データベース連携の拡張

組み込みデータベースへの接続性によって、Maple は、エンジニアや科学者が、企業の大規模なデータベースに Maple の最先端の解析と可視化ツールを組み合わせる強力なアプリケーションを迅速に開発し、導入することを可能にします。SQL に関する詳細な知識がなくても、Maple のデータベースを簡単に検索、作成、および更新できます。Maple 18 では、SQLite のデータベースへのネイティブサポートを含む範囲まで、データベースへの接続性が拡張されています。SQLite は、アプリに内蔵されサーバを構成しない Transact-SQL のデータベースエンジンを実装する、ソフトウェアライブラリです。SQLite は、世界で最も広く導入されている SQL のデータベースエンジンです。SQLite の詳細については、http://www.sqlite.org/ を参照してください。

以下の例では、Maple の Database パッケージを使用して SQLite データベースに G20 の国と地域の人口データをインポートし、このデータベースにアクセスして人口のテーブルとチャートを作成します。

.csv ファイルから SQLite のデータベースへの人口データのインポート

```
> with(ArrayTools):
> with(Database[SQLite]):
> with(StringTools):
```

データファイルのインポート

```
> csv := FileTools:-JoinPath([kernelopts('mapledir'), "data", "SQLite", "G20-Population.csv"]):
> data := ImportMatrix(csv):
```

メモリデータベースでの作成

Open コマンドを使用して、新しいデータベース接続を開くことができます。

```
> db := Open(":memory:")
```

$$db := \text{"SQLite database"}, table([("main") = ""])$$ (1

Population テーブルの作成

```
> sql := sprintf("CREATE TABLE population (%s)", Join(convert(data[1], 'list'), ", "))
```

$$sql := \text{"CREATE TABLE population (Date, USA, CHN, JPN, DEU, FRA, BRA, GBR, ITA, RUS, IND, CAN, AUS, ESP, MEX, KOR, IDN, TUR, SAU, ARG, ZAF)"} \quad (2$$

Execute コマンドにより、提供されたデータベース接続を使用して SQL ステートメントを実行できます。

```
> Execute(db, sql)
```

Population テーブルへのデータの挿入

```
> nrows := Size(data, 1):
> ncols := Size(data, 2):
> sql := sprintf("INSERT INTO population VALUES (%s)", Join(["?"$ncols], ", "))
```

$$sql := \text{"INSERT INTO population VALUES (?, ?)"} \quad (3$$

Prepare コマンドにより、実行用の SQL ステートメントが準備されます。

```
> stmt := Prepare(db, sql)
```

$$stmt := \text{"SQLite statement"}, \text{"INSERT INTO population VALUES (?, ?)"} \quad (4$$

```
> for i from 2 to nrows do
    for j to ncols do
      Bind(stmt, j, data[i, j])
    end do;
    Step(stmt);
    Reset(stmt, 'clear' = true)
  end do:
```

Finalize コマンドにより、準備されたステートメントが終了します。

```
> Finalize(stmt):
```

データの再読み込み

> *sql* := "SELECT · FROM population"

sql := "SELECT * FROM population" (5

> *stmt* := *Prepare*(*db*, *sql*)

stmt := "SQLite statement", "SELECT * FROM population" (6

FetchAll により、準備されたステートメントのすべての行が返されます。

> *FetchAll*(*stmt*)

$$\begin{bmatrix} 53 \text{ x } 21 \text{ Matrix} \\ \text{Data Type: anything} \\ \text{Storage: rectangular} \\ \text{Order: C\_order} \end{bmatrix} \quad (7$$

> *Finalize*(*stmt*) :

1975 年から 1990 年のカナダの人口データのプロット

> *sql* := "SELECT date, CAN FROM population WHERE date >= '1975-12-31' AND date <= '1990-12-31'"

sql := "SELECT date, CAN FROM population WHERE date >= '1975-12-31' AND date <= '1990-12-31'" (8

> *stmt* := *Prepare*(*db*, *sql*)

stmt := "SQLite statement", "SELECT date, CAN FROM population WHERE date >= '1975-12-31' AND date <= '1990-12-31'" (9

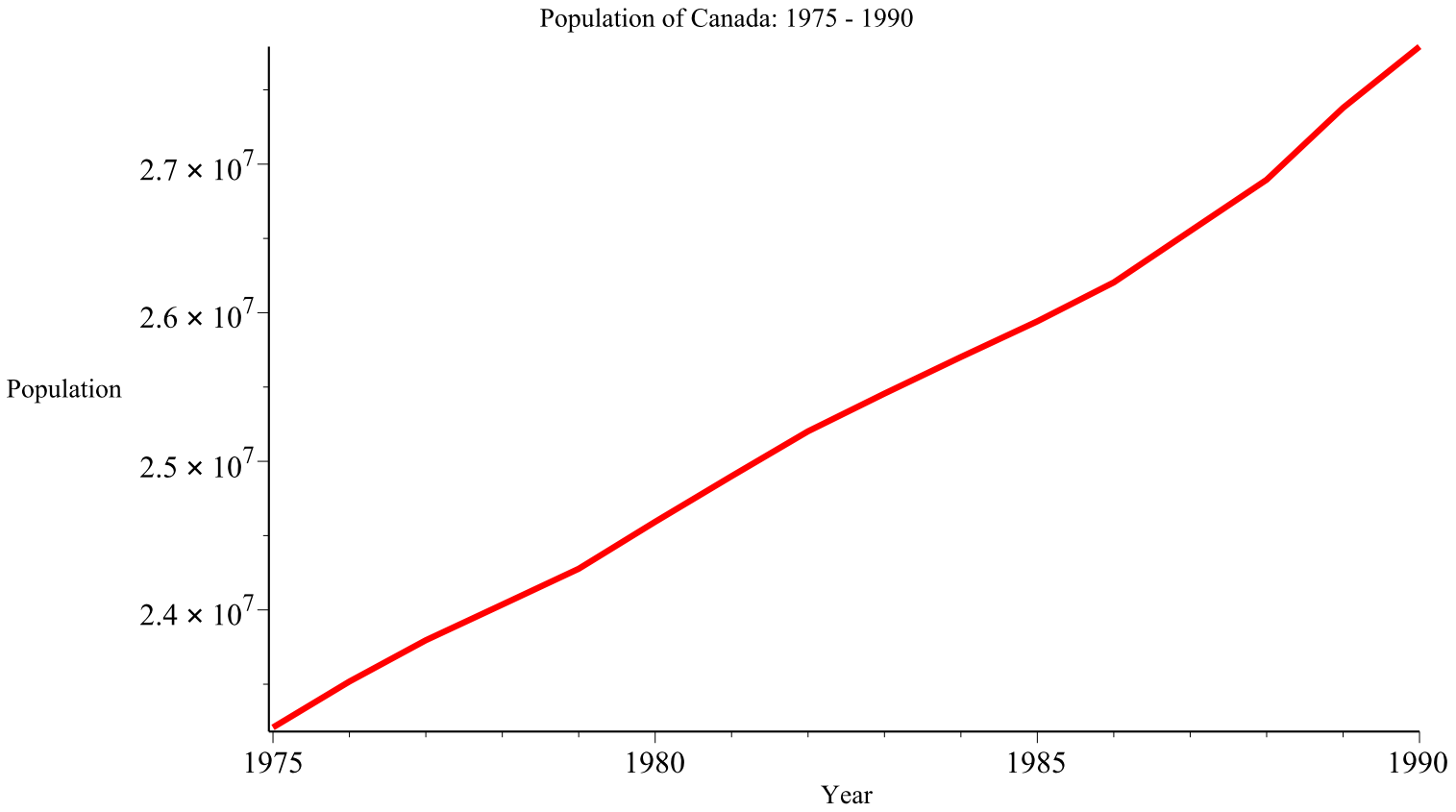
> *population* := *Matrix*(1 ..0, 1 ..2) :

> *row* := 1 :

> **while** *Step*(*stmt*) = *RESULT_ROW* **do**
  population(*row*, 1) := *sscanf*(*Fetch*(*stmt*, 0), "%d-%d-%d")$_1$;
  population(*row*, 2) := *Fetch*(*stmt*, 1); *row* := *row* + 1
  end do:

> *Finalize*(*stmt*) :

```
> plots:-pointplot(population, color = "Red", title = "Population of Canada: 1975 – 1990", labels = ["Year", "Population"], style = line, size = [800, 400])
```



参照

Database、Database[SQLite][Execute]、Database[SQLite][Finalize]、Database[SQLite][FetchAll]、Database[SQLite][Open]、Database[SQLite][Prepare]