**データベースの正規化**

**1. 非正規形（UNF）**

非正規形では、データが整理されておらず、1つのセルに複数の値が含まれています。この状態では、データ管理が非効率で、検索や更新に手間がかかります。

**例:**

| **生徒ID** | **生徒名** | **科目** | **成績** |
| --- | --- | --- | --- |
| S001 | 山田 | 数学, 英語 | 90, 85 |
| S002 | 鈴木 | 数学 | 78 |

**問題点:**

* **冗長性**: 1セルに複数の値が含まれ、データを個別に扱えない。
* **操作の困難さ**: 数学だけの成績を変更する場合、部分的な編集が必要。

**対策:**

各データを1セル1データに分け、行ごとに整理します。これにより、\*\*第1正規形（1NF）\*\*に進みます。

**2. 第1正規形（1NF）**

第1正規形では、すべてのセルを「原子値」（分割不可能な単位）に整理します。

**例:**

| **生徒ID** | **生徒名** | **科目** | **成績** |
| --- | --- | --- | --- |
| S001 | 山田 | 数学 | 90 |
| S001 | 山田 | 英語 | 85 |
| S002 | 鈴木 | 数学 | 78 |

**正規形: 第1正規形（1NF）**

**理由: データが原子値で構成され、1セル1データを実現しています。**

**課題:**

* **冗長性**: 「生徒名」が繰り返し記録され、重複が発生。
* **更新異常**: 例えば、生徒名を変更する場合、すべての行を変更する必要がある。

**対策:**

複合主キーの一部にのみ依存するデータを分割し、\*\*第2正規形（2NF）\*\*に進みます。

**3. 第2正規形（2NF）**

第2正規形では、**部分的関数従属**を排除します。複合主キーの一部に依存する非キー列を分割し、独立したテーブルに整理します。

**分割の基準:**

* **生徒ID** → **生徒名**
* **科目ID** → **科目名**

**例:**

**生徒-科目-成績テーブル**

| **生徒ID** | **科目ID** | **成績** |
| --- | --- | --- |
| S001 | C001 | 90 |
| S001 | C002 | 85 |
| S002 | C001 | 78 |

**生徒テーブル**

| **生徒ID** | **生徒名** |
| --- | --- |
| S001 | 山田 |
| S002 | 鈴木 |

**科目テーブル**

| **科目ID** | **科目名** |
| --- | --- |
| C001 | 数学 |
| C002 | 英語 |

**正規形: 第2正規形（2NF）**

**部分的関数従属:**

* **生徒名**は、生徒IDのみに依存していたため、生徒テーブルに分割。
* **科目名**は、科目IDのみに依存していたため、科目テーブルに分割。

**課題:**

* **推移的関数従属**が残っている場合は、さらに分割が必要。

**4. 第3正規形（3NF）**

第3正規形では、**推移的関数従属**を排除します。非キー列同士が依存しないようにデータを整理します。

**分割の基準:**

* **科目ID** → **科目名**（直接依存）
* **成績**は主キー（生徒ID + 科目ID）にのみ依存し、他の非キー列に依存しない。

**例:**

**生徒-科目-成績テーブル**

| **生徒ID** | **科目ID** | **成績** |
| --- | --- | --- |
| S001 | C001 | 90 |
| S001 | C002 | 85 |
| S002 | C001 | 78 |

**科目テーブル**

| **科目ID** | **科目名** |
| --- | --- |
| C001 | 数学 |
| C002 | 英語 |

**正規形: 第3正規形（3NF）**

**推移的関数従属: なし**

* **成績**は主キーに直接依存しており、他の非キー列（科目名など）に依存していない。

**正規化の目的と理由**

1. **冗長性の削減**: データの重複を最小化し、ストレージの効率を向上。
2. **データの一貫性**: 更新異常や削除異常を回避し、データ整合性を維持。
3. **データ操作の簡易化**: 検索や更新が効率的になる。

**まとめ: 正規化の覚え方**

* **1NF**: データを「1つ1つ」に分ける（1セル1データ）。
* **2NF**: 主キーの「全部」に依存させる（部分的関数従属をなくす）。
* **3NF**: 主キー「だけ」に依存させる（非キー列同士の依存をなくす）。

この形式を採用することで、テーブルの見やすさと説明のわかりやすさを両立させられます。

4o

### ****第2正規形（2NF）と部分関数従属の排除の解説****

第2正規形における「部分関数従属の排除」とは、データベースの設計において、主キーの**一部にのみ依存する列**を分離することを指します。この概念を理解するために、「複合キー」「候補キー」についての説明と具体例を使いながら解説します。

### ****用語の説明****

1. **主キー（Primary Key）**
   * テーブル内の各行を一意に識別するためのキー。
   * 主キーは1つの列でも複数の列（複合キー）でも構成可能です。
2. **候補キー（Candidate Key）**
   * 主キーの候補となるキー。テーブル内の行を一意に識別できるすべての列または列の組み合わせが該当します。
   * 主キーは候補キーの中から選ばれる。
3. **複合キー（Composite Key）**
   * 主キーが**複数の列**で構成される場合のキー。
   * 例: 「生徒ID」と「科目ID」を組み合わせて1つの行を一意に識別する。
4. **部分関数従属**
   * 複合キーの**一部の列に依存**している列のことです。
   * この依存を残したままでは冗長性やデータの不整合が生じやすくなるため、第2正規形で解消します。

### ****具体例を使った解説****

#### ****正規化前のテーブル（第1正規形: 1NF）****

| **生徒ID** | **科目ID** | **科目名** | **担任ID** | **担任名** |
| --- | --- | --- | --- | --- |
| 101 | S01 | 数学 | T01 | 佐藤先生 |
| 101 | S02 | 英語 | T02 | 高橋先生 |
| 102 | S01 | 数学 | T01 | 佐藤先生 |

* **主キー**: 生徒ID と 科目ID の組み合わせ（複合キー）。
* **問題点**:
  + 列「科目名」「担任ID」「担任名」は、科目ID のみで決定されており、生徒ID には依存していません。
  + これは「部分関数従属」に該当します。

#### ****部分関数従属の問題点****

「科目名」「担任ID」「担任名」は 科目ID のみで決定されるため、冗長なデータが発生しやすくなります。

* **冗長性の例**:
  + 「S01」に関連する「数学」「T01」「佐藤先生」が複数回出現しており、重複データが増えています。
* **更新時の問題**:
  + 例えば、「佐藤先生」の名前を変更する場合、すべての行を更新しなければ不整合が発生します。

#### ****第2正規形（2NF）への変換****

部分関数従属を解消するため、「科目ID」に依存する列を別のテーブルに分割します。

##### **(1) 生徒-科目対応テーブル**

| **生徒ID** | **科目ID** |
| --- | --- |
| 101 | S01 |
| 101 | S02 |
| 102 | S01 |

##### **(2) 科目テーブル**

| **科目ID** | **科目名** | **担任ID** | **担任名** |
| --- | --- | --- | --- |
| S01 | 数学 | T01 | 佐藤先生 |
| S02 | 英語 | T02 | 高橋先生 |

### ****ポイントのまとめ****

1. **部分関数従属の排除**:
   * 複合キーの一部（この場合 科目ID）にのみ依存する列（科目名、担任ID、担任名）を別テーブルに分離します。
2. **冗長性の削減**:
   * 「科目ID」ごとに情報を一意に管理することで、重複データを減らします。
3. **候補キーの選定**:
   * 生徒-科目対応テーブルでは、「生徒ID」と「科目ID」の組み合わせが主キーであり候補キーです。

これにより、データベースが効率的で一貫性を保ちやすくなります！

**主キーの選定基準**

主キーの選定は、データベース設計において非常に重要です。主キーはテーブル内の行を一意に識別するためのキーであり、データの整合性や一貫性を保証します。以下に、主キーを選定する際の基準と「生徒-科目対応テーブル」での選定理由を説明します。

**主キー選定の一般的な基準**

1. **一意性**
   * 主キーはテーブル内のすべての行を一意に識別できる必要があります。
2. **非NULL制約**
   * 主キーは必ず値を持つ（NULL値を許容しない）必要があります。
3. **安定性**
   * 主キーの値は、データのライフサイクルを通じて変更される可能性が極めて低い必要があります。
4. **シンプルさ**
   * 主キーは可能な限り簡単で短いものが望ましいです。
5. **冗長性の排除**
   * 主キーはテーブルの他の列から推測できない（独立性を持つ）必要があります。

**候補キーとは**

候補キーとは、上記の基準を満たす列または列の組み合わせのうち、**主キーの候補となるもの**を指します。テーブルには複数の候補キーが存在する場合がありますが、その中から1つを主キーとして選定します。

**「生徒-科目対応テーブル」の候補キーの選定**

**「生徒-科目対応テーブル」**

| **生徒ID** | **科目ID** |
| --- | --- |
| 101 | S01 |
| 101 | S02 |
| 102 | S01 |

**候補キーの考え方**

* **列「生徒ID」のみ**では一意性を保証できません。  
  例: 生徒ID「101」が複数の科目（S01、S02）を受講している。
* **列「科目ID」のみ**でも一意性を保証できません。  
  例: 科目ID「S01」を複数の生徒（101、102）が受講している。
* **「生徒ID」と「科目ID」の組み合わせ**であれば一意性を保証できます。  
  例: 組み合わせ「101-S01」「101-S02」「102-S01」が各行を一意に識別します。

**主キーの選定理由**

「生徒ID」と「科目ID」の組み合わせを主キーとして選定した理由は次の通りです。

1. **一意性**:
   * 「生徒ID」と「科目ID」の組み合わせが、各行を一意に識別できる。
2. **シンプルさ**:
   * 必要最小限の列で一意性を実現している。
3. **非NULL制約の満足**:
   * 「生徒ID」と「科目ID」はどちらもNULL値を許容しない。
4. **安定性**:
   * 「生徒ID」と「科目ID」はそれぞれ独立した情報であり、頻繁に変更される可能性が低い。

**主キー選定時の注意点**

1. **冗長性に注意**:  
   主キーを複数列で構成する場合、不要な列を含めないようにします。
2. **候補キーが複数ある場合**:  
   候補キーが複数存在する場合は、以下を考慮して選定します。
   * シンプルな構成か（列数が少ない方が望ましい）。
   * 更新頻度が低いか。

**補足: 代理キーの利用**

場合によっては、自然キー（元のデータから構成される主キー）ではなく、**代理キー（Surrogate Key）** を導入することもあります。例えば、「生徒-科目対応テーブル」に新しい一意のID（例: 登録ID）を追加し、主キーとして使用する方法です。

**まとめ**

「生徒-科目対応テーブル」における主キーの選定では、「生徒ID」と「科目ID」の組み合わせが一意性を保証するため、候補キーとして適切です。この選定は、主キーとして必要な基準（特に一意性とシンプルさ）を満たしているため、主キーとして採用されます。

### ****第3正規形（3NF）と推移的関数従属の排除について****

第3正規形（3NF）では、「推移的関数従属」を排除することを目的とします。以下に詳細な解説を提供します。

### ****1. 推移的関数従属とは？****

**定義**:  
ある列（非キー列）が主キーに依存しているが、間接的に他の非キー列を介して依存している場合を「推移的関数従属」と呼びます。

#### ****記述例****:

1. 主キー → 非キー列 A
2. 非キー列 A → 非キー列 B

この場合、非キー列 B は主キーに間接的に依存しています。このような関係を「推移的関数従属」といいます。

### ****2. なぜ推移的関数従属を排除するのか？****

推移的関数従属が存在すると、次のような問題が発生します。

1. **データの冗長性**  
   非キー列が他の非キー列に依存するため、データが重複しやすくなります。
2. **更新時異常**  
   一部の非キー列を更新する際に、複数の行を修正しなければならなくなり、整合性が損なわれる可能性があります。
3. **削除時異常**  
   特定の行を削除すると、それに関連するデータが失われる危険があります。

### ****3. 第3正規形（3NF）のルール****

第3正規形では、次の条件を満たすようにテーブルを分割します。

1. 第2正規形（2NF）の条件を満たしている。
2. 主キーに依存しない非キー列が存在しない（推移的関数従属がない）。

### ****4. 第3正規形の具体例****

#### ****例: 正規化前のテーブル****

以下の「科目と担当者」テーブルを考えます。

| **科目ID** | **科目名** | **担任ID** | **担任名** |
| --- | --- | --- | --- |
| S01 | 数学 | T01 | 佐藤 |
| S02 | 英語 | T02 | 高橋 |
| S03 | 理科 | T01 | 佐藤 |

#### ****分析****

* 主キーは「科目ID」です。
* **関数従属性**:
  + 科目ID → 科目名
  + 科目ID → 担任ID
  + 担任ID → 担任名

ここで、「担任名」は主キーである「科目ID」に直接依存しておらず、「担任ID」を介して間接的に依存しています。この関係が「推移的関数従属」です。

#### ****第3正規形に分割する****

1. 「科目ID → 担任ID」という直接的な関係を保つために、「科目ID」「科目名」「担任ID」を含むテーブルを作成します。
2. 「担任ID → 担任名」という独立した関係を別のテーブルに分離します。

##### **正規化後のテーブル**

1. **科目テーブル**

| **科目ID** | **科目名** | **担任ID** |
| --- | --- | --- |
| S01 | 数学 | T01 |
| S02 | 英語 | T02 |
| S03 | 理科 | T01 |

1. **担任テーブル**

| **担任ID** | **担任名** |
| --- | --- |
| T01 | 佐藤 |
| T02 | 高橋 |

### ****5. 推移的関数従属を排除するメリット****

1. **冗長性の削減**
   * 担任名「佐藤」の情報が1か所にのみ記録されるため、重複がなくなります。
2. **データの一貫性向上**
   * もし「佐藤」の名前が変更された場合でも、担任テーブル内のデータを1か所変更するだけで済みます。
3. **操作の効率化**
   * データの更新・削除・挿入時に異常が起きにくくなります。

### ****6. 第3正規形の注意点****

* 第3正規形まで正規化することで、データが分散されるため、**クエリの性能**が低下する場合があります。
* 実際のシステム設計では、データの使用頻度や速度要求に応じて、正規化の度合いを調整することが重要です。

### ****7. まとめ****

* **推移的関数従属**とは、非キー列が他の非キー列を介して主キーに依存する状態を指します。
* 第3正規形では、推移的関数従属を排除し、非キー列が主キーに直接依存するようにテーブルを分割します。
* 第3正規形を採用することで、データの冗長性を削減し、データベースの効率性や整合性を向上させます。

正規化を実施する際は、システムの要件や性能を考慮し、過剰な分割を避けることも重要です。

**部分関数従属と推移的関数従属の違い**

データベースの正規化における「部分関数従属」と「推移的関数従属」は、それぞれ異なる問題を扱っています。それぞれを詳しく説明し、その違いを明確にします。

**1. 部分関数従属（Partial Dependency）**

**定義**

* 複合主キーが存在する場合、主キーの一部の列に依存している非キー列のことを指します。
* \*\*第1正規形（1NF）**から**第2正規形（2NF）\*\*へ正規化する際に解消します。

**具体例**

以下のようなテーブルを考えます：

| **生徒ID** | **科目ID** | **科目名** | **成績** |
| --- | --- | --- | --- |
| S001 | C001 | 数学 | 90 |
| S002 | C002 | 英語 | 85 |

**主キー**は「生徒ID」と「科目ID」の組み合わせです。

* \*\*「科目名」\*\*は、「科目ID」によって一意に決定できます（「科目ID → 科目名」）。  
  つまり、「科目名」は複合主キー「生徒ID, 科目ID」の一部「科目ID」に依存しています。
* この状態が「部分関数従属」です。

**解消方法**

「科目ID」と「科目名」の関係を別テーブルに分割します：

* **成績テーブル**

| **生徒ID** | **科目ID** | **成績** |
| --- | --- | --- |
| S001 | C001 | 90 |
| S002 | C002 | 85 |

* **科目テーブル**

| **科目ID** | **科目名** |
| --- | --- |
| C001 | 数学 |
| C002 | 英語 |

**2. 推移的関数従属（Transitive Dependency）**

**定義**

* 非キー列が他の非キー列を通じて主キーに依存している状態を指します。
* \*\*第2正規形（2NF）**から**第3正規形（3NF）\*\*へ正規化する際に解消します。

**具体例**

以下のようなテーブルを考えます：

| **商品ID** | **商品名** | **メーカーID** | **メーカー名** |
| --- | --- | --- | --- |
| P001 | 洗濯機 | M001 | メーカーA |
| P002 | 冷蔵庫 | M002 | メーカーB |

**主キー**は「商品ID」です。

* \*\*「メーカー名」\*\*は、「メーカーID」に依存しています（「メーカーID → メーカー名」）。
* さらに、「メーカーID」は「商品ID」に依存しています（「商品ID → メーカーID」）。
* このように、「メーカー名」は主キー「商品ID」を\*\*間接的に（推移的に）\*\*参照している状態です。

**解消方法**

「メーカーID」と「メーカー名」の関係を別テーブルに分割します：

* **商品テーブル**

| **商品ID** | **商品名** | **メーカーID** |
| --- | --- | --- |
| P001 | 洗濯機 | M001 |
| P002 | 冷蔵庫 | M002 |

* **メーカーテーブル**

| **メーカーID** | **メーカー名** |
| --- | --- |
| M001 | メーカーA |
| M002 | メーカーB |

**違いのまとめ**

| **項目** | **部分関数従属** | **推移的関数従属** |
| --- | --- | --- |
| **定義** | 複合主キーの一部に依存する非キー列の問題 | 非キー列が他の非キー列を通じて主キーに依存 |
| **正規化の段階** | 第1正規形（1NF）→ 第2正規形（2NF） | 第2正規形（2NF）→ 第3正規形（3NF） |
| **原因** | 複合主キーの存在 | 非キー列間の依存関係 |
| **解決方法** | 主キーの一部に依存する列を別テーブルに分ける | 非キー列間の依存関係を別テーブルに分ける |
| **例** | 科目ID → 科目名 | メーカーID → メーカー名 |

第二正規形では複合主キー以外のキーに依存する非キーが存在しない　第3正規形では主キー以外のキーに依存する非キーが存在しない　という理解であってるでしょうか

###### ChatGPT:

ChatGPT

はい、基本的にその理解で正しいです。以下で詳細に解説します。

### ****第二正規形（2NF）の理解****

**条件**

1. 第一正規形（1NF）を満たしていること。
2. 非キー列が**主キー全体**に完全に依存していること（部分関数従属がないこと）。

**重要なポイント**

* **複合主キー**が存在する場合、非キー列が主キーの一部にだけ依存する「部分関数従属」を解消します。
* 複合主キーの一部だけに依存する列を別テーブルに分離することで、非キー列が**主キー全体**に完全に依存する形にします。

#### ****具体例****

| **生徒ID** | **科目ID** | **科目名** | **成績** |
| --- | --- | --- | --- |
| S001 | C001 | 数学 | 90 |
| S002 | C002 | 英語 | 85 |

* 主キーは「生徒ID」と「科目ID」の複合主キーです。
* \*\*「科目名」\*\*は「科目ID」に依存しており、「生徒ID」には依存していません。
  + これが**部分関数従属**です。
* このため、「科目ID → 科目名」の部分を切り出して分ける必要があります。

### ****第三正規形（3NF）の理解****

**条件**

1. 第二正規形（2NF）を満たしていること。
2. 非キー列が**主キー以外のキー（非キー列）に依存しない**こと（推移的関数従属がないこと）。

**重要なポイント**

* 非キー列が他の非キー列を通じて主キーに依存している場合、それを解消します。

#### ****具体例****

| **商品ID** | **商品名** | **メーカーID** | **メーカー名** |
| --- | --- | --- | --- |
| P001 | 洗濯機 | M001 | メーカーA |
| P002 | 冷蔵庫 | M002 | メーカーB |

* 主キーは「商品ID」です。
* \*\*「メーカー名」\*\*は「メーカーID」に依存し、「メーカーID」は「商品ID」に依存しています。
  + これが**推移的関数従属**です。
* このため、「メーカーID → メーカー名」の部分を切り出して分ける必要があります。

### ****2NFと3NFの違いを整理****

| **項目** | **第二正規形（2NF）** | **第三正規形（3NF）** |
| --- | --- | --- |
| **問題の焦点** | 複合主キーの一部に依存する非キー列の解消 | 非キー列同士の依存関係の解消 |
| **依存関係** | 主キーの一部に依存する非キー列を解消 | 主キー以外のキーに依存する非キー列を解消 |
| **正規化の目的** | 主キー全体に完全に依存させる | 非キー列が直接主キーに依存するようにする |

### ****覚え方****

* **2NF（第二正規形）**: 「主キーの一部だけに依存するデータを分ける」
  + **例**: 複合主キーが親だとしたら、片親だけに依存するデータを分けて整理。
* **3NF（第三正規形）**: 「非キー同士の間接的な依存を分ける」
  + **例**: データの中に「他の非キーを経由して」しか意味が決まらないものがあれば、それを分ける。

このように整理することで、2NFと3NFの違いがはっきりします。

**第二正規形（2NF）と第3正規形（3NF）の違いについて**

* **第二正規形（2NF）**: 主キーが複合キー（複数の列を組み合わせたキー）である場合、**複合キーの一部に依存する非キー列を排除**することが求められます。つまり、**複合主キーの一部に依存する列**があれば、それを新しいテーブルに分ける必要があります。
* **第三正規形（3NF）**: 主キー以外の非キー列が、他の非キー列に依存しないようにすることが求められます。具体的には、**推移的関数従属**を排除します。

**ご質問の表について**

以下の表は、**複合主キーを使っており**、かつ**科目名が科目IDに依存している**ため、まだ**第二正規形（2NF）には達していない**ということです。

| **生徒ID** | **科目ID** | **科目名** | **成績** |
| --- | --- | --- | --- |
| S001 | C001 | 数学 | 90 |
| S002 | C002 | 英語 | 85 |

**第二正規形（2NF）に適合するための修正**

この表は、**科目名**が**科目ID**に依存しているので、これを別のテーブルに分ける必要があります。これにより、\*\*第二正規形（2NF）\*\*を満たすことができます。

**修正後のテーブル（2NFに適合）**

1. **生徒-科目-成績テーブル**（複合主キー：生徒ID + 科目ID）

| **生徒ID** | **科目ID** | **成績** |
| --- | --- | --- |
| S001 | C001 | 90 |
| S002 | C002 | 85 |

1. **科目テーブル**（科目ID -> 科目名）

| **科目ID** | **科目名** |
| --- | --- |
| C001 | 数学 |
| C002 | 英語 |

このように、**科目名が科目IDに依存している**状態を解消するために、**科目ID**と**科目名**を別テーブルに分けることで、\*\*第二正規形（2NF）\*\*を満たすようになります。

**3NF（第三正規形）の適用**

3NFを満たすためには、さらに**非キー列同士が依存していないこと**を確認します。既に**科目テーブル**で「科目ID → 科目名」の依存関係が整理されていますので、**これで3NFにも適合**しています。

**結論**

最初の状態では**第二正規形（2NF）には達していませんが、科目名を科目テーブル**に分けることで、**2NF**と**3NF**を満たします。

したがって、最初に記載された表は**2NFを満たしていない状態**ですが、適切に分けることで、2NFおよび3NFを満たすことができます。

3NF（第三正規形）の適用に関して、以下のように整理します。3NFを満たすためには、非キー列同士の依存関係を排除する必要があります。すでに科目テーブルにおいて「科目ID → 科目名」の依存関係が分割されているので、次のように整理されたテーブルが3NFに適合しています。

**3NFに適合したテーブルの例**

**1. 生徒-科目-成績テーブル（複合主キー：生徒ID + 科目ID）**

ここでは、**生徒ID**と**科目ID**の組み合わせを主キーとして使用し、成績を記録します。

| **生徒ID** | **科目ID** | **成績** |
| --- | --- | --- |
| S001 | C001 | 90 |
| S002 | C002 | 85 |

* **主キー**: 生徒ID + 科目ID（複合主キー）
* **非キー列**: 成績（主キーに完全に依存している）

**2. 科目テーブル（科目ID -> 科目名）**

このテーブルでは、**科目ID**に対応する**科目名**を記録しています。科目IDに依存する科目名の情報は、このテーブルで管理されます。

| **科目ID** | **科目名** |
| --- | --- |
| C001 | 数学 |
| C002 | 英語 |

* **主キー**: 科目ID
* **非キー列**: 科目名（科目IDに依存）

**3NFの適用理由**

* **生徒-科目-成績テーブル**では、**非キー列である成績**が、**主キーである生徒IDと科目IDに依存**しており、他の非キー列との依存関係は存在しません。このため、3NFを満たしています。
* **科目テーブル**では、**科目ID**が主キーであり、**科目名**が科目IDに依存しています。このテーブル内で非キー列同士（例えば、科目名が成績に依存するなど）に依存関係がないため、3NFを満たしています。

**まとめ**

* **3NF**では、非キー列同士の依存関係を排除することが求められますが、上記の2つのテーブルでは、すでに非キー列同士に依存関係は存在しないため、これで3NFが満たされます。

このように、複合主キーのテーブルでも、適切に非キー列の依存関係を整理することで、3NFを適用することができます。

**正規化の目的と理由**

1. **冗長性の削減**: データの重複をなくし、効率的なデータ管理を実現します。
2. **データの一貫性**: 更新異常や削除異常を回避します。
3. **可読性の向上**: データ構造を明確化し、保守性を向上させます。

**覚え方:**

* **1NF**: データを「1つ1つ」に分ける（1セル1データ）。
* **2NF**: 主キーの「全部」に依存させる（部分的依存をなくす）。
* **3NF**: 主キー「だけ」に依存させる（非キー列同士の依存をなくす）。

このプロセスを覚えておくことで、正規化の意味をより深く理解できるようになります。

以下に、それぞれの質問に回答します。

**1. 2NFでは、非キー列同士の依存があってもいいのですか？**

**回答:**  
はい、**2NFでは非キー列同士の依存があっても問題ありません**。2NFの目的は、主キーの一部にのみ依存している非キー列（**部分的関数従属**）を排除することです。非キー列同士の依存（推移的関数従属）は、**3NFで排除**することになります。

**例:**

以下は、まだ2NFの段階にあるテーブルです。

| **生徒ID** | **科目ID** | **科目名** | **成績** |
| --- | --- | --- | --- |
| S001 | C001 | 数学 | 90 |
| S001 | C002 | 英語 | 85 |

* **主キー**: 生徒ID + 科目ID
* **部分的関数従属**: なし（すべての非キー列が複合主キー「生徒ID + 科目ID」に完全に依存）。
* **非キー列同士の依存**:
  + 科目名は科目IDに依存している（非キー列同士の依存）。

このように、2NFの段階では非キー列同士の依存が残っていますが、3NFに進む際に科目テーブルを分割して排除します。

**2. 主キーの「全部」に依存させるの点について、全部とは何ですか？**

**回答:**  
主キーの「全部に依存させる」とは、**主キーのすべての属性に基づいて値が決まること**を指します。もし、主キーの一部だけで非キー列が決まる場合、それは**部分的関数従属**と呼ばれ、2NF違反となります。

**具体例:**

次のテーブルを考えます。

| **生徒ID** | **科目ID** | **生徒名** | **成績** |
| --- | --- | --- | --- |
| S001 | C001 | 山田 | 90 |
| S002 | C002 | 鈴木 | 85 |

* **主キー**: 生徒ID + 科目ID
* **部分的関数従属**:
  + 生徒名は「生徒ID」だけで決まる（科目IDに依存していない）。
  + 一方、成績は「生徒ID + 科目ID」の両方に依存している。

この状態では、主キー「全部」（生徒ID + 科目ID）に依存していないデータ（生徒名）が含まれているため、2NF違反となります。

**解決策:**

* **生徒名**を分割して、生徒IDを主キーとする「生徒テーブル」を作成します。
  + 生徒テーブル: 生徒ID → 生徒名
  + 生徒IDが一意に生徒名を決定する関係を持つため、部分的関数従属が解消されます。

**まとめ**

* **2NFの定義**: 主キーの一部にのみ依存する非キー列（部分的関数従属）を排除する。非キー列同士の依存は残っていてもよい。
* **「主キーの全部に依存する」とは**: 主キーのすべての属性を利用しなければ、非キー列の値が決まらないことを意味する。

これにより、データの正規化が進み、冗長性が減り、更新や削除の異常が防げるようになります。