ER/Studio 6.6.1 の新機能「非正規化マッピング」を活用した物理データモデルの最適化

概要

データモデリングにおいては、データ重複といった無駄を無くすため、業務で扱うデータの意味を明確にするため、正規化が重要になります。しかしながら、物理データモデルではデータベース実装時の処理性能を考慮する必要があるため、非正規化についても検討を行います。正規化と非正規化、双方のメリットを比較しながら必要に応じて非正規化を行います。

ER/Studio6.6.1 では、物理データモデル設計時の非正規化を支援する機能として「非正規化マッピング」が新たに追加されました。この機能は、テーブルを統合して第 2 正規形や第 1 正規形にするなどの非正規化をウィザードにより簡単に実現する機能です。

本稿では、非正規化マッピング機能を活用した非正規化の具体例を紹介しながら、非正規化の手法とその留意点について説明します。

【注記】

本稿では、データモデルの表記法として IDEF1X を使用しています。IDEF1X の詳細につきましては、弊社 J's PORTAL で公開している『ER/Studio チュートリアル』などのドキュメントを参照してください。

J's PORTAL http://jsys-soft.biz/coreport/

※ 『ER/Studio チュートリアル』は、［ダウンロード］→［評価版：Embarcadero Technologies］→［ER/Studio］からダウンロードできます。

※ エンティティと属性の説明は「Lesson2 エンティティと属性の作成および編集」、リレーションシップの説明は「Lesson3 リレーションシップの作成および編集」に記載されています。

1．正規化の重要性

データモデリングにおける重要な概念としてOne Fact in One Placeという原則があります。一つの事実は一箇所にだけ存在するようにして、データ重複といった無駄を無くすことを目指すものです。そして、この原則を守るための実践的な手法が正規化です。正規化されていないエンティティでは図 1 に示す 3 つの不整合（更新不整合・挿入不整合・削除不整合）が生じてしまい、

◇ 重複更新が必要になる（更新不整合）

◇ 適切なタイミングで適切なデータ管理ができなくなる（挿入・削除不整合）

などの問題が発生します。

注文番号	製品番号	製品名	製品単価	注文数量
1001	100	ER/Studio	500,000	5
1001	200	Rapid SQL	135,000	5
1001	300	DBAirtisan	240,000	1
1002	100	ER/Studio	500,000	10

更新不整合	「製品番号100はER/Studio、500,000である」との関係が重複して登録されるため、製品名または製品単価を変更する際に重複更新が必要となる
挿入不整合	注文を登録する前に「製品番号100はER/Studio、 500,000である」との関係を登録しておくことができない
削除不整合	注文を削除してしまうと「製品番号100はER/Studio、 500,000である」との関係も喪失してしまう

図 1：正規化されていないエンティティにおける不整合

正規化では、第 1 正規化、第 2 正規化、第 3 正規化と段階的にエンティティを分解し、不整合の問題を排除していきます。図 2 は正規化に伴いエンティティが分解されていく様子を表しています。一見するとエンティティの数が増えて複雑になっていくようにも見えますが、注文を管理するためには注文データ、注文明細データ、顧客データ、製品データを管理する必要があることが鳥瞰図的に読み取れるようになります。また、エンティティ間の関連（エンティティ・リレーションシップ）、「1 対多」などのエンティティ間の対応（カーディナリティ）により、様々なデータ管理ルールが読み取れるようになります。データモデリングの主旨である「業務で扱うデータの意味を明確にする」といった点においても正規化が重要です。

非正規形

注文番号	注文日付	顧客番号	顧客名	顧客住所	製品番号	製品名	製品単価	注文数量
1001	2004/3/1	001	鈴木	東京都	100 200 300	ER/Studio Rapid SQL DBAirtisan	500,000 135,000 240,000	5 5 1
1002	2004/3/1	002	佐藤	神奈川県	100	ER/Studio	500,000	10
1003	2004/3/2	003	高橋	千葉県	200 300	Rapid SQL DBAirtisan	135,000 240,000	3 3
1004	2004/3/3	001	鈴木	東京都	300	DBAirtisan	240,000	4

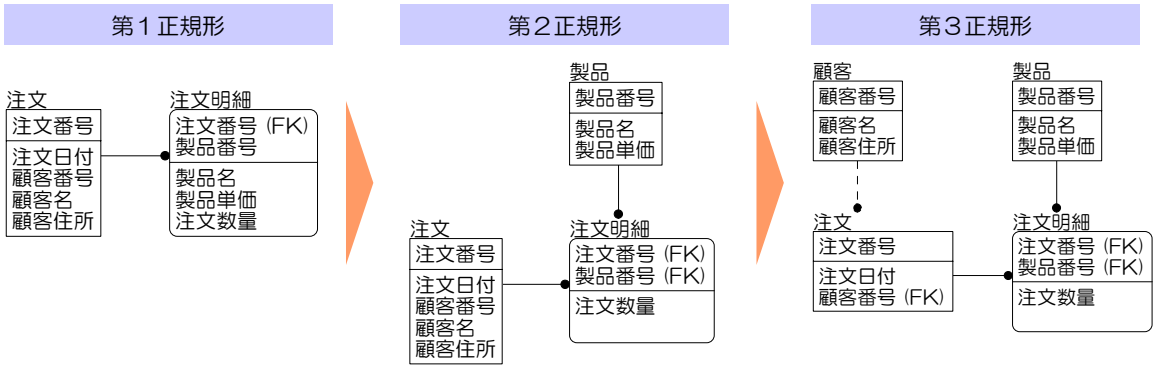

図２：正規化の流れ

2．物理データモデルの最適化

論理データモデルではデータベースでの実装を考慮せず、無駄なデータを無くすこと、データの意味を明確にすることを目指します。正規化を行いながら不整合の問題を取り除き、エンティティを分解していきます。一方、物理データモデルではデータベースでの実装のためにモデリングを行います。実装設計の一要素であるパフォーマンス設計を行いながら、必要に応じて性能のためにモデルを見直していきます。

正規化されて多数に分解されたエンティティをそのままテーブルに対応させていくと、テーブルの結合処理が頻繁に発生するなど、処理性能上の問題が生じる可能性があります。そこで、正規化により分解されたテーブルを一つのテーブルに統合して結合処理を不要にするなどの非正規化を検討します。

ただし、非正規化してしまうと正規化で回避していた不整合が再び問題になります。以下の点に留意し、非正規化を必要最小限に抑えるようにします。

◇ 重複更新漏れが発生するとデータ管理の必須条件である整合性を失うため、論理データモデルでは必ず正規化を行い、更新不整合が起こる箇所を把握しながら非正規化を行うとの手順を守ること

◇ 非正規化ではデータ参照性能は向上するが重複更新により更新性能が低下するとの傾向があるため、参照性能と更新性能のトレードオフを考慮し、非正規化すべきかを決定すること

◇ 非正規化による挿入不整合、削除不整合が業務ルール（データ管理ルール）の視点で問題がないことを確認し、非正規化すべきかを決定すること

3．非正規化マッピング機能

（1）非正規化マッピングの概要

ER/Studio6.6.1 では、物理データモデル設計時の非正規化を支援する機能として「非正規化マッピング」が新たに追加されました。この機能は、テーブルを統合して第 2 正規形や第 1 正規形にするなどの非正規化をウィザードにより簡単に実現する機能です（図 3）。親テーブルに統合、テーブルの横分割、カラムのコピーなど、表 1 に示す 6 つの非正規化手法に対応しています。

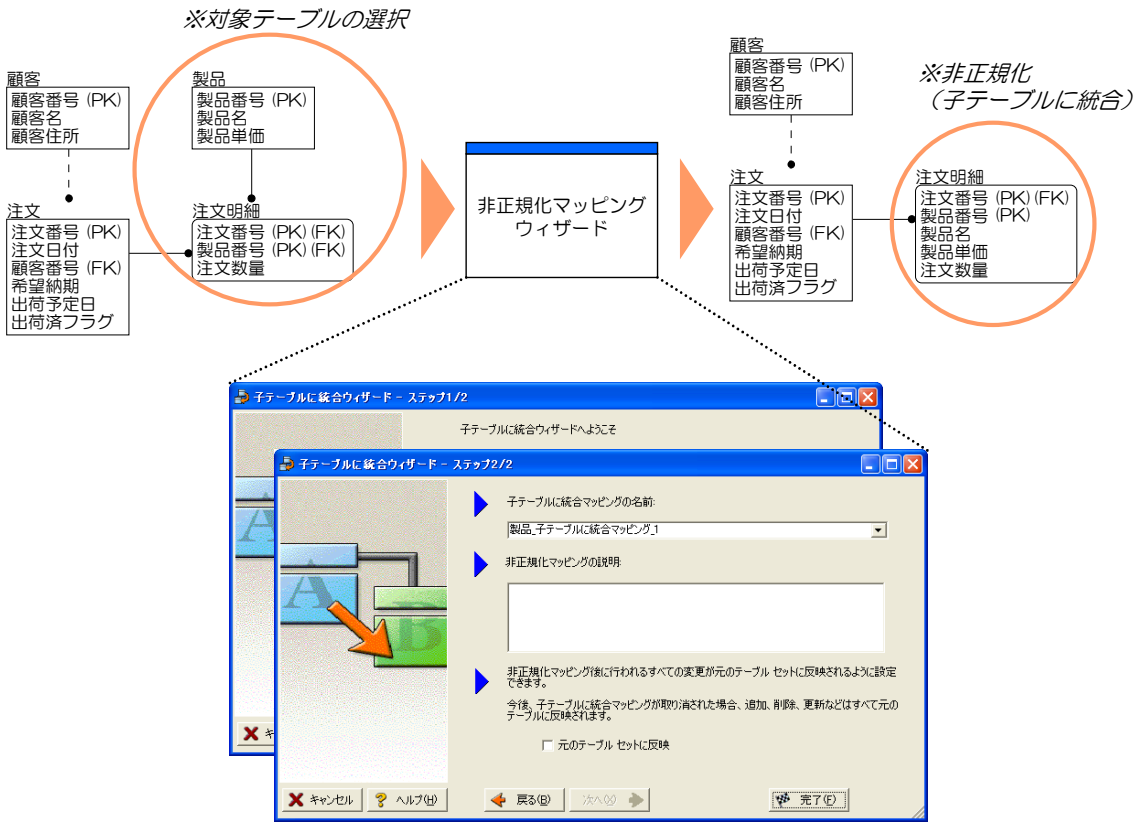

図 3：非正規化マッピング機能による非正規化手順

表 1：非正規化マッピング機能と主な活用例

手法	概要	主な活用例
親テーブルに統合	子テーブルの属性を親テーブルの繰り返しカラムとして取り込み、親テーブルに統合する	繰り返しグループの横持ち スーパータイプへのロールアップ
子テーブルに統合	親テーブルの属性を子テーブルのカラムとして取り込み、子テーブルに統合する	第１正規形への非正規化 サブタイプへのロールダウン
横分割	1 つのテーブルから同じカラムセットを持つ独立した複数のテーブルを作成し、行を分割して管理できるようにする	テーブルの横分割
縦分割	1 つのテーブルを複数のテーブルに分割して元テーブルのカラムを分配し、カラム数の少ないサブセットにする	テーブルの縦分割
カラムのコピー	あるテーブルのカラムを他のテーブルのカラムとしてコピーする	第２正規形への非正規化 第１正規形への非正規化
テーブルマージ	同一の主キーを持つ関連性のない２つ以上のテーブルを１つのテーブルにマージする	

（2）非正規化マッピングの実行手順

手順①：非正規化するテーブルまたはカラムを選択

モデルエクスプローラ上でテーブルまたはカラムを選択します。（テーブルとカラムの何れを選択するのか、テーブルを選択する場合は単一テーブルと複数テーブルの何れを選択するのかは、表 2 に示すように非正規化マッピングの手法によって異なります。）

表 2：非正規化対象の選択

手法	非正規化対象の選択
親テーブルに統合	統合する複数テーブル
子テーブルに統合	統合する複数テーブル
横分割	分割する単一テーブル
縦分割	分割する単一テーブル
カラムのコピー	コピーする単一カラム
テーブルマージ	マージする複数テーブル

手順②：非正規化マッピング ウィザードを起動

テーブルまたはカラムを選択後、右クリックで表示されるショートカットメニューから非正規化マッピングウィザードを起動します（図 4、図 5）。

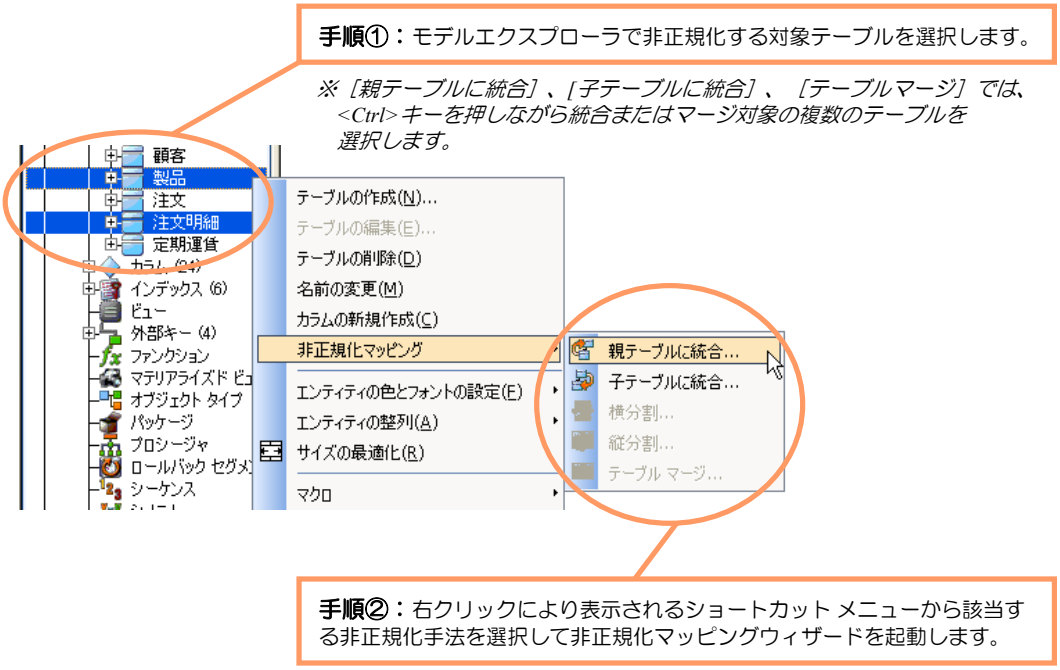

図 4：非正規化マッピングウィザードの起動方法（テーブルを選択する場合）

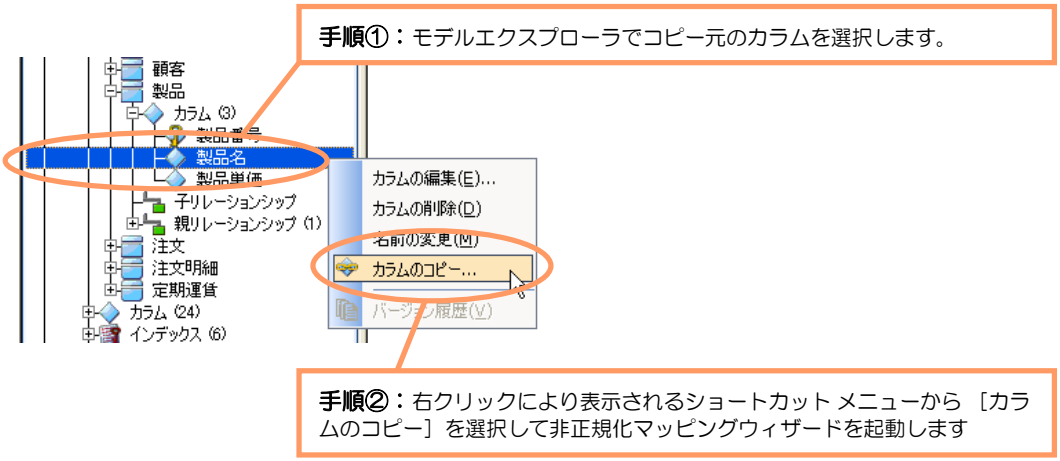

図 5：非正規化マッピングウィザードの起動方法（カラムを選択する場合）

手順③：表示されたウィザード上に表示される指示に従い操作

ウィザードでの操作は非正規化マッピングの手法によって異なります。ウィザード上に表示される説明に従い操作します。

【参考】非正規化マッピングウィザードを起動する他の操作方法

非正規化するテーブルまたはカラムを［ダイアグラム］ウィンドウで選択すること、非正規化マッピングウィザードをメニューバーまたは［モデリング］ツールバーから起動することもできます。

※<Shift>キーを押しながらコピー元カラムを選択します。

（3）非正規化マッピングの確認

非正規化するとデータの重複登録が可能になり、データを更新するアプリケーションにおいて重複更新の仕組みが必要になります。重複更新漏れが発生するとデータの不整合が生じてしまうため、アプリケーション開発時に非正規化された箇所を全て把握しておくことが重要です。

また、非正規化すると論理データモデルと物理データモデルが 1 対 1 の単純な対応ではなくなります。アプリケーション開発後の情報資源管理においても、非正規化されたテーブルやカラムが把握できること、それらのテーブルやカラムが論理データモデルのどのエンティティや属性と対応するかを把握できることが重要です。

非正規化マッピング機能では、これらの要求に応えるため、正規化状態と非正規化状態とのマッピングを保持して確認できるようにしています。ER/Studio6.6.1 で新たに追加されたエンティティエディタとテーブルエディタの［論物関連］タブ（図 6）、またはモデルエクスプローラの［非正規マッピング］セクション（図 7）において、保持されているマッピングを確認することができます。

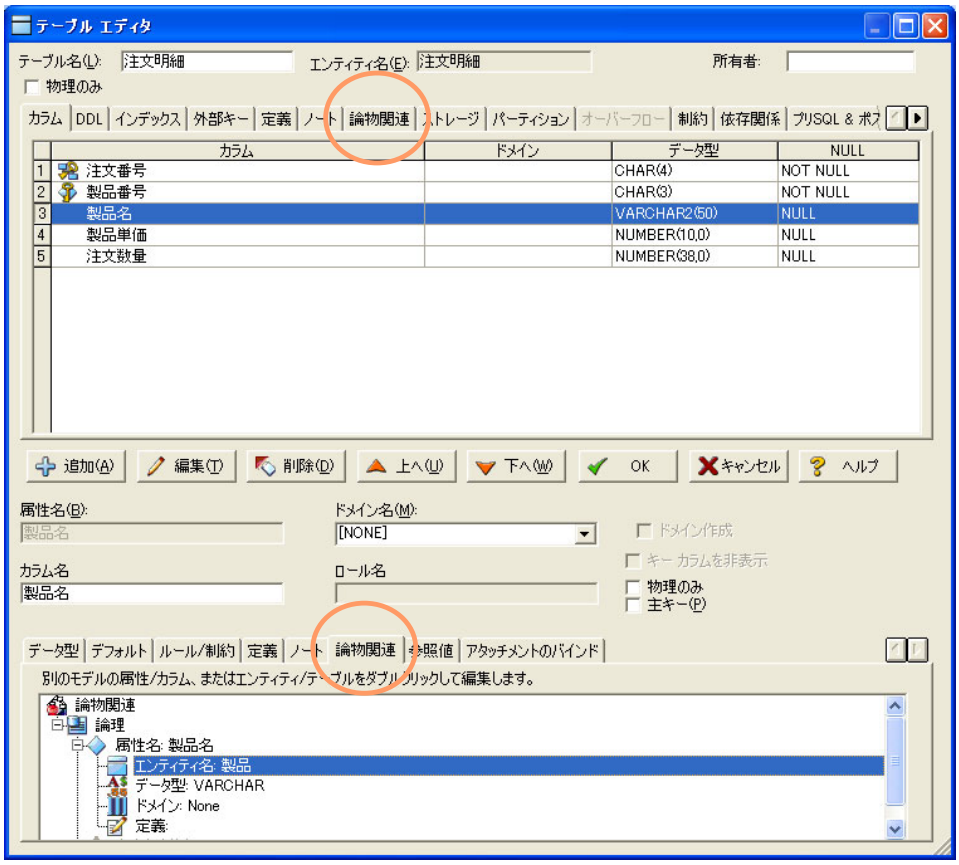

図 6：テーブルエディタの［論物関連］タブ

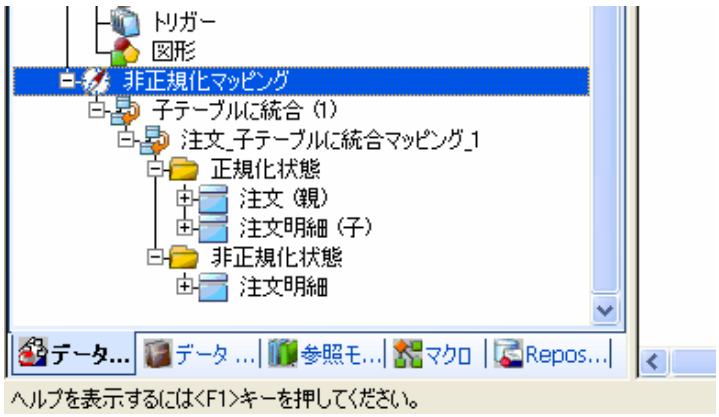

図 7：モデルエクスプローラの［非正規マッピング］セクション

（4）非正規化マッピングの取り消し

非正規化マッピング機能では、非正規化マッピングを取り消して非正規化前の元の状態に戻すこともできます。非正規化マッピングを取り消す場合は、モデルエクスプローラの［非正規マッピング］セクションにおいて、該当する非正規化マッピングを選択し、右クリックにより表示されるショートカット メニューから「非正規化マッピングの取り消し」を実行します（図 8）。

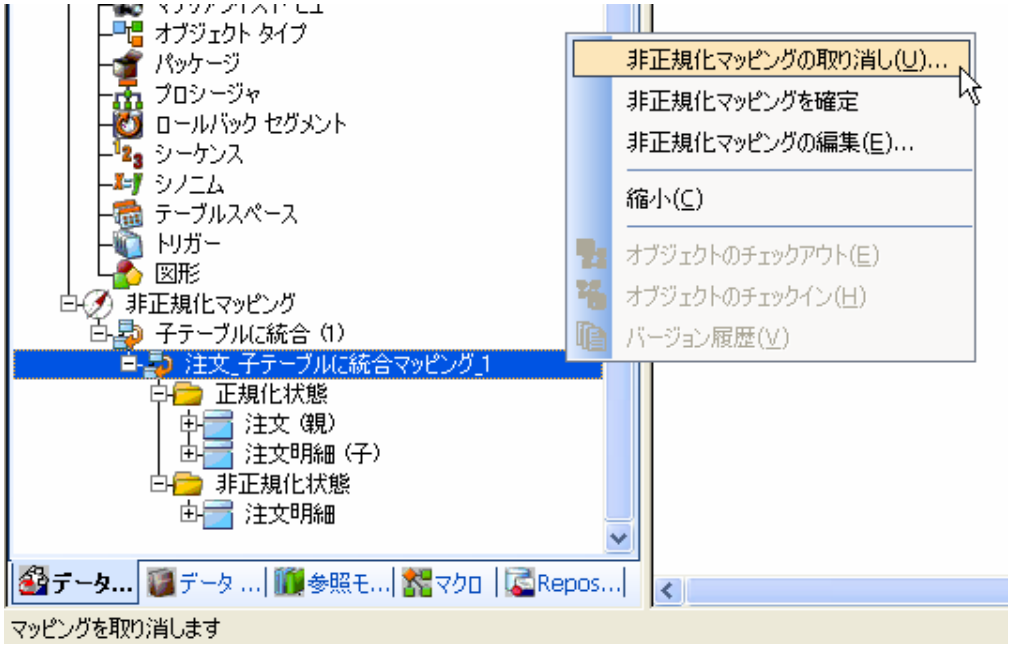

図 8：非正規化マッピングの取り消し

4．非正規化の具体例①：テーブル統合とカラムコピー

テーブルの結合処理が性能面で問題となる場合、結合を不要にするために非正規化を検討します。結合を不要にする非正規化手法としてはテーブル統合とカラムコピーの 2 つがあります。非正規化により生じる不整合（表 3）を考慮し、適切な手法を選択します。

表 3：テーブル統合/カラムコピーにより生じる不整合

手法	更新不整合		挿入不整合/削除不整合	
テーブル統合	△	統合したテーブル内で更新不整合が生じる	×	一つに統合するため、挿入/削除不整合が生じる
カラムコピー	×	コピー先のテーブル内、コピー元とコピー先のテーブル間で更新不整合が生じる	○	コピー元テーブルを残すため、挿入/削除不整合が生じない

（1）第２正規形への非正規化

非依存型リレーションシップで結ばれている親子のテーブルを子テーブル側に統合すると第 2 正規形になります。ただし、テーブル間の依存度が低く、両テーブルが異なるタイミングで挿入、削除されるケースが多いので、通常はテーブル統合ではなくカラムコピーにより非正規化を行います。

図 9 は［注文］テーブルと［顧客］テーブルの結合処理を不要にするため、第 2 正規形に非正規化する例です。［注文］テーブルを［顧客］テーブルにテーブル統合してしまうと、顧客データを注文テーブルで管理することになり、注文データを登録する前にあらかじめ顧客データを登録しておくことができなくなります。そこで、［顧客］テーブルの”顧客名”カラムと”顧客住所”カラムを非正規化マッピング機能の「カラムのコピー」を使用して［注文］テーブルにコピーします。［顧客］テーブルが残るので挿入不整合と削除不整合は問題になりませんが、以下の更新不整合に対応する重複更新の仕組みを作り込む必要があります。

◇　［顧客］テーブルと［注文］テーブルとの間で顧客番号、顧客名、顧客住所が重複して登録される
◇　［注文］テーブル内で顧客番号、顧客名、顧客住所が重複して登録される

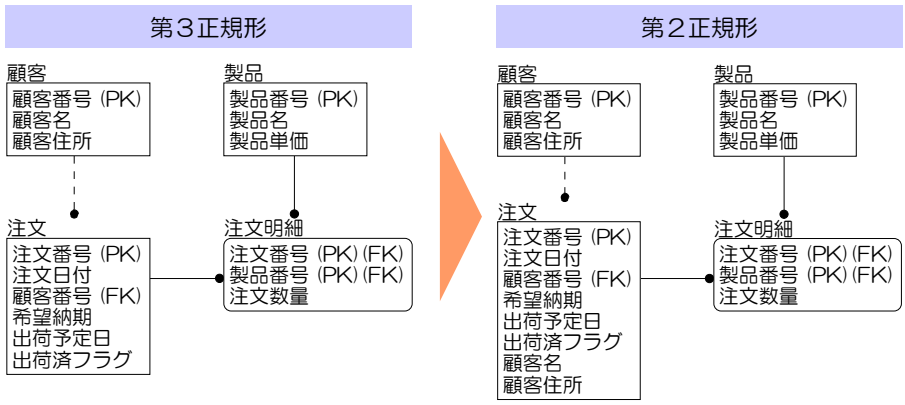

図 9：第 2 正規形への非正規化の例

（2）第１正規形への非正規化

依存型リレーションシップで結ばれている親子のテーブルを子テーブル側に統合すると第 1 正規形になります。両テーブルが同じタイミングで挿入、削除され、挿入不整合と削除不整合が問題にならない場合、テーブル統合により非正規化を行います。異なるタイミングで挿入、削除され、挿入不整合と削除不整合が問題になる場合、カラムコピーにより非正規化を行います。

図 10 は、［注文］テーブルと［注文明細］テーブルの結合処理を不要にするため、第 1 正規形に非正規化する例です。［注文］テーブルと［注文明細］テーブルは同じタイミングで挿入、削除されるため、［注文］テーブルを非正規化マッピング機能の「子テーブルに統合」を使用して［注文明細］テーブルに統合しています。以下の更新不整合に対応する重複更新の仕組みを作り込む必要があります。

◇　［注文明細］テーブル内で注文番号、注文日付、顧客番号が重複して登録される

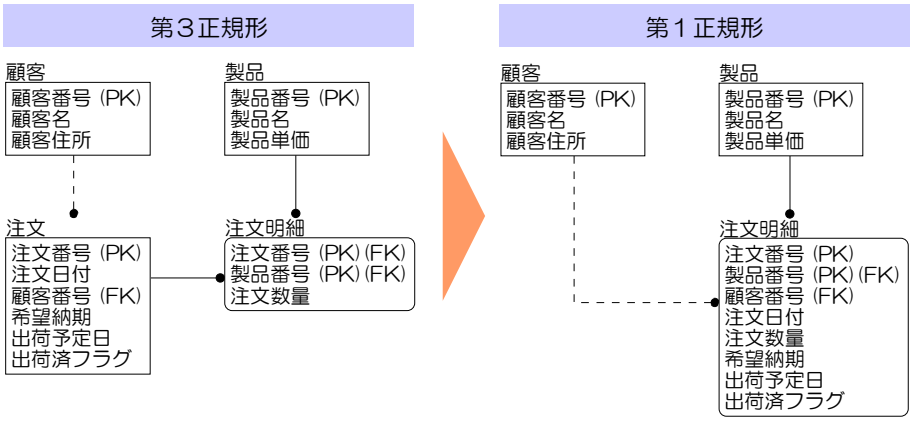

図 10：第 1 正規形への非正規化の例（子テーブルに統合）

図 11 は、[注文明細] テーブルと [製品] テーブルの結合処理を不要にするため、第 1 正規形に非正規化する例です。前述の「第 2 正規化への非正規化」の場合と同様、挿入不整合と削除不整合が問題になるため、非正規化マッピング機能の「カラムのコピー」を使用し、[製品] テーブルの”製品名”カラムと”製品単価”カラムを [注文明細] テーブルにコピーしています。

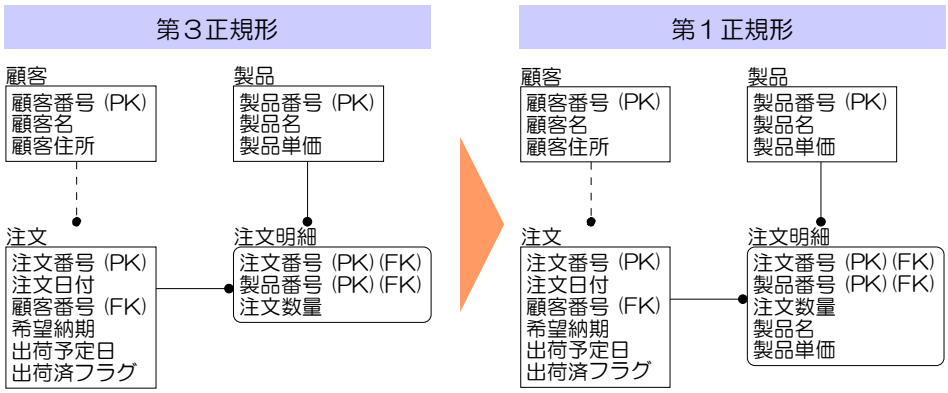

図 11：第 1 正規形への非正規化の例（カラムのコピー）

（3）繰り返しグループの横持ち

依存型リレーションシップで結ばれている親子のテーブルを親テーブル側に統合すると、繰り返しの横持ちと呼ばれる構造になります。この構造では、子テーブルの属性を親テーブル内に繰り返しカラムグループとして持つことにより、親子のテーブルを 1 つに統合します。そのため、以下の点に留意する必要があります。

- ◇ 繰り返しカラムグループ数の上限が規定できること
- ◇ 繰り返しカラムグループに対する SQL 文は複雑になり易いため、データ操作や検索に問題がないかを確認すること

図 12 は、[区間] テーブルと [定期運賃] テーブルの結合処理を不要とするため、[区間] テーブルに統合する例です。非正規化マッピング機能の「親テーブルに統合」により [定期運賃] テーブルを [区間] テーブルに統合しています。[定期運賃] テーブルの”定期運賃”カラムが図 13 のウィザードで指定する繰り返しカラムグループ数だけ繰り返され、[区間] テーブルに統合されています。依存型リレーションシップで結ばれているテーブルの結合処理を不要にする手法として、前述の「第 1 正規形への非正規化」もありますが、繰り返しグループの横持ちでは更新不整合、挿入不整合、削除不整合が起こらないため、上記の留意点に問題がなければこの手法を選択します。

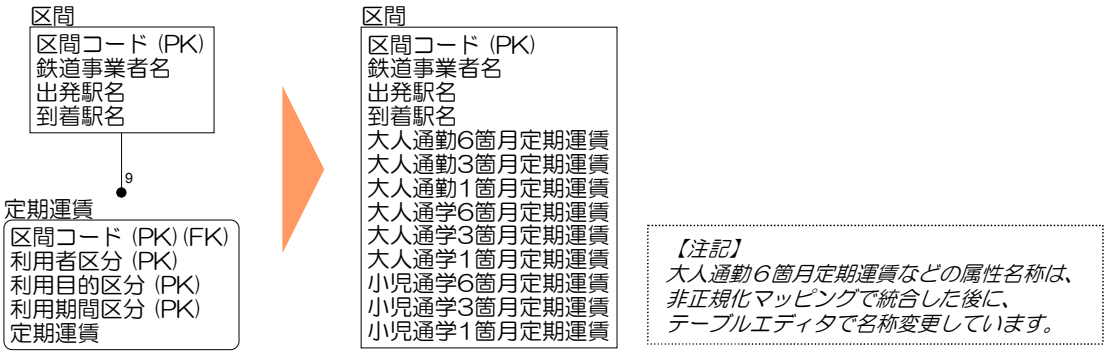

図 12：繰り返しグループの横持ちの例

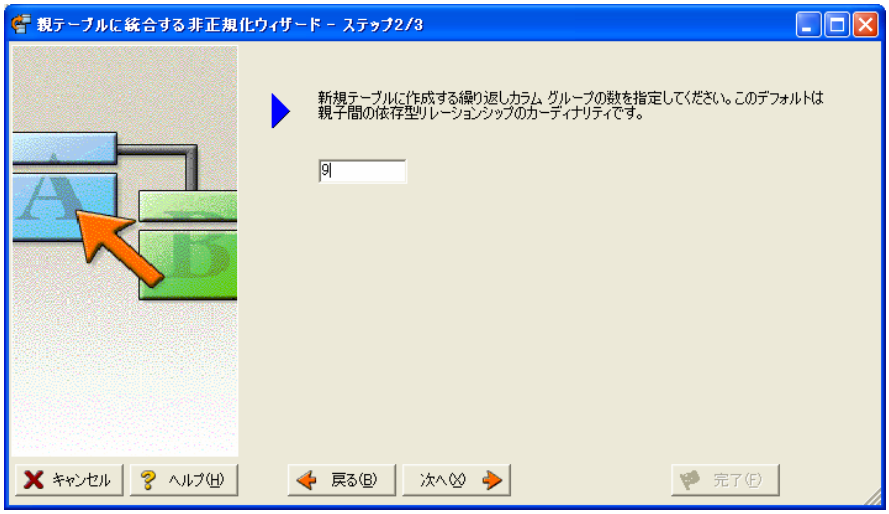

図 13：繰り返しカラムグループ数を指定するウィザード

5．非正規化の具体例②：テーブルの分割

大量のデータを格納する大規模なテーブルでは、検索性能に問題が生じることがあります。特に、以下に挙げる状況により性能が問題となっている場合には、テーブルを分割して性能を向上させることを検討します。

- ◇ 文字列の部分一致検索などにより全件検索が頻繁に行われている
- ◇ 上位検索など、大規模なソートが必要となる処理が頻繁に行われている
- ◇ ロックの競合が多発している

テーブルの分割には、図 14 に示すようにテーブルの横方向と縦方向に分割する手法があります。どちらの手法においても更新不整合、挿入不整合、削除不整合は生じませんが、以下に挙げるデメリットがあるため、これらのデメリットを考慮して分割すべきか、分割する場合は横方向と縦方向の何れの方向に分割すべきかを決定します。

- ◇ 分割したテーブルのうち、どのテーブルにアクセスすべきかを判定するロジックが必要になる
- ◇ 分割したテーブル間の結合が必要な場合は検索性能が低下する

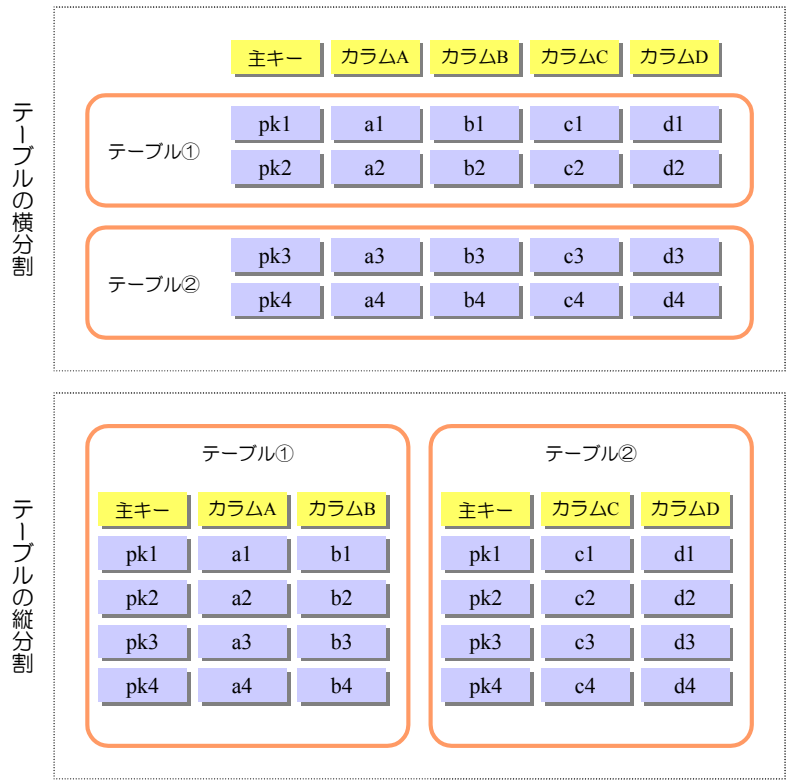

図 14：テーブルの横分割と縦分割

（1）テーブルの横分割

図 15 は、非正規化マッピング機能の「横分割」を使用してテーブルを分割する例です。営業所別に注文データが管理されることを前提として、[注文明細] テーブルを営業所別に 2 つのテーブルに分割しています。分割されたテーブル間においてデータの独立性があるため、分割のデメリットが問題とならず、性能の向上が期待できます。なお、図 15 では 2 つのテーブルに分割していますが、非正規化マッピング機能の「横分割」では図 16 のウィザードにより分割数を指定することで任意の数にテーブルを分割できます。

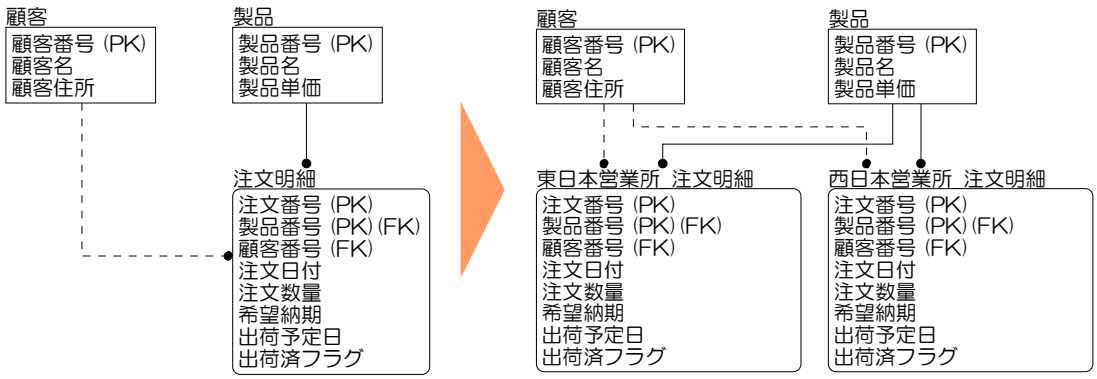

図 15：テーブルの横分割の例

図 16：分割数を指定するウィザード

（2）テーブルの縦分割

図 17 は、非正規化マッピング機能の「縦分割」を使用して［注文］テーブルを［注文］テーブルと［出荷］テーブルに分割する例です。図 18 のウィザードにおいて、注文時に登録されるカラムを［注文］テーブル、出荷を管理するために使用するカラムを［出荷］テーブルに振り分けています。非正規化マッピング機能の「縦分割」においても「横分割」と同様に任意の数にテーブルを分割することができます。

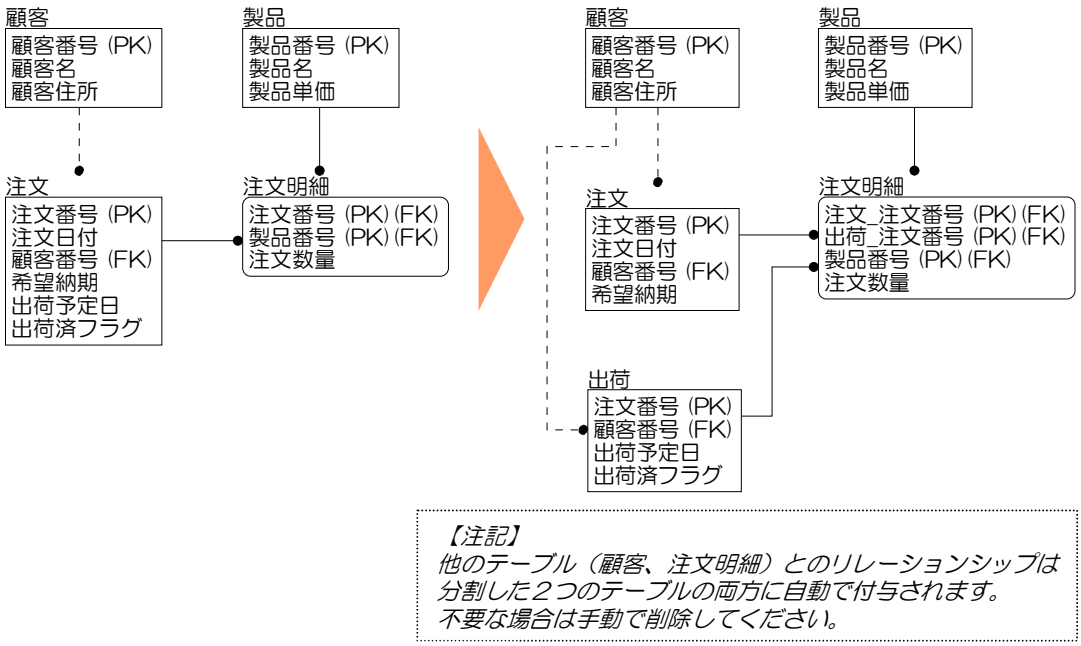

図 17：テーブルの縦分割の例

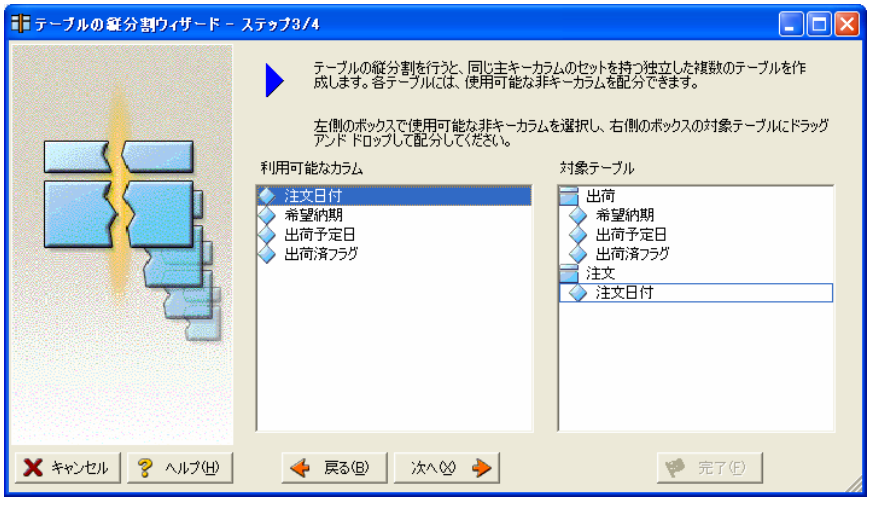

図 18：非キーカラムを振り分けるウィザード

6．非正規化の具体例③：スーパータイプ/サブタイプの論物変換

論理データモデルのスーパータイプとサブタイプは、表 4 に示す手法のうちの何れかを用いてテーブルに変換します。各手法のメリットとデメリットを比較しながら最適な手法を選択します。

ER/Studio では「物理モデルの作成」によりスーパータイプとサブタイプが個別のテーブルに自動変換されます（図 19）。他の手法を選択する場合は非正規化マッピング機能を使用し、ロールアップする場合は「親テーブルに統合」、ロールダウンする場合は「子テーブルに統合」を実行します（図 20）。

表 4：スーパータイプ/サブタイプの論理/物理変換手法のメリット/デメリット

変換手法	メリット	デメリット
個別のテーブルに変換	格納効率が良い	テーブルの結合が必要
スーパータイプへのロールアップ	テーブルの結合が不要	サブタイプの属性が多い場合に格納効率が悪くなる
サブタイプへのロールダウン	1 つのサブタイプ内での検索ではテーブルの結合が不要	スーパータイプの属性が多い場合に格納効率が悪くなる

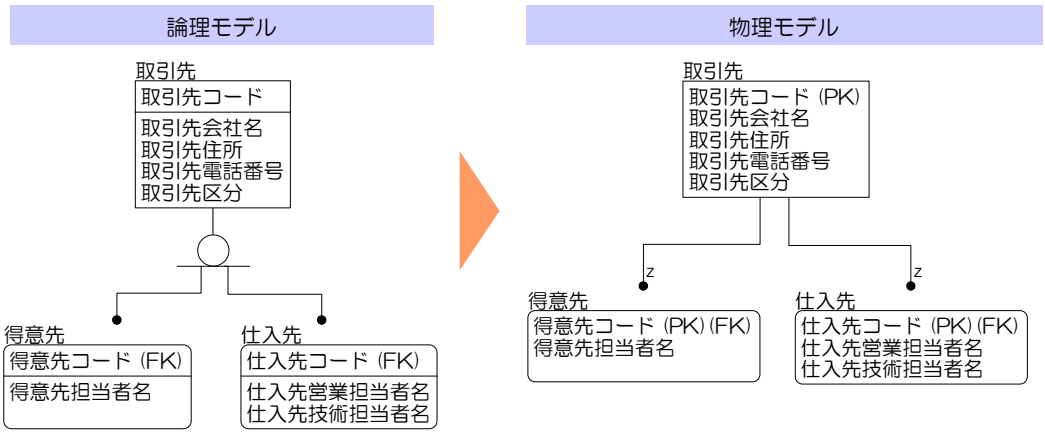

図 19：スーパータイプ/サブタイプの論物変換（個別のテーブルに変換）

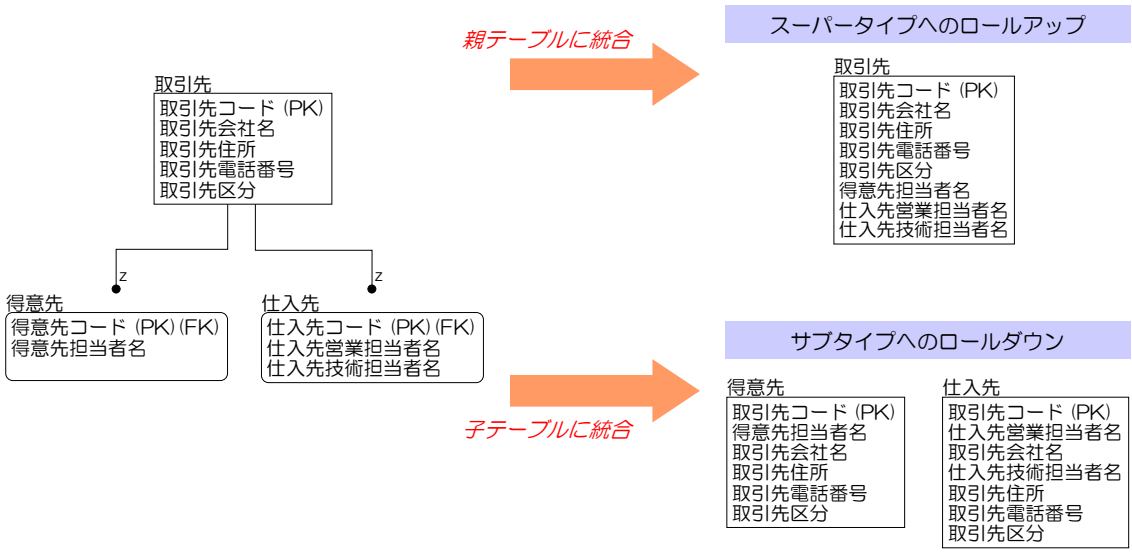

図 20：スーパータイプ/サブタイプの論物変換（ロールアップとロールダウン）

7．導出項目と継承項目の扱い

他の項目から算出される導出項目、他の項目のコピーである継承項目は、正規化との観点からは不要な項目ですが、論理データモデルまたは物理データモデルを設計する際に、以下の 3 つの視点で非正規化すべきかを検討しておく必要があります。

① データの意味を明確にするといったモデルの役割において重要な項目

図 21 の［顧客別月次注文実績］テーブル、［受注］テーブルの”受注金額”カラムは、実際の業務で取り扱われる重要な導出項目であるため、モデル上に図示することを検討します。この視点での非正規化はデータの意味を明確にすることが目的であり、論理データモデルの設計時に行います。

② データの更新タイミングが異なる項目

図 21 の［注文明細］テーブルの”注文時_製品単価”カラムは、［製品］テーブルの”製品単価”カラムを継承する項目ですが、あくまでも注文時の単価であり、注文を受けた後に製品単価が変更されても変わるものではありません。非正規化して項目を追加するか、履歴管理のための仕組みを追加（例えば、［製品履歴］テーブルを追加）するかの二者択一になります。この視点での非正規化はデータ管理ルールを明確にするものであり、論理データモデルの設計時に行います。

③ 性能を向上させるために追加する項目

図 21 の［注文］テーブルの”注文月”カラムは注文日付から算出できる導出項目ですが、追加することで顧客別月次注文実績の集計における処理性能の向上が期待できます。この視点での非正規化はデータベース実装を考慮するものであり、物理データモデル設計時に行います。

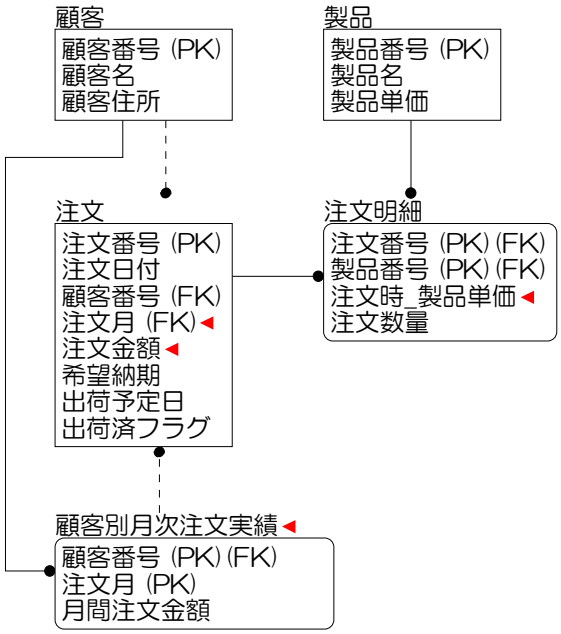

図 21：導出項目と継承項目の例

8．代理キーの使用

（1）代理キーの使用目的

主キーが多数のカラムで構成される複合キーである場合、単一カラムのシンプルな代理キー（Surrogate Key）に主キーを置き換えます。代理キーで置き換えられる主キーには一意性を確保するための代替キー（Alternate Key）を付与し、同じデータが登録されることを防ぎます。代理キーでの置き換えは不整合の問題も起こらず非正規化ではないのですが、以下に示す効果が期待できるため、データベースでの実装を考慮してモデルを見直す手法の一つとして、物理データモデルの設計では重要と言われています。

◇ 結合操作をするSQL文の簡易化が図れ、操作性の向上と実装ミスの低減を図れる
◇ 代理キーで置き換えられた元の主キーの値が容易に変更できるようになる

論理データモデルにおいて代理キーを使用するケースも見受けられますが、以下の理由により、データの意味を明確にするための論理データモデルでは使用せず、物理データモデルで使用することを推奨します。

◇ 代理キーを使用すると、データの粒度（データの管理単位）を表すといった主キー本来の役割を失ってしまう
◇ 代理キーは実装における容易性を確保するものであり、業務上必要な属性ではない

図 22 は、代理キーを使用して主キーを置き換える例です。［製品］テーブルの主キーを代理キーである”製品 ID”カラムに置き換え、元の主キーである”製品番号”カラム、”製品バージョン”カラム、”製品エディション”カラムを代替キーとしています。代理キーを使用することで、［製品］テーブルと［注文明細］テーブルとの結合操作において結合条件の記述が簡易になり、また、［注文明細］テーブルのデータを削除、再登録せずに”製品バージョン”カラムなどの更新ができるようになります。

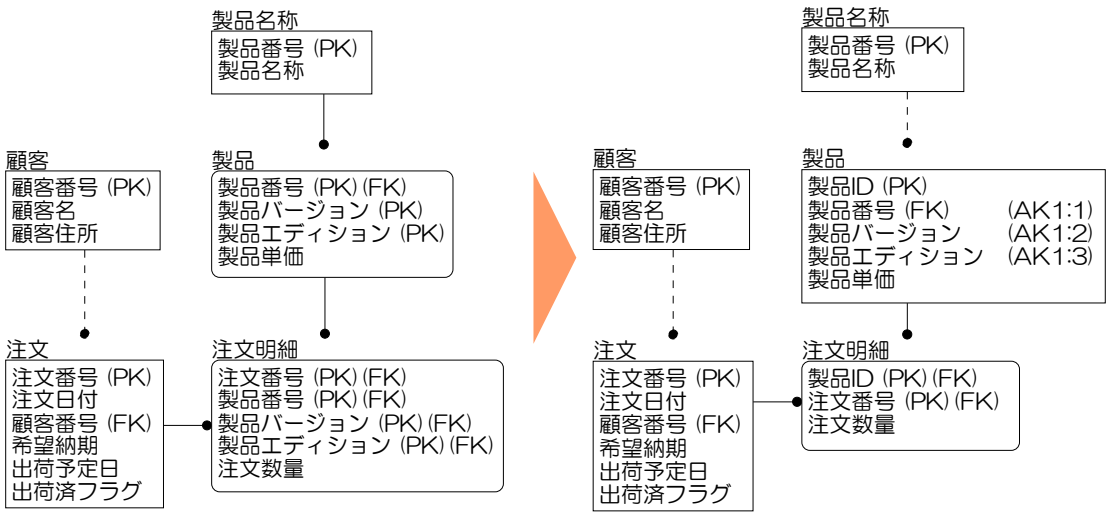

図 22：代理キーの使用例

（2）代理キーの生成手順

手順①：依存型リレーションシップで関連している親テーブルがある場合には非依存型リレーションシップに変更する

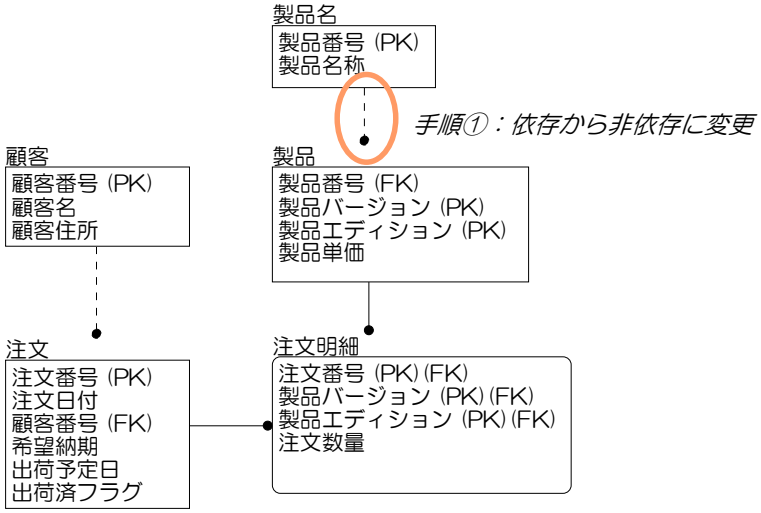

図 23：代理キーの生成手順（手順①）

手順②：元の主キーを代替キーに変更する

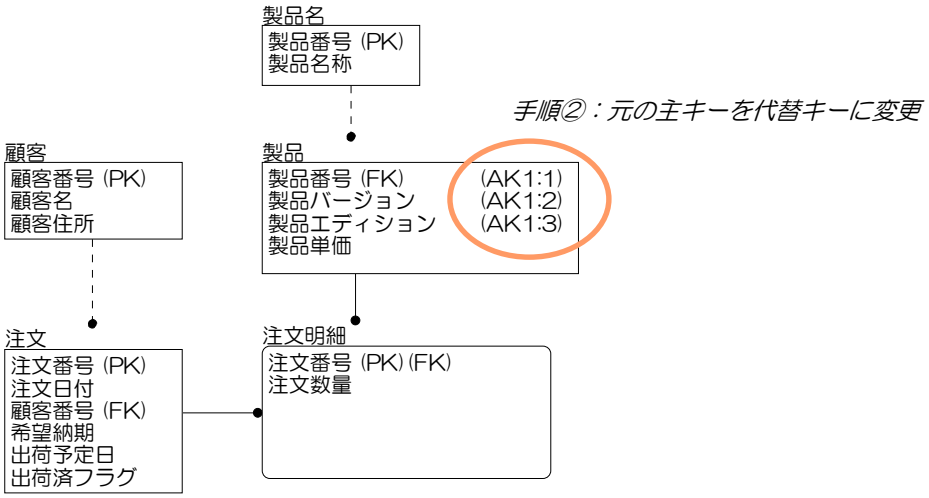

図 24：代理キーの生成手順（手順②）

手順③：代理キーを追加し、新たな主キーとして設定する

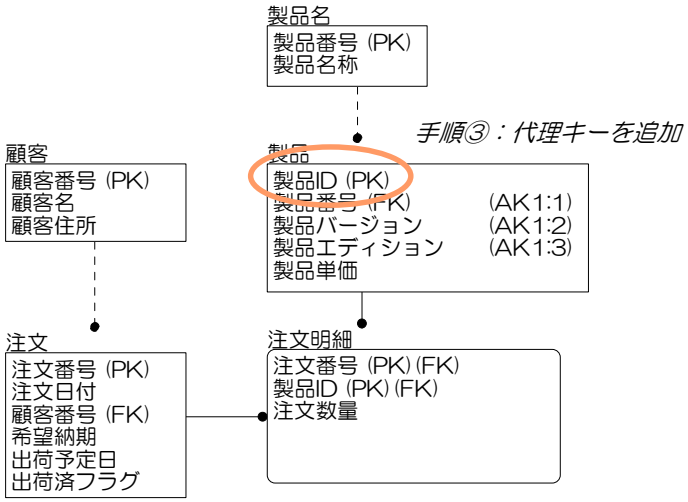

図 25：代理キーの生成手順（手順③）

（3）代理キーの使用における注意事項

注意事項①：（代替キーに変更した）元の主キーを参照していた外部キーは削除される

ER/Studio では、リレーションシップに連動して外部キーが自動で追加されたり、削除されたりします。そのため、前述の手順②において、［注文明細］テーブルの外部キーである”製品番号”カラム、”製品バージョン”カラム、”製品エディション”カラムが自動で削除され、手順③において［注文明細］テーブルに新たな外部キー（”製品 ID”カラム）が自動で追加されます。［注文明細］テーブルから削除された”製品番号”カラム、”製品バージョン”カラム、”製品エディション”カラムは、［製品］テーブルで管理されるカラムであり、［注文明細］テーブルに残すと冗長なデータになるので正しい動きなのですが、あえて残すべきだと判断する場合は、非正規化マッピングの「カラムのコピー」により継承項目として追加します。

注意事項②：代理キーは親テーブル側から生成すること

図 26 は、［注文明細］テーブルに代理キーを生成してから、［製品］テーブルに代理キーを生成した例です。一方、図 27 は、［製品］テーブルに代理キーを生成してから、［注文明細］テーブルに代理キーを生成した例です。［注文明細］テーブルでは、”注文番号”カラムと”製品 ID”カラムに対して代替キーが付与されている状態が正しいので、親テーブル側から代理キーを生成した図 27 が正しい結果となっています。

子テーブル側から代理キーを生成した図 26 では、［製品］テーブルに代理キーを生成した際、［注文明細］テーブルの外部キーが削除されてから新たに追加されるため、［注文明細］テーブルの”製品 ID”カラムが代替キー（AK1）ではなくなっています。親子の関係にある複数のテーブルに代理キーを生成する場合は、親テーブル側から生成します。

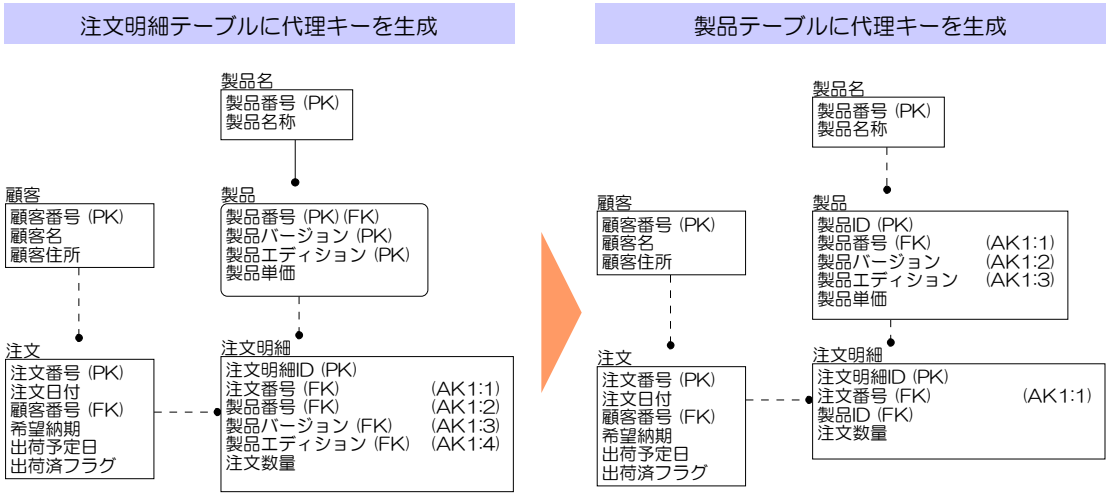

図 26：代理キーを子テーブル側から生成した例

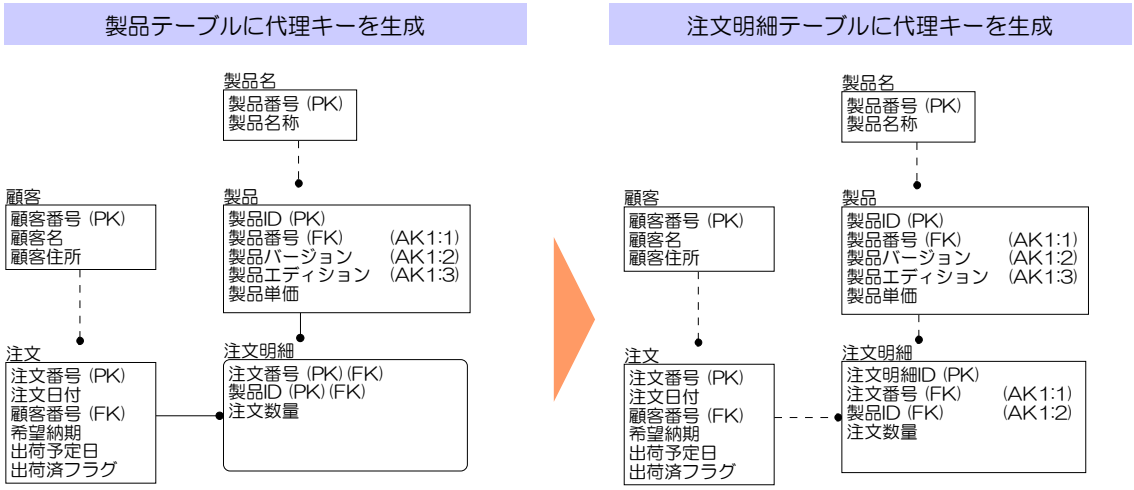

図 27：代理キーを親テーブル側から生成した例

注意事項③：インデックスを追加する前に代理キーを生成すること

図 28 は、［製品］テーブルと［注文明細］テーブルを結合した検索の処理性能を考慮し、［製品］テーブルを参照している外部キー（”製品番号”カラム、”製品バージョン”カラム、”製品エディション”カラム）にインデックスを追加したケースを表しています。［製品］テーブルにおいて代理キーを生成した際、［注文明細］テーブルの外部キーが削除され、同時に外部キーのインデックス（IE1）も削除されています。図 28 のように代理キーの生成に関連する外部キーにインデックスを追加する場合は、まずは代理キーを生成してからインデックスを追加します。

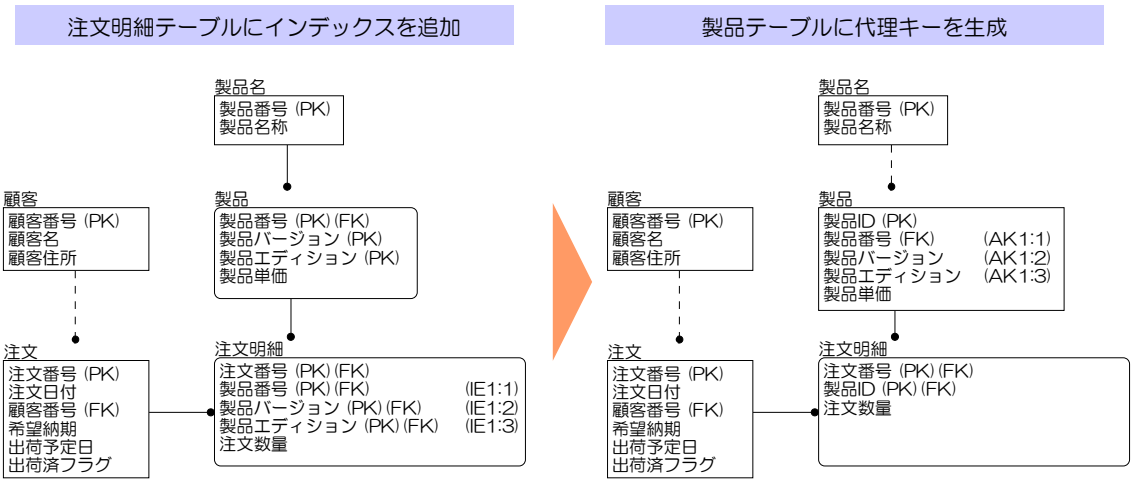

図 28：インデックスを追加してから代理キーを生成した例

【参考】代理キー生成マクロ

ER/Studio は、強力なオートメーション・インターフェース機能を持っています。この機能は、Sax Basic（VBA 文法に準拠した Basic 言語）を使用して ER/Studio のメタデータを操作するための機能で、定型的な作業を自動化することができます。

弊社の Web サイト（下記 URL）において、オートメーション・インターフェース機能を活用した「代理キー生成マクロ」を公開しています。このマクロを使用すると、「（２）代理キーの生成手順」で示した操作がダイアログ（下図）での指定に従い自動化され、簡易に代理キーを生成することができます。

※　このマクロを使用した場合でも、「（３）代理キーの使用における注意事項」に示した内容に注意してください。

ER/Studio カスタム マクロ ライブラリ

http://www.jsys-products.com/product/erstudio/macro.html