外部キー

多くのテーブルは「主キー」でインデックス付けされますが、 「外部キー」と呼ばれるものもあります。これは2つのテーブル間のリンクとして機能します。例えば、新しいテーブルを作成してみましょう。

```
CREATE TABLEメッセージ ( id int
        NOT NULL AUTO_INCREMENT, content text
        NOT NULL, createdAt datetime
        DEFAULT CURRENT_TIMESTAMP,
        AuthorId int NOT NULL,
        主キー（ID）、
        ユニーク(ID)
        制約fk_author外部キー(AuthorId)参照 users(id)
) ;
```

- 「CONSTRAINT」では、「制約」の名前を指定できます。 （オプションですが、デフォルト名は複雑になる可能性があります）

- 「FOREIGN KEY」は、テーブルのどの列が影響を受けるかを示します。
- 「REFERENCES」は、この外部キーにリンクされているテーブルと列を示します。

テーブルにメッセージを追加してみましょう。

```
INSERT INTOメッセージ(コンテンツ, AuthorId) VALUES ("blah blah blah blah", (SELECT id
FROM users WHERE username = "Basile"));
```

ここでユーザーを削除しようとすると、次のようになります。

```
DELETE FROM users WHERE username = "Basile";
```

このユーザーを削除できないというエラーが発生しています。外部キーは安全策として使用できます。これにより、別のテーブルで必要なデータを削除してしまうことを防ぐことができます。

しかし、ここで私が望んでいるのはそれではないので、この「制約」を削除しましょう。

```
ALTER TABLEメッセージDROP FOREIGN KEY fk_author;
```

削除には「制約」に指定した名前を使用します。

新しいものを追加してみましょう：

```
ALTER TABLEメッセージに、制約fk_authorと外部キー(AuthorId)の参照 users(id) を追加して、削除をカスケードで実行します。
```

- 「ON DELETE」では、削除の際に何を行うかを指定できます。
  - 「ON UPDATE」を使用して、更新の場合に実行する操作を示すこともできます。
- 「CASCADE」は、変更をこのテーブルに反映することを示します。その他のオプションは以下のとおりです。
  - 「SET NULL」はフィールドを「NULL」に設定し、 「NO
  - ACTION」は何も行わず、
  - 「RESTRICT」はデフォルト値を変更しないようにします。

ここで再度ユーザーを削除しようとすると、次のようになります。

```
DELETE FROM users WHERE username = "Basile";
```

彼は姿を消し、彼のメッセージもすべて消え去った。

「RESTRICT」 、 「CASCADE」 、 「SET NULL」の使用は、データベースの種類、プロジェクト、会社、状況によって異なります。