

## 赤レンガ倉庫6丁目2番地



### 第5回 消費者信用の分析ソリューション・テンプレート（金融サービス編）

#### はじめに

第5回目の今回は、ビジネスソリューション・テンプレートをご紹介します。3回目として、金融分野のシステムを構築する上でご利用いただける、ソリューション・テンプレートをご紹介します。

今回ご紹介するテンプレートは、消費者の信用度について分析を行うシステム構築を想定した、テンプレートとなります。それでは、始めに金融サービス - 消費者信用の分析ソリューション・テンプレートを構成する各コンポーネントから、ご紹介していきたいと思えます。

#### 金融サービス - 消費者信用ソリューション・テンプレートの基本コンポーネント

##### ビジネス上の質問のセット

どのような金融サービス用の分析システムを構築したいかを、質問の形で表現しています。ここで紹介しているビジネス上の質問は、テンプレートをご紹介しますために作成するシステムの範囲であり、ユーザからの要求を反映させた形で書かれています。

##### スキーマの設計及び実装例

今回ご紹介する金融サービス - 顧客の信用度分析ソリューションテンプレートは、ビジネス上の質問から判断して、口座の利用状況の分析、顧客/世帯の分析、プロモーション効果の分析、および収益の分析を行うシステム構築用のテンプレートになります。このシステムを構築するため、この実装例では、顧客のクレジット取引に関する分析用スキーマと、顧客の月間のクレジット口座要約用スキーマという2種類のスキーマ構成例を提案しています。またそのために、3つのファクト表を持つスキーマ構成をとっています。

##### テーブルの項目についての説明

前述したビジネス上の質問に答えるために作成した、各テーブルのデータ項目についての説明を行っています。

##### テーブルの作成とスターインデックスの設定

スキーマの設計及び実装例で示したスキーマ構成を実際に作成するDDLの記述例を示します。またINDEXの設定、特にスタースキーマ構成に最適化されたインデックスである、スターインデックスの定義例も提供しています。スタースキーマ構成で、RedBrickを使用する場合、まず最初に考慮するのが、スターインデックスを設定することであり、そのことが、IBM Red Brick Warehouseのパフォーマンスの良さを享受するための必須条件となります。

#### コンテンツ

- ↓ [はじめに](#)
- ↓ [構築手順](#)
- ↓ [サンプルのビジネス上の質問のセット](#)
- ↓ [論理データモデルとテーブルスキーマの設計](#)
- ↓ [各テーブルの項目内容について](#)
- ↓ [テーブルの作成とスターインデックスの設定](#)
- ↓ [ビジネス上の質問に答えるためのSQLステートメント](#)

#### 関連リンク

- [IBM Red Brick Warehouse 製品ページ](#)

#### 執筆者

データ・マネジメント・ソリューション技術部

堤 保晴



## ビジネス上の質問に答えるための SQL ステートメント

作成したスタースキーマ構造のデータベースに対して、ビジネス上の質問を行う場合に使用されるSQLステートメントの例を提供しています。SQLの記述については、この例以外にも色々と考えられると思いますが、このDMLを利用して、より最適なクエリーの記述の考察とか、要求以外の処理を実施する場合の元になるSQLとして使用することも可能です。

ここに挙げたような、基本コンポーネントの全部、あるいは一部を利用して、金融サービス向けの分析アプリケーションを作成することが可能となります。それでは、次に今までご紹介してきました、ソリューションテンプレートを使ったシステムの構築について、構築の手順に沿って見て行きましょう。

## 構築手順

金融サービス向けソリューション・テンプレートをより効果的に使用するためには、以下の手順で作業を行うことを、推奨しています。ここでは、実際のシステム構築手順の流れを示しながら、金融サービス向けソリューションテンプレートの利用ポイントとその使用方法を説明いたします。

**Step 1** エンティティと属性の定義を含む論理データ・モデルについて調べます。

ビジネス上の質問セットの内容から、システムを構築する上で必要なデータ項目を洗い出し、データを検証しながら、論理データモデルを作成していきます。次に作成した論理データモデルを構成するエンティティや属性情報を、ビジネス上の質問セットに答えることを考慮しながら、各テーブルのデータ項目としてそれらの情報を反映させたテーブルの作成と、作成したテーブルをスター型のスキーマ構成として定義します。

**Step 2** スキーマについて調べます。

今回、ソリューションテンプレートとして提供していますテーブルスキーマ構成は、すでに、ビジネス上の質問セットに答えるために、作成されたテーブルのスキーマ構成とその各テーブルごとの項目情報になります。また、ここでご紹介しているビジネス上の質問セットは、今回のソリューションテンプレートを説明する中で構築するシステムの範囲を規定しています。

**Step 3** データベースに格納するデータソースを識別する。

設計が完了したテーブルスキーマの各テーブルに対して、必要とするデータの内容についての検証を行います。この場合、事前に用意したサンプルデータを利用することで、このステップでの作業効率が図れます。

**Step 4** 電子媒体からサーバーにファイルをインストールします。

IBM Red Brick Warehouse 製品をインストールして、実行可能な状態にします。

**Step5** 具体的なデータ要件を正確に表わすよう、必用であればスキーマを変更します。

再度、スキーマ定義を検証し、データ要件の正確さを高め、必要であればスキーマ構造の変更を行います。

**Step6** スクリプトを実行し、データベースを作成します。

ソリューションテンプレートで提供しているデータベース作成スクリプトを、現行システム用にカスタマイズし、実行してデータベースを作成します。ここでは、テーブルの作成とスターインデックスの設定として、テーブル作成用のスクリプトとインデックスの設定を行うDDLファイルを提供しています。

**Step7** データベースにデータをロードします。

サンプルのデータ、あるいは実際に作成したデータをデータベースにロードします。

**Step8** 選択したクライアント・ツールを利用してビジネス上の質問を実施します。

ソリューションテンプレートとして提供しています、ビジネス上の質問に答えるためのSQLステートメントを使って、ビジネス上の質問を、クライアントツールを使って実行します。このステップではクエリーパフォーマンスを考慮するインデックス作成用のスクリプトと、最終目的であるビジネス上の質問に答えるためのSQLステートメントを用いて、実際のビジネス上の質問に答えていきます。この二つのソリューション・テンプレート（インデックス作成用のスクリプトと、ビジネス上の質問に答えるためのSQLステートメント）を有効に使うことで、最終的に出来上がったシステムのパフォーマンスと製品のクオリティを上げる効果があります。

それでは、最後に今回、ご紹介しますソリューション・テンプレート個々の詳細について、お話しさせていただきます。前述いたしました構築手順に従って、以下に示しますテンプレートを利用することが、システム開発の過程で有効であることを御確認いただければと思います。

## サンプルのビジネス上の質問のセット

ここで、記述していますビジネス上の質問セットは、今回のテンプレートを利用して構築するシステムを想定して記述されています。

### 口座の利用状況の分析

**質問1** 昨年末の各口座のタイプごとの口座所有者の総数は何人か？

**質問2** 多額の残高 (1,000 ドル以上) を持つ顧客が多いのは、どのようなタイプの口座か？

**質問3** 本年第1四半期における、VISAカードでの商品購入取引とキャッシング取引の比率はどうなっているか？

**質問4** 取引数が多く(30回を超える)、残高が少ない(500ドル未満)口座はどのようなタイプか？

**質問5** 本年第1四半期における、フロリダ州と比較した場合のカリフォルニア州の平均口座残高は？

**質問6** 口座残高が最も多い上位5つの州は？

**質問7** 昨年第4四半期に受領した金利収入の額は？

**質問8** 顧客/世帯の分析  
年収3万5,000ドル~5万ドルの男性顧客の昨年末の平均残高は？

**質問9** 独身の顧客と既婚の顧客の平均残高を比較するとどのような違いが見られるか？

**質問10** 信用度「A」の顧客のうち、残高を繰り越しているのは何パーセントか？これは、信用度「B」の顧客と比較した場合はどうか？

**質問11** プロモーションに関する分析  
前回のプロモーション中に口座の平均残高は増加したか？

**質問12** 前回のプロモーション中に顧客は何人増加したか？

## 論理データモデルとテーブルスキーマの設計

テーブル・スキーマの設計をする場合、ビジネス上の質問セットの内容から、システムを構築する上で必要なデータ項目を洗い出し、データを検証しながら、論理データモデルを作成します。

次にその作成した論理データモデルを元に、テーブルスキーマの設計を行います。この時、一番の注意点は、スタースキーマ構成を考慮し、定義するテーブルをFact表とDimension表に分けるように設計します。また、論理データモデルのエンティティや属性が各テーブルのデータ項目に該当するかどうかを判断、考慮しながら設計する必要があります。今回のソリューションテンプレートでは、上記手順を踏んで作成したテーブルとそのスキーマ構成を提供しています。

金融サービス - 顧客の信用度分析ソリューション・テンプレートとして提供いたしますテーブルスキーマでは、ビジネス上の質問である"口座の利用状況の分析、顧客/世帯の分析、プロモーションの効果の分析、および収益の分析"を反映させることができます。このスキーマには、Account Transaction (口座取引)、Monthly Account Summary (月間の口座取引要約)、および

Monthly Payments (月間の支払い) という3つのファクト表が含まれています。

Account Transaction (口座取引) ファクト表には、月次の預金取引明細書に記載される各勘定科目のレコードが含まれています。このファクト表には、Household (世帯)、Customer Account (顧客の口座)、Account Type (口座のタイプ)、Period (期間)、Merchant (加盟店)、Promotion (プロモーション)、および Transaction (取引) などのディメンション表が関連付けられています。

Monthly Account Summary (月間の利用明細書)

The Monthly Account Summary (月間の利用明細書) ファクト表には、請求サイクルの間における、各稼動口座についてのレコードが含まれています。各レコードには、現在の支払い請求サイクルにおける口座請求の内訳が含まれています。このファクト表には、Household (世帯)、Customer Account (顧客の口座)、Account Type (口座のタイプ)、Period (期間)、および Promotion (プロモーション) の各ディメンション表が関連付けられています。

Monthly Payments (月間の支払い) ファクト表には、顧客から受領した各支払いのレコードが含まれています。この表には、各稼動口座についての月次ベースの収益の内訳が含まれています。このファクト表には、Monthly Account Summary (月間の利用明細書) ファクト表と同じディメンション表を共有しています。このファクト表にはさらに、Account Status (口座の状況) 表と Payment Type (支払いのタイプ) 表も関連付けられています。

ここでご紹介しますテーブルスキーマは、考えられる数多くのテーブルスキーマ定義の1つにすぎません。また、テーブルスキーマへの一般的な拡張としては、集約表の使用がありません。Customer (顧客) のディメンション表が非常に大きく、Customer (顧客) の制約条件を使わずに必要なディメンションの分析を行うような場合には、集約表の利用が有効になります。

## 各テーブルの項目内容について

ソリューションテンプレートで提供します各テーブルの項目についての詳細を以下に示します。

### Account Status (口座の状況)

口座に割り当てられた、口座の状況が含まれています。これは、顧客の支払い実績を示します。

Status\_Id - 特定の口座状況を示す固有の整数です。

On\_Time\_Status - 口座の支払いが期日どおりに実施されたかどうかを示す True/False のフラグです。

Payed\_In\_Full\_Status - 口座の支払いが全額だったかどうかを示す True/False のフラグです。

### Account Transaction (口座の取引)

口座に適用された、貸方と借方それぞれのレコードが含まれています。

Date\_Key - 取引が発生した日付です。

Line\_Item\_Num - 取引内の特定の内訳を示す固有の整数です。

Cust\_Account\_Id - Customer Account (顧客の口座) から取得されます。

Acct\_Type\_Id - Account Type (口座のタイプ) から取得されます。

Promo\_Id - Promotion (プロモーション) から取得されます。

Merchant\_Id - Merchant (加盟店) から取得されます。  
Transaction\_Id - Transaction (取引) から取得されます。  
Amount - 特定の取引で口座に請求または入金される金額の合計です。

## Account Type (口座のタイプ)

顧客に発行された特定のクレジット・カードの情報を示します。

Acct\_Type\_Id - 特定の口座のタイプ (一般、ゴールドなど) を示すための固有の ID です。  
Card\_Name - カードの名前です (VISA、Master Card、Mobil など)。  
Acct\_Annual\_Fee - 顧客に請求される年会費です。  
Acct\_Bill\_Cycle - 支払い期間の日数です。  
Acct\_Cash\_Advance\_Fee - 顧客に請求されるキャッシング手数料です。  
Acct\_Type\_Description - 特定のアカウントのタイプについての、テキストによる説明です。

## Customer Account (顧客の口座)

クレジット・カード口座を所有する個人の情報を示します。

HH\_Id - Household (世帯) から取得されます。  
Cust\_Account\_Id - 特定の口座と顧客を示す固有の整数です。  
Cust\_Account\_Number - 特定の口座と顧客を示す固有の文字ストリングです。  
Cust\_Address\_Established\_Date - カード所有者が現住所での居住を開始した日付です。  
Cust\_Age - カード所有者の年齢です。  
Cust\_City - カード所有者が居住している都市の名前です。  
Cust\_Co\_Name - 口座のカードの共有所有者の氏名です。  
Cust\_County - カード所有者が居住している国の名前です。  
Cust\_Credit\_Limit - 口座に設定されている現在の与信限度額です。  
Cust\_Credit\_Rating - 信用度に基づいて顧客に割り当てられている評価です。  
Cust\_Date\_Bankrupt - カード所有者が破産を宣言した日付です (該当する場合)。  
Cust\_Gender - カード所有者の性別です。  
Cust\_Home\_Owner\_Flag - カード所有者が自宅を所有しているかどうかを示す True/False のフラグです。  
Cust\_Income - カード所有者のおおよその所得額です。  
Cust\_Insurance\_Premium - 保険料率です。  
Cust\_Marital\_Status - カード所有者の婚姻状況です。  
Cust\_Name - 口座を所有しているカード所有者の氏名です。  
Cust\_Nationality - カード所有者の国籍です。  
Cust\_New\_Cust\_Date - 口座が開設された日付です。  
Cust\_Num\_Dependents - カード所有者の扶養者数です。  
Cust\_Occupation - カード所有者の職業です。  
Cust\_Orig\_Credit\_Limit - 口座に対して最初に設定された与信限度額です。  
Cust\_Postal\_Code - 国内の各地域を示すコード (郵便番号) です。米国郵政公社および国際郵便サービスが使用します。  
Cust\_State - カード所有者が居住している州の名前です。  
Cust\_Waive\_Annual\_Fee\_Flag - カード所有者の年会費が免除されているかどうかを示す True/False のフラグです。

## Household (世帯)

1人以上のクレジット・カード所有者がいる世帯の情報です。

HH\_City - 特定の世帯が居住している都市の名前です。

HH\_County - 特定の世帯が居住している国の名前です。

HH\_Id - 特定の世帯を示す固有の整数です。

HH\_Income - 世帯のおおよその所得合計額です。

HH\_Name - 世帯主の名前です。

HH\_Num\_Children - 世帯内の子供の数です。

HH\_Num\_People - 世帯の総人数です。

HH\_Num\_Adults - 世帯内の成人の数です。

HH\_Children\_Over\_6 - 世帯内の6歳より上の子供の数です。

HH\_Children\_Over\_11 - 世帯内の11歳より上の子供の数です。

HH\_Num\_Males - 世帯内の男性の数です。

HH\_Num\_Females - 世帯内の女性の数です。

HH\_Num\_White - 世帯内の白人系住民の数です。

HH\_Num\_Hispanics - 世帯内のラテンアメリカ系住民の数です。

HH\_Num\_Afroamerican - 世帯内の黒人系住民の数です。

HH\_Num\_Other - 世帯内の上記以外の民族の住民の数です。

HH\_Income\_Range - 所得範囲のレベルを示す整数です (0 = 不明、1 = 0 ~ 1万5,000ドル、2 = 1万5,000 ~ 3万5,000ドル、3 = 3万5,000 ~ 5万ドル、4 = 5万 ~ 7万5,000ドル、5 = 7万5,000 ~ 10万ドル、6 = 10万 ~ 20万ドル、7 = 20万ドル超)

HH\_Num\_Employed - 世帯内の就労者の数です。

HH\_Num\_Over\_65 - 世帯内の65歳より上の人の数です。

HH\_Num\_Disabled - 世帯内の障害者の数です。

HH\_Num\_Retired - 世帯内の退職者の数です。

HH\_Num\_Drivers - 世帯内の運転免許保有者の数です。

HH\_Num\_Cars - 世帯が所有する車の数です。

HH\_Postal\_Code - 特定の世帯の居住地の郵便番号です。

HH\_State - 特定の世帯が居住している州の名前です。

## Merchant (加盟店)

当社のクレジット・カードによる支払いを受け付け、払い戻しを受ける特定の小売店を示します。

Merchant\_City - 加盟店がビジネスを行っている都市を示します。

Merchant\_Country - 加盟店がビジネスを行っている国を示します。

Merchant\_County - 加盟店がビジネスを行っている郡を示します。

Merchant\_Id - 商品を顧客に販売している特定の加盟店を示すために使用する固有の整数です。

Merchant\_Postal\_Code - 加盟店がビジネスを行っている場所の郵便番号を示します。

Merchant\_State - 加盟店がビジネスを行っている州を示します。

Merchant\_Type - 加盟店のビジネスのタイプを示します。

## Monthly Account Summary (月間の利用明細書)

各口座における取引の月間の要約が含まれています。

Bill\_Date - 顧客に対する請求日です。

Cust\_Account\_Id - Customer Account (顧客の口座) から取得されます。

Acct\_Type\_Id - Account Type (口座のタイプ) から取得されます。  
Promo\_Id - Promotion (プロモーション) から取得されます。  
Previous\_Balance - 前回の請求サイクルから繰り越された口座の残高です。  
Current\_Balance - 現在の口座の残高です。  
New\_Purchase\_Amount - 前回の請求サイクルでの、口座を利用した購入の総額です。  
Credit\_Amount - 口座に設定されている与信限度額です。  
Annual\_Percentage\_Rate - 顧客の未払い残高に対して課される金利の年間利率です。  
Interest\_Charge - 金利の金額です。  
Fees - 口座に請求される取引手数料の金額です。  
Insurance\_Premium - 口座保険の料金です。  
Num\_Transactions - その月の取引総数です。

## Monthly Payments (月間の支払い)

各口座で受領した月間の支払いが含まれています。

Payment\_Date - 支払いを受けた日付です。  
Cust\_Account\_Id - Customer Account (顧客の口座) から取得されます。  
Acct\_Type\_Id - Account Type (口座のタイプ) から取得されます。  
Promo\_Id - Promotion (プロモーション) から取得されます。  
Status\_Id - Account Status (口座の状況) から取得されます。  
Payment\_Type\_Id - Payment Type (支払いのタイプ) から取得されます。  
Payment\_Amount - 顧客から受領した支払い金額の合計です。  
Interest - 金利としての支払い金額です。  
Fees - 取引手数料としての支払い金額です。  
Insurance\_Premium - 口座保険料としての支払い金額です。

## Payment Type (支払いのタイプ)

顧客が使用する支払い方法を示します。この表には、支払いに使われる特定の金融期間の口座 (存在する場合) も含まれています。

Payment\_Type\_Id - 特定の支払いタイプを識別する整数です。  
Payment\_Method - 顧客が使用する支払い方法 (小切手、口座振替、自動引き落としなど) です。  
Financial\_Institution - 支払い元の金融機関の名前です。  
Account\_Number - 支払い元の金融機関の口座番号です。  
Account\_Type - 支払い元の口座の種類 (当座預金、普通預金など) です。

## Promotion (プロモーション)

指定された期間において顧客に提供される特定のプロモーションを示します。プロモーションはオファーの1つであり、新規および既存の顧客が申し込みを行います。

Promo\_Description - テキストによるプロモーションの説明です。  
Promo\_End\_Date - プロモーションの終了日です。  
Promo\_Id - 特定のプロモーションを示す固有の整数です。  
Promo\_Start\_Date - プロモーションの開始日です。

## Period (期間)

その年の日次のレコードが含まれています。

Date\_Key - 特定の日を示す整数です。

Calendar\_Day\_Num - 暦年中の特定の日を示す整数です。

Calendar\_Month\_Abbbr - 暦年中の特定の月を示す、月名の省略名です。

Calendar\_Month\_Name - 暦年中の特定の月を示す名前です。

Calendar\_Month\_Num - 暦年中の特定の月を示す整数です。

Calendar\_Quarter\_Num - 暦年中の、1月から始まる各3か月間を示す整数です (単一の暦年内では、暦四半期は固有です)

Calendar\_Year\_Num - ユリウス暦の年で定義される年 (1994、1995、1996 など) です。

Fiscal\_Month\_Abbbr - 単一の会計年度を12分割した、企業の時間グループを示す月名の省略名です (会計月は、複数の完全な週 (4週または5週) で構成され、単一の会計年度内で固有です)。

Fiscal\_Month\_Name - 単一の会計年度を12分割した、企業の時間グループを示す名前です (会計月は、複数の完全な週 (4週または5週) で構成され、単一の会計年度内で固有です)。

Fiscal\_Month\_Num - 単一の会計年度を12分割した、企業の時間グループを示す整数です (会計月は、複数の完全な週 (4週または5週) で構成される必要があり、単一の会計年度内で固有です)。

Fiscal\_Quarter\_Num - 単一の会計年度を4分割した単位で構成される、企業の時間グループです。

Fiscal\_Week\_Num - 会計年度中の連続する7日ごとにグループ化した、企業の時間グループです。会計年度中の固有の週を表します。

Fiscal\_Year\_Num - 会計報告年に相当する、企業の時間グループです。

Full\_Date - ユリウス暦の年を基にした、午前0時から始まる特定の24時間です。

Day\_Name - 曜日の名前 (Monday (月曜日)、Tuesday (火曜日)、Wednesday (水曜日) など) です。

Day\_Of\_Week\_Num - 週内の特定の日を示す整数です。

Day\_Type - 平日、週末、祭日などの日のタイプです。

## Transaction (取引)

顧客が開始した取引のタイプ (購入、キャッシング、キャッシング手数料など) を説明します。

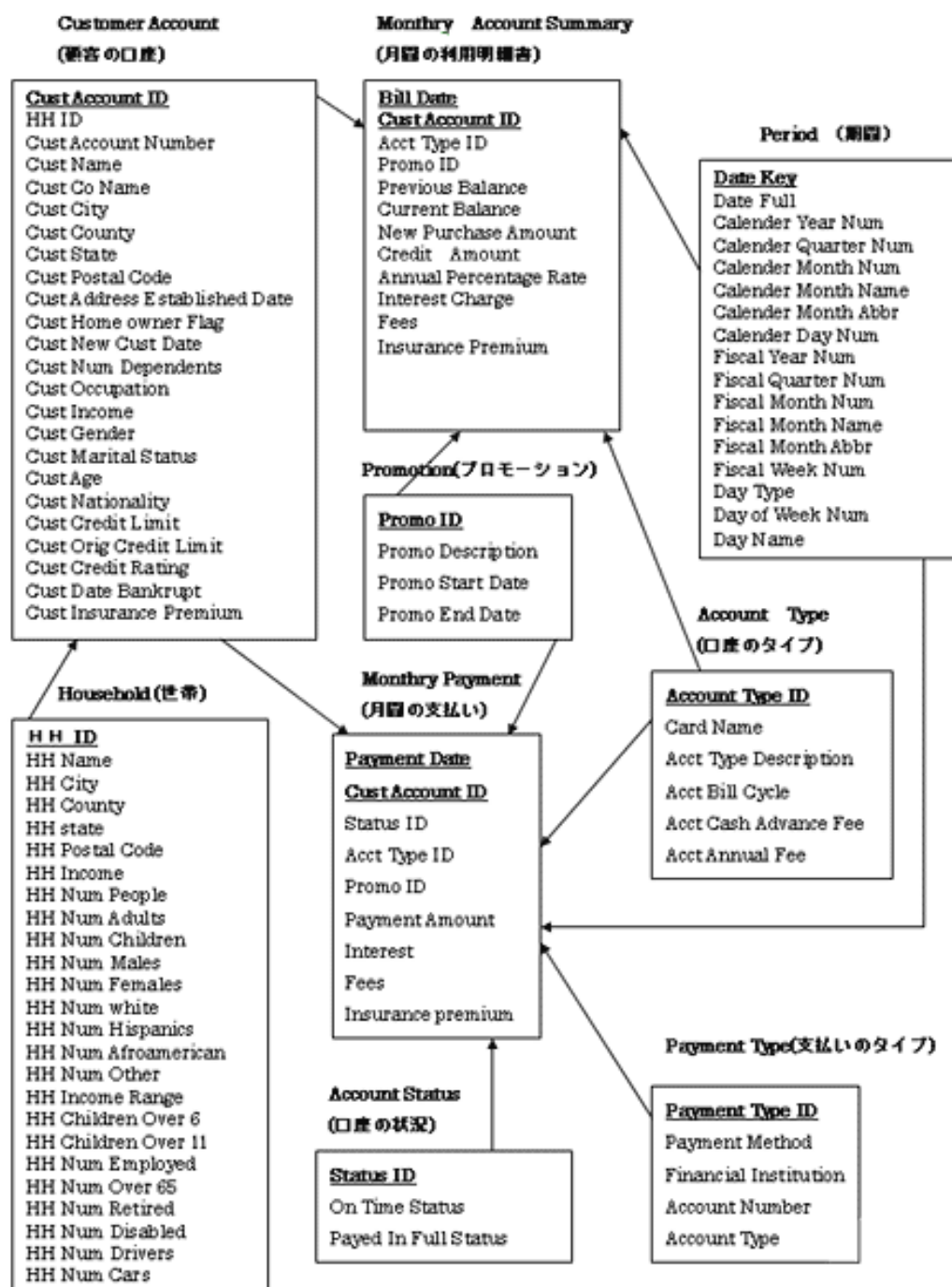
Transaction\_Id - 特定の取引のタイプを示す固有の整数です。

Transaction\_Description - テキストによる取引の説明です。

Transaction\_Type - 顧客の取引のタイプ (購入、返済、キャッシング手数料) を示します。

## 顧客の月間のクレジット口座要約スキーマ

## 顧客の月間のクレジット口座要約スキーマ





## Period Table

```
create table period (  
    date_key            integer not null,  
    date_full          date,  
    fiscal_year_num    integer,  
    fiscal_quarter_num integer,  
    fiscal_month_num   integer,  
    fiscal_month_name  char(10),  
    fiscal_month_abbr  char(3),  
    fiscal_week_num    integer,  
    calendar_year_num  integer,  
    calendar_quarter_num integer,  
    calendar_month_num integer,  
    calendar_month_name char(10),  
    calendar_month_abbr char(3),  
    calendar_day_num   integer,  
    day_of_week_num    integer,  
    day_name           char(10),  
    day_type           char(4),  
    primary key (date_key))  
maxrows per segment 1000 ;
```

## Promotion Table

```
create table promotion (  
    promo_id            integer not null,  
    promo_desc         char(30),  
    promo_start_date   date,  
    promo_end_date     date,  
    primary key (promo_id))  
maxrows per segment 1000 ;
```

## Account\_type Table

```
create table account_type (  
    acct_type_id        integer not null,  
    card_name          char(30),  
    acct_type_descr    char(40),  
    acct_bill_cycle    integer,  
    acct_cash_advance_fee decimal(4,2),  
    acct_annual_fee    decimal(5,2),  
    primary key (acct_type_id))  
maxrows per segment 1000 ;
```

## Merchant Table

```
create table merchant (  
    merchant_id          integer not null,  
    merchant_name        char(30),  
    merchant_type        char(10),  
    merchant_city        char(20),  
    merchant_county      char(20),  
    merchant_state       char(2),  
    merchant_postal_code integer,  
    merchant_country     char(20),  
    primary key (merchant_id))  
maxrows per segment 1000 ;  
create index merchant_index on merchant (merchant_name);
```

## Transaction Table

```
create table transaction (  
    trans_id            integer not null,  
    trans_type          char(15),  
    trans_desc          char(40),  
    primary key (trans_id))  
maxrows per segment 1000 ;
```

## Payment Type Table

```
create table payment_type (  
    payment_type_id     integer not null,  
    payment_method      char(10),  
    financial_institution char(30),  
    account_number      char(15),  
    account_type        char(10),  
    primary key (payment_type_id))  
maxrows per segment 1000;
```

## Account\_Status Table

```
create table account_status (  
    status_id           integer not null,  
    on_time_status     char(1),  
    payed_in_full_status char(1),  
    primary key (status_id))  
maxrows per segment 1000;
```

## Household Table

```
create table household (  
    hh_id            integer not null,  
    hh_name          char(40),  
    hh_city          char(20),  
    hh_county        char(20),  
    hh_state         char(2),  
    hh_postal_code   integer,  
    hh_income        decimal(8,2),  
    hh_num_people    integer,  
    hh_num_adults    integer,  
    hh_num_children  integer,  
    hh_num_males     integer,  
    hh_num_females   integer,  
    hh_num_white     integer,  
    hh_num_hispanics integer,  
    hh_num_afroamerican integer,  
    hh_num_other     integer,  
    hh_children_over_6 integer,  
    hh_children_over_11 integer,  
    hh_num_employed  integer,  
    hh_num_over_65   integer,  
    hh_num_retired   integer,  
    hh_num_disabled  integer,  
    hh_num_drivers   integer,  
    hh_num_cars      integer,  
    primary key (hh_id))  
maxrows per segment 1000 ;
```

## Customer\_account Table

```
create table customer_account (  
    cust_account_id  integer not null,  
    hh_id            integer not null,  
    cust_account_number char(16) not null,  
    cust_name        char(40) not null,  
    cust_co_name     integer,  
    cust_city        char(20),  
    cust_county      char(20),  
    cust_state       char(2),  
    cust_postal_code integer,  
    cust_address_established_date date,  
    cust_home_owner_flag char(1),  
    cust_new_cust_date date,
```

```

cust_num_dependents    integer,
cust_occupation        char(20),
cust_income            decimal(8,2),
cust_gender            char(1),
cust_marital_status   char(1),
cust_age               integer,
cust_nationality       char(10),
cust_credit_limit      integer,
cust_orig_credit_limit integer,
cust_credit_rating     char(1),
cust_date_bankrupt    date,
cust_insurance_premium decimal(6,2),
cust_waive_annual_fee_flag char(1),

primary key (cust_account_id),
foreign key (hh_id) references household (hh_id)
maxrows per segment 1000 ;
create index cust_index1 on customer_account (cust_state);

```

#### Account Transaction Table

```

create table account_transaction (
  cust_account_id    integer not null,
  date_key           integer not null,
  line_item_num     integer not null,
  acct_type_id      integer not null,
  promo_id          integer not null,
  merchant_id       integer not null,
  trans_id          integer not null,
  amount            decimal (7,2),

  primary key (date_key, cust_account_id, line_item_num),
  foreign key (cust_account_id) references customer_account (cust_account_id),
  foreign key (date_key) references period (date_key),
  foreign key (acct_type_id) references account_type (acct_type_id),
  foreign key (merchant_id) references merchant (merchant_id),
  foreign key (promo_id) references promotion (promo_id),
  foreign key (trans_id) references transaction (trans_id) );
create star index acct_trans_index1 on account_transaction
(date_key, acct_type_id, trans_id);

```

## Monthly Account Summary Table

```

create table monthly_account_summary (
  cust_account_id      integer not null,
  bill_date_id         integer not null,
  acct_type_id         integer not null,
  promo_id             integer not null,
  previous_balance     decimal (7,2),
  current_balance      decimal (7,2),
  new_purchase_amount  decimal (7,2),
  credit_amount        decimal (7,2),
  annual_percentage_rate decimal (3,2),
  interest_charge      decimal (4,2),
  fees                 decimal (4,2),
  insurance_premium    decimal (4,2),
  num_transactions     integer,

  primary key (cust_account_id, bill_date_id),
  foreign key (cust_account_id) references customer_account (cust_account_id),
  foreign key (bill_date_id) references period (date_key),
  foreign key (acct_type_id) references account_type (acct_type_id),
  foreign key (promo_id) references promotion (promo_id) );
create star index acct_sum_index1 on monthly_account_summary
  (bill_date_id, promo_id);

```

## Monthly Payment Summary Table

```

create table monthly_payment_summary (
  cust_account_id      integer not null,
  payment_date_id      integer not null,
  status_id            integer not null,
  acct_type_id         integer not null,
  promo_id             integer not null,
  payment_type_id      integer not null,
  payment_amount       decimal (7,2),
  interest              decimal (4,2),
  fees                 decimal (4,2),
  insurance_premium    decimal (4,2),

  primary key (cust_account_id, payment_date_id),
  foreign key (cust_account_id) references customer_account (cust_account_id),
  foreign key (payment_date_id) references period (date_key),
  foreign key (status_id) references account_status (status_id),
  foreign key (acct_type_id) references account_type (acct_type_id),
  foreign key (promo_id) references promotion (promo_id),
  foreign key (payment_type_id) references payment_type (payment_type_id) );
create star index acct_pay_sum_index1 on monthly_payment_summary

```

```
(payment_date_id, acct_type_id);
```

## ビジネス上の質問に答えるための SQL ステートメント

それでは、最後にビジネス上の質問に答えるためのSQLステートメントをご紹介します。以下のSQLはビジネス上の質問に答えるための一つの例としてご紹介いたします。

質問1

昨年末の各口座のタイプごとの口座所有者の総数は何人か？

```
select card_name,
       count (monthly_account_summary.cust_account_id) as num_accounts
from account_type, customer_account, period, monthly_account_summary
where account_type.acct_type_id = monthly_account_summary.acct_type_id
      and customer_account.cust_account_id = monthly_account_summary.cust_account_id
      and period.date_key = monthly_account_summary.bill_date_id
      and period.calendar_year_num = 2001
      and period.calendar_month_num = 12
group by card_name;
```

質問2

多額の残高 (1,000 ドル以上) を持つ顧客が多いのは、どのようなタイプの口座か？

```
select card_name,
       sum(previous_balance) as balance,
       rank (sum(previous_balance)) as bal_rank
from account_type, period, customer_account, monthly_account_summary
where account_type.acct_type_id = monthly_account_summary.acct_type_id
      and period.date_key = monthly_account_summary.bill_date_id
      and customer_account.cust_account_id = monthly_account_summary.cust_account_id
      and monthly_account_summary.previous_balance > 1000.00
      and period.calendar_year_num = 2002
      and period.calendar_month_num = 9
group by card_name
order by balance desc;
```

質問3

本年第1四半期における、VISA カードでの商品購入取引とキャッシング取引の比率はどうなっているか？

```
set arithignore;
select num_purchase, num_cash, (num_purchase / num_cash) as cash_ratio
from
  (select card_name, count (*)
   from period p1, account_type a1, transaction t1, account_transaction at1
   where a1.acct_type_id = at1.acct_type_id
        and p1.date_key = at1.date_key
        and p1.calendar_year_num = 2002
        and p1.calendar_quarter_num = 1
        and a1.card_name = 'VISA'
        and t1.trans_id = at1.trans_id
        and t1.trans_type = 'cash advance'
```

```

group by card_name) as cash_advance(card_name, num_cash)
natural join
(select card_name, count (*)
from period p2, account_type a2, transaction t2, account_transaction at2
where a2.acct_type_id = at2.acct_type_id
      and p2.date_key = at2.date_key
      and p2.calendar_year_num = 2002
      and p2.calendar_quarter_num = 1
      and a2.card_name = 'VISA'
      and t2.trans_id = at2.trans_id
      and t2.trans_type = 'purchase'
group by card_name) as purchase(card_name, num_purchase) ;

```

## 質問4

取引数が多く (30 回を超える)、残高が少ない (500 ドル未満) 口座はどのようなタイプか？

```

select card_name,
       sum(previous_balance) as balance,
       rank (sum(previous_balance)) as bal_rank
from account_type, period, customer_account, monthly_account_summary
where account_type.acct_type_id = monthly_account_summary.acct_type_id
      and period.date_key = monthly_account_summary.bill_date_id
      and customer_account.cust_account_id = monthly_account_summary.cust_account_id
      and monthly_account_summary.num_transactions > 20
      and monthly_account_summary.previous_balance < 500.00
      and period.calendar_year_num = 2002
      and period.calendar_month_num = 9
group by card_name
order by balance;

```

## 質問5

本年第 1 四半期における、フロリダ州と比較した場合のカリフォルニア州の平均口座残高は？

```

select avg (current_balance) as avg_calif,
       (select avg (current_balance)
        from customer_account, period, monthly_account_summary
        where customer_account.cust_account_id
              =monthly_account_summary.cust_account_id
              and customer_account.cust_state = 'FL'
              and period.date_key = monthly_account_summary.bill_date_id
              and period.calendar_quarter_num = 1
              and period.calendar_year_num = 2002) as avg_florida
from customer_account, period, monthly_account_summary
where customer_account.cust_account_id = monthly_account_summary.cust_account_id
      and customer_account.cust_state = 'CA'
      and period.date_key = monthly_account_summary.bill_date_id
      and period.calendar_quarter_num = 1

```

```
and period.calendar_year_num = 2002;
```

質問6

口座残高が最も多い上位5つの州は？

```
select cust_state as state,  
       sum(current_balance) as balance,  
       rank (sum(current_balance)) as bal_rank  
from customer_account, period, monthly_account_summary  
where customer_account.cust_account_id = monthly_account_summary.cust_account_id  
       and period.date_key = monthly_account_summary.bill_date_id  
       and period.calendar_month_num = 8  
       and period.calendar_year_num = 2002  
group by cust_state  
when bal_rank <= 5  
order by balance desc;
```

質問7

昨年第4四半期に受領した金利収入の額は？

```
select sum (interest) as total_interest  
from period, account_type, monthly_payment_summary  
where period.date_key = monthly_payment_summary.payment_date_id  
       and account_type.acct_type_id = monthly_payment_summary.acct_type_id  
       and account_type.card_name = 'VISA'  
       and period.calendar_quarter_num = 4  
       and period.calendar_year_num = 2002;
```

質問8

年収3万5,000ドル～5万ドルの男性顧客の昨年末の平均残高は？

```
select count (*) as num_cust,  
       avg (current_balance) as avg_bal  
from customer_account, period, monthly_account_summary  
where customer_account.cust_account_id = monthly_account_summary.cust_account_id  
       and period.date_key = monthly_account_summary.bill_date_id  
       and customer_account.cust_gender = 'm'  
       and cust_income >= 35000.00  
       and cust_income <= 50000.00  
       and period.calendar_month_num = 12  
       and period.calendar_year_num = 2002;
```

## 質問9

独身の顧客と既婚の顧客の平均残高を比較するとどのような違いが見られるか？

```
select cust_marital_status,
       avg (current_balance) as avg_bal
from customer_account, period, monthly_account_summary
where customer_account.cust_account_id = monthly_account_summary.cust_account_id
      and period.date_key = monthly_account_summary.bill_date_id
      and (cust_marital_status = 's' or cust_marital_status = 'm')
      and period.calendar_month_num = 8
      and period.calendar_year_num = 2002
group by cust_marital_status;
```

## 質問10

信用度「A」の顧客のうち、残高を繰り越しているのは何パーセントか？これは、信用度「B」の顧客と比較した場合はどうか？

```
select num_A, num_A_wbal, (num_A_wbal / num_A) as percent_w_bal
from
  (select cust_credit_rating, count (previous_balance)
   from customer_account, period, monthly_account_summary
   where customer_account.cust_account_id = monthly_account_summary.cust_account_id
         and period.date_key = monthly_account_summary.bill_date_id
         and cust_credit_rating = 'A'
         and monthly_account_summary.previous_balance > 0.0
         and period.calendar_month_num = 8
         and period.calendar_year_num = 2002
   group by cust_credit_rating) as balance(cust_credit_rating, num_A_wbal)
natural join
  (select cust_credit_rating, count (previous_balance)
   from customer_account, period, monthly_account_summary
   where customer_account.cust_account_id = monthly_account_summary.cust_account_id
         and period.date_key = monthly_account_summary.bill_date_id
         and cust_credit_rating = 'A'
         and period.calendar_month_num = 8
         and period.calendar_year_num = 2002
   group by cust_credit_rating) as no_balance(cust_credit_rating, num_A) ;
```

## 質問11

前回のプロモーション中に口座の平均残高は増加したか？

```
select promo_id,
       avg (previous_balance) as avg_bal
from period, monthly_account_summary
where period.date_key = monthly_account_summary.bill_date_id
      and period.calendar_year_num = 2002
      and (promo_id = 4
          or promo_id = 0)
group by promo_id;
```

## 質問 12

前回のプロモーション中に顧客は何人増加したか？

```
select num_prev_cust,num_cur_cust, (num_cur_cust - num_prev_cust) as num_new_cust
from
  (select calendar_year_num, count (cust_account_id)
   from period, monthly_account_summary
   where period.date_key = monthly_account_summary.bill_date_id
         and period.calendar_month_num = 9
         and period.calendar_year_num = 2002
   group by calendar_year_num) as cur_cust(calendar_year_num, num_cur_cust)
natural join
  (select calendar_year_num, count (cust_account_id)
   from period, monthly_account_summary
   where period.date_key = monthly_account_summary.bill_date_id
         and period.calendar_month_num = 5
         and period.calendar_year_num = 2002
   group by calendar_year_num) as prev_cust(calendar_year_num, num_prev_cust) ;
```