2025年IT総合研修

はじめてのSQL

基本をマスター！！！

SQL入門

三重大学附属図書館　花原稔

楽しもう！

この講義のゴール

2025年 IT総合研修

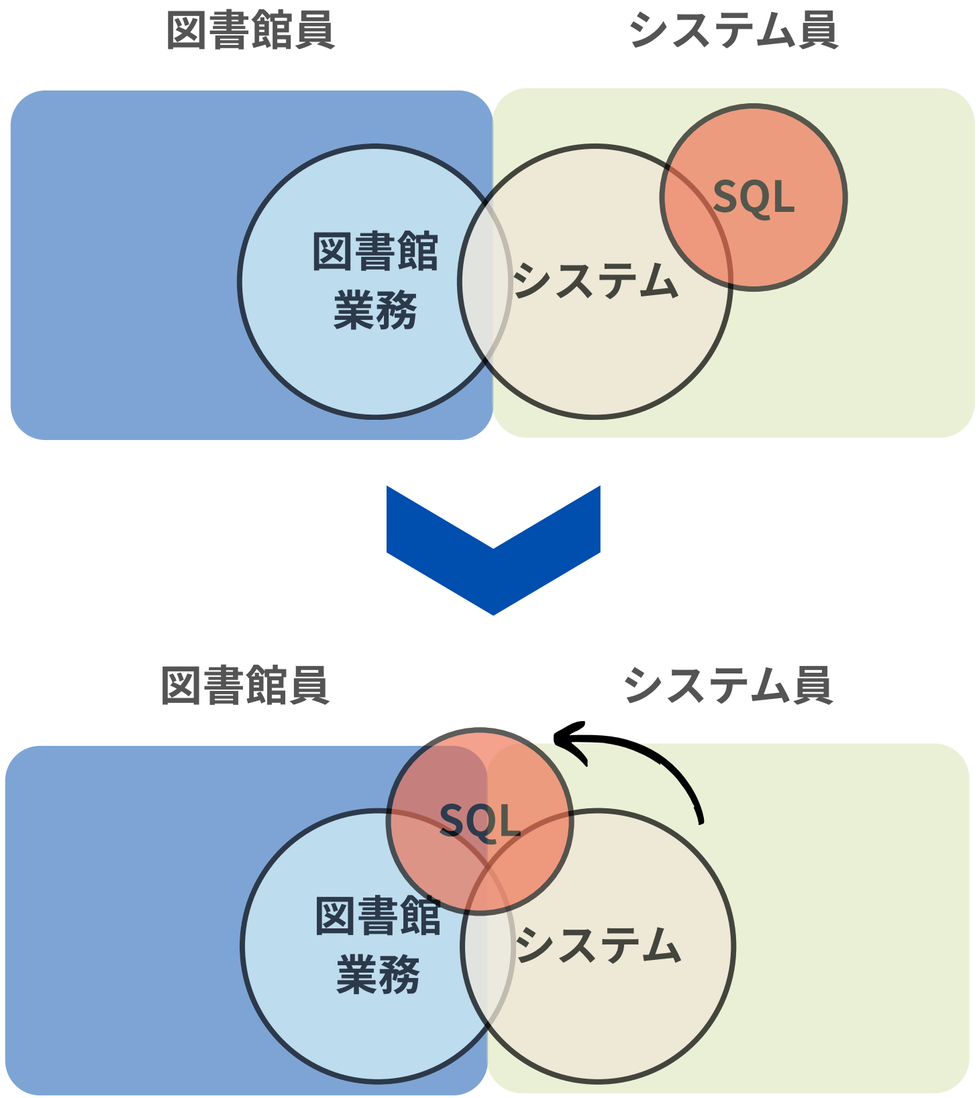
こんなイメージを持ってませんか？

- 何なのか分からない
- 事務職員には使いこなせない
- 図書館員には不要なスキル
- 簡単に読み解けない

この講義の中で全部払拭します！

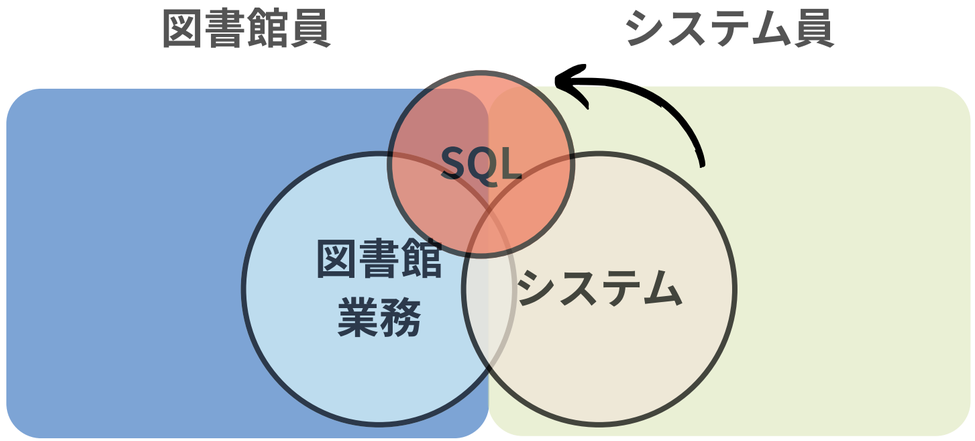
この講義のゴール

このイメージを

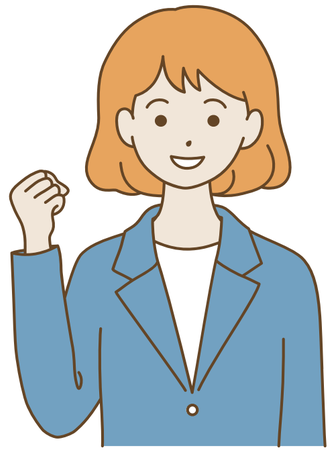


この形に！

この講義のゴール



最後にテストもありません。
安心して楽しみましょう！



この講義の流れ

2025年 IT総合研修

この講義のフロー

この次で実践！

SQLの基礎知識

図書館でのSQL

SQLとデータベース

SQLと仲良くなろう

SQLと仲良くなろう

2025年 IT総合研修

SQLってなに？

ＳＱＬとは、データを**簡単**に**好きな形**に加工する呪文です。

テーブル名：近畿都道府県一覧表

県名	面積	人口	地区
兵庫県	8,401	5,315,447	北部
大阪府	1,905	8,773,979	中部
京都府	4,612	2,506,796	北部
滋賀県	4,017	1,413,610	北部
奈良県	3,691	1,324,473	中部
和歌山県	4,726	922,584	南部

これがSQL

↓

近畿人口のTOP３
になーれ♪

順位	県名	人口	人口割合
1	大阪府	8,773,979	43%
2	兵庫県	5,315,447	26%
3	京都府	2,506,796	12%

SQLってなに？

ＳＱＬとは、データを**簡単**に**好きな形**に加工する呪文です。

例えば……図書館運営費で買った**ジャーナル年間金額の増加率**を出したい！

SQLなし

1.支払表①から去年の金額をコピー
2.支払表②から今年の金額をコピー
3.エクセルにそれらを張り付け
4.各年の合計価格を合算
5.その二つの増加率を計算

SQLあり

```
SELECT
    SUM(A.KAKAKU)/SUM(B.KAKAKU)
FROM
    SHIHARAI1 A,SHIHARAI2 B
```

呪文ってことは難しいんでしょ？

SQLの開発思想は「専門的なトレーニングなしで理解できる」です！

文字	英語のキーワード構成される
言語特性	自然言語に似た“walk-up-and-read”

SQL言語開発の際、プログラミング経験のない大学生に検証。
その結果、数時間で習熟できることを確認している

この研修で習熟できる！

50 Years of Queries – Communications of the ACM

改めてSQLを見てみよう。

複雑な呪文じゃないよ。

```
SELECT
   SUM(A.KINGAKU)/SUM(B.KINGAKU)
FROM
   KAKAKU1 A,KAKAKU2 B
```

```
選択せよ
   合計(A.KAKAKU)/合計(B.KAKAKU)
ここから
   SHIHARAI1 A,SHIHARAI2 B
```

ね。簡単でしょ。

けどプログラミングって難しそう……

SQLはプログラム言語とは違うよ。

```
PASS_SCORE = 60
students = [
 {"name": "佐藤", "score": 75},
 {"name": "鈴木", "score": 58},

def check_pass(score):
 if score >= PASS_SCORE:
 return "合格"
 else:
 return "不合格"

print("=== 成績一覧 ===")
for student in students:
 name = student["name"]
 score = student["score"]
 result = check_pass(score)
 print(f"{name} さん：{score} 点 → {result}")
```

手続きを
記載する

```
SELECT
   SUM(A.KAKAKU)/SUM(B.KAKAKU)
FROM
 SHIHARAI1 A,SHIHARAI2 B
```

問い合わせを
記載する

プログラムほど難しくはないです
恐れずに一緒に接していきましょう！

Excelじゃだめなの？

でもデータ加工ならエクセルでいいじゃん。

部分的にそう。

簡単なものはExcelのがよい。
なんでも得意と不得意があるよ。だから使える武器が多いほうがいい。

SQLに対する認識

メリット

再現性の高さ
- 同じ条件でデータを取得できる

修正の容易さ
- 修正箇所が特定しやすい

処理の高速さ
- 大量データの操作が容易

デメリット

システム依存性の高さ
- システム入れ替えなどで全滅することも

可読性の低さ
- 高難度になるほど可読性が低下する

エクセルの3ステップをSQL一撃で終わらせるか。

SQLは万能ではない。だけど手段を増やして武器にしよう！

AIが日本語でデータ取ってくる時代が来てるのに。

その分、別のスキルが必要になるだけだよ。

AIで完璧なデータを取り出すには、プロンプトチューニングなど
そういうテクニックが必要。SQLじゃなくそれを極めるのも一つ。

ただ、SQLはAIと違い**「再現性」「冪等性」**がある。
数値を扱う場合、この点はかなり重要なポイントになるよ。
あと、まだまだこの先のシステムはSQLで動くよ。

コラム：SQLって何の略？

WikipediaはじめSQLは「Structured Query Language」＝「構造化参照言語」と記載されています。

しかしSQLのルールは国際規格ISO/IEC 9075-1:2023て定義されており、以下記載があります。

4 Concepts

4.1 What is SQL?

The name "SQL" is correctly pronounced "ess cue ell," instead of the somewhat common "sequel". SQL, specified in the parts of the ISO/IEC 9075 series, is a *database language* (more precisely, a *data sublanguage*) used for access to pseudo-relational databases that are managed by pseudo-relational database management systems (RDBMS).

NOTE 1 — Many books and articles "define" SQL by parenthetically claiming that the letters stand for "Structured Query Language". While this was undoubtedly true for the original prototypes, it is not true of the standard. When the letters appear in product names, they have often been assigned this meaning by the product implementors, but users are ill-served by claims that the word "structured" accurately describes the language. In the SQL standard, the letters do not stand for anything at all. There are afficionados who assert that the letters stand for "SQL Query Language", a recursive acronym in the tradition of GNU (GNU's Not Unix) and SPARQL (SPARQL Protocol And RDF Query Language).

SQL is *based on*, but is not a strict implementation of, the relational model of data Relational Model, making SQL "pseudo-relational" instead of truly relational. There are two principal ways in which SQL breaks from the relational model.

ISO/IEC 9075-1:2023 -4.Concept

コラム：SQLって何の略？

ISO/IEC 9075-1:2023 -4.Concept Note1（日本語訳）

> **注1** — 多くの書籍や記事では、「SQL」を「Structured Query Language（構造化問い合わせ言語）」の略であると、かっこ付きで“定義”しています。これは、初期のプロトタイプにおいては確かに正しかったのでしょうが、標準仕様においてはそうではありません。製品名においては、この意味が製品の開発者によって付けられていることが多いものの、「structured（構造化された）」という語がこの言語を正確に表していると主張するのは、ユーザーの利益になりません。SQL標準では、これらの文字がそのような意味を持つとは規定されていません。

SQLの前身であるSEQUELは（Structured Query Language）という略称です。
しかし、後継のSQLに略称は無いです。SQLはSQLです。
この後説明しますが、最近はNewSQLなどStructure（構造）を持たないものも増えています。

だから俺は『ＳＱＬはStructured Query Languageの略』という奴を絶対に許さない。

「SQLと仲良くなる」おしまい。

SQLと仲良くなる

SQLとデータベース

図書館でのSQL

SQLの基礎知識

SQLと仲良くなれましたか？

SQLとデータベース

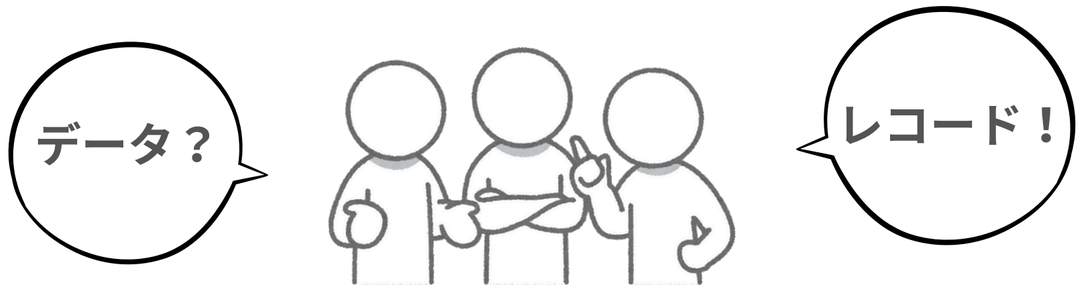
2025年 IT総合研修

定義や用語のお勉強セクション

ここのセクションではSQLを学ぶ前に定義や用語のお勉強をします。

目的と理由

- この先のグループワークでメンバー間で言葉の定義のずれをなくすため



- この話の中でもっと興味を持ってくれる人がでてくるかもしれないため
- 概念的が役立つタイミングが出てくるため

SQLとデータベース

SQLとは

データベースからデータを取得する呪文

データベースとは

一定のルールで整理保管された情報の集まり

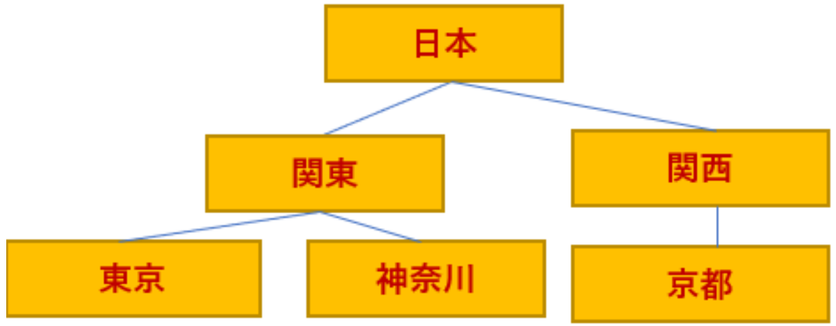
※　JapanKnowledgeのような図書館でいうデータベースとは別の意味です

情報を保管する場所としての「データベース」であれば、意味は共通ですが、
今日お話しする「データベース」は「情報を保存・体系的に管理する仕組み」です。

データベースの種類

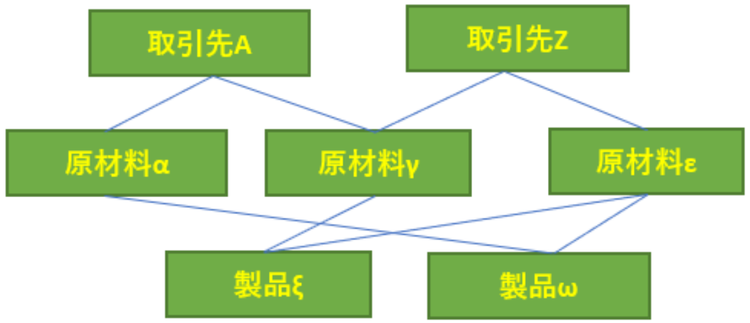
データベースの種類

階層型データベース



データの追加削除が苦手・・・

ネットワーク型データベース



構造が複雑になる・・・

リレーショナルデータベース（RDB）

部署コード	部署名	住所
A	本社	東京都中央区
B	左遷支店	シベリア市中央区
C	日本首都支店	京都府京都市

従業員番号	氏名	部署コード	口座情報
000001	山口重雄	A	123-4567
000002	マルコロドリゲス	A	123-4568
000003	丸本美幸	B	123-4569

口座情報	区分	支店コード
123-4567	当座	893
123-4568	当座	884
123-4569	普通	883

業務システムのメインストリーム。今日はこのRDBを学んでいきます。

そのリレーショナルデータベース（RDB）って何よ？

RDBとは

データを表（テーブル）形式で管理し、項目同士の関係を活用して効率よく扱えるデータベース

RDB製品とは

オープンソース：　PostgreSQL　/　MySQL　/　SQLite
商　　用：　OracleDB　/　SQL Server　/　DB2　/　HiRDB
クラウド：　AmazonAurora　/　GoogleCloudSQL　/　Azure SQL Database

- ＳＱＬのルールは（前述の通り）国際規格で定義されている
  - そのため、基本的な書法はどのＲＤＢ製品でも共通している。

- 製品独自の便利関数や標準外の部分で違いもあります。

困ったときはポケットリファレンスを開いてみよう！

部署コード	部署名	住所
A	本社	東京都中央区
B	左遷支店	シベリア市中央区
C	日本首都支店	京都府京都市

従業員番号	氏名	部署コード	口座情報
000001	山口重雄	A	123-4567
000002	マルコロドリゲス	A	123-4568
000003	丸本美幸	B	123-4569

口座情報	区分	支店コード
123-4567	当座	893
123-4568	当座	884
123-4569	普通	883

用語の勉強、まとめ。

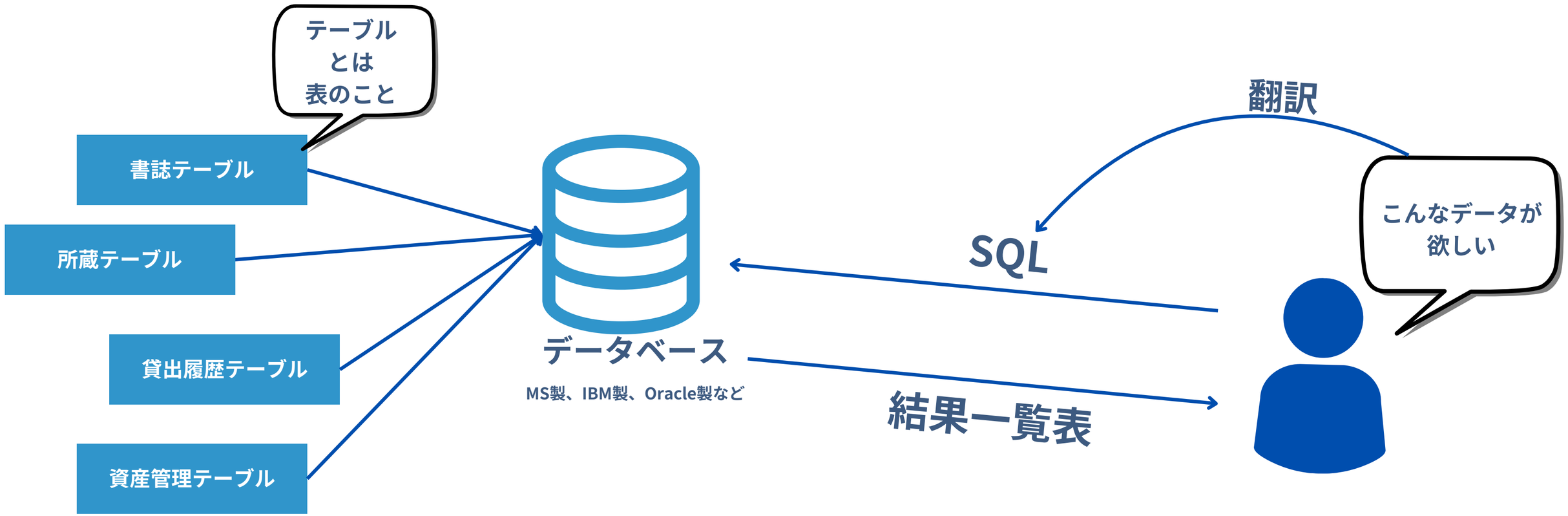
SQLとは

データベースからデータを取得する呪文

データベースとは

一定のルールで整理保管された情報の集まり

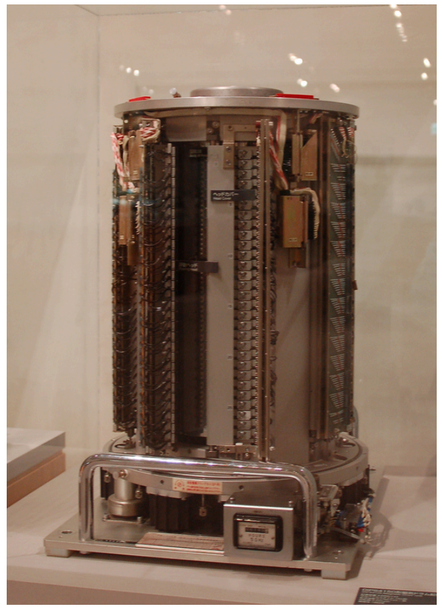
中でもＲＤＢはエクセルのようなの表がたくさん管理されているデータベース



ちなみに…

1970年代、日本の通信の未来を担った大型コンピューターシステム「DIPS-1」。
その心臓部とも言えるのが、DIPS4150形 磁気ドラム記憶装置。
当時の最先端技術「浮動ヘッド方式」を採用し、従来の記録方式の10倍もの密度でデータを記録。
この画期的な装置は、日本電信電話公社と日立製作所の共同開発によって誕生。
東京・芝局での導入を皮切りに、科学技術計算やバンキングシステムなど、様々な商用システムで活躍した。

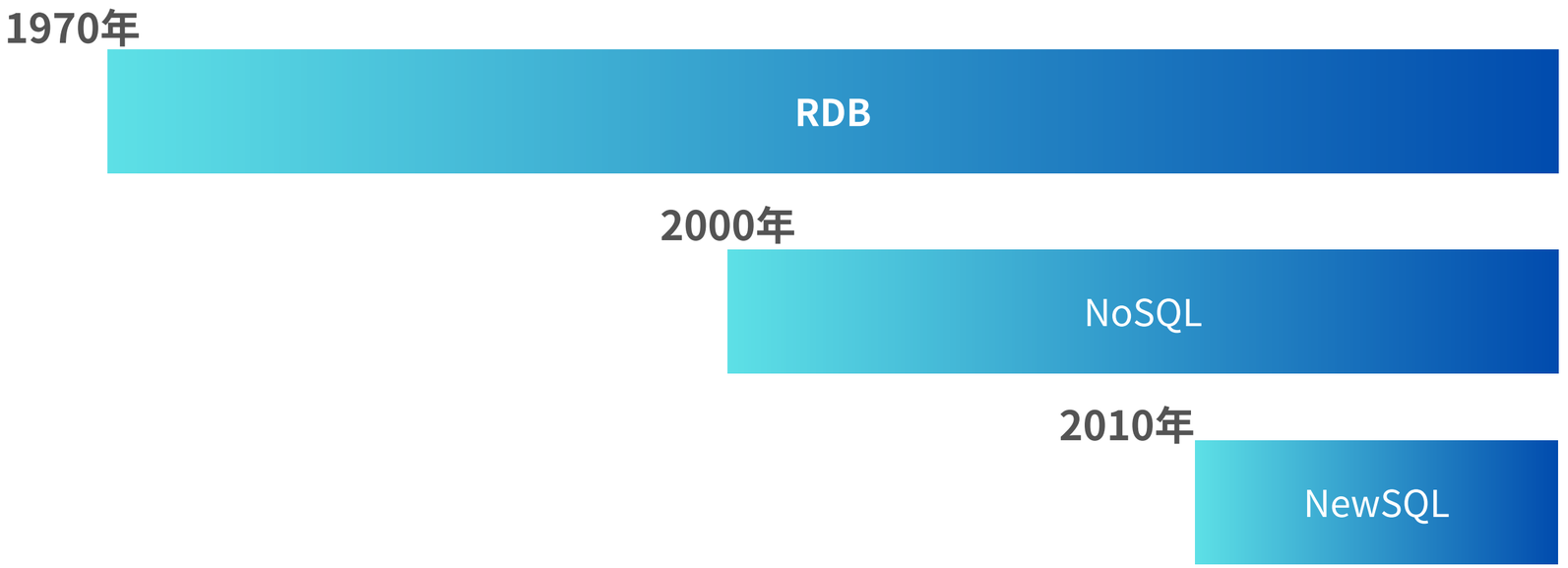
データベース



磁気ドラム

NTT技術史料館, 2016/3/10　DIPS4150形磁気ドラム記憶装置が「情報処理技術遺産」に認定されました。 | その他 | トピックス一覧, https://hct.lab.gvm-jp.groupis-ex.ntt/topics/other/o_20160310.html (2025/8/4閲覧).

コラム：SQLとデータベース変遷



柔軟なデータ構造を持ち、スケーラビリティや可用性に優れた非RDBの総称。
MongoDBやRedis、Cassandraなど。

従来のRDBと同様にSQLを使用しながら、NoSQL同様高いスケーラビリティや処理性能を実現するDB
Google SpannerやCockroachDB、TiDBなど。

RDBはデータの一貫性に非常に優れており、2025年現在も業務システムでは代替不可な要素です。デメリットとしては速度です。
これはデータベースソフトの改良、ハードウエアの向上、（あとはDBAの血と汗と涙）により何とかクリアしていました。

しかし、ビッグデータの時代が到来しデータの一貫性を犠牲にしてでも速度を求める時代が到達しました。
そこで生まれたのが「NoSQL」です。分散性、高速性に特化した「NoSQL」はデータ分析などで躍進をしました。

2010年頃からはNoSQLの高速性を持たせながら、RDBの特徴、データの一貫性を持たせ、SQLを使用する技術が生まれました。
「NewSQL」と呼ばれるデータベースです。

現在のトレンドはこんな状況ですが、この先もRDBに限らずSQLはまだまだ使われていきます。

「SQLとデータベース」おしまい

SQLの基礎知識

図書館でのSQL

SQLとデータベース

SQLと仲良くなる

何となくわかりましたか？

図書館でのSQL

2025年 IT総合研修

システムはデータの集合体である

全てのシステムは

・データをどう更新して
・データをどう表示するか

だけです。

システムとはデータの集合体なのです。
もちろんそれは図書館システムも。

SQLができるようになったときの最大のメリット

SQLができると好きな形でデータを取り出せる

書誌テーブル

貸出履歴テーブル

操作履歴テーブル

蔵書点検履歴テーブル

所蔵テーブル

利用者情報テーブル

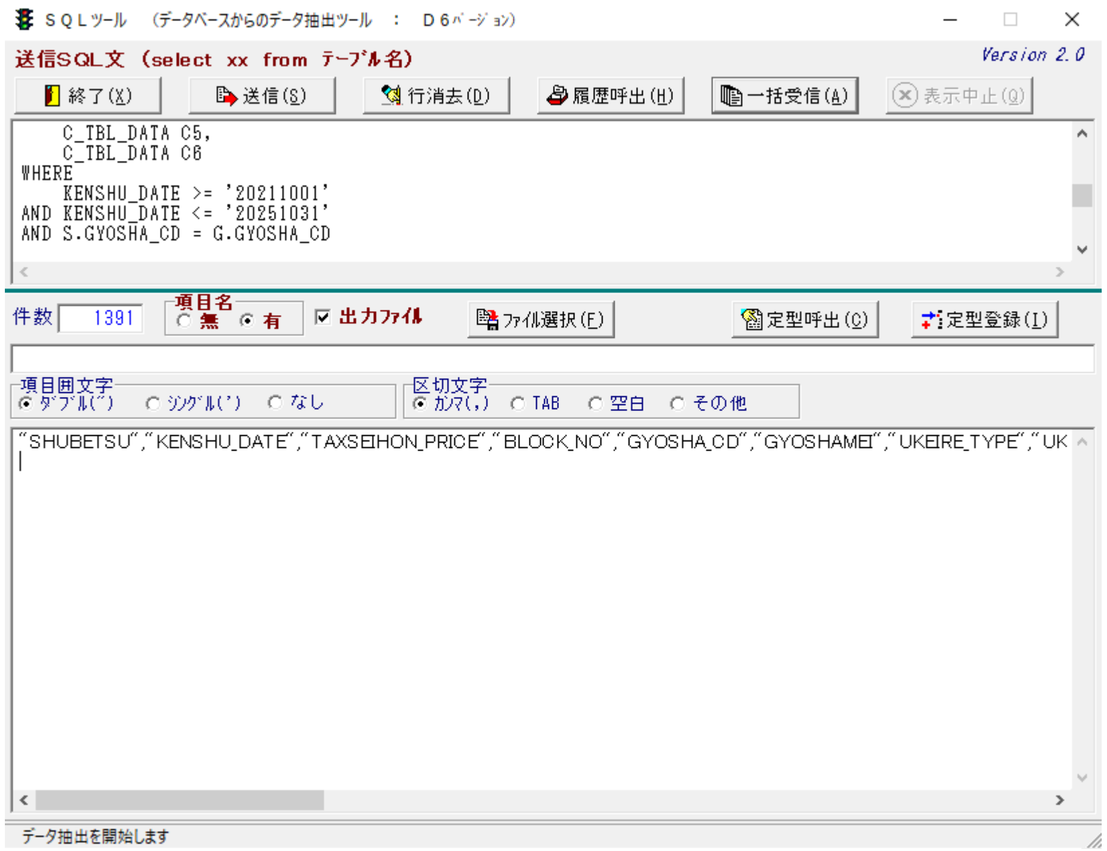
ILL情報テーブル

色々やりたいこと浮かんできませんか？

上司から急に**「こんな実績だして」**って言われたことないですか？

（俺は、ある）

図書館システムとSQL



お使いの図書館システムでSQL機能ありませんか？

あればそこからSQLでデータを取得できます。

いやー、イメージできないなぁ。って人のために。

例えばこんな使い方してます。

入力間違いを取得するＳＱＬ	不正データの組み合わせを取得し、データを修正
所蔵、受入数等基本情報ＳＱＬ	日図協調査などの作業省力化
重複図書一覧	抜き取り業務のリスト利用
当日登録データ一覧	入力データのエラーチェック、日報データ
～年以前の所蔵雑誌一覧	電子公開調査対象リスト
入退館者数データ解析	入退館システムの標準分析機能以上の詳細な調査

まとめ

システムはデータの集合体である。

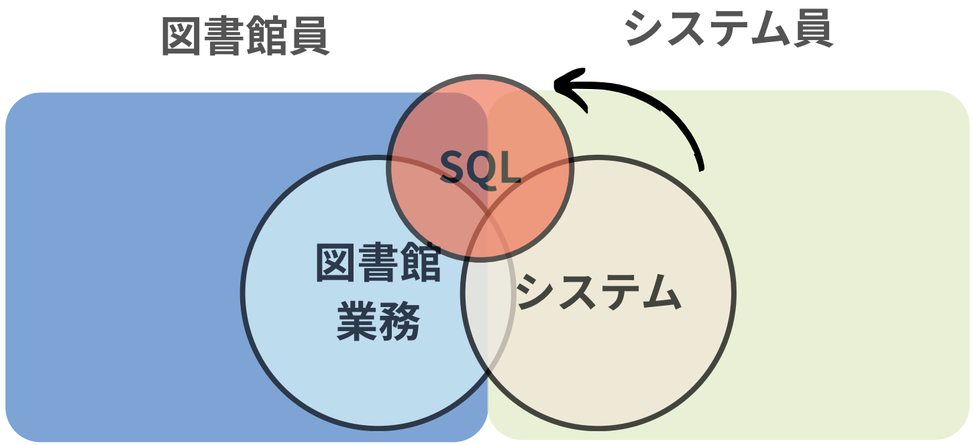
もちろん、図書館システムも。

仕事の中でExcelのデータをコネコネしている時間はどれぐらいありますか？

SQLができれば、その時間が短くなるだけじゃなく、精度も上がります。

SQLはプログラムの中だけで動くものじゃないです。
日々の業務でも活用していきましょう！

覚えてますか？　この図。



SQLができればデータの分析の大きな武器になる！
SQLができれば集計作業の大きな省力化になる！

だからこそ、図書館業務の中でSQLを活用しよう！

「図書館でのSQL」おしまい。

SQLの基礎知識

図書館でのSQL

SQLとデータベース

SQLと仲良くなる

具体的なやりたいことをイメージしよう！

SQLを読み解こう！

2025年 IT総合研修

今からSQLの構文を説明します。

その前に大事なこと。

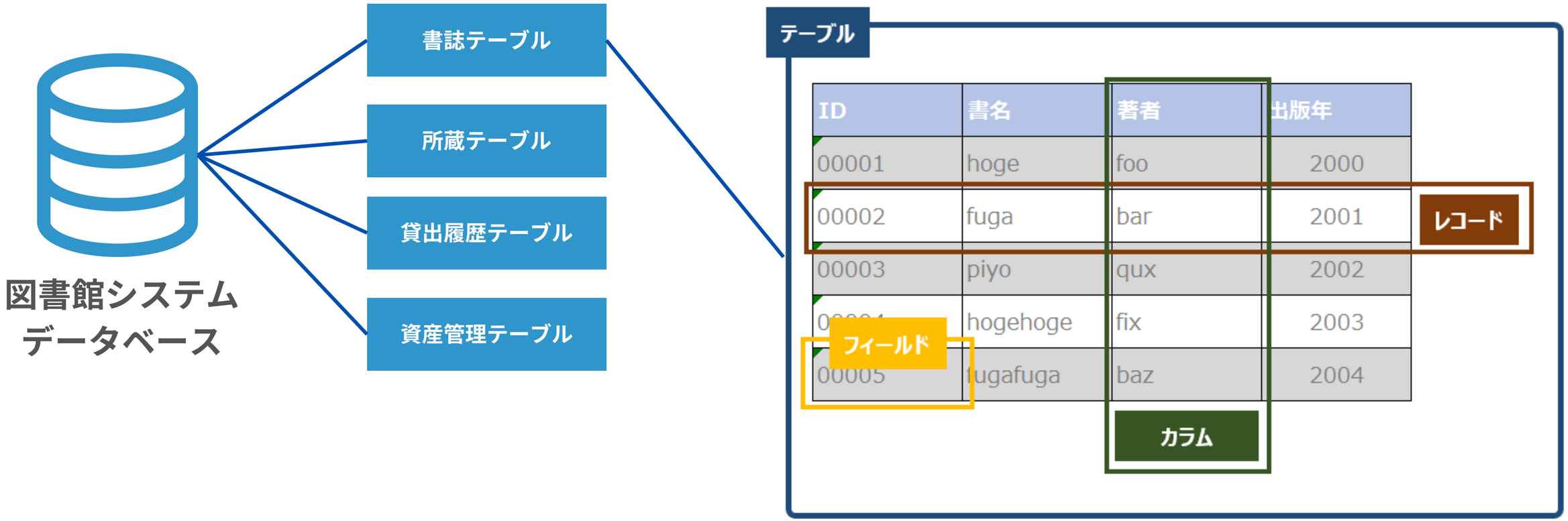
理解の仕方は人それぞれです。
実際に手を動かしてみないと分からない人もいれば、
まずは基礎知識をしっかり理解してからでないとピンとこない人もいます。

だから、これからの説明で「よく分からないな」と感じても、あきらめないでください。
大切なのは「イメージすること」です。
「表からこうやってにデータを抜くんだな」とイメージを持って聞いてみてください。

細かい文法や書き方は、あとで調べればいくらでも出てきます。
でも、イメージをつかむチャンスは“今この瞬間”だけです。

言葉の定義が分からなくなったら戻ってくる場所。

図書館システム



この図では、図書館システムと図書館データベースが１：１となっているが、そうじゃないこともあるよ。

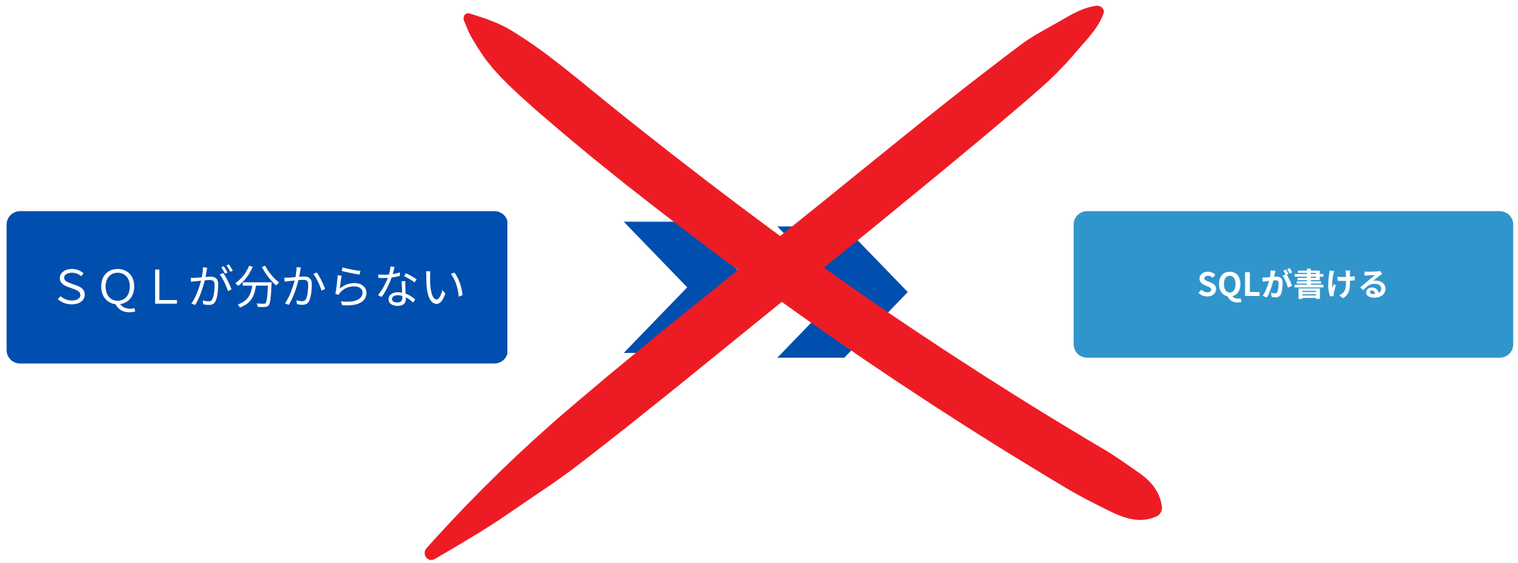
使うSQLの種類

ここで学ぶSQLの種類は大きく４つだけ！

SELECT	SELECT＝参照、データを取得する指示に使う
UPDATE	UPDATE＝更新、データを書き換えるときに使う
INSERT	INSERT＝挿入、データを新規追加するときに使う
DELETE	DELETE＝削除、データを消去するときに使う

まずは学び方を理解しよう

人はいきなり達人にはなれません。



SELECTの基本形

（基本構成）

```
SELECT      カラム名１,カラム名2,カラム名3
FROM        テーブル名
WHERE       任意：絞込条件
GROUP BY    任意：グループ化の条件
ORDER BY    任意：ソート順の条件
```

（解説）

```
SELECT      ：何を取り出すか。どの列を取り出すか。
FROM        ：どのテーブルから取り出すか
WHERE       ：取り出すデータの条件。（and,or）
GROUP BY    ：データを集計・グループ化する条件
ORDER BY    ： 表示するときの並び順
```

SELECTの基本形

```
SELECT　県名,面積,人口
FROM　近畿都道府県一覧表
WHERE　面積＞4000 and 人口<3,000,000
```

テーブル名：近畿都道府県一覧表

県名	面積	人口	地区
兵庫県	8,401	5,315,447	北部
大阪府	1,905	8,773,979	中部
京都府	4,612	2,506,796	北部
滋賀県	4,017	1,413,610	北部
奈良県	3,691	1,324,473	中部
和歌山県	4,726	922,584	南部

実行結果

県名	面積	人口
京都府	4,612	2,506,796
滋賀県	4,017	1,413,610
和歌山県	4,726	922,584

SELECTの基本形

```
SELECT　県名,面積,人口
FROM　　近畿都道府県一覧表
```

テーブル名：近畿都道府県一覧表

県名	面積	人口	地区
兵庫県	8,401	5,315,447	北部
大阪府	1,905	8,773,979	中部
京都府	4,612	2,506,796	北部
滋賀県	4,017	1,413,610	北部
奈良県	3,691	1,324,473	中部
和歌山県	4,726	922,584	南部

県名	面積	人口
兵庫県	8,401	5,315,447
大阪府	1,905	8,773,979
京都府	4,612	2,506,796
滋賀県	4,017	1,413,610
奈良県	3,691	1,324,473
和歌山県	4,726	922,584

SELECTの基本形

```
SELECT　県名,面積,人口
FROM　近畿都道府県一覧表
WHERE　面積 > 4000 and 人口 < 3000000
```

テーブル名：近畿都道府県一覧表

県名	面積	人口	地区
兵庫県	8,401	5,315,447	北部
大阪府	1,905	8,773,979	中部
京都府	4,612	2,506,796	北部
滋賀県	4,017	1,413,610	北部
奈良県	3,691	1,324,473	中部
和歌山県	4,726	922,584	南部

実行結果

県名	面積	人口
京都府	4,612	2,506,796
滋賀県	4,017	1,413,610
和歌山県	4,726	922,584

SELECTの基本形

```
SELECT   県名,面積,人口
FROM     近畿都道府県一覧表
WHERE    面積 > 4000 and 人口 < 3000000
```

テーブル名：近畿都道府県一覧表

県名	面積	人口	地区
兵庫県	8,401	5,315,447	北部
大阪府	1,905	8,773,979	中部
京都府	4,612	2,506,796	北部
滋賀県	4,017	1,413,610	北部
奈良県	3,691	1,324,473	中部
和歌山県	4,726	922,584	南部

実行結果

県名	面積	人口
京都府	4,612	2,506,796
滋賀県	4,017	1,413,610
和歌山県	4,726	922,584

SELECTの基本形

```
SELECT  地区,MAX(人口)
FROM    近畿都道府県一覧表
GROUP BY  地区
ORDER BY  地区 ASC
```

テーブル名：近畿都道府県一覧表

県名	面積	人口	地区
兵庫県	8,401	5,315,447	北部
大阪府	1,905	8,773,979	中部
京都府	4,612	2,506,796	北部
滋賀県	4,017	1,413,610	北部
奈良県	3,691	1,324,473	中部
和歌山県	4,726	922,584	南部

実行結果

地区	MAX(人口)
北部	5,315,447
中部	8,773,979
南部	922,584

あれ？　結果が返ってこないぞ

大量のデータを一括取得したとき、複雑な条件や索引（インデックス）が未付与のなどの場合、処理結果が戻ってこないことがあります。

基本的には待ってください。それでもだめなら強制終了してください。

もしそれを本番サーバでやってしまったときは、システム担当に相談してください。

実行中のSQLは完了まで残り続けます。何度も何度も強制終了するとシステムが大変なことになります。

UPDATEの基本形

（基本構成）

```
UPDATE    テーブル名
   SET    カラム名１    ＝    （更新内容）
         ,カラム名２    ＝    （更新内容）
WHERE  任意：絞込条件
```

（解説）

UPDATE　　：どのテーブルを更新するのか
SET　　：更新カラムとその内容
WHERE　　：どの条件で一括更新をするのか

UPDATEの基本形

```
UPDATE　近畿都道府県一覧表
SET　面積　= 5000
WHERE　地区　= ’北部‘
```

テーブル名：近畿都道府県一覧表

県名	面積	人口	地区
兵庫県	8,401	5,315,447	北部
大阪府	1,905	8,773,979	中部
京都府	4,612	2,506,796	北部
滋賀県	4,017	1,413,610	北部
奈良県	3,691	1,324,473	中部
和歌山県	4,726	922,584	南部

実行結果

県名	面積	人口	地区
兵庫県	8,401	5,315,447	北部
大阪府	1,905	8,773,979	中部
京都府	5,000	2,506,796	北部
滋賀県	5,000	1,413,610	北部
奈良県	3,691	1,324,473	中部
和歌山県	4,726	922,584	南部

INSERTの基本形

（基本構成）

```
INSERT INTO　　テーブル名
（カラム名１,カラム名２）
VALUES（値１,値２）
```

（解説）

INSERT INTO　：どのテーブルに挿入するのか

Q.カラム名はテーブルの全カラム書かないといけないのか？
A.全項目であれば、省略可能。ただし値の数が足りなければエラー

Q.カラム名に指定しなかったカラムには何がはいるの？
A.何も入らないです。空っぽです。ただし、テーブルの制約で
　空っぽをNGにしているカラムがあれば、エラーになります（ＩＤとか）

INSERTの基本形

```
INSERT INTO 近畿都道府県一覧表
(県名,面積)
VALUES ('広野県',5000)
```

テーブル名：近畿都道府県一覧表

県名	面積	人口	地区
兵庫県	8,401	5,315,447	北部
大阪府	1,905	8,773,979	中部
京都府	4,612	2,506,796	北部
滋賀県	4,017	1,413,610	北部
奈良県	3,691	1,324,473	中部
和歌山県	4,726	922,584	南部

実行結果

県名	面積	人口	地区
兵庫県	8,401	5,315,447	北部
大阪府	1,905	8,773,979	中部
京都府	4,612	2,506,796	北部
滋賀県	4,017	1,413,610	北部
奈良県	3,691	1,324,473	中部
和歌山県	4,726	922,584	南部
広野県	5,000		

INSERTの応用

INSERTで項目を並べるのが面倒だ

分かる。（わかる）

テーブル名：近畿都道府県一覧表

県名	面積	人口	地区
兵庫県	8,401	5,315,447	北部
大阪府	1,905	8,773,979	中部

テーブル名：新県候補

県名	面積
広野県	5,000
奥山県	6,200

```
INSERT INTO 近畿都道府県一覧表
(県名, 面積)
SELECT 県名, 面積
    FROM 新県候補;
```

県名	面積	人口	地区
兵庫県	8,401	5,315,447	北部
大阪府	1,905	8,773,979	中部
広野県	5,000		
奥山県	6,200		

DELETEの基本形

（基本構成）

DELETE　FROM　テーブル名
WHERE　任意：絞込条件

（解説）

テーブルじゃなくて行を指定して消すSQL

DELETEの基本形

```
DELETE FROM 近畿都道府県一覧表
WHERE 人口 < 2000000
```

テーブル名：近畿都道府県一覧表

県名	面積	人口	地区
兵庫県	8,401	5,315,447	北部
大阪府	1,905	8,773,979	中部
京都府	4,612	2,506,796	北部
滋賀県	4,017	1,413,610	北部
奈良県	3,691	1,324,473	中部
和歌山県	4,726	922,584	南部

テーブル名：近畿都道府県一覧表

県名	面積	人口	地区
兵庫県	8,401	5,315,447	北部
大阪府	1,905	8,773,979	中部
京都府	4,612	2,506,796	北部

大体わかりましたか？

分からないですよね。大丈夫。

やりたいことをイメージしよう

やりたいことを思い描かないと、SQLは身につきません。
参考書を読むよりも、問題集を解くイメージです。

・不明となった日付が5年前以前の図書を一括で除籍するにはどうしたらいいか？

・貸出延滞日数が30日を超えている利用者のメールアドレスを抽出したい。

思い描いて形にしよう！

生成AIにSQLを投げてみて。

最近の生成AIは賢いので、テーブル情報なしのSQLだけでも
エラーを推測してきます。

所属機関のルールに則る範囲で使ってみるのも一つです。

ただし、テーブル構成はシステムにおける重要な情報です。
あまり外に流さないようにしましょう。

空文字ってなんだ？

例えばINSERTで指定しなかった項目はNULL(≒空文字)が設定されます。

DBによっては
　①Where 地区　＝ NULL　②where 地区　is NULL ③Where 地区　＝ ‘’
で結果が変わってきます。
最初は実際のデータを見ながら確かめてください。

また古いシステムではNULLが使えず、△（＝‘ ‘(1バイトスペース)）を使うケースもあります。
SQLをかけたとしても、実際のデータを結果を見比べることも大事です。

日付型ってなんだ？

Excelで日付の形にイライラしたことはありませんか？
データベースも同じです。テーブルのフィールドには、「ここには日付が入るよ！」と指定されています。
そして日付の形はデータベースの設定によってバラバラです。

なので

where 除籍日付 = '20250702'

としても、HITする可能性は低いです。じゃあどうしたらいいのか？

日付型ってなんだ？

WHERE 日付　= TO_DATE(’20250702’,’YYYYMMDD’)

で検索してください。 これにより、’20250702’はyyyymmddの日付の形として扱うよ、という魔法をかけたことになります。
セルの書式設定みたいなやつです。

ある程度はデータベースが自動的に判断してくれる（暗黙的型変換）のですが、このようにこちらが「日付やで！！」と言ってあげる（明示的型変換）をする必要があることもあります。
そういう時は、この話を思い出して、調べてみてください。

コミットってなんだ？

UPDATE、INSERTの処理は、実行するとメモリ（中間領域）でデータを作ります。
そして、COMMITを実行することによって本番データを更新したり、挿入したりします。

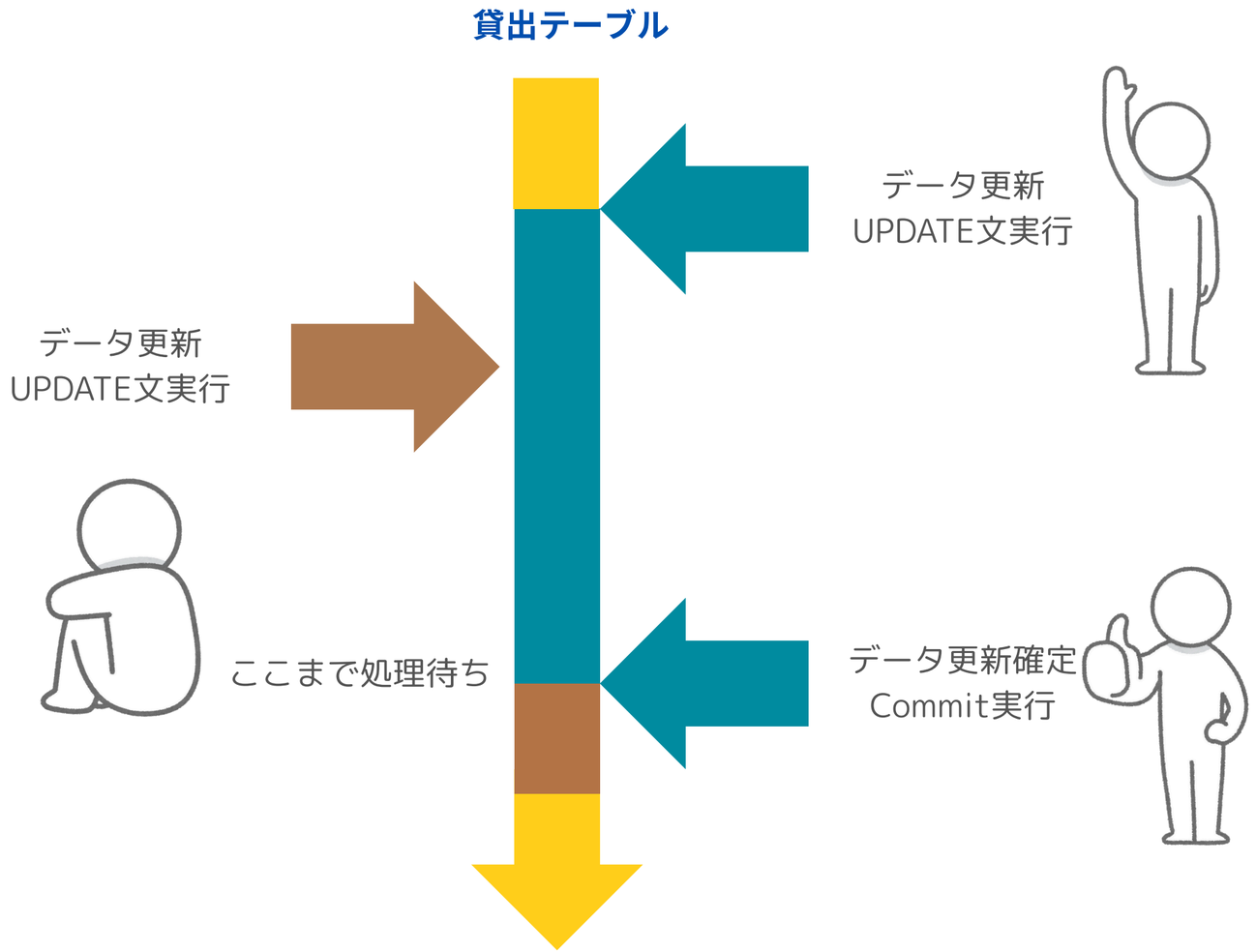
逆に、COMMIT前であれば、ROLLBACKを実行することでなかったことにできます。
利用するソフトによっては「オートコミット」があるものもあります。

ちなみにDELETEもCOMMITが必要です。それで救われた技術者がたくさんいます。

ふーん、って思っているでしょ。**これ、大事な機能です。**

補足したい大事なこと。(３／３)

コミットってなんだ？



コミットってなんだ？

じゃあ2名で同じデータを触っていて、一人がUPDATE後、COMMITをしなかったらどうなるでしょうか。
正解は、COMMITがされるまで、もう一人のSQLは実行中のままになります。

これが本番データの場合、どうなるでしょうか。システムが待つことになります。

データを触るということは、システムを触るということです。
本番データの更新はシステム会社では基本2名以上の立ち合いや事前にSQLを承認取るなどして実行します。
危険なことであることを理解してデータを触るようにしてください。

時間があれば話す予習内容。

2025年 IT総合研修

複数の表を一つの表として取得する

システム的な利便性などから表を分割することがあります。
例：多重的に値を持たない、速度を持たせる、などなど

ですが、人間が表を扱うときはそんなもんは不便でしかないです。
なので、SELECTで複数の表を結合して一つの表として取得できます。

詳しくはこの後の実習で説明するので、そんなことができるんだ、ぐらいで
この講義はOKです。

SELECTの応用形（結合）

従業員TBL

従業員番号	氏名	部署コード
000001	山口重雄	A
000002	マルコロドリゲス	C
000003	丸本美幸	B
000004	ニコライボルコフ	D

部署TBL

部署コード	部署名	住所
A	本社	東京都中央区
B	日本首都支店	京都府京都市
C	営業所	東部ごみ処理センター
E	左遷支店	シベリア市中央区

二つのテーブルの情報を一回で取得したいときは結合することで

例えば、従業員テーブルの部署コードから、部署の住所を結合させて
部署住所の表を作ってみましょう。

SELECTの応用形（等価結合）

従業員TBL

従業員番号	氏名	部署コード
000001	山口重雄	A
000002	マルコロドリゲス	C
000003	丸本美幸	B
000004	ニコライボルコフ	D

部署TBL

部署コード	部署名	住所
A	本社	東京都中央区
B	日本首都支店	京都府京都市
C	営業所	東部ごみ処理センター
E	左遷支店	シベリア市中央区

```
SELECT a.氏名,a.部署コード,b.部署名
FROM   従業員TBL a, 部署TBL b
WHERE  a.部署コード   =  b.部署コード
```

結果

氏名	部署コード	部署名
山口重雄	A	本社
マルコロドリゲス	C	営業所
丸本美幸	B	日本首都支店

２つのテーブルがある場合、どちらの表の項目か分からないので、別名（aやb）を付けて表記します。

等価結合 はどちらにもデータがあるものだけでてくるので、部署コードDの人は、部署TBLにデータがないので、取得されません。

SELECTの応用形（左結合）

従業員TBL

従業員番号	氏名	部署コード
000001	山口重雄	A
000002	マルコロドリゲス	C
000003	丸本美幸	B
000004	ニコライボルコフ	D

部署TBL

部署コード	部署名	住所
A	本社	東京都中央区
B	日本首都支店	京都府京都市
C	営業所	東部ごみ処理センター
E	左遷支店	シベリア市中央区

```
SELECT a.氏名,a.部署コード,b.部署名
FROM
    従業員TBL a
LEFT JOIN
    部署TBL b
ON
    a.部署コード = b.部署コード;
```

結果

氏名	部署コード	部署名
山口重雄	A	本社
マルコロドリゲス	C	営業所
丸本美幸	B	日本首都支店
ニコライボルコフ	D	(NULL)

左結合をすると、左の表の情報をすべて結果に出します。
その際、右側に指定したテーブルに値が無ければNULLで返却されます。
絶対にメインテーブルを全件出したい場合は左結合を使います。

SELECTの応用形（左結合）

従業員TBL

従業員番号	氏名	部署コード
000001	山口重雄	A
000002	マルコロドリゲス	C
000003	丸本美幸	B
000004	ニコライボルコフ	D

部署TBL

部署コード	部署名	住所
A	本社	東京都中央区
B	日本首都支店	京都府京都市
C	営業所	東部ごみ処理センター
E	左遷支店	シベリア市中央区
A	新本社２	東京都千代田区

```
SELECT a.氏名,a.部署コード,b.部署名
FROM
    従業員TBL a
LEFT JOIN
    部署TBL b
ON
    a.部署コード = b.部署コード;
```

結果

氏名	部署コード	部署名
山口重雄	A	本社
山口重雄	A	新本社２
マルコロドリゲス	C	営業所
丸本美幸	B	日本首都支店
ニコライボルコフ	D	(NULL)

ただし、左側を全部出すというのは、左側をメインテーブルとして、全パターン出すということです。

なので、部署TBLに間違って部署コードAを２行作ると、山口さんが分身します。ヤバイ。結合条件や部署テーブルの条件を見直しましょう。

SELECTの応用形（右結合）

従業員TBL

従業員番号	氏名	部署コード
000001	山口重雄	A
000002	マルコロドリゲス	C
000003	丸本美幸	B
000004	ニコライボルコフ	D

部署TBL

部署コード	部署名	住所
A	本社	東京都中央区
B	日本首都支店	京都府京都市
C	営業所	東部ごみ処理センター
E	左遷支店	シベリア市中央区

```
SELECT a.氏名,a.部署コード,b.部署名
FROM
    従業員TBL a
RIGHT JOIN
    部署TBL b
ON
    a.部署コード = b.部署コード;
```

結果

氏名	部署コード	部署名
山口重雄	A	本社
マルコロドリゲス	C	営業所
丸本美幸	B	日本首都支店
(NULL)	E	左遷支店

左があれば右もあります。
右結合をすると、右のテーブルの組み合わせを全件表示します。

SELECTの応用形（おまけ：等結合）

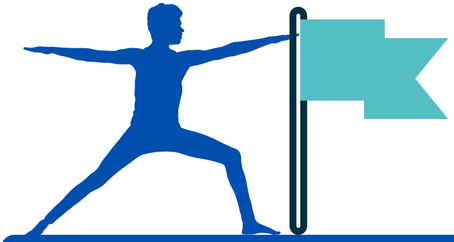
```
SELECT a.氏名,a.部署コード,b.部署名
FROM       従業員TBL a, 部署TBL b
WHERE  a.部署コード       ＝    b.部署コード
```

```
SELECT a.氏名,a.部署コード,b.部署名
FROM
   従業員TBL a
INNER JOIN
   部署TBL b
ON
   a.部署コード = b.部署コード;
```

等結合だけ書き方が変って思いましたよね。

SQL99以降の標準書法では　等結合はinner joinで書くべきと言われています。左はその当時を戦ってきた古い人間の書き方です。

「SQLの基礎」おしまい。



SQLの基礎知識

図書館でのSQL

SQLとデータベース

SQLと仲良くなる

次は実際に試してみよう！

まとめ

2025年 IT総合研修

UPDATE/DELETEは細心の注意を

本来本番環境へ更新は事前WF、SQLレビュー、立ち合いなどかなり厳しいです。
理由は何か。

１．監査上の理由

書店への支払金額を更新したらどうなるか。

２．追跡上の理由

画面更新の履歴に残らない更新内容があればどうなるか。

もう一つ大きな理由があります。

SQL大体理解した！

「システムはデータの集合体である」

データを更新するということはシステムを更新するということ。
データが壊れるということはシステムが壊れるということ。
簡単にデータが、画面が表示されなくなります。

部署コード	部署名	住所
A	本社	東京都中央区
B	左遷支店	シベリア市中央区
C	日本首都支店	京都府京都市

従業員番号	氏名	部署コード	口座情報
000001	山口重雄	A	123-4567
000002	マルコロドリゲス	A	123-4568
000003	丸本美幸	B	123-4569

口座情報	区分	支店コード
123-4567	当座	893
123-4568	当座	884
123-4569	普通	883

取引先コード	取引先名	担当部署
123-000	新日本企画	A
123-001	ｶﾌﾟｾﾙｺﾝﾋﾟｭｰﾀ	B
123-002	中村製作所	C

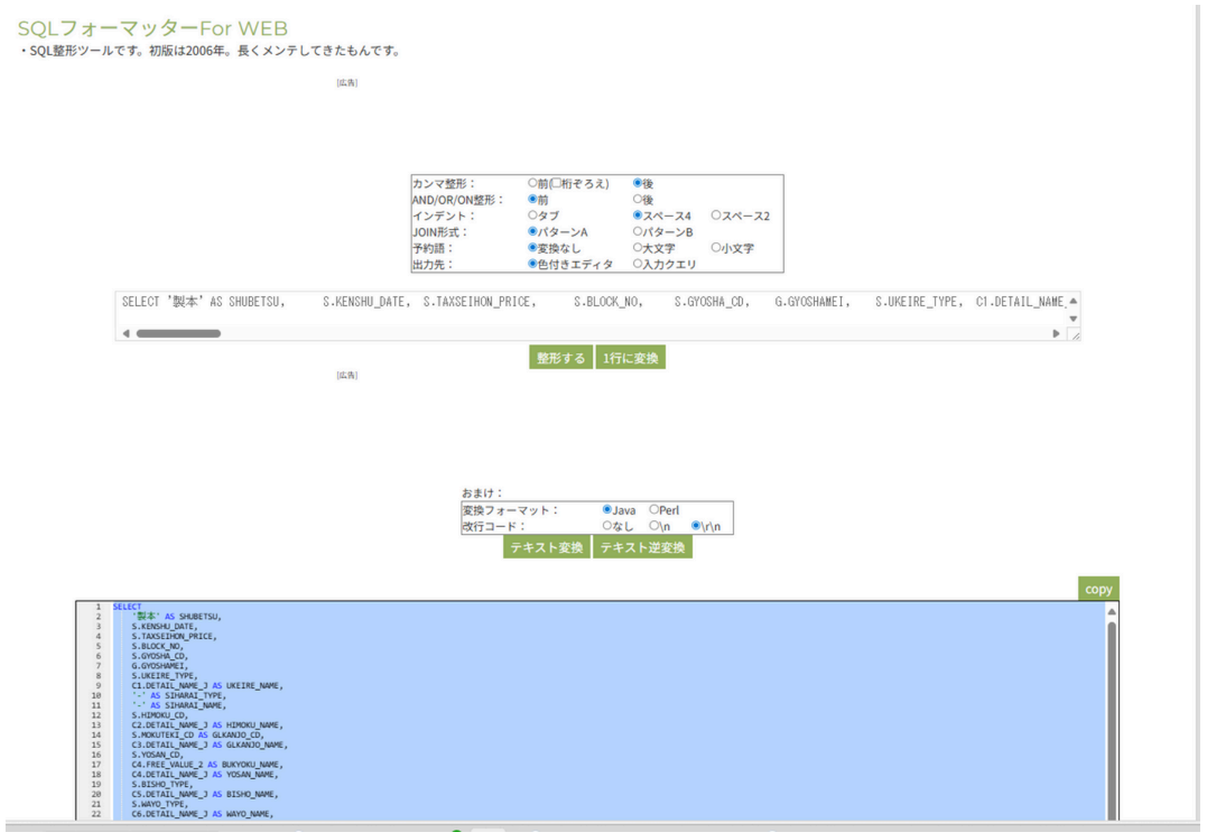
部署コード	年度	売上
A	2004	2,000,000
B	2004	4,000,000
C	2004	600,000
A	2005	2,000,000
B	2005	4,000,000
C	2005	600,000
A	2006	2,000,000
B	2006	4,000,000
C	2006	600,000

「たった１項目更新するSQLで大げさな」
そう思っている方、システム部門の人に聞いてみてください。
必ず悲しい事故の武勇伝を持ってます。**そのSQL、影響調査大丈夫？**

SQL大体理解した！

SQLが長くて読みにくい…

SQLフォーマッターなど成形ツールがあります。是非使ってみてください。



ドットツールズ, SQLフォーマッターFor WEB, https://atl2.net/webtool/sqlフォーマッターfor-web/ (2025/8/14閲覧).

まとめ

SQLに対する認識

こんなイメージを持ってませんか？

- 何なのか分からない
- 事務職員には使いこなせない
- 図書館員には不要なスキル
- 簡単に読み解けない

この講義の中で全部払拭します！

どうでしたか？全部払拭されましたか？

まだ不安な人、大丈夫です。実際に触ることで理解できることもあります！

じゃあ次は実践だ！

2025年 IT総合研修