電力会社向け「資材 DWH(Data Warehouse)システム」の構築




関西営業所　上田昌行　阿路川 智　石田晶久　山本幸恵

1．はじめに

関西電力株式会社殿（以下、「関西電力殿」と略す）では、資材管理システムの ERP（Enterprise Resource Planning）化に伴って、「資材 DWH（Data Warehouse）システム」を構築することになり、当社がその開発を担当した。

今回、「資材 DWH システム」を構築する上で、業務要件を満たすための開発期間としては、決して長いものではなく、規模も大きい上に、ホストのデータ抽出から Web でのデータアクセスまで、開発範囲は多岐にわたった。また、業務要件は既存システムに加えて新システムのデータの連携も含まれるので、既存・新を考慮した上での設計となるため通常より設計作業は複雑であった。

本稿では、上記のような条件での「資材 DWH システム」構築プロジェクトの事例を紹介する。

2．システム概要

2.1　システムの目的

資材管理システムの ERP 化に伴い、基幹業務が既存システム「ホスト」と新システム「SAP（R/3）」とに変わることによって、既存 EUC（End User Computing）ファイルは作成できなくなる。EUC ファイル情報には、既存システム・新システムの両方の情報が必要だからである。

そこで、既存システムのデータと新システムのデータを取り込んで、資材 DWH を構築し、DM（Data Mart）に資材 EUC ファイルを作成する。本システムを稼働することによって、既存の定型分析 EUC に加えて、非定形分析を DM から容易に行うことを実現する。さらに、Web 画面から DM へアクセスできるようにし、既存ホストでのデータの取り扱いに比べて、操作性を高め、出力データの２次加工を容易に行うことができるようにする。

また、将来的にも DM へ加工する内容を整理すれば、CWH（Central Warehouse）データから目的に応じた定型分析業務を追加していくことが可能である。

今回の対象となった資材管理システムには、購買業務、貯蔵品業務（入出庫・在庫管理）、取引先管理業務の３つの業務が含まれる。

2.2　システムの概要・特徴

本システムは、既存システム・新システムの両方からインタフェース・データを取得して、同期を取り CWH へ蓄積していく。CWH に蓄積されたデータを DM で加工し EUC のファイルを作成できるようにする。具体的な処理の流れとしては、次のとおりである。

①ホスト処理では、各支店分散システムおよび本店分散システムで当日発生したデータを抽出し、ホストに転送する。
②ホストに転送されたデータを抽出および集約する。集約結果を転送ファイルに加工し、CWH へ転送する。
③CWH では、ホストから転送されたデータを DB のワークテーブルにロードする。
④さらに、新システム（R/3）から取り込んできたデータを DB のワークテーブルにロードする。
⑤CWH の内部処理としては、ホストデータ・R/3データの整合性チェックを行い、蓄積テーブルへ格納する。
⑥格納されたデータは DM 側で１次加工される。
⑦１次加工されたデータは、日次・月次・期次・年次処理で２次加工され、最終的にエンドユーザーが使用する EUC ファイルを出力用のテーブルとして作成する。
⑧作成されたテーブルは、エンドユーザーがブラウザか

らツール（本システムではWebFOCUS[*1]）を使用して参照する。また、データはExcel等に取り込んで保存することもでき、独自に編集可能である。

上記の処理の流れのうち、既存システム・新システムのデータの同期を取る部分の処理の難易度が高く、処理タイミング・処理時間も考慮しなければならない点である。

特に購買系の処理においては、契約情報などは既存システム（ホスト）側でデータが発生し、後続の検収データ、計上・支払データは新システム（R/3）側から取り込むため、データの整合性がポイントとなる。新システム（R/3）の基幹業務で発生する伝票などは、必ずしも時系列にデータが転送されず、過去日付、未来日付、追加入力、データの取消・削除、業務途中から発生するデータなど、データの発生パターンが複雑なため、DWHでのデータ加工処理も増加する。また、過去データが存在するため、過去からのデータ整合処理を行わなければならなかった。これは、データが増加するとともに処理時間も増加するので、処理時間がデータ量に依存しないように処理を考慮する必要がある。

その他、処理時間に影響するものとしては、上流データで毎日9割程度の変更が入る大量データが存在した。ほぼ全面更新されるようなデータについては、差分抽出および差分に対するマージ処理ではなく、全体を毎日取り直す方法で対応する必要がある。

3．開発の概要と指針

3.1　システム開発の目標

システム開発の目標としては、納期・予算・品質など多々あるが、特に品質保証に関しては、他システムとのインタフェース整合性・蓄積データ正確性の保証と、ユーザー・インタフェース／EUCの正確性の保証を最重要課題とした。

3.2　開発基本方針

本システムにおいて、品質保証の最重要課題であるデータの正確性を達成できるように、スケジュールも考慮して次のとおり2段階でシステムをリリースする方針で開発に当たった。

①CWHまでのデータ蓄積とDMの1次加工を第1段階としてリリース

②DMの2次加工（EUCファイルの作成）を第2段階としてリリース

このような開発形態を取った目的は、上記①によって、開発範囲を少なくしてデータの正確性を保証できるようにすることにある。また、期間が短いことおよび各インタフェース提供側システムも同時開発で実施しているため、リリース後のトラブル影響範囲を最小限にすることもねらいとした。

上記②では、第1段階リリースデータを使用して第2段階の開発を行えるので、最終加工データの精度をあげることができる。つまり、第2段階リリース分のシステムテストは本番データそのものを利用することができるということである。

3.3　開発範囲

本システム開発の対象範囲は、ホスト（COBOL）～クライアント/サーバー（Shell・PL/SQL）～Web（HTML・JavaScript）の多岐にわたった。

図1「資材DWHシステム構成図」中に番号で示すように、データの流れは、下記の①～⑥となる。

①既存ホストの最上流データを取得する。

②ホストで取得したデータをCWHへ転送する（各分散システムから、日次で発生している変更データの差分を抽出してホストへ転送し、ホストでは、各分散システムから送られたデータを集約してCWHへ転送する）。

③新システムで出力されているインタフェース・ファイルを取得する（新システム＝R/3は、R/3サーバー内に連携データを出力しているのでCWH側から対象インタフェース・ファイルを取得する）。

④CWHにてホスト・新システム両方のインタフェース・データを蓄積する（既存・新の連携データの整合性をチェックし、正常であれば蓄積する）。

⑤DMにてデータの1次加工を行う。

⑥DMにてデータの2次加工を行う（DMでエンドユーザーが参照したい形式に変更するなどして加工する）。

3.4　本番構成と開発言語

開発言語で考慮した点は、UNIX内の処理だが、ShellとPL/SQLだけに絞った。その他は必要である言語をそのまま使用することとなった。

実際に使用した言語を表1に示す（本番構成は図1参照）。

＊1）WebFOCUS：開発元は米国Information Builders, Inc.、日本での供給元は㈱アシスト。企業にとって必要不可欠な情報の共有化を、既存の環境を活かしたままで、低コストで容易に実現するツール。エンドユーザーは手近なWebブラウザから、企業内のあらゆるデータを定型・非定型にかかわらず自由にデータ検索可能となる。

表１　開発に使用した言語

項番	ホスト	C/S	Web
1	COBOL	K-Shell	HTML
2	JCL	PL/SQL	JavaScript

3.5　開発スケジュール

開発スケジュールは図２のとおりであった。

４．業務設計

4.1　ユーザー要件の確定

本システムの業務設計を行う上で、ユーザー要件での変更は影響が少なくなかった。結果から振り返ると、具体的にユーザー要件は決めているが、仕様変更・要件変更は頻繁に発生し、手戻り作業が大きくなった。これは、もともと作業工程もラップしているため、ある程度の変更による影響は事前にわかっていたので、最小限の変更対応を行う

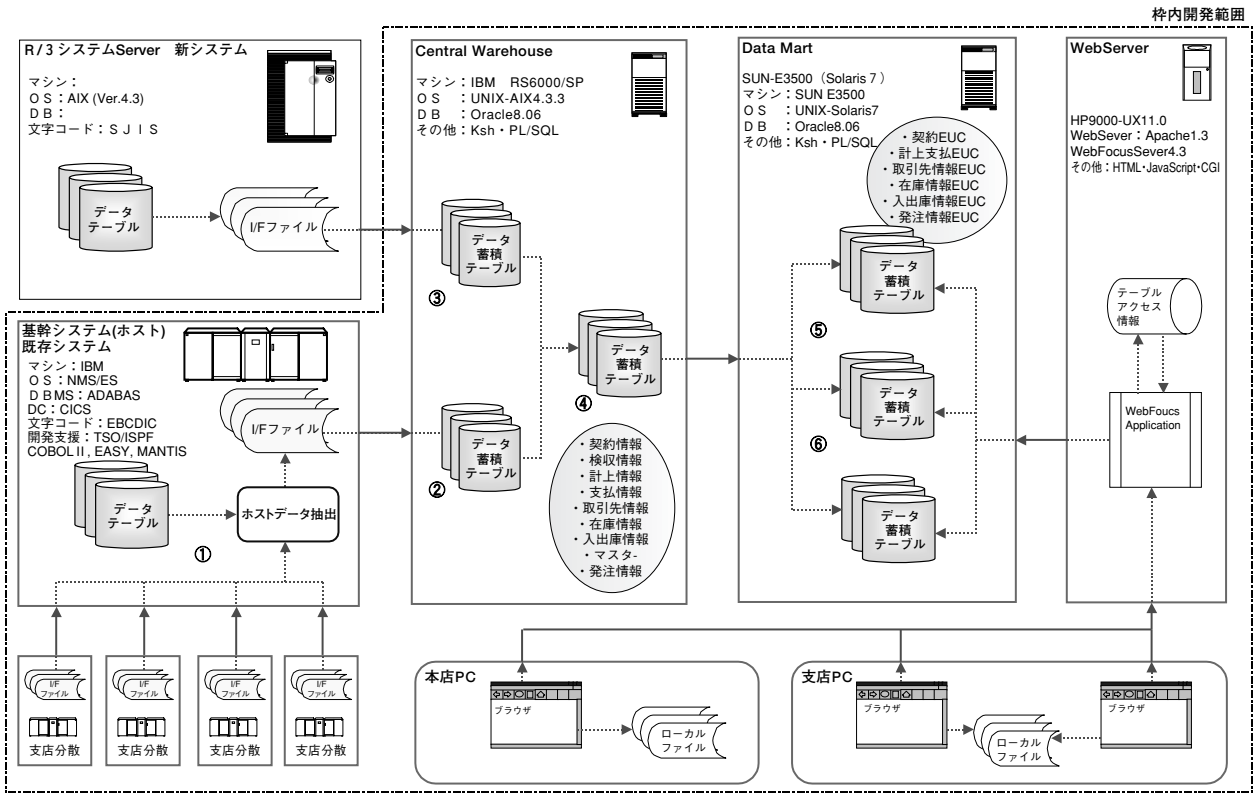

図１　資材 DWH システム構成図

マスター・スケジュール		２００１年								２００２年						
	工程	５月	６月	７月	８月	９月	１０月	１１月	１２月	１月	２月	３月	４月	５月	６月	７月
３月末リリース	概要設計															
	基本設計															
	詳細設計															
	開発															
	結合試験															
	システムテスト															
	移行															
７月末リリース	基本設計															
	詳細設計															
	開発															
	結合試験															
	システムテスト															
	移行															
	作業要員数	1	2	3	4	4	7	14	16	23	26	26	18	8	8	8

図２　開発スケジュール

ことができている。とはいえ、予測通り、手戻りの影響は小さくはなかった。

しかし、要件変更・仕様変更の発生要因として、決して要件定義が甘かったわけではない。本来、既存システムだけのインタフェースであれば、取得・連携・タイミングなど問題なくできるのだが、基幹業務の一部が新システムに移行されること、既存も全くなくならずに残ること、さらには同時開発であるため上流のデータ取得元のシステム・業務仕様にも影響されている。新システムの業務データの詳細部分まで全て把握した上でインタフェースを考慮しても、上流側での変更の影響は当然起こりうるのである。このあたりも事前予測の下に作業を進めているとはいえ、全体への影響が小さくなかったのも事実である。

4.2 業務設計

購買業務処理・貯蔵品業務（在庫処理）のDMを作成するためには、当然のことではあるが、既存処理を理解した上で作成する必要がある。そこで、多少時間はかかるが、スケジュールの許す範囲で、不明部分の業務理解には十分に時間を取るようにして進めた。また、作業の役割分担をしっかり行っていないと円滑に進捗させることはできない。そこで、開発者も業務の理解に必要に応じて時間を取った。

業務に関しては「既存踏襲型」、新システムに合わせた「新規開発型」、および「既存移行のみ型」が存在した。既存踏襲型は主に購買業務である。基本的な流れは変わることなく処理を行うからである。新規開発型は貯蔵品業務（在庫処理）であり、既存では使用頻度が少なかったため、基本的な考え方は既存と同じであるが、全面的に新しく考えての設計となった。既存移行のみ型は取引先管理業務であり、この業務は基本的に現行ホストで生成されているデータそのものを蓄積する。大きくはこの3パターンでの業務設計となった。

4.3 処理サイクル

処理サイクルは、日次・月次・期次・年次である。JOBネットとしては、全体の方針として、全ての処理は日次で動作させ、月次・期次・年次はインタフェース・ファイル内にデータが存在すれば通常に処理される。日次のタイミングでは、0件データでインタフェースして毎日動作するようにしている。

4.4 処理時間

バッチ全体の処理可能時間は当日の20時から翌日7時であるが、オンライン時間が延長される場合があるので、実際には22時から翌日7時の9時間が与えられている。

システム稼働後の実際の処理時間は、全体の処理開始時間が20時とすると、日次処理は平均翌日午前2時前後に終了している。月次処理に関しても遅くとも午前4時ぐらいには完了している。

設計時点の処理時間見積およびパフォーマンス試験においては結果がよくなかった。日次で翌日7時ぐらいであり、月次においては時間内に収まりがつかない状態であった。最終的には、下記の3つの方法によって解決した。

①パフォーマンス・チューニングによって処理時間を飛躍的に短縮した。
②今後のシステムの発展のことも考慮して関西電力殿がDMマシンのスペックアップ（本来の2～3倍の速度）を行った。
③JOBネット構成を考慮して対応した。

上記①および③については詳細を後述するが、③のJOBネット構成の考慮は今後にも対応できる対策である。

5．ツールの選定

5.1 ツールの選定

本システムを開発するに当たり、期間が短いことや開発生産性向上などを考慮し、開発の上で各種ETL（Extract Transformation Loading）ツールの導入を検討した。結果としては今回のDWH開発では、開発期間・システム環境要件に合うものがなく、システム構築でのツールの利用は断念した。ただし、エンドユーザーがブラウザで使用するWebFOCUSは関西電力殿が他部門でも導入されており、使用実績があることから資材部門でも使用することになった。その他使用しているツールや検討したツールに関しては本稿では割愛する。

5.2 ツールの役割

WebFOCUSは、内部的にはWebブラウザへデータベース内容を出力する、定形・非定形アプリケーション構築用の製品である。定型・非定形レポートの作成は開発者が行う。エンドユーザーはWebブラウザから検索するため、RDBMSの知識は全く必要ない。また、作成された定型レポートは一元管理され、多種多様な非定形レポートもWebブラウザからGUI（Graphical User Interface）ベースで作成できる機能を備えている。

今回の開発では定型レポートの作成、非定形レポート作成に必要なデータベース定義の作成などを行った。定型レポートは、エンドユーザーがWebブラウザの条件入力画面で指定した条件を元にデータ検索を行い、検索結果データを提供する固定のレポートであり、検索結果データの出力形式は、HTML表示とExcel形式での二種類がある。

なお、WebFOCUSに関して作成を行った部分は、条件入力画面（HTMLファイル）と、エンドユーザーがレポー

ト作成時に必要とするデータベース項目だけを定義したマスター定義ファイルの作成だけであった。このように、ツール使用は場合によっては、開発部分が非常に簡素になる。

5.3 ツールの効果

今回、関西電力殿の側にWebFOCUSの利用実績があったことによって、環境構築作業の軽減、関西電力殿との開発作業の分担など、開発に際してメリットが多々あった。

しかし、一方ではWebFOCUS側で対応不可能なために、データベース側にビュー作成等の作業が生じるなど、WebFOCUSの機能の限界によって問題もたびたび発生した。また、動作環境に関しても、Webブラウザのサポート対象範囲など、エンドユーザー側の環境があまり統一されておらず、いろいろな環境下で使用される場合には不都合を生じることも多い。本システムに関して、短期間の開発で大きな問題もなくWebFOCUSをうまく導入できたのは、業務面だけでなく環境面でも関西電力殿にご協力いただけたことが大きな要因と言えるだろう。

6．パフォーマンス・チューニング

6.1 チューニングの論旨

ここでは、PL/SQLにおけるパフォーマンス・チューニングに着目していきたい。

十分にチューニングされた高性能なシステムによって、高速な応答時間と優れたスループットが得られ、チューニングを鑑みない設計および開発よりもジョブ全体の所要時間を大幅に短縮させることが可能である。本システムでは、PL/SQLにおけるパフォーマンス・チューニングにより処理所要時間を大幅に短縮させることに成功した。

6.2 SQLチューニング

SQLチューニングを開発後のプログラム一部修正と考える人が意外に多いが、それは間違った認識である。なぜなら、設計および開発時にチューニングを意識していないとデータモデルの変更を強いられる可能性があり、パフォーマンスを意識しない設計および開発は非常に危険である。以下に、本システムにおける実施作業の流れと作業内容を⑴〜⑺の順に記述する。

⑴ チューニング前の実測値計測

CWHサーバーおよびDMサーバーで使用される処理を対象に、懸案対象の処理における処理所要時間を計測した。ここでの処理のI／Oは想定最大件数としてテストを実施し、SQL*LOADERやPL／SQLなど、全てを対象とした。

⑵ 検索条件文の洗い出し

処理で使用している検索条件（WHERE句、GROUP BY句、ORDER BY句）を全て調査し、プライマリキーやインデックスを使用しないと想定される条件を洗い出した。当作業は、自分が書いたプログラム以外に他人の書いたプログラムも入念に調査する作業であるため、非常に労力を要する作業である。しかし、本システムに限らず、この作業がパフォーマンス・チューニングのキーポイントであると断言できる。なぜなら、条件文の洗い出し漏れや基礎的なSQLの間違い（算術演算子や関数の使用方法）などを発見できないと、チューニング後の実測値計測などの手戻り作業（重い処理の計測には数時間掛かることもある）が発生する可能性があるからである。よって、慎重に調査を遂行することが肝要である。

⑶ SQL修正

SQLの基礎的な間違いの修正を実施した。SQLの記述方法が大きくオーバーヘッドに影響することは周知の事実であるが、全てのプログラマがSQLを熟知しているわけではないため、熟練者がチェックすることは大変重要である。ここでは、以下のような間違いがよく見受けられた。

- 更新キーなどにROWIDを使用可能な場面で使用していない
  →レコードに存在するROWIDは、最速のレコード検索方法である。
- INDEX（索引）がオプティマイザの実行計画で使用されない条件文をWHERE句で使用している。
  →算術演算子や関数を右辺でなく左辺に使用しているなど
- INDEX（索引）のついていない条件で負荷の高い条件文がWHERE句の先頭に配置されていない。
  →ORACLEでは、最初の句から評価されていくので、指定順序を意識しなければならない。
- その他
  →COMMITする件数が少ない（＝データサイズ）処理に対して、COMMIT件数を増やす変更を行った。

⑷ SQL実行計画の確認

上記⑵の作業で見つかった問題のSQL文（上記⑴で処理時間を多く要する処理など）を対象に、「EXPLAIN PLAN」コマンドを使用し、オプティマイザが最良の実行計画を選択し実行しているかを確認した（キーでの検索やインデックスでの検索を実行しているかなど）。これによって、望ましくない検索パスをオプティマイザが選択しているSQL文を発見し、ヒント句などを使用し修正する。

⑸ ワークテーブルの使用

ヒント句などを使用してもディスクI/Oが極端に必要となるようなSQLでは、データモデルを見直し、ワークテーブルを用いることでディスクI/Oを減らし、実行速度の改善に努めた（本システムではパーティションは使用できない環境であった）。これによって、大幅に処理を短縮することが可能となった。

⑹ **INDEX（索引）の検討**

テーブルにおける予想データ量やデータ分布を調査し、INDEX（索引）を定義するかを検討した。INDEX（索引）は常にデータベースのパフォーマンスの問題の解決になるとは限らないし、使用することによるコスト（記憶領域、保守、INSERT・UPDATE・DELETE時のオーバーヘッドなど）もかかるため、慎重な検討が必要である。

⑺ **チューニング後の実測値計測**

I/Oは想定最大件数としてテストを実施し、チューニングの成果を確認した。上記の⑴〜⑺の作業を実施することにより、数時間を所要した処理が数分に短縮された。

6.3 チューニングの総則

⑴ **設計（基本〜詳細設計工程）**

PL/SQLを中心としたバッチ処理システムでは、設計者は常にパフォーマンスを意識した設計を行うことが必須である。製造工程に入ってからでは手遅れになる場合もあるため、設計者は設計＝プログラムと考えるべきである。いかに優秀なプログラマといえども、データモデルにおいて設計不良であるアプリケーションに対し、オーバーヘッドなしにプログラミングすることは不可能である。

⑵ **開発（製造工程）**

プログラマは期待した結果が得られたのだから、それが正しいSQLだと信じてしまう傾向があるが、それは間違いである。SQL文の選択が正しいといえる条件は、他のシステム・リソースのパフォーマンスを妨げずに、最速の時間で正しい結果を作成する場合だけである。プログラマは担当分全てのアプリケーションが効率よく動作し、最良のスループットを得ることができるようにチューニングする権限と責任がある。また、SQLの記述次第では、多大なオーバーヘッドを生じさせる可能性があることを十分に認識することが肝要である。

⑶ **パフォーマンス・テスト（結合試験工程）**

DWHシステム開発では、パフォーマンスがプロジェクトの命運を左右する1つのリスクである。システムが取り扱うデータ量が多い場合はなおさらである。可能な限り早い段階でパフォーマンス問題を発見し、解決することによって、リスクヘッジを行うことが重要ではないだろうか。

7．移　　行

システム全体の最終関門が移行である。ここでは本システムの移行において注力した事項について紹介しておこう。

7.1 移行実施計画

移行に当たって、事前リハーサルを2回行い、問題点を確認できるように計画した。移行手順の確認、作業時間の確認、データの確認などを実施し、本番移行時に時間内に手順が収まるような計画とした。

通常の場合も、上記のような手順は踏まえるものであるが、特に本システムでは、既存システムの改修、新システムの切替、DWHシステムの稼働といったシステム切替を一斉に実施するので、念入りな計画を立てた。

結果的には、移行作業自体は予定通り実施・完了できた。その後、データが部分的に不正であったりするので、何度となく移行手順を繰り返して、データの整備を行うことになった。やはり計画段階では予想されないデータが存在し、そのデータを整備するために、繰り返し作業を行った。徐々にデータが整備され、最終的にデータ量が少なくなると修正は直にパッチを当てるなどして収束していった。

7.2 移行ポイント項目

移行計画においては、まず移行対象の抽出作業を行う。この作業の中では、移行する対象データは過去に取り決めた範囲のデータに加えて、新システムR/3へ移行される部分は新システムからの移行を行わなければならない。移行においてポイントとなった4つの項目、「ホストデータの扱い」「初期登録」「変換」「データ・クレンジング」の内容を以下に紹介する。既存システムと新システムとの同期を取らなければならないことが、全体的にかかわっている。

⑴ **ホストデータ**

ホストシステムでは長年使われてきたコード体系があり、新システム（R/3）では新たなコード体系が作成されている。資材DWHシステムでもこれをふまえ、ホストから資材DWHにデータを取り込む際にはホストの旧コードを新システムに対応したコードに変換する対応を行った。

⑵ **初期登録**

新DWHシステムが既存のホストEUCシステムの置き換えとなることから、既存ホストシステムで作成されていたEUCファイルの移行としては、既存ホストで過去に作成されたEUCファイルを過去5年分移行することとした。

⑶ **変換**

既存ホストシステムから新DWHシステムに移行する際、新システムのコード体系に合わせ、変換を行った。ただし、移行時において一部変換処理がうまく行えなかったため、本番稼働後においても一部移行処理を再度行っているデータがある。

⑷ **移行（データ・クレンジングについて）**

通常DWHシステムでは、上記初期登録で述べたように、移行データは本システムで過去5年分程度を移行する必要がある。しかし、現行ホストシステムのデータをDHW移行する際、次のような問題があり、対応を必要とした。

①新システムでは外字を使用しないため、既存ホストEUCファイル内の外字データをスペースに変換した。

②新システムにおいて、DWH のテーブルと構造が異なるため重複データが発生した。このため、ホスト側にて一部重複データをカットするなどの対応をした。

7.3 システムの稼働

システム稼働後、データ不正が上流で発生すれば下流側の DWH にも影響が出るので、その都度対応することとなった。システム稼働後、1 週間弱で JOB は安定したが、データの内容に関しては 1 カ月～2 カ月は落ち着かなかった。直接的な影響はないものの予定外のデータが潜んでいることによってエラーは発生する。その都度対処し、データも安定していった。

8．JOB ネット

本システムの JOB ネットについて紹介する。

基本的には他のシステムでも組まれている一般的な方法であり、特に問題になる部分はなかったが、注意した点として 2 つあげる。

1 つは、既存システムと新システムの両方からデータが連携され、DWH 側で同期を取って処理することである。これは既存システムを基準としてホストから運用日付を取得し、その運用日付が取得されることと、各インタフェースが連携されることを条件に CWH の各 JOB ラインが起動する仕組みとした。

2 つ目は、JOB 量が多いことに対し、情報系システムの DWH は基幹業務の処理後に起動されるので、夜間の与えられる処理時間が短くなる。試算した JOB 処理時間を考慮して JOB を組み合わせる上で、日次・月次・期次・年次と一時的に JOB 量が増える部分に関しては、業務に影響が少ないように複数日で処理を実行できるように制御情報の設定で簡易に変更できる仕組みを組み入れた。

次に、JOB ネット・トラブル時の対応としては、関西電力殿の基準として DWH システムは翌日対応と決定していることと、不正データが CWH に流れた場合の復旧には時間がかかる可能性が高いことによって、DWH 側関連 JOB は後続 JOB 全て中断することとした。

9．効　果

本システム導入前は、基幹システムで作成されている EUC ファイルは月次で処理されていたが、本システム導入により日次で処理を行うこととなった。EUC ファイルを取り扱って定常作業や依頼作業を行う場合に、長いもので数時間かかっていた。本システム導入後は、同様の作業を数分でできるようになった。

また、ブラウザから Excel などで加工して作業できるようになったことによって、従来ホストで処理するときに必要であった知識も不要になり、より柔軟にデータを加工することができるようになった。一定の形でしか見られなかったものが、2 次加工できるため使い勝手が非常に良くなった。

従来と比較して使い勝手はかなり向上しているので、使用するユーザー数も増加して、今後さらに仕事に役に立つことが期待できる。

10．おわりに

資材の DWH システムは今回の構築が最初となるが、CWH･DM は、その他システムのデータ取得元(参照元)であるので、今後、必要に応じて情報は追加されるであろう。

本システム開発は、短期間で開発範囲の幅が広かったが、関西電力殿を先頭として当社の開発・設計担当者がシステム構築完了という共通の目的の下に認識を統一して作業を行うことができたので、無事完了することができたと思う。

本システムの開発に関してご協力いただいた関西電力殿、当社開発メンバーおよび関連各位に、この場を借りて心からお礼申し上げたい。