
このとき、該当レコードに対して明示的なロックを行う

クライアント B

3. COMMIT コマンドを実行する
4. 同様に生徒テーブルから生徒番号が 5 のレコードを参照する
このとき、該当レコードに対して明示的なロックを行う
クライアント A によって、明示的なロックが該当レコードに対してかけられており、クライアント B の処理が待たされている（処理結果が表示されない）ことを確認する

クライアント A

5. 生徒テーブルから生徒番号が 5 のレコードのクラスを「3A」に更新する
6. COMMIT コマンドを実行する
7. 参照系 SQL を実行し、5 の更新が反映されていることを確認する

クライアント B

8. クライアント A によるロックが解除され、4 の処理が実行されたことを確認する
このとき、5 の更新が反映されていることを確認する
9. 生徒テーブルから生徒番号が 5 のレコードのクラスを「3C」に更新する
10. COMMIT コマンドを実行する

11. 参照系 SQL を実行し、9 の更新が反映されていることを確認する

クライアント A

12. 参照系 SQL を実行し、9 の更新が反映されていることを確認する

以上