2.6節「データベース」補足資料

(2024年8月30日版)

A.2.6.1 正規化理論

本文の図 2.47 (図 A.2.1 に再掲載) に示したデータ構造は, データの挿入や更 新, 削除をおこなうと整合性が損なわれる. そこで, エドガー・フランク・コッ ド (E.F.Codd) は関数従属の概念により, 整合性が損なわれないデータ構造 (関 係)を導く正規化理論を提唱した.

以下では、まず、図 A.2.1 の関係に潜む問題を整合性の観点から考察する. つ ぎに、関数従属にもとづいて正規化する手順を理解する. そして、十分に正規 化された関係は、挿入や更新、削除がおこなわれても整合性が損なわれないこ とを理解する.

(1) 完全従属にもとづく正規化—第2正規形

図 A.2.1 の関係を, HclubID と AclubID を主キーとする**正規形** (以下,**第1 正規形** (1st normal form)) という.一方,表の1つのセルに複数の値が記入さ れるような形式を非正規形という. RDB (relational database) を構成する関 係は,第1正規形でなければならない.

さて、この第1正規形のままでは、データの追加や修正、削除などの操作が おこなわれるとき、以下のような整合性を損なう**更新不整合**が起こる。

date	$\underline{\mathbf{H}}_{\mathrm{clubID}}^{*}$	\mathbf{H} team	$\mathbf{H}\mathbf{s}\mathbf{c}\mathbf{o}\mathbf{r}\mathbf{e}$	$\underline{\mathbf{A}}_{\mathrm{clubID}}^*$	$\mathbf{A} \mathrm{team}$	\mathbf{A} score	$\mathbf{stad}\mathrm{ID}^*$	$\operatorname{stadium}$	visitors
2022-05-07	SAPPORO	札幌	1	KYOTO	京都	0	SAPPOROD	札幌ドーム	9527
2022-05-14	KYOTO	京都	0	SHIMIZU	清水	0	SANGA	サンガ S	12398
2022-05-25	SAPPORO	札幌	1	KASHIWA	柏	6	SAPPOROD	札幌ドーム	5791
				1	* H はH	ome. A	は Away sta	d は stadu	ım の略

図 A.2.1 J1 リーグの試合を表した関係 (第1正規形). 下線は主キーを示す. (本文の 図 2.47 を再掲載)

- 挿入不整合: クラブ (チーム) やスタジアムの情報のみを挿入したくても、 対戦相手が決まるまで主キー制約により挿入できないという不整合。
- 修正不整合: クラブやスタジアムに重複するデータがあるため, すべての データが正しく修正されなかったとき, 一部に誤ったデータが残る不整合.
- ・ 削除不整合:一度しか使用されなかったスタジアムの試合が削除される ことによって、そのスタジアムの情報までを削除する不整合。

これらの更新不整合を解決するために,以下に定義する**関数従属**,部分従属, 完全従属にもとづいて,図A.2.1の第1正規形を分解する.

- ・ 関数従属: X と Y を関係 R の属性の部分集合とする. X の属性値がただ
 1つの Y の属性値を決めるとき, Y は X に関数従属するといい, X → Y
 と記す.
- 完全従属: 関数従属 $X \to Y$ において、X の真部分集合 $X' (X' \subset X)$ と Y との間に $X' \to Y$ が成立しないとき、Y は X に完全従属するという.
- 部分従属:関数従属 X → Y において, X の真部分集合 X' (X' ⊂ X) と
 Y の真部分集合 Y' (Y' ⊂ Y) との間に, X' → Y' が成立するとき, Y'
 は X' に部分従属するという.

部分従属はデータの重複をもたらす要因となることに着目して,以下のよう にして部分従属する属性を分離する.まず,図A.2.1の第1正規形の関数従属 を調べると,主キーとそれ以外の属性間に,つぎの関数従属F1が成立する.

F1 HclubID, AclubID \rightarrow date, Hteam, Hscore, Ateam, Ascore, stadID, stadium, visitors

この関数従属 F1 には、主キーの属性集合 (HclubID, AclubID) の真部分集合

HclubID と AclubID に関して、つぎの部分従属 F2, F3 が確認できる.

- F2 $HclubID \rightarrow Hteam$
- F3 AclubID \rightarrow Ateam

そこで、F1から部分従属する Hteam と Ateam を分離して F4とする.

F4 **HclubID**, **AclubID** \rightarrow date, Hscore, Ascore, stadID, stadium, visitors

これにより, F4 は完全従属が成り立っていることがわかる.また, F2, F3 も完全従属が成立している.この完全従属が成立している関係を**第2正規形**と (2nd normal form) いう.ここで,各クラブは,1つのホームスタジアムを所有 することから,HclubID \rightarrow stadID, stadium も成り立つのではないかと考えた 読者がいるかもしれない.しかし,国立競技場のようにどのクラブのホームス タジアムにも属さない競技場があるため部分従属は成立しない.

このように図 A.2.1 の第1正規形を, 関数従属 (部分従属, 完全従属含む) に もとづいて正規化し, F2, F3, F4 の第2 正規形に分解できた.

これらの第2正規形の関係名を、それぞれ Home, Away, Game として、つぎの ように RDB を構成する関係を表すことにする.なお、カッコ内は属性を表し、 下線は主キーを表している.

- Home (<u>HclubID</u>, Hteam)
- \bigcirc Away (<u>AclubID</u>, Ateam)
- Game (date, <u>HclubID</u>, Hscore, <u>AclubID</u>, Ascore, stadID, stadium, visitors)

これらの正規化された関係を自然結合すると、元の第1正規形 (図 A.2.1) に 復元できる.このように元の関係に復元できる分解を無損失分解あるいは情報 無損失分解といい、正規化により分解された関係は無損失分解となる.

さて,第2正規形へ正規化したが,第2正規形では更新不整合を解決するこ とができない.たとえば,関係 Game をみると,スタジアムに関して,つぎの ように更新不整合が残っている.

• **挿入不整合**:スタジアムの情報のみを挿入したくても,対戦試合が決ま

るまで挿入ができないという不整合.

- 修正不整合:スタジアムにデータの重複があるため、すべてのデータが 正しく修正されなかったとき、一部に誤ったデータが残る不整合。
- ・ 削除不整合:一度しか使用されなかったスタジアムの試合が削除される ことによって、そのスタジアムの情報までを削除する不整合。

これらの不整合を解決するためには、つぎの**推移従属**にもとづいて、さらに 正規化しなければならない。

(2) 推移従属にもとづく正規化--第3正規形

第2正規形に、つぎの推移従属が成立するとき、更新不整合が起こる.

• 推移従属: $X, Y, Z を関係 R の属性の部分集合とする. <math>X \to Y, Y \to X$ かつ $Y \to Z$ ならば推移律より $X \to Z$ であり, Z は X に推移従属する という.

Game に推移従属が成り立っていないか調べてみると,つぎの関数従属 F5 が 成り立つため,推移従属 F6 が成立する.

F5 stadID \rightarrow stadium

F6 HclubID, AclubID \rightarrow stadium

そこで, Game の関数従属 F4 から推移従属する属性 stadium を分離して, つぎの F7 とする.

F7 HclubID, AclubID \rightarrow date, Hscore, Ascore, stadID, visitors

F5, F7 には完全従属が成立する. それぞれの関係名を Stadium, Game として, つぎのように関係を分解する.

○ Stadium (stadID, stadium)

○ Game (date, <u>HclubID</u>, Hscore, <u>AclubID</u>, Ascore, stadID, visitors)

推移従属にもとづいて分解された関係は、完全従属が成り立つ第2正規形で あり、かつ推移従属が成り立たない関係になっている.このような関係を**第3** 正規形 (3rd normal form) という、第3正規形に正規化したことで、これまで の更新不整合が解決されることを確かめてみよう.

さて, Home と Away は, ともに1つのクラブである. そこで, これらを1つ

の Club として, つぎのように定義する. 属性 stadID は, ホームスタジアムを 表すために加えることにする. また, team はクラブ名, home は住所とする.

- Club (<u>clubID</u>, team, home, stadID)
 また, Stadium も定員 (capacity) や,所在地 (location) を加えて、つぎのように定義する.
- Stadium (stadID, stadium, capacity, location)
 Game は変更がなく、つぎのようになる。

○ Game (date, <u>HclubID</u>, Hscore, <u>AclubID</u>, Ascore, stadID, visitors)

ここで、Gameの主キーは、HclubID と AclubID の組合せによるため、1 シー ズンの試合データの管理に限定される. つまり、複数シーズンの試合データを 管理する場合は、この主キーでは一意に行を識別できない. そこで、1 つの解決 策として date を主キーに加える方法が考えられる.

さて、本文の図 2.53 の ER モデルと比べてみよう. 同じように定義できてい ることが確認でき、ER モデルによる設計手法を評価することができる.

(3) ボイス-コッド正規形

第3正規形まで正規化すれば更新不整合は起こらないと考えられていたが,つ ぎのような場合に更新不整合が起こることが指摘された.

■ *X*, *Y*, *Z*を関係 *R*の属性の部分集合とする. *X*, *Y*を主キーとして, *X*, *Y* → *Z*が成立しているとき, *Z* → *Y*も成立している場合.

このような場合,関係 R を正規化して, $S(\underline{X}, \underline{Z}) \ge T(\underline{Z}, Y)$ に無損失分解す る. このように正規化された関係 S, T を,ボイス-コッド正規形 (Boyce-Codd normal form) という.第3正規形において,キーとなる属性が複数ある場合 (これらを候補キーとよぶ),ボイス-コッド正規形について考察する必要がある.

(4) 多値従属と第4正規形

本文の ER モデルでは,各クラブのホームスタジアムを1つに限定して,1対 1の関連を定義した.しかし,実際にはセレッソ大阪のように2つのホームスタ ジアムをもつケースがある.また,Jリーグの理念の一つに,クラブが拠点を置

く地域を豊かにするための「ホームタウン活動」がある.多くのクラブが複数 の都市をホームタウンに指定している.そこで,このような場合の正規化につ いて考えてみる.各クラブのホームスタジアムとホームタウンを,1つの行で表 すとき,図A.2.2のように記述することが多い.

クラブ	ホームスタジアム	ホームタウン
鹿島アントラーズ	県立カシマサッカースタジアム	鹿嶋市, 潮来市
セレッソ大阪	ヤンマースタジアム長居,ヨドコウ桜スタジアム	大阪市,堺市

図 A.2.2 非正規形

このように表の1つのセルに複数の値が記入されている表を非正規形といった.しかし, RDBの関係は正規形でなければならないため, つぎのように第1 正規形にしなければならない.

クラブ	ホームスタジアム	ホームタウン
鹿島アントラーズ	県立カシマサッカースタジアム	鹿嶋市
鹿島アントラーズ	県立カシマサッカースタジアム	潮来市
セレッソ大阪	ヤンマースタジアム長居	大阪市
セレッソ大阪	ヤンマースタジアム長居	堺市
セレッソ大阪	ヨドコウ桜スタジアム	大阪市
セレッソ大阪	ヨドコウ桜スタジアム	堺市

図 A.2.3 第1正規形

ところが、この第1正規形のままでは更新不整合が起こる.たとえば、主キー の1つであるホームスタジアムが決まるまで、ホームタウンを挿入できない.そ の逆もあり挿入不整合となる.また、重複データがあるため、修正漏れによる 修正不整合が起こる.さらに、鹿島アントラーズのホームスタジアムを削除す ると、主キー制約によりホームタウンまで削除される削除不整合が起こる.

このような更新不整合は、つぎの多値従属が原因となっている.

• 多値従属: X, Y, Zを関係 Rの属性の部分集合とする. ここで, $Y \ge Z$ の集合は独立しているとする. この関係 Rにおいて, X の1つの値が Yの複数の値を決めるとき, Y は X に多値従属するといい, $X \to Y \ge$ 記す. さらに, $X \to Z$ も成り立つとき, 自明でない多値従属といい, $X \to Y|Z$ と記す. ここで, $Y \ge Z$ が独立しているとは, Yの値が Z

の値に依存せず,かつ Z の値が Y の値に依存しないことである.

そこで、多値従属 $X \rightarrow Y \ge X \rightarrow Z$ にもとづいて正規化し、 $S(\underline{X}, \underline{Y})$ $\ge T(\underline{X}, \underline{Z})$ に分解する. このように正規化された関係 S, T を第4正規形 (4th normal form) という.

多値従属に着目して,図A.2.3の第1正規形を調べてみよう.つぎの多値従 属F8,F9が成立していることがわかる.

そこで,図A.2.4のように自明な多値従属F8,F9にもとづいて分解した第4 正規形にする.

カラブ	ホートフタジアト	<u>クラブ</u>	ホームタウン
		鹿島アントラーズ	鹿嶋市
鹿島アントラース	県立カシマサッカースタジアム	鹿島アントラーズ	潮来市
セレッソ大阪	ヤンマースタジアム長居	セレッソ大阪	大阪市
セレッソ大阪	ヨドコウ桜スタジアム	セレッソ大阪	堺市
차	ニームスタジアム	ホームタ	ウン

図 A.2.4 第4正規形

関係ホームスタジアムと関係ホームタウンを属性クラブで自然結合すると,図 A.2.3の元の関係(第1正規形)が復元できる無損失分解であることを確かめて みよう.また,更新不整合が起こらないことも確かめてみよう.さらに,1つ のクラブが複数のホームスタジアムやホームタウンを有する表現が可能になる.

A.2.6.2 ビッグデータと NoSQL

オンラインショッピングなどの web サービスやソーシャルメディア (SNS), さらに, IC カードやセンサーなどのモノが相互に情報交換する IoT の拡大によ り, テキストや数値のみならず音声や画像などの多様 (Variety) なタイプのデー タが, 高速 ・高頻度 (Velocity) に生成され, 膨大な量 (Volume) となって処理・ 管理されている. このような特性をもつデータをビッグデータとよび, 新たな 価値の創造や社会システムの最適化など, データ駆動型社会の原動力を担って いる.

(1) 分散システムと NoSQL データベース

ビッグデータの多様性は、表形式に収まらない非構造化データも含まれる点 にある.そのため、正規化された構造化データを扱う RDB では、それらのデー タに対応することが困難となった.また、高速・高頻度なデータ処理は、1 台の コンピュータを高性能化するか、図 A.2.5 のような複数のコンピュータが連携 して動作する分散システムにしなければならない.

本節の説明に用いる分散システムは、1台のマスター・コンピュータ(以下、 マスターと記す)と多数のスレーブ・コンピュータ(以下、スレーブと記す)で 構成される、そして、データベースは分割されてスレーブで管理される。

多数のスレーブが処理を分担するため、総合的に負荷が分散される. さらに、 1つのスレーブは、同じデータベースを管理する数台のコンピュータで構成さ れるため、さらに負荷が分散される. あわせて、1台のコンピュータが故障して も、他のコンピュータが対応するため耐障害性を高める.

ビッグデータの拡大はとどまるところを知らないが,分散システムはスレーブ を拡張して,ビッグデータの拡大に対応することができるという特徴がある.こ のようなシステムの拡張方式をスケールアウトという.一方,1台のコンピュー タを高性能化して対応する方式をスケールアップというが,一般的にコストが 高くなる問題がある.したがって,ビッグデータの処理には分散システムが必 須の技術となる.



図 A.2.5 n 基のスレーブで構成される分散システムの例

分散システムによる RDB の分散データベースは、1 つの処理要求 (トランザ クションという)を実行するとき、データベースの一貫性を保障するために同時 に受け付けたトランザクションの実行制御をおこなう、その実行制御により処 理待ち時間が発生するため高速な処理には限界がある。

そこで、RDB 以外のデータベースを意味する NoSQL (Not Only SQL) デー タベースが開発され、キー・バリュー型、カラム指向型、ドキュメント型など のデータモデルが提案されている.ただし、NoSQL はデータベースの一貫性よ りもトランザクションの高速な実行を優先するため、在庫や空き状況を厳密に 管理する必要がある販売や予約システム、または銀行など一貫性が必要となる システムには適さない.以下、データモデルの概要をみてみよう.

(2) キー・バリュー型

図 A.2.6 のように一意識別の役割をもつキー (key) とその値を表現するバ リュー (value) だけで構成される単純なデータモデルである. ここでは, IoT の センサーデータを管理する単純な例とした.

$\pm - (\text{key})$	バリュー (value)
sensor_a+timestamp_1	temperature:28.5, humidity:70.0, pressure:1000
<pre>sensor_b+timestamp_2</pre>	X:1.50, Y:1.85, Z:1.55
sensor_a+timestamp_3	temperature:30.0, humidity:75.0, pressure:1005
•	•

図 A.2.6 キー・バリュー型の例

このデータモデルについて、分散システムの動作をみてみよう、クライアントから、あるキーについてバリューの問い合わせがあると、マスターはキーから そのデータを管理するスレーブを割りだして指示を与える、指示されたスレー ブは、データベースからキーに対応するバリューを読みだして、マスターに返 し、そしてクライアントが受け取る、バリューのデータ構造は決まっていない ため、クライアントのプログラムが、自由に決めることができる特徴がある、

このように単純なデータモデルであるが, Amazon は, 自社のショッピング システムにおいて, 主キーによるアクセスしか必要としない現状を踏まえて, キー・バリュー型を採用しているといわれる.

(3) カラム指向型

キー・バリュー型は、キーとバリューが1対1となるデータモデルであるが、 カラム指向型は、1つのキーに複数のカラムのバリューが対となるデータモデル である.さらに、カラムファミリーを定義することができ、データの共通性を 反映したデータベースの設計が可能になる.たとえば、図A.2.7のような多数 のセンサーからリアルタイムに送られてくるデータを処理するデータベースを 考えてみる.

3つのカラムファミリーが定義され、1つ目の Sensor Column Family は、セ ンサーの情報を記録している。カラムが固定されていて、RDBの関係に似た 管理ができる。2つ目の Record Column Family は、センサー ID をキーにし て、データの送信時刻 (timestamp) をカラム名にするデータ ID を追加してい く. このようにカラムを自由に追加できる点が、カラム指向型の特徴である。3 つ目の Data Column Family は、データ ID をキーにしてバリューのデータを 記録している。

カラムファミリーは、その名前のみを事前に定義するだけでよく、カラム名や カラムの追加などは、クライアントのプログラムがデータを追加するとき、自 由に決定できる.このように、クライアントが要求する多様なデータ構造を表



図A.2.7 カラム指向型の例

現するデータモデルを提供している.

(4) ドキュメント指向型

キーの対となるバリューが,JSON や XML などで記述されたドキュメント になっているデータモデルである.JSON は,プログラミング言語 JavaScript で用いられるデータ構造である.つぎのように,{}のなかに,コロン(":")で 区切ったキーとバリューのセットを記述する.さらに,homeのように,オブ ジェクトとよばれるJSON 形式の入れ子データ構造を記述することができる.

```
{
```

```
"id" : "SAPPORO",
```

```
"club": "コンサドーレ札幌",
```

"**home**": {

```
"street": "3-4-1 西区宮の沢 2",
```

```
"city": "札幌市",
```

"prefecture": "北海道"

```
},
```

```
"stadium": "札幌ドーム"
```

```
}
```

ドキュメントの構造は、クライアントがデータを追加するとき決めることが できるため、さまざまなドキュメントを記録できる.また、ドキュメント内の キーに対して条件検索が可能である.

A.2.6.3 データベースの作成と利用

(1) MySQL Workbench のインストール (Windows の場合)

① ソフトウェアのダウンロード
 https://dev.mysql.com/downloads/workbench/
 ダウンロードファイル例:mysql-installer-community-8.0.34.0.msi

② Visual C++ 2019 Redistributable の確認
 MySQL Workbench は Visual C++ 2019 Redistributable package を必
 要とする、そこで、Windowsの「設定(③))」から「インストールされ

ているアプリ」を表示して,2019 以降のバージョンがインストールさ れていることを確認する.たとえば,Microsoft Visual C++ 2015-2022 Redistributable (x64) があればよい.該当するバージョンがない場合は, つぎの web サイトからダウンロードしてインストールする. https://learn.microsoft.com/ja-JP/cpp/windows/latestsupported-vc-redist?view=msvc-170 ダウンロードファイル:vc_redist.x64.exe

- ③ インストール 最初に表示されるインストールタイプの選択 (Set Up Type)は FULL で よいだろう、以下、表示される内容を確認してインストールを進めれば よい。
- (2) MySQL Workbench のホーム画面



図A.2.8 ホーム画面の構成

 ●をホームボタンとよぶことにする.このホーム画面に戻る.
 ②を SQL モードとよぶことにする.SQL と GUI によるデータベースの 作成,データの挿入や更新,削除,問い合わせなどの操作ができる.
 ③を ER モードとよぶことにする.ER モデルを定義する.

(3) MySQL Workbench の環境設定

図 A.2.8 の④ Edit メニューから Preferences...を選択して,図 A.2.9 の日本 語フォントと図 A.2.10 の SQL Editor を設定する.

① 日本語フォントの設定

▲ Appearance をクリックして、 ● Japanese を選択する.

SQL Editor Query Editor	Color Presets Colors available when cre	eating tables, views etc	Colors available when creat	ing layers, notes etc
Object Editors SQL Execution Administration Modeling Defaults MySQL Diagram	#988FDA #FEDE58 #9808A5 #FE9898 #FE987E #FFFFFF	~	#FFEEEC #FEFDED #EAFFE5 #ECFDFF #FCFDFF #FOF1FE #FFEBFA	
Appearance Fonts & Colors	Fonts	0		
Others	Configure Fonts For:	Japanese		~
	Location Text Figure Text Connection Caption Layer Title Routine Group Figure Routine Group Figure Routine Group Figure Table Figure 1tems Table Figure Stetion Table Figure Title	Japanase Korean Simplified Chinese Cyrillic Title Arial Unicode MSBold 12 Arial Unicode MSBold 11 Arial Unicode MSBold 11 Arial Unicode MSBold 12		

図 A.2.9 日本語フォントの設定

② SQL Editor の設定

図 A.2.10 の SQL Editor を選択して、 Safe Updates…のチェックを はずす. SQL の UPDATE と DELETE の制限を解除するためである.

General Editors						
SOL Editor	SQL Editor					
Query Editor Object Editors SOL Execution	Save snapshot of open editors on open editors on open editors.	dose		Interval to perform auto-saving of all open		
Administration	Auto-save scripts interval:	10 seconds	~	script tabs. The scripts will be restored from		
Modeling				the last auto-saved version if Workbench		
Defaults				anexpected y quick		
MySQL	Create new tabs as Query tabs in	nstead of File				
Diagram	Restore expanded state of the acti	iva schama ohia	rte			
Appearance		ive scheme obje				
SCH SCH						
Others	Sidebar					
	Show Schema Contents in Schem	a Tree				
	Show Metadata and Internal School	emas				
	MySQL Session					
				Time interval between sending		
	DBMS connection keep-alive interval (in	n seconds):	600	keep-alive messages to DBMS. Set to to not send keep-alive messages		
				to not send keep dive messagesi		
	DBMS connection read timeout interval	(in seconds):	30	The maximum amount of time the quer		
		(in accorday).		DBMS.Set 0 to skip the read timeout.		
			c 0	Maximum time to wait before a		
	DBMS connection timeout interval (in se	econas):	00	connection attempt is aborted.		
	Other					
	Internal Workbench Schema:	mysqlworkbench	1	This schema will be used by MySQL Workbench to store information required		
				for certain operations.		
	Safa Lindates (rejects LIPDATEs and	d DELETER with	no restrictions)			

図 A.2.10 SQL Editor の設定

A.2.6.4 ER モデルの作成

新規 ER モデルの設定

File Edit View Arrange Mode	Database Tools Scripting Help	
Description Editor	Model Overview	
No Selection V	▼ Physical Schemas w mydb	2 :: ;;;;] ≡ (+
	Table () Long) a) 45 Table Wers () Long)	
	Add View Routines (Diterra)	
	 Add Routine 	
	Routine Groups (0 items)	
Description	Add Group	
User Types List	J 1league - Schema 🛛 🕹	
Type Definition Flag	Name: Decourse Rename References	here. You can use any combination of ANSI letters, numbers and the underscore cee found in view, triggers, stored procedures and functions from the old schema name
	Charset/Collation: Default Charset V Default Collation V The character set and its collation	selected here will be used when no other charset/collation is set for a database

図 A.2.11 ER モデルの新規作成

図 A.2.11 の● File メニューから New Model を選択する. ②をクリックして、 で、③の新規データベースを作成する. これをダブルクリックして、④の名前 を、たとえば J1league に変更する. そして、⑤をダブルクリックして、ER モ デルの作成を開始する.

(2) 実体の定義

MySQL Model* × EER Di	agram ×			
File Edit View Arrange Model	Database Tools Scripting	Help		
🖹 🔌 🚅 🐃 🍊 🖉 e	1 🔁			
Bird's Eye	Diagram			
Zoom: 100% 🔹 🔍 🔍				1
-	0 2			
	@ table1	-8		
		ō		
Catalog Tree				
▶ 😫 mydb				
V 🔁 Jileague	8			
table1	0			
Priews	table1 - Table 🛛 🗙			
	Table Name:	table 1	Schema: J1league	*
	Column Name	Datatype PK	NN UQ B UN ZF AI G	Default/Expression
Catalog Layers User Types				

図A.2.12 実体の作成

図 A.2.12 の● Place a New Table を選択し、つづけて作図画面をクリックする. 2の実体 table1 が作成される. これをダブルクリックして、図 A.2.13 のように、 3の実体名、 ④の属性を入力する. 図 A.2.9 で日本語フォントの設定ができていれば、実体名や属性名を日本語で入力することができる. ただし、組織のルールがある場合は、それに従おう.

MySQL Model* × EER Di	agram × Database Tools	Scripting	Help					
🖹 🔌 🐺 🗢 🦘 🖉 🖻	17							
Bird's Eye	Diagram							
Zoom: 100% - 🔍 🔍	L3							
	0	6						
	Ø	0 Home	_					
		? clubID Cl	HAR(12) RCHAR(30) C					
		◇ home VA	RCHAR(100)					
Catalon Tree		Indexes	▶					
6 alog nee		Ψ	<u> </u>					
▶ 📑 mydb ▼ 🔁 J1league	-							
Tables	6							
Tiews								
PRoutine Groups	Home - Table $~\times~$		3					
	W	Table Name:	Home		Schem	a: J1league		*
	Column Name		Datatype	PK	NN UQ B	UN ZF	AI G	Default/Expression
	♦ team		VARCHAR(30)	0			ĬÖ	
Catalog Layers User Types	v nonte		VARCHAR(100)				5 0	

図A.2.13 実体の属性定義

属性のデータ型 (DataType) には、整数型、実数型、文字列型、日付型など 多くの種類がある.入力するデータに合わせてデータ型とサイズを決める.た とえば、主キーの clubID は、最大 12 文字の文字列データと決めているので CHAR(12) とした.文字列には固定長 (CHAR) と可変長 (VARCHAR) がある が、固定長のほうが高速な処理を期待できる.そこで、インデックスが作成さ れて頻繁にアクセスされる主キーは固定長にした. ⑤に定義内容が随時反映さ れるが、1 番目の属性は主キーとみなされ、黄色の鍵マークが付く.また、⑥に は作成したテーブルが表示される.図 A.2.14 の ER 図を参考にして、Away と Stadium も定義してみよう.

ところで、MySQL の CHAR 型や VARCHAR 型は、英数字や日本語を問わ ず、指定した文字数を格納できる. たとえば、VARCHAR(30) 型の属性 team には、30 文字以内の日本語のチーム名を入力できる. ただし、UTF-8 での日本 語1文字は3バイトで表現されるため、実際には文字数の3倍のメモリサイズ が使用される.

Bird's Eye	Diagram					
Zoom: 100% • 🔍 🔍	L3					
	Ø					
	0	🔲 Home 🔻			Away	v
		clubID CHAR(12) team VARCHAR(30) home VARCHAR(100)			ClubiD CHAR(12) 	0)
Catalog Tree		Indexes >			Indexes	•
V 😝 Jileague	-					
Away	6					
Stadium	1:1		_			
Boutine Groups	1:0			Stadium 🔻		
			7 s	tadID CHAR(12)		
			○ s	tadium VARCHAR(60)		
	1:0		00	apacity INI		
	n:m		lar			
	1 :n		-			

図A.2.14 Away と Stadium の作成

(3) 関連の定義

1対1の関連



図A.2.15 Home と Stadium の1対1の関連

Home と Stadium は依存関係がない 1 対 1 の関連を定義する. そこ で,図 A.2.15 の ① の 1 対 1(1:1 <u>Non-Identifying</u> Relationship) を選択 して,外部キーを設定する ② の Home,外部キーと関連づけられる ③ の Stadium の順にクリックする. ④ のように依存関係がないことを示す点 線で Home と Stadium を接続して,両端の「++」が 1 対 1 の多重度を表 す.同時に,⑤に外部キー Stadium_stadID が設定されて,Stadium の 主キーと関連づけられる外部キー制約が定義される. Away と Stadium の関連も定義してみよう.

関連の定義を変更あるいは削除したいときは、④の結線を右クリック するとよい. ポップメニューが表示されて,変更 (Edit Relationship) あ るいは削除 (Delete) ができるようになる.



多対多の関連

図 A.2.16 Home と Away の多対多の関連

ホーム&アウェイの試合は、Home と Away の多対多の関連で表現す る. そこで、図 A.2.16 の の多対多 (n:m <u>Identifying</u> Relationship) を 選択して、 **2** Home と **3** Away をクリックする. **4** の Home_has_Away という弱実体が作成され, Home および Away と実線で接続される. 実線は依存関係があることを示し,両端の「++」と「→<」は1対多の多重度を表す.

④の Home_has_Away をダブルクリックして、図A.2.17の ER 図のように実体名を Game に変更し、試合日 (date) とスコア (Hscore, Ascore),
 観客数 (visitors)の属性を定義しよう.ここで、DATE は日付型、INT は整数型を表す.

③ 1対多の関連



図 A.2.17 Stadium と Game の 1 対多の関連

Stadium と Game は 1 対多の関連を定義する. そこで, **①**の 1 対 多 (1:m <u>Non-Identifying</u> Relationship) を選択して, **②**の Game, **③**の Stadium の順にクリックする. **④**のように接続され, Game に外部キー stadium stadID が設定される.

ここで,依存関係がある1対多の関連を定義することもできる.その 場合は,Stadium_stadIDがGameの主キーとしても定義される.以上

の作業で ER モデルが完成する.

A.2.6.5 物理モデルの作成

① フォワードエンジニアリングによる方法

フォワードエンジニアリングを利用して, ER モデルから物理モデルを 生成する方法をみてみよう.



図 A.2.18 フォワードエンジニアリングの操作

図 A.2.18 の① Database メニューをクリックし, ②ポップメニューから Forward Engineer...を選択して処理を開始する. Review SQL Script の画面で, データ定義言語 (DDL) で記述されたデータベースの定義が, つぎのように表示される.

```
CREATE SCHEMA IF NOT EXISTS 'J1league';
USE 'J1league';
```

CREATE SCHEMA は、データベース J1league が存在していなけれ ば作成する.そして、USE は、データベースの利用を宣言している. つぎの CREATE TABLE は、テーブル Stadium を定義している.

```
CREATE TABLE IF NOT EXISTS 'Jlleague'.'Stadium' (
    'stadID' CHAR(12) NOT NULL,
    'stadium' VARCHAR(60) NULL,
    'capacity' INT NULL,
    'location' VARCHAR(100) NULL,
PRIMARY KEY('stadID'))
ENGINE = InnoDB;
```

外部キー制約が定義されているテーブル Home の記述もみてみよう. INDEX 句は,外部キー Stadium_stadID のインデックスを作成する. さらに,FOREIGN KEY 句と REFERENCES 句により,外部キー Stadium_stadID がテーブル Stadium の属性 stadID と関連 (を参照) する外部キー制約を定義している. ON DELETE (UPDATE)句は,参照 されるテーブル Stadium の行が削除 (更新) されるとき,それを参照する テーブル Home の行も一緒に削除 (更新) するかどうかを定義する. NO ACTION は,削除 (更新) せずに外部キー制約違反として処理する.

```
CREATE TABLE IF NOT EXISTS 'J1league'.'Home' (
 'clubID' CHAR(12) NOT NULL,
 'team' VARCHAR(30) NULL,
 'home' VARCHAR(100) NULL,
 'Stadium_stadID' CHAR(12) NOT NULL,
PRIMARY KEY('clubID'),
INDEX 'fk_Home_Stadium_idx' ('Stadium_stadID' ASC) VISIBLE,
CONSTRAINT 'fk_Home_Stadium'
 FOREIGN KEY('Stadium_stadID')
 REFERENCES 'J1league'.'Stadium' ('stadID')
ON DELETE NO ACTION
ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

フォワードエンジニアリングによって生成されたテーブル定義から, ER モデルの実体と関連の定義が忠実に記述されていることがわかる.

ところが、この ER モデルでは、ホーム&アウェイの対戦試合をわか りやすく表現するために、Home および Away を定義した.そのため、 フォワードエンジニアリングはそれぞれのテーブルを作成し、冗長な表 現になっていることに気づく、RDB では、A.2.6.6 項(4)の問い合わせ 例 6 のように、1 つのテーブル club に 2 つの別名 Home と Away を付け て SQL を記述することができる. したがって, これらを1つのテーブル として定義すればよい.

② SQLによるデータベースの作成 フォワードエンジニアリングによって生成された記述を編集して、つぎ のようにする.なお、省略できる記述は削除してわかりやすくした.ま た、J1league DDL.txt として、サポートサイトで提供している.

```
/* データベースの作成 */
CREATE SCHEMA 'J1league';
USE 'J1league';
/* スタジアムの作成 */
CREATE TABLE 'J1league'. 'Stadium' (
  'stadID' CHAR(12) NOT NULL,
 'stadium' VARCHAR(60) NULL,
  'capacity' INT NULL,
  'location' VARCHAR(100) NULL,
PRIMARY KEY('stadID'));
/* クラブ(チーム)の作成 */
CREATE TABLE 'J1league'.'Club' (
  'clubID' CHAR(12) NOT NULL,
  'team' VARCHAR(30) NULL,
  'home' VARCHAR(100) NULL,
  'stadID' CHAR(12) NOT NULL,
PRIMARY KEY('clubID'),
FOREIGN KEY('stadID')
REFERENCES 'J1league'.'Stadium' ('stadID'));
/* 試合の作成 */
CREATE TABLE 'J1league'.'Game' (
  'date' DATE NULL,
  'HclubID' CHAR(12) NOT NULL,
  'Hscore' INT NULL,
  'AclubID' CHAR(12) NOT NULL,
  'Ascore' INT NULL.
  'stadID' CHAR(12) NOT NULL,
  'visitors' INT NULL,
PRIMARY KEY('HclubID', 'AclubID'),
FOREIGN KEY('HclubID')
REFERENCES 'J1league'.'Club' ('clubID'),
FOREIGN KEY('AclubID')
REFERENCES 'J1league'.'Club' ('clubID').
FOREIGN KEY('stadID')
REFERENCES 'J1league'.'Stadium' ('stadID'));
```

ところで、このようなデータベースを定義する SQL(DDL) 文では、

バッククォート「'」を使用するので注意しよう.ただし,必要に応じて 省略してもよい.

この SQL を図 A.2.19, 図 A.2.20 のように実行して, データベースと テーブルを実装してみよう.



図 A.2.19 MySQL サーバの接続

●のホームボタンをクリックして、ホーム画面に戻る. ②の SQL モードを選択して、③の MySQL サーバに接続する. 接続にはインストール時に設定したパスワード入力が要求される.

File Edit View Query Databas	Server Tools Scripting Help	
8 5 8 8 8 8 8	60 10 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	
Navigator	Query 1 x 5	
SCHEMAS 7	🗀 🗖 🖉 🖗 🔕 🕒 🚱 🚳 🖓 🛛 Imèto 1000 muse 🔹 📥 🚅 G	
9, Filter objects		
V illeague	1 /* デーダベースの11FRX */	
Tables	2 • CREATE SCHEMA 'J1league';	
Club game	3 • USE 'J1league';	
▶ IIII stadium	4 /* スタジアムの作成 */	
Views	5 • CREATE TABLE 'J1league'. 'Stadium' (
Functions	6 'stadID' CHAR(12) NOT NULL,	
	7 'stadium' VARCHAR(60) NULL,	
▶ 📄 world	8 'capacity' INT NULL	
	9 Jocation' VARCHAR(100) NULL	
	10 PRIMARY KEY ("stadID"))	
	11 (* 251 (I-1) o/c c */	
Administration Schemas	12 • O CREATE TABLE Theague . Club (
Information	13 ClubID' CHAR(12) NOT NULL,	
	14 `team` VARCHAR(30) NULL,	
Schema: j1league	15 'home' VARCHAR(100) NULL,	
	L	1.7 ····
	Output	
	 Action Output 	
0	# Time Action	Message
6	1 16:47:08 CREATE SCHEMA 'Ulleague'	1 row(s) affected
	2 16:47:08 USE Jileague 2 16:47:08 CREATE TABLE 'Illeague' 'Onders' ('stadD' CHAR(10) NOT NULL 'stade	U row(s) affected
	A DRAFIN DEALE LODIE JURANE SACIO SACIU UNARGIZINUI NULL STACI	
	A 1647-08 CREATE TABLE 'Illeague' Club'/ ClubID' CHAR/12 NOT NULL Jaam' VA	0 row(s) affected

図 A.2.20 データベースの実装

④の SQL エディタに、編集した SQL を貼り付け、⑤の実行をクリックする。⑥に実行した SQL とその処理結果が表示される。すべて正常に実行されれば、⑦のデータベース表示の更新をクリックすると、⑧に作成されたデータベースとテーブルが表示される。以上により、A.2.6.6 項で利用するデータベースが完成する。

 ③ GUIによるデータベースの作成
 図 A.2.21 と図 A.2.22 のように、GUIによるデータベースとテーブルの 定義もできる。

Navigator	Query 1 j'lleague - Schema ×		SQLAdditions
SCHEMAS (%	Name: jileague	Specify the name of the schema here. You can	・ 原 売 CREATE TA
Q Filter objects jilleague sakila Sakila sakila Sys world	Rename References Charset/Collation: Default Charset V Default Collation V	Refactor model, changing all references found in The character set and its collation selected here	No Context He
Administration Schemas			
No object selected	Schema		
			Context Help Snippets
	Couple		
	Time Action 1 18:36:05 Apply changes to J lleague	Message Changes applied	Duration / Fe

図 A.2.21 GUI によるデータベースの作成

まず,データベースを作成する. ● Create a new schema...をクリックして, ②のデータベース名を J1league に変更する. ③ Apply をクリックするとデータベースが作成される. ④に表示された J1league をダブルクリックして,データベースの利用を宣言する.

	Query 1	j1league - S	chema cl											
SCHEMAS	49	6 Tab	e Name: dub				Schen	na:	j1lea	gue				1
V 🗧 j1league	_ 💵	Charset/0	collation: utf8	mb4 ∨	utf8mb4	0! ~	Engine	e:	InnoD	в			~	
Tables Idub Views Stored Procedures		Commen	ts:										*	
Functions	Column N	lame	Data	type	Pk	NN	UQ	в	UN	ZF	AI	G	Default	t/Ex
sakila	7 ? clubI	D	CHA	R(12)	2									
world	♦ team		VAR	CHAR(30)			H	Н	H	Н	H	Н	NULL	
	♦ stadi	ID	CHA	R(12)	Č	iŏ	ŏ	ŏ	ŏ	ŏ	ö	ŏ	NULL	
Administration Schemas	Colum	nn Name: dubIE	Defei V	Data T	ype: CH	AR(12)								_
Schema: j1league	Charset/0	Collation: Derau	v Delac v	Def	ault:				- 111	12				
	Co	omments:		Stor	age: 🔘	Virtual		9) Stor	ed	0			
						Primar	Key		Not	Null	0	Unique	2	
						Pipper			Unsi	gned		Zero F		
					U	Diridi y								

図 A.2.22 GUI によるテーブルの作成

つぎに、テーブルを作成する. ⑤ Create a new table...をクリックし、 ⑥のテーブル名、⑦の属性を定義して、⑧ Apply をクリックする. ⑨ に作成したテーブル club が表示される. 外部キー制約は、⑩の Foreign Keys をクリックして手動で定義する. 各自で試みてみよう.

A.2.6.6 データベースの利用

(1) 行の挿入

① INSERT 文で挿入する方法

図 A.2.23 のように, INSERT 文を入力して, テーブル stadium に, 京都 のサンガスタジアム (スタジアム名はサンガSとする)を挿入してみよう.



図 A.2.23 INSERT 文によるデータ入力

●の SQL エディタに INSERT 文を記入し, ②の実行をクリックする.
 ③の処理結果をみると,正常に1行が挿入されている.

挿入されていることを,図A.2.24のSELECT 文で確かめてみよう.

File Edit View Query Databa	se Server Tools Scri	ipting Help
1 🖏 🛱 🧳 🖧 🚛 🖧	. d 🧔 🐙 2	
Navigator	Query 1 SQL File 2*	×
SCHEMAS 🚸	🖿 🖬 💆 🖅 🖉	🕻 🕐 💁 🥥 🛞 📓 Limit to 1000 rows 🔹 🏡 🛫 🔍 🗻 🖃
v → Inter objects v → jileague v → Tables b → dub b → game b → stadium ∀ Vews Stored Procedures ⊕ Functions b → skila	1 • SELECT *	FROM stadium;
sys 🖉	stadID sta	dium capacity location Result
Administration Schemas	▶ SANGA サン	ガS 21623 龟岡市 Grid
Information	*	
No object selected		Form Editor
	all a divised to be	Decision Decision

図 A.2.24 SELECT 文によるデータ検索

Create a new SQL tab をクリックして、②の新規タブを開き、③
 のように SELECT 文を入力して実行する.④の結果表 (Result Grid) が
 表示され、サンガ S の挿入が確認できる.

■ 主キー制約の役割

ここで, 主キー制約の役割をみてみよう. 図 A.2.25 のように札幌ドームを stadium に挿入する. ただし, 主キーを SAPPOROD ではなく, 誤って挿入済みの SANGA にしたとする.



図 A.2.25 主キー制約とエラー処理

●を実行すると、 ②に主キー SANGA が重複しているというエラー メッセージが表示され、この挿入は拒否される. つまり、主キー制約に よりテーブルの行の一意性が保証される.

■ 外部キー制約の役割

つづけて,外部キー制約の役割もみてみよう.外部キー制約が定義さ れているテーブル club に,図 A.2.26 のように京都サンガとコンサドー レ札幌を挿入する.



図 A.2.26 外部キー制約とエラー処理

メッセージをみると,京都サンガは正常に挿入できたが,コンサドー レ札幌は,外部キー制約のため挿入あるいは更新ができないと表示され ている.テーブル club の外部キーが参照する stadium に札幌ドームの 主キー SAPPOROD が存在しないためである.

外部キー制約のもう一つの役割として,外部キーが参照するテーブル を更新あるいは削除するケースをみてみよう.図A.2.27のように,テー ブル stadium からサンガSを DELETE 文で削除してみる.

File Edit View Query Databa	se Server Tools Scripting Help
Navigator	
SCHEMAS 🚸	🗀 🖬 🖉 🖉 👰 🕐 🚱 💿 🚳 Limit to 1000 rows 🔹 📩 🛫 🔍 👖 🖘
 q. Filter objects ♥ ♥ j1league ♥ ♥ Tables ▶ ■ dub 	1 • DELETE FROM stadium WHERE stadium="サンガS;
stadium	Action Output
 Stored Procedures Functions sakila 	Time Action Message 1 18.35.18 DELETE FROM stadium WHERE stadium="ty Error Code: 1451. Cannot delete or update a parent row: a foreign key constraint fails

XA.2.27

メッセージをみると、親 (parent) とよばれているテーブル stadium の サンガ S の行が,外部キー制約で定義した,ON DELETE NO ACTION または ON UPDATE NO ACTION のため更新あるいは削除ができな いと表示されている.うっかり削除してしまうケースであるが,参照さ れるテーブルの操作に対しても,外部キー制約によってデータの整合性 が保たれる.このケースの削除をおこなう場合は,参照する側 (game も 参照している)の行を先に削除する.

② GUI によるデータ入力



図 A.2.28 GUI によるデータ入力

図 A.2.28 の❶のテーブル表示をクリックすると, ❷の SELECT 文が 実行されて, ❸のように結果表が表示される. ④ Insert new row をク リックすると, 結果表に入力できるようになる. 1 行あるいは複数行を

入力後,
Apply をクリックし、つづいて表示される INSERT 文を確認 して実行する.

③ LOAD DATA INFILE によるテキストデータファイルの入力

LOAD DATA INFILE は大量のデータを高速に入力することができ る. つぎのテキストデータファイルをサポートサイトからダウンロード して入力してみよう. なお, データはカンマで区切られた CSV 形式で, 文字コードは UTF-8 である.

- club-UTF8.csv
- stadium-UTF8.csv
- game-UTF8.csv

LOAD DATA INFILE の場合,これらの入力ファイルを MySQL サー バが許可するディレクトリに保存しなければならない.つぎの SELECT 文で, MySQL サーバが許可するディレクトリを調べることができる.

SELECT @@global.secure_file_priv;

たとえば、つぎのように表示される.

C:\ProgramData\MySQL\MySQL Server 8.0\Uploads\

そこで、ダウンロードしたファイルを上記パスの Uploads フォルダに 保存する.

つぎに、3つのファイルの処理順について、主キー制約と外部キー制約 に基づいて、つぎの点を確認しよう。

- 主キー制約に関して:①と②の操作でテーブル club と stadium に いくつかの行を挿入している場合、テキストデータファイルは、そ れらの行も含むため主キーが重複する.そこで、重複する行は、以 下の LOAD 文の3行目のように IGNORE を指定して置き換えない ことにする.なお、DUPLICATE を指定すれば置き換えることがで きる.
- 外部キー制約に関して:テーブル game の外部キー (HclubID, Aclu-

bID, StadID) はテーブル club と stadium の主キーを参照する.また,テーブル club の外部キー stadID はテーブル stadium の主キーを参照する.したがって,最初に stadium を処理し,つぎに club,最後に game の順に処理する.

以下の LOAD DATA INFILE 文を参考にして,図 A.2.29 のように stadium, club, game の順にデータを入力しよう.



ここで、Windows ユーザが、(\'を入力する際には '¥'を押せばよい.



図 A.2.29 テキストデータファイルの入力

メッセージから,22行が入力され,SANGA が主キーに重複している ことがわかる.

ところで、図 A.2.30 のように SELECT… INTO OUTFILE を使う と、問い合わせ結果をファイルに書き出すことができる. 複数のテーブ ルを結合した問い合わせ結果から、分析データセットを作成する場合に 有用である.

File Edit View Query Databas	e Server Tools Scripting Help	
Navigator	Query 1 SQL File 3* SQL File 9* ×	
SCHEMAS 🚸	🗀 🖬 デ 😥 👰 🔘 🗞 📀 😳 🔞 Limit to 1000 rows 🔹 🌟 🛫 🍳	1 7
Q Filter objects	1 • SELECT * FROM club NATURAL JOIN stadium INTO OUTFILE	
V ileague V Tables	2 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/home_stadium	n_list.csv'
Views	Output	
Functions	🗇 Action Output 🔹	
 ▶ sys ▶ world Administration Schemas 	Time Action Time Action 1 12:01:45 SELECT * FROM dub NATURAL JOIN stadium INTO OUTFILE C:/ProgramDat	Message 18 row(s) affected

図 A.2.30 分析データセットの作成

 ④ MySQL Workbench のインポートによるテキストデータファイル入力
 ③の方法では、MySQL サーバが許可するディレクトリにテキストデー タファイルを保存しなければならない、以下で説明する方法では、任意の ディレクトリに保存することができる。正確には、MySQL Workbench が動作するコンピュータ (クライアントという)に保存することができる。 また、文字コードは Shift-JIS でよい、game-SJIS.csv をサポートサイト からダウンロードして図 A.2.31 から図 A.2.33 にしたがって入力してみ よう、なお、③の方法で、テーブル game にデータを入力している場合 は削除してからおこなう。



図 A.2.31 インポートによるデータ入力

●のテーブル表示をクリックして、②の結果表を表示する. ③の Import

をクリックして、以下のファイルの選択とテーブルの選択をする.

ファイルの選択

図 A.2.32 の画面でファイルを選択する. この例では、フォルダ MySQL-W に保存している.

2	Table Data Import	-		×
	Select File to Import			
	Table Data Import allows you to easily import CSV, JSON datafiles. You can also create destination table on the fly. File Path: C:WMySQL-WWgame-SJIS.csv	(Browse.	

- 図A.2.32 ファイルの選択
- テーブルの選択

既存のテーブルに入力する方法と、テキストデータファイルのファ イル名をテーブル名、1 行目の列見出しを列名にして、テーブルの作 成とデータ入力を同時におこなう方法が選択できる.

> > ベル	1 1 1 9 9 9 0 1 3	に肥友のこ	ブルノア	レーナフ
$(1d.$	凶 A.2.33 ワよう	に広任りノー	ノルレー	$\sqrt{19}$
,	,			

2	Table Data Import		-	×
	Select Destination			
	Select destination ta	ble and additional options.		0
0	Use existing table:	j1league.game		\sim
	O Create new table:	jileague 🗸 . game		

図 A.2.33 テーブルの選択

 ● Use existing table を選択して、②から既存のテーブルを選択 する. ③にチェックを入れて、入力前にテーブルを空にすることも できる. つづく操作は省略する.

ところで、MySQL Workbench には、図 A.2.31 で示した4の Export

があり、問い合わせ結果をファイルに書き出すことができる.

(2) データの更新 (UPDATE) は省略する

(3) データの削除 (DELETE) は省略する

(4) データの問い合わせ

本文の問い合わせ例1から6のSQLとその結果表を掲載する.

例1 クラブのデータを見たい.

SELECT * FROM club;

clubID	team	home	stadID
COSAKA	C 大阪	大阪市	YODOKO
FCTOKYO	F C 東京	東京都	AJINOMOTO
FUKUOKA	福岡	福岡市	BEST
GOSAKA	G 大阪	吹田市	PANA
HIROSHIMA	広島	広島市	EDION
IWATA	磐田	磐田市	YAMAHA
KASHIMA	鹿島	鹿嶋市	KASHIMA
KASHIWA	柏	柏市	SANKYO
KAWASAKIF	川崎F	川崎市	TODOROKI
KOBE	神戸	神戸市	NOEVIR
куото	京都	京都市	SANGA
NAGOYA	名古屋	豊田市	TOYOTA
SAPPORO	札幌	札幌市	SAPPOROD
SHIMIZU	清水	静岡市	IAI
SHONAN	湘南	平塚市	LEMON
TOSU	鳥栖	鳥栖市	EKIMAE
URAWA	浦和	さいたま	SAITAMA
YOKOHAMAFM	横浜FM	横浜市	NISSAN

図A.2.34 例1の結果

例2 収容人数 (capacity) が3万人以上,5万人以下のスタジアムを調べたい.

```
SELECT * FROM stadium
WHERE capacity>=30000 AND capacity<=50000;</pre>
```

stadID	stadium	capacity	location
AJINOMO	味スタ	47851	調布市
EDION	ЕХЯ	35909	広島市
KASHIMA	カシマ	38669	鹿嶋市
PANA	パナスタ	39694	吹田市
SAPPOROD	札幌ド	38794	札幌市
TOYOTA	豊田ス	43739	豊田市
YANMAR	ヤンマー	47816	大阪市

図A.2.35 例2の結果

例3 ホームスタジアムを50%に入場制限した場合の収容人数を求めたい.

SELECT team,stadium,capacity*0.5 AS 入場制限,location FROM club AS R, stadium AS S WHERE R.stadID=S.stadID;

つぎの, NATURAL JOIN を使った記述も試してみよう.

```
SELECT team,stadium,capacity*0.5 AS 入場制限,location FROM club NATURAL JOIN stadium;
```

team	stadium	入場 制限	location
C大阪	ヨドコウ	12240.5	大阪市
F C 東京	味スタ	23925.5	調布市
福岡	ベススタ	10781.0	福岡市
G 大阪	パナスタ	19847.0	吹田市
広島	EZA	17954.5	広島市
磐田	ヤマハ	7582.5	磐田市
鹿島	カシマ	19334.5	鹿嶋市
柏	三協F柏	7554.5	柏市
川崎F	等々力	13413.5	川崎市
神戸	JIZA	14498.0	神戸市
京都	サンガS	10811.5	龟岡市
名古屋	豊田ス	21869.5	豊田市
札幌	札幌ド	19397.0	札幌市
清水	アイスタ	9797.0	静岡市
湘南	LEVS	7690.0	平塚市
鳥栖	駅スタ	12065.0	鳥栖市
浦和	埼玉	31005.0	さいたま市
横浜FM	日産ス	35911.0	横浜市

図A.2.36 例3の結果

例4 ホーム試合で勝った試合数と、そのときの平均ゴール数を求めたい、

SELECT COUNT(*) AS 勝試合数, AVG(Hscore) AS 平均ゴール数 FROM game WHERE Hscore>Ascore;

勝試合数	平均ゴール数
109	2.3670

図A.2.37 例4の結果

ちなみに,アウェイチームが勝った試合数と,そのときの平均ゴール数は,以下のとおりであった.ホーム試合のほうが有利といえるだろうか.



図A.2.38 アウェイチームの試合結果

例5 例4の結果をチーム別に集計し,勝試合数の多いチームから順に並べたい.

SELECT team, COUNT(*) AS 勝試合数, AVG(Hscore) AS 平均ゴール数 FROM club JOIN game ON clubID=HclubID WHERE Hscore>Ascore GROUP BY clubID ORDER BY 勝試合数 DESC;

team	勝試合数	平均ゴール数	
川崎F	11	2.5455	
横浜FM	11	2.4545	
広島	10	2.6000	
C 大阪	7	2.2857	
F C 東京	7	2.4286	
鹿島	7	2.1429	
柏	6	2.1667	
名古屋	6	1.6667	
札幌	6	2.1667	
浦和	6	3.1667	
神戸	5	2.8000	
京都	5	1.8000	
鳥栖	5	3.0000	
福岡	4	2.5000	
磐田	4	2.0000	
湘南	4	1.7500	
G大阪	3	2.3333	
清水	2 2.0000		

図A.2.39 例5の結果

例6 2万人以上の観客が入った試合の日程,対戦チームとそのスコア,そしてスタジアムの観客数を調べたい.

SELECT date, Home.team, Hscore, Away.team, Ascore, stadium, visitors FROM stadium NATURAL JOIN game JOIN club AS Home ON Home.clubID =HclubID JOIN club AS Away ON Away.clubID=AclubID WHERE visitors>=20000;

date	team	Hscore	team	Ascore	stadium	visitors
2022-02-23	横浜FM	4	川崎F	2	日産ス	20433
2022-02-26	鹿島	0	川崎F	2	カシマ	27234
2022-03-19	浦和	4	磐田	1	埼玉	24207
2022-04-02	横浜FM	2	F C 東京	1	日産ス	20823
2022-04-10	F C 東京	0	浦和	0	味スタ	22429
2022-04-29	F C 東京	2	G 大阪	0	国立	43125
2022-05-03	鹿島	3	磐田	1	カシマ	26493
2022-05-03	名古屋	1	京都	1	豊田ス	37068
2022-05-07	横浜FM	2	名古屋	1	日産ス	22178
2022-05-08	G 大阪	2	神戸	0	パナスタ	26490
2022-05-21	浦和	1	鹿島	1	埼玉	37144
2022-05-29	F C 東京	3	鹿島	1	味スタ	28436
2022-06-18	浦和	3	名古屋	0	埼玉	28699
2022-06-25	横浜FM	4	柏	0	日産ス	23368
2022-06-26	札幌	1	G 大阪	0	札幌ドーム	21599
2022-06-26	神戸	0	浦和	1	JIZA	21456
2022-07-02	清水	3	横浜FM	5	国立	56131

図 A.2.40 例 6 の結果(日付順に並べ替えた)

【付録1】 データベースをバックアップする方法

PC の故障や誤操作などによるデータベースの障害に備えて,図 A.2.41 のようにデータベースのバックアップを作成しておくとよいだろう.

	0
File Edit View Query Databas	se Server Tools Scripting Help
Navigator	Query 1 Administration - Data Export ×
SCHEMAS 🚸	Local Instance MuSQL80
Q Filter objects	Advanced Options
V 🗐 j1league	Object Selection Export Progress
 Iables club 	Tables to Evnort
▶ game	Event Schema B
Views	2 ileague A 2 ileague
Stored Procedures	🗌 📴 sakila 🔤 🔤 game
sakila	world
sys	
Administration Schemas	
Information	
No object selected	Refresh 3 tables selected Dump Structure and Dat Select Views Select Tables Unselect All
	Objects to Export
	Dump Stored Procedures and Functions Dump Events Dump Triggers
	Export Options
	Export to Dump Project Folder
	Each table will be exported into a separate file. This allows a selective restore, but may be slower.
	C:\#MySQL-W\#J!league_backup.sql C:\#MySQL-W\#J!league_backup.sql
	All selected database objects will be exported into a single, self-contained file.
	Create Dump in a Single Transaction (self-contained file only)
	Press [Start Export] to start

図 A.2.41 データベースのバックアップ

● Server のポップメニューから Data Export を選択する. ●でバックアップするデータベースをチェックする. ●の J1league をクリックして, ●に表示されるテーブルを確認する. テーブルを選択してバックアップすることもできる. ● Export to Self-Contained File を選択し、 ●にバックアップデータベース名を入力する. ここでは、J1league_backup.sql とした. ● Start Export をクリックして処理を開始する.

【付録2】バックアップしたデータベースを復元する方法

	0
File Edit View Query Databas	se Server Tools Scripting Help
Navigator	Query 1 Administration - Data Import/Res ×
SCHEMAS 🚺 🔹	Local Instance MrSO(80
Q Filter objects	Data Import
j1league sakila	Import from Disk Import Progress
sys	Import Options
world	O Import from Dump Project Folder
	Select the Dump Project Folder to import. You can do a selective restore.
	Load Folder Contents
6	2 o Import from Self-Contained File
	Select the SOL /dumo file to import. Please pote that the whole file will be imported.
Administration Schemas	denses are additionally use to import of requestions and the minute meaning an important
Information	Default Schema to be Imported To
No object selected	Default Target Schema: Illeague_bak Verw V
	Select Database Objects to I performance_schema
	Imp.,, Schema Sys
	Dumo Structure and Data val. Colorabilities
	Dump Structure and Dat V Select Views Select Tables Unselect All
	Press [Start Import] to start

図 A.2.42 データベースの復元

 ● Server のポップメニューから Data Import を選択する.
 ● Import from Self-Contained File を選択し、
 ●でバックアップした J1league_backup.sql を探 して指定する。
 ● New...をクリックして表示される画面に、復元するデータベース 名を入力する。
 この例では、正常な J1league があるので、J1league_bak とする。
 そして、
 ●の中から●で作成した J1league_bak を選択して、
 ● Start Import で処 理を開始する。
 ●のデータベース情報を更新すると、J1league_bak が表示される。