FileMaker

FileMaker データベースと SQL データベースとの統合活用

– 基幹系データベースにもっとワークグループの自由度を –

木下 雄一朗　著

目　次

■目的

このドキュメントは、下記のような方々を対象としています。

・FileMaker と基幹システムを持っていながら、有効な連携ができていない方。

・基幹システムを持っているが、情報の二次利用ができていない方。

・FileMaker による Web 公開のサイトをスケールアップしたい方。

・Web サイト上のデータの管理、帳票作成に困っている方。

こうした状況で困っている方は、FileMaker Pro の外部 SQL データソースの機能を活用することで、一気に問題を解決できる可能性があります。

筆者は、FileMaker と基幹システムとのデータ連携、Web データベースの管理ツールなどを長年開発してきましたが、外部 SQL データソースが使えるようになって以来、開発の生産性やシステムの柔軟性を大幅に向上させることができました。

このドキュメントでは、外部 SQL データソースを活用することで、生産性の向上が期待できる４つの典型的なモデルケースを想定して、ケースごとにメリット、導入方法、注意点などを紹介します。視点としては、機能の説明というよりは、現場での活用方法を軸にした内容になっています。

FileMaker、基幹システム、Web システム、それぞれの長所を生かした、機能性と柔軟性を持つシステムを実現するための参考資料としてご利用いただければ幸いです。

■外部 SQL データソース（ESS）とは

外部 SQL データソースは、FileMaker Pro 9 で追加された比較的新しい機能で、英文では「External SQL Data Source」と表記されています。正式名称は長いため、このドキュメントでは、略称の「ESS」の方を使いたいと思います。

ESS は、ひとことを説明すると、「SQL データベースのテーブルを FileMaker 内にある仮想的なテーブルとして扱うことができる機能」です。

もう少しかみ砕くと、「SQL データベース上のデータを、あたかも FileMaker 上のデータと同じように、閲覧、更新ができる機能」と言うこともできるでしょう。

通常、Oracle、MS SQL Server、MySQL など他のデータベースで定義されているデータにアクセスするためには、専用のクライアント、ライブラリ、ODBC/JDBC などの汎用データ・インターフェースなどを通じて、SQL という問い合わせ言語を実行します（FileMaker の［ODBC インポート］や［SQL を実行］スクリプトはその一例です）。

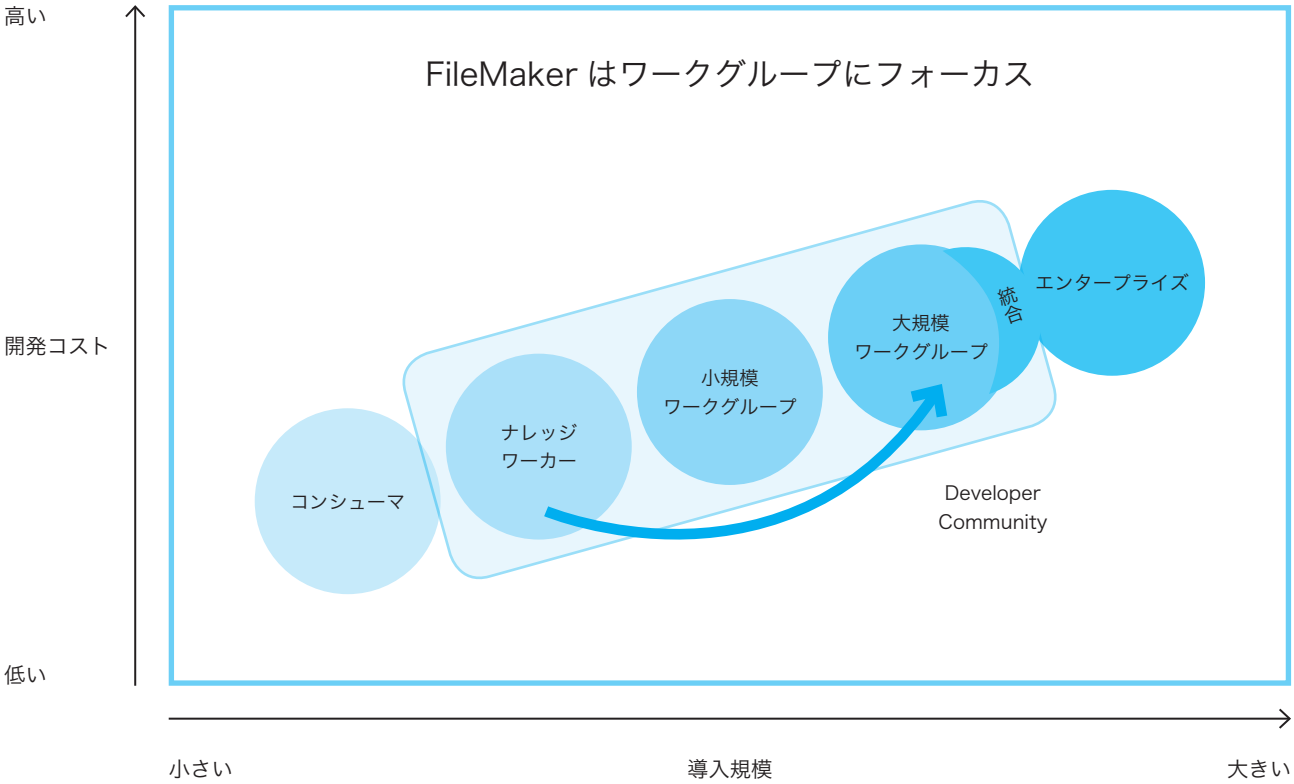
SQL はリレーショナル･データベースの開発者であれば必要な知識ですが、その障壁によって、現場の一般利用者がデータを柔軟に活用することが難しくなっている側面があることも否めません。ESS を使えば、FileMaker を使う知識があれば、SQL データベース上のデータを利用することができるため、データ活用の敷居を大きく引き下げることができるのです。

■ ESS の位置づけ

FileMaker は、ユーザフレンドリーなデータベース製品として、個人レベルから企業のワークグループ、基幹のサブシステムのレベルにいたる幅広い環境で利用されています。

これは、FileMaker が「誰にでも分かりやすい、使いやすい」という間口の広さと、かなりの規模のネットワーク利用に対応できるスケーラビリティや機能性を兼ね備えているからでしょう。また、Windows、Mac OS X の両方で動作するクロスプラットホーム製品であることも見逃すことができない利点のひとつです。

2009 年 10 月 30 日（金）に開催された「FileMaker カンファレンス 2009」では、米 FileMaker, Inc. 社長の Dominique Goupil 氏による基調講演が行なわれましたが、その中では、「FileMaker のターゲットはワークグループ市場」と冠したチャート図を使用して、FileMaker のターゲット層についての説明されました。



チャート図では、ターゲット層の帯が、ナレッジワーカー、小規模ワークグループ、大規模ワークグループの領域はもちろん、エンタープライズ領域の一部にもかかっていて、FileMaker は「エンタープライズとワークグループの橋渡し」にも対応している点が強調されました。

ESS は、まさにこの「エンタープライズとワークグループの橋渡し」を具現化する機能と言って良いでしょう。

FileMaker は、ナレッジワーカーから部や課のようなワークグループで利用されていることに対して、一般に、エンタープライズ用途（基幹システム、Web アプリケーションなど）では、SQL データベースが利用されます。

FileMaker、SQL データベースには、それぞれの良い点があるのですが、従来は、両方のデータベースを持っていても、相互のデータの連携は図られておらず、データの活用や運用には制限があることが大半でした。

従来から、テキストファイルの交換や ODBC インポートや［SQL を実行］スクリプトなどで、定期的なデータの連携を行うことはできましたが、そのための技術的な課題、手間の問題などによって、それぞれが独立して運用されているのが一般的だったのです。

ESS は、「SQL データベース上のデータを、あたかも FileMaker 上のデータと同じように」扱うことができるため、上記のようなシステム間のデータ分断問題を一気に解消することができるのです。

ただし、注意点もあります。個々の注意点は後で触れますが、ESS が開発された目的は「FileMaker によるシステムに SQL データベースをシームレスに統合すること」にある点です（SQL データベースのための完全なフロントエンド環境を実現することが目的ではない点がポイントです）。

つまり、各 SQL データベースの特性に合わせて、処理速度を最適化することよりも、FileMaker における操作手順や操作結果の一貫性が優先されているのです。

そのため、SQL データベースの機能を生かすという点では、必ずしも最適な結果が得られるとは限りませんので、ESS の特徴や仕様上の制限を理解して、適材適所で使うことが活用のポイントになります。

ESS 登場前の解決方法

FileMaker Pro 9 が登場して、ESS を使えるようになる前でも、基幹システムと FileMaker のデータ連携を実現する目的で使える機能はありました。

具体的には、下記のような機能になります。

1. テキストファイルのインポート / エクスポート
2. ODBC インポート、［SQL を実行］スクリプト
3. FileMaker の ODBC/JDBC 公開
4. Web 公開系 API（XML 公開、PHP）

テキストファイルの交換は、一番歴史のある方法です。例えば、基幹システムからバッチ処理で書き出したテキストファイルを FileMaker から定期的にインポートするといったパターンです。

そのパターンのメリットは、データを書き出すプログラムとインポートするプログラムが完全に分離されているため、開発時の役割分担や障害時の問題の切り分けがしやすい、という点があります。

ただし、定期的にデータの同期を行うだけの仕組みになるため、リアルタイム性はありません。基幹システムに入力をして、すぐに FileMaker 上でも利用したいような用途では、反映のタイムラグは大きな問題になります。

「バッチ処理のタイミングを頻繁にすれば良いのではないか？」というと、実はそれほど単純なものでもありません。

例えば、深夜の誰もアクセスをしていない時間帯に、それほど大きくないテーブルのデータを丸ごと更新するだけであれば、該当のテーブルで全レコード削除→全件が含まれるテキストファイルをインポート、ということも可能ですが、データが大きかったり、常に利用されている可能性があるシステムでは、排他制御を含むエラー処理、差分更新の方法を検討する必要が生じます。FileMaker 上での更新を SQL データベースに書き戻したい場合には、さらに複雑なロジックが必要になります。

また、インポート / エクスポート後のテキストファイルも、トラブル対応時の参考情報として一定の期間は残しておきたいものですが、放置しておくとディスク容量を消費するため、定期的なファイル操作（古いものを削除など）も必要になります。

つまり、取っ付きやすそうに見えて、データ連携の方法によってはさまざまな課題が出てくることがあります。

ODBC インポートは、テキストファイルではなく、ODBC を通じてインポートするパターンです。このパターンでも、リアルタイム性がないこと、エラー処理や差分更新への配慮が必要な点などは同じですが、物理的なファイルを経由しないので、処理後のファイルの取り扱いを心配する必要はありません。また、SQL データベースへの更新が必要な場合

には、[SQL を実行] スクリプトを使うことができます。

ODBC インポート、[SQL を実行]スクリプトには、「SQL 文を自由に書くことができる」という特徴があります。これは、ESS にはないメリットで、場面、用途によって使い分けや併用が効果的です。

ODBC/JDBC 公開、Web 公開系 API は、FileMaker がデータベースを公開して、外部からデータ操作を行うパターンです。プログラムが必要になるため、システム部署や開発会社に依頼することになるでしょう。エラー処理や差分更新については、IT 部署や開発会社がケアしてくれるでしょうが、作成のための期間やコストは必要になります。

例に挙げた４つの方法は、それぞれのメリット、デメリットがあるのですが、共通するのはリアルタイム性がないこと、一般のエンドユーザにとって簡単な方法ではないことです。

ESS でのデータ連携のメリット

ESS 登場前の解決方法と比べて、ESS を使った場合のメリットは下記のようになります。

1. リアルタイム性
2. 双方向性
3. SQL やプログラムの知識が不要

ESS は、外部データソースとして定義された SQL データベースのテーブルやフィールドの値を、動的に取得し、FileMaker から利用できるようにします。SQL データベースのテーブルやフィールドは、FileMaker のリレーションシップグラフ上ではシャドウがかかって表示され「シャドウテーブル」「シャドウフィールド」と呼ばれています。

シャドウフィールドが、レイアウト上で表示された、計算フィールドやスクリプトから値を参照された、などのタイミングでデータを取得するため、リアルタイム性があります（SQL データベース上の値の変更を常時監視しているわけではないので、“ ほぼ ” リアルタイムのレベルです）。

シャドウフィールドは、通常の FileMaker フィールドを同じように、更新も可能です（SQL データベース上の更新権限が必要）。他のデータ連携方法では、双方向の更新を実現することは難しいことが多いので、双方向のデータ連携ができる点は大きなメリットになります。

そして、こうした機能を実現するために、SQL やプログラムの知識は必要ありません。シャドウテーブルのレコードを、FileMaker のレコードと同じように操作することができます。ODBC インポート、[SQL を実行] スクリプトを駆使すれば、擬似的に ESS のような動作を実現することも可能ですが、そのために必要な知識、時間は ESS と比べて格段にハイレベルなものになります。

ファイルを経由しないため、テキストファイルの交換の場合のように、バッチ、シェルスクリプトなどを使ったファイル操作のためのプログラムも必要ありません。

つまり、ESS を使うことで、リアルタイムで、更新も可能なデータ連携を、特別な知識がなくても、実現することができるのです。

■ ESS の設定方法

ESS の基本的な設定手順は、下記のようになります。

□ SQL データベース側の準備

1. ESS として接続するための、テーブルやビュー、アクセス権限、アカウントを準備。

□ FileMaker 側の設定手順

1. 接続先データベースに対応した ODBC ドライバのインストール
2. DSN（データソース名）の設定
3. FileMaker 上での外部データソースの定義
4. リレーションシップグラフへの「テーブルの別の名前」の追加

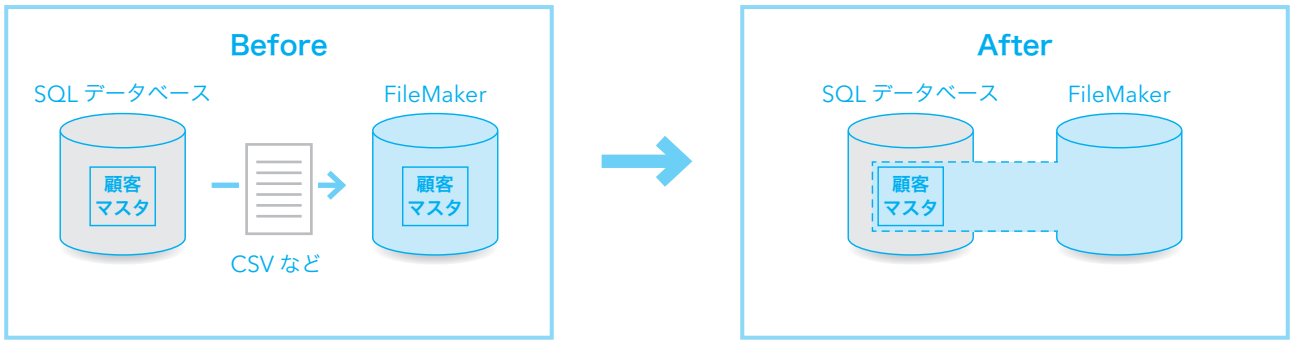
※ FileMaker Server を使用する場合は、ODBC ドライバのインストール、DSN の設定はサーバ上で行うだけで、各クライアント上では必要ありません。

ESS は、ODBC を経由して SQL データベースにアクセスします。ODBC ドライバは、接続先のデータベースに対応したものを使用することに注意して、DSN の設定については、ドライバごとに設定内容が異なるので、ドライバのメーカーのホームページやインターネット上の情報を探せば、具体的な設定例を見つけることができるでしょう。

なお、詳しい設定手順は、FileMaker オンラインヘルプのほか、巻末で紹介する資料などに掲載されているので、そちらをご参照ください。

■ ESS 導入ケーススタディ

ケース 1 （基幹システムと FileMaker のマスタ情報の連携）



背景

規模の小さな組織では、FileMaker 製品ラインのみで社内の全システムを構築していることもありますが、中規模以上の組織であれば、勘定系システム、CRM、ERP といった基幹システムは、SQL データベースで構築されたシステムが主流となっています。

基幹システムは、安定性、信頼性などが重視され、開発や導入には大きなコストがかかるため、通常、頻繁に変更やリプレースは行なわれません。しかし、現場でのビジネスは、それこそ毎日のように、お客様の都合であったり、社内の都合であったり、経済環境の変化であったり、さまざまな理由で、変化を求められます。

そのため、現場では、FileMaker のように、自分たちで開発、管理ができるソフトウェアを駆使して、基幹システムに足りない機能はそちらで補完するということがよく行なわれます。ケース 1 では、そのようなパターンを想定してみたいと思います。

こうした利用形態は、組織内にいるパソコン好きでスキルの高い人が、自分を楽にするために作り始めることが多いようですが、実際の業務を行っている知識をもとに作成するので、現場のニーズや業務手順にあわせた工夫が凝らされていたりします。

その便利さに惹かれて、他の人も使い始めると、作る人のやる気が刺激されて、機能強化を繰り返し、部署単位で承認が得られれば、FileMaker Server 導入、ボリュームライセンス購入といった流れでシステムが成長して、利用者も数十人からそれ以上の規模に増えていくこともあります。

筆者も仕事柄、そのような経緯で成長したシステムを見たことが何度もあります。専門家が作ったシステムと比べると、設計面での問題を持っていることはありますが、現場の知識をもとに試行錯誤しながら機能の強化を図っていけるため、使い勝手ではプロが開発したシステムを凌駕する仕上がりになっているものもあります。

ただ、実際には、この規模に成長するまでの間に遭遇する問題があります。

それは、FileMaker と基幹システムとのデータ連携の問題です。

例えば、すでに基幹システムには、顧客や社員、その他の基本情報がマスタとして登録されているにもかかわらず、FileMaker では、あらためて登録し直さなければならないということになると、数人で使っている間は我慢していても、利用者が増えると、「基幹システムのデータを FileMaker に自動的に登録されるようにできないのか？」という声が大きくなってきます。

ESS の導入手順

ケース 1 で想定した、基幹システムと FileMaker のマスタ情報の連携では、多くの場合では、基幹システム上のデータがマスタ（オリジナル）になりますので、FileMaker からはそのデータを閲覧することができれば十分です。

□ SQL データベース側の準備

1. ESS で使用するマスタテーブルに対して、閲覧可能な権限の設定を行なう。

□ FileMaker 側の設定手順

1. ODBC ドライバのインストール
2. DSN の設定
3. FileMaker 上での外部データソースの定義
4. リレーションシップグラフへの「テーブルの別の名前」の追加
5. シャドウテーブルに対するリレーションやルックアップの設定。
6. レイアウトに必要なフィールドの追加。

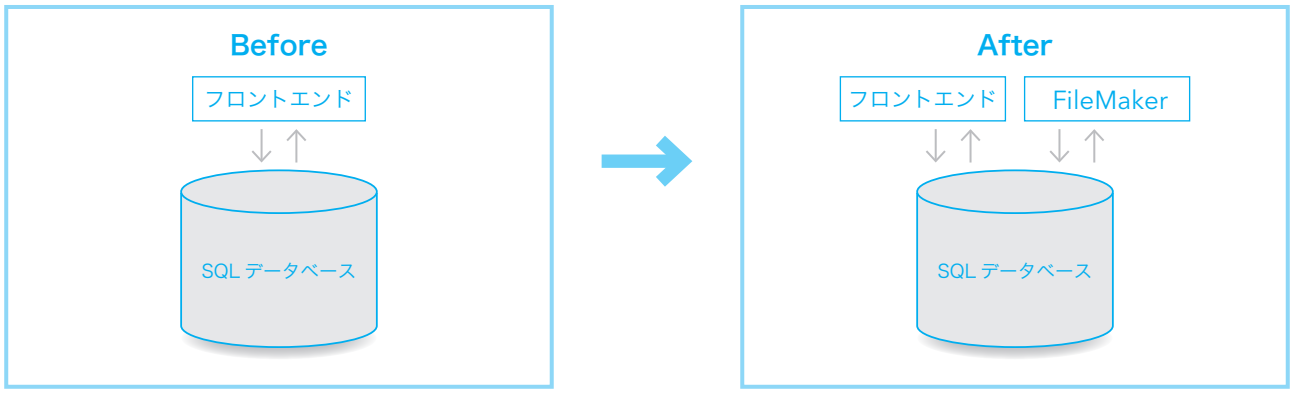
といった流れになります。

ケース 1 は、ESS を導入しやすく、効果も大きい例ですが、下記のような環境では導入が難しい場合があります。

1. サーバやネットワークの負荷から、SQL データベースへの直接アクセスを認めない。
2. セキュリティ的な理由で、SQL データベースへの直接アクセスを認めない。
3. パッケージや CRM などの仕様で、SQL データベースへの直接アクセスができない。

その場合は、他の方法で代替することとのメリット、デメリットを比較して検討することになるでしょう。

ケース２（エンドユーザコンピューティング環境としての FileMaker）



背景

SQL データベース、プログラム言語などによって開発された基幹システムでは、頻繁に変更、機能拡張を繰り返すようなことは稀です。よく「基幹システムは融通が利かない」という意見を耳にしますが、基幹システムは、安定性、信頼性を高いレベルで要求され、そして、業務の標準化にも配慮されていることを考えると、やむを得ない部分もあります。

しかし、そうしたシステム側の事情とは関係なく、ビジネスの現場では、どんどん新しい要求が突きつけられます。新しい情報項目、新しい印刷帳票、新しい集計レポート、さまざまな要求に対して、「こういうわけだから、できません」というだけでは能がありません。

こういう場合は、利用部署がアクセスすることができる情報については、それを使うシステムを提供するのではなく、データへのアクセス権限を提供して、ユーザが FileMaker を使って必要な機能を実現するという方法があります。

以前はこうした取り組みはまれでしたが、最近では、ますます迅速なビジネスの対応力が求められるようになり、数年単位のアップデートといった時間的な区切りでは対応できない事態も珍しくありません。

多くの組織では、利用者のあらゆる要求に応えることができるだけの資源が IT 部署に与えられていないため、全社的な価値観で、優先順位の高いものから順番に対応していくことしかできないのが現実なのです。

もちろん、こうしたシステムの開発や変更を開発会社に依頼することもできますが、開発規模が小さかったり、変更が頻繁だったりすると、コストパフォーマンスは悪くなります。開発会社のコストには、純粋な開発コストだけでなく、打ち合わせや商談、契約、納品、検収といった手続きを行なうための管理コストが含まれているからです。

ESS の導入手順

FileMaker の特徴は、誰にでも分かりやすい、使いやすいデータベースであることです。

ESS は、特別な知識がなくても、SQL データベースに対するリアルタイムで、更新も可能なデータ連携を実現できる機能です。

この長所の組み合わせにより、FileMaker + ESS は、基幹システムのデータを活用したエンドユーザコンピューティングを実現する環境として利用することができます。

ケース２では、必ずしも、既存の FileMaker システムは存在していませんので、ESS を導入するステップの後には実際に必要な機能を開発するステップが存在することになります。

□ SQL データベース側の準備

1. ESS で使用するテーブル / ビューに対して、必要な権限の設定を行なう。

□ FileMaker 側の設定手順

1. ODBC ドライバのインストール
2. DSN の設定
3. FileMaker 上での外部データソースの定義
4. リレーションシップグラフへの「テーブルの別の名前」の追加
5. FileMaker レイアウト上で必要な機能の作り込み。

といった流れになります。

SQL データベースに存在しない情報を追加したい場合には、FileMaker 内でだけ情報を追加することも可能です。

1. 計算フィールド、集計フィールドの追加

2. FileMaker 内のリレーションテーブルによるフィールドの追加

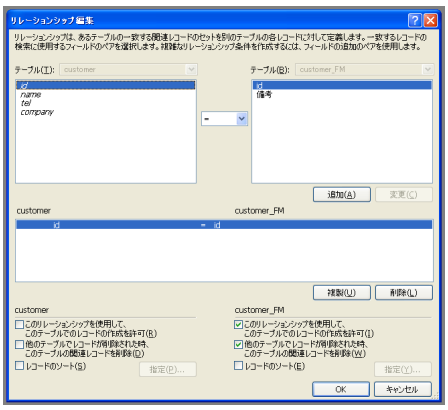
SQL データベース上に、「性」、「名」のフィールドがあり、それに「様」を付加したテキストを出力したいといった場合では、性 & 名 & " 様 " という計算式で計算フィールドを追加することで、SQL データベース側に何も変更を加えずに対応することができます。

小計や総計のカウント、合計などを求めたい時には、SQL の集計関数を使わずに、FileMaker の使い方に沿った集計パート、集計フィールドの追加で対応することもできます（シャドウテーブル内に追加できる計算フィールド、集計フィールドは「補助フィールド」と呼ばれています）。

データを記入して、保存するフィールドを追加したい場合には、少し工夫が必要です。

「customer」というシャドウテーブルに対して、「備考」というフィールドを追加したい場合、まず FileMaker 内に新しくテーブル（ここでは「customer _FM」テーブルとします）を作成して、「備考」フィールドを追加します。

「customer」シャドウテーブルと「customer _FM」FileMaker テーブルの間に、ID フィールドをキーにして、レコードの新規作成を許可、関連レコードの削除をオンにをしたリレーションを定義します。



「customer」というシャドウテーブルに関連づけた FileMaker のレイアウト上に、「customer_FM」の「備考」フィールドを配置します。

これで、「備考」フィールドに値が入った時点で自動的に「customer _FM」テーブルにレコードが追加され、「customer」テーブルのレコードが FileMaker 上で削除されれば、関連する「備考」の情報が削除される設定が出来上がります。

SQL データベース上のデータに FileMaker から更新をかけるフィールドを追加することは、排他制御などの問題を引き起こす可能性もあって、IT 部署や開発会社から歓迎されないこともありますが、この方法であれば、そうした懸念は必要ありません。

ESS 利用上の注意点

ケース１のようにマスタのデータ参照だけの用途であれば、ESS 関連の問題は起こりにくいのですが、ケース２のような用途では、ESS の仕様に起因する問題に遭遇する可能性が出てきます。

問題になりやすいポイントは下記のようになります。

1. 期待したパフォーマンスが得られない
2. サーバの負荷増大
3. ネットワークトラフィックの増大

ESS の特徴のひとつは、「SQL の知識が不要」という点にあります。ユーザは FileMaker 上で通常の手順で操作を行なえば、「あたかも FileMaker 内にあるデータと同じような」操作結果をえることができます。これは FileMaker がユーザの操作を自動的に SQL に翻訳して SQL データベースに問い合わせていることで実現されています。

SQL クエリーへの翻訳が完全自動であるために、SQL を知らなくても SQL データベース上のデータを操作することができるのですが、一方で、FileMaker が生成する SQL をカスタマイズすることができないため、期待したパフォーマンスが得られない場合に打つ手が限られているという制約があります。

パフォーマンス上の問題が生じやすい典型的な原因は下記のようなものです。

1. SQL データベース上で適切な索引が設定されていない。
2. 大量データの操作を伴う処理。
3. FileMaker 独特の機能の利用。
4. その他

まず、索引の問題です。FileMaker では、グローバルフィールド、関連レコードを参照する計算フィールドなどの例外を除けば、索引を設定しなくても、検索を行うことで自動的に索引が作成されます。それに対して、SQL データベースでは、明示的に索引を定義しない限り、自動的に索引が設定されることはありません。

ESS で利用している外部データソースであるか、FileMaker のテーブルであるか、レイアウト上でユーザにはほどんど見分けがつきませんので、特に検索が制限されていない限り、検索モードに入って、検索を実行することができます。

索引は、検索やソートを高速化させる効果だけではなく、レコード更新時の負荷や処理速度の増大などのデメリットもあって、SQL データベースではやみくもには設定しないものですが、FileMaker を使い慣れている人ほど、このギャップにはまりやすいため、注意が必要です。

対策は、検索やリレーションキーなどで利用される SQL データベースのフィールドには索引を設定することですが、前述のように、索引にはデメリットもあるので、場合によっては FileMaker 上で検索を制限することも必要になるでしょう。

大量データのソート、集計を伴う処理の問題は、FileMaker の動作方法を知ることで、理解しやすくなります。FileMaker は、ファイル共有するという形でネットワーク上の複数ゲストが同じデータベースを共有できるようになっています。

索引が効く検索や一部の関数の計算などは、FileMaker Server 上で実行されますが、処理の大半はゲスト側で実行されています。例えば、集計の処理では、FileMaker Server 上で集計された結果が表示されているわけではなく、集計に必要な全データがネットワークを流れて、FileMaker Pro 上で計算、表示が実行されています。

ESS は、「あたかも FileMaker 内にあるデータと同じような」動作を実現するため、SQL データベースとの間でも同じようなデータの転送が行われます。SQL データベース → FileMaker Server → FileMaker Pro という、バケツリレーのようにデータが流れるため、ネットワーク上のデータ転送に時間がかかってしまいます。

この問題への対処は、FileMaker 内ではできるだけ対象となるデータ範囲を絞り込むことです。もうひとつは、問題と

なる処理はできるだけ SQL データベース上で実行させるようにすることです。具体的には、あらかじめ集計結果やソート結果を持つビュー（view）を作っておいて、FileMaker からはそれを参照する構成にすることです。

ESS では、SQL データベースのテーブルだけでなく、ビューを取り込むこともできます。FileMaker 上で大量のデータを処理しなくても済むように、SQL データベース上で処理を済ませるようにすることで、劇的に処理速度を改善することができる場合があります。

FileMaker 独特の機能を ESS 上のデータに対して実行する場合にも、大量データの操作問題と同じような問題が生じやすくなります。例えば、シャドウフィールドに対して、ゆるやか検索の演算子「~」を使った検索を実行するケースを考えてみます。

ゆるやか検索という概念は、FileMaker 独自のもので、SQL に変換することができません。そのため、FileMaker は一旦、該当フィールドの全レコードの値を取得して、キャッシュします。そして、次にそのキャッシュに対してゆるやか検索を実行するような動作を行ないます。

この場合も、大量のデータ転送が生じることと、ESS への問い合わせと FileMaker 内の検索という二段構えの動作になるため、レコード数の多いテーブルに対して実行すると、処理速度が遅くなります。

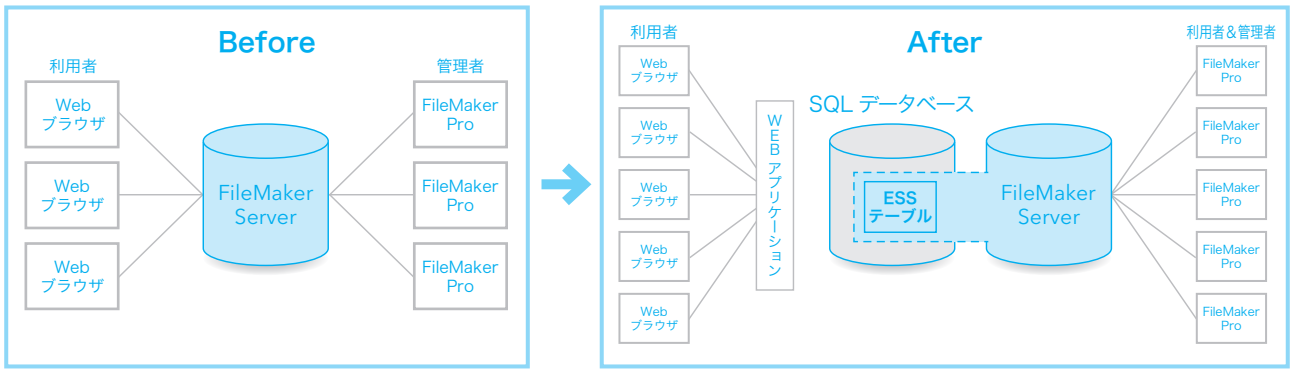
FileMaker 独特の機能に該当するものは他にもいろいろありますが、ESS に直接関係する部分では、補助フィールド（計算フィールド）に対する検索、異なる SQL データベースに属する複数のシャドウテーブルをリレーションで定義して検索、といった操作などがあります。

SQL をカスタマイズできないという ESS の制限から、この現象に対する特効薬はありません。FileMaker の独自機能という利便性を取るか、負荷を伴う操作を制限するか、という取捨選択の検討を行うことになります。

SQL への翻訳が暗黙裏に行なわれるため、すべての問題を予見して、対策を講じることは現実的には困難です。ESS は設定や利用は簡単に行なうことができるというメリットがあるので、実際のソリューションに近い環境を用意して、テストを行い、問題になりそうな機能に対して対策を検討することをお勧めします。

ESS の開発目的は「FileMaker によるシステムに SQL データベースをシームレスに統合すること」という前提があるので、完全に定型化された業務フローは基幹システムとして実装して、それ以外の現場の柔軟な対応が必要な業務については、ESS を利用した FileMaker システムを利用するのが適材適所な使い分けになるでしょう。

ケース３（FileMaker Web 公開のスケールアップ）



背景

FileMaker はバージョン４で Web コンパニオンを搭載して、Web 公開を手軽に実現できる環境を実現して、それ以降も、バージョンアップに伴って、XML 公開、PHP 対応など、Web 公開の機能が強化されてきました。

インスタント Web や FileMaker Site Assistant を使ったテンプレートを自動的に生成する機能などが活用できるため、高度な知識がなくてもデータベースを Web に公開できる FileMaker は、ユーザにとって貴重な選択肢となっています。

しかし、実際に Web サイトを運用していくと、時間の経過とともに、利用者も増加し、利便性の向上のための機能追加も必要になっていきます。回線環境の改善、コンテンツのリッチ化、利用者の高速レスポンスへの慣れなどの要因もあって、今日では Web サイトにはますます高いレベルの処理能力が求められるようになっています。

FileMaker は手軽な Web 公開では優位性がありますが、同時接続数などの制限もあって、同時アクセス数の大きなサイト、大量のデータを処理するサイトなどでは、十分な応答速度を実現することができない場合があります。

こうした場合、ESS を使うことで、Web の応答は SQL データベースに任せて高速レスポンスを実現し、それ以外の部分では FileMaker を使うことで、既存システムの利便性を維持することができます。

また、FileMaker は利用できるサーバ環境に制限があるため、安価なレンタルサーバなどのサービスを使うことができませんが、ESS を使うことで、Web 公開のデータベースは外部のレンタルサーバに置き、社内の FileMaker システムと連携させて使用するといった使い方もできるようになります。

ESS の導入手順

FileMaker の既存データベースとそこで管理しているデータがある場合、ESS への移行には、下記のような準備が必要になります。

1. オブジェクト名（テーブル名、フィールド名）の見直し
2. FileMaker と ESS で扱うデータ項目の整理
3. SQL データベースのテーブル定義
4. ESS の設定
5. データ移行

FileMaker ユーザの方は、分かりやすさを優先して、テーブル名、フィールド名に日本語などのマルチバイト文字を使用する傾向が強いようです。FileMaker 内だけであれば問題ないのですが、多くの SQL データベースやそのデータを利用する Web アプリケーション環境では、マルチバイト文字を使ったオブジェクト名はトラブルの原因になります。

ESS として利用するためのテーブル名、フィールド名などのオブジェクト名では、マルチバイト文字を避けて、半角の英数文字を使用することで、無用なトラブルを避けるようにしましょう。

次に、項目ごとに FileMaker と SQL データベースのどちらで取り扱うのかを決めていきます。基本的に Web から直接参照されるテーブル、フィールドは、SQL データベースに定義します。社内の管理情報として利用するものは、FileMaker に定義することで、FileMaker のメリットを生かし続けることができます。

筆者の関係したケース３のパターンの案件では、1、2 の分類作業は Excel シートで表にして、整理するようにしました。FileMaker ユーザの方は、すぐにデータベースに向き合って作業を開始したがる傾向がありますが、SQL データベースを使い、しかも、情報によって FileMaker を使い分けるような用途では、事前の整理と確認がスムーズな移行のポイントになります。

ここで多くの方が遭遇する課題は、SQL データベース上でのデータ型の選択です。FileMaker では、文字を扱うデータ型は「テキスト」タイプのみですが、SQL データベースでは、固定長文字、可変長文字、大きなテキストを扱うためのデータ型や最大文字数など、分類がより細かくなっています。数字型についても同様です。

安直なのは、その SQL データベースで利用できるもっとも制限の緩い文字型、数字型にしてしまうことですが、最適化という点では、実際に保存されるデータの内容に合わせたデータ型を選択することが望ましいでしょう。

FileMaker の繰り返しフィールドを利用している場合は、そのままでは SQL データベースに移行できないので、リレーションの関係に展開するように設計を変更することになります。

数字タイプのフィールドに値一覧を設定していると、数字が改行区切りのデータ形式で保存されているため、やはりそのままでは SQL データベースに移行できません。リレーションなどに展開して格納できるように設計を変更するか、あるいは文字型に変更してテキストとして扱うようにするなどの対策が必要になります。

データ項目の整理ができたら、SQL データベースでのデータベース、テーブル、フィールドの定義を行います。ESS では SQL データベースの構造定義の操作はできませんので、使用する SQL データベースに対応した管理ツールなどを使います。

最近の管理ツールでは、テーブルやフィールドの追加は、GUI 環境でボタンをクリックしてオブジェクトを追加、それのタイプや名前を変更といった操作を直感的に行なうことができるようになっています。また、この前の段階で整理した項目、データ型の資料があれば、SQL 文に変換して、一気にテーブルの定義を行うことも可能です。

複雑な問い合わせを行う SQL とは異なり、テーブル定義で使う要素は限られていますので、この機にマスターすることも悪くないでしょう。SQL で自在に定義できるようになれば、テーブルの削除、再定義なども簡単にできるようになりますので、データの移行などの場面でも知識が生きてきます。

SQL データベースの定義が済んだら、ESS の定義をします。この手順は他のケースと変わりありません。これで、FileMaker から SQL データベースに対して、データを流し込むことができるようになります。

次に、データの移行です。基本的には、FileMaker にあったデータをエクスポートして、シャドウテーブルにインポートする流れになります。

データに問題がなければ、簡単に移行できますが、実際にはなかなかそうはいかないようです。FileMaker ではデータ型に対する制限が緩いため、数字タイプのフィールドにもテキストを入力できしまいます。一方、SQL データベースはデータ型に対しての検証が厳密に行なわれるので、FileMaker で問題がなかったデータだからといって、SQL データベースにそのままインポートできるとは限らないのです。

SQL データベースのデータ型に合わないデータが混入していると、インポートがエラーになります。FileMaker のエラー表示では、どのレコードのどのデータが原因でエラーになったのかは通知されません。無効な日付や時刻データについては「?」演算子で検索が可能ですが、FileMaker と SQL データベースの型やサイズの相違に起因するものは、目視でチェックするか、意図しないデータを見分けるための計算フィールドを作って、それで調べるような形になるでしょう。

SQL データベースの管理ツールでは、インポートエラーの原因を詳しく通知してくれるものもあるので、管理ツールに抵抗がない人はそちらを使うのも良いでしょう。

なお、データチェックの際、Excel でファイルを開くことを避けるのが賢明です。ゼロの左詰めになっているテキストが数字型と解釈されて、思わぬトラブル（ゼロがなくなる）に見舞われることがあります。ファイルを開く時は、テキストエディタなどデータを変換しないものを使います。

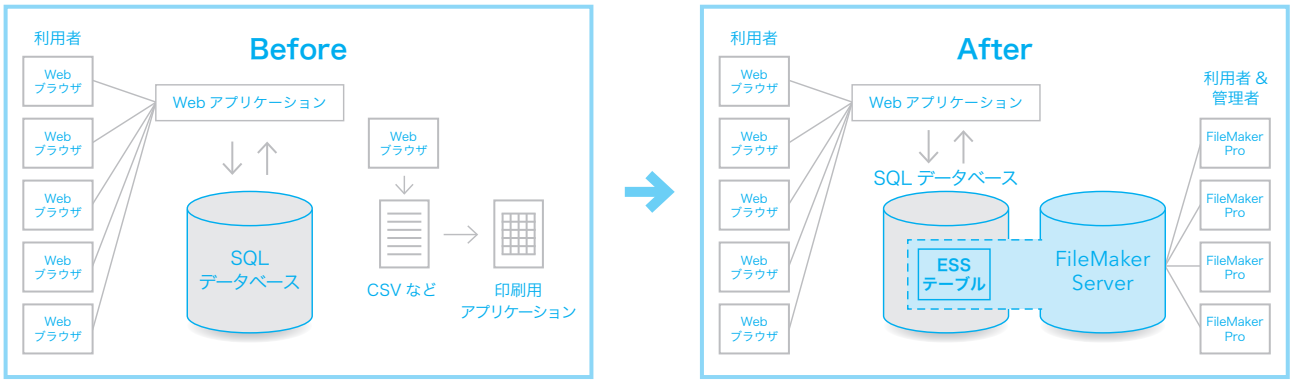
データの移行が無事に済んだら、ESS を含めた FileMaker 上の検証、Web アプリケーションとのデータの連携を確認します。

ESS 利用上の注意点

ケース３も、ケース２と同様に、パフォーマンスへの配慮が必要な構成になります。特に、SQL データベースが外部のサーバにおいてある場合には、経路のネットワーク帯域が小さくなるため、問題が顕在化しやすくなります。対処法については、ケース２と同様になるので、実際の環境に近い環境でテストを行ない、事前に対策を立てておくと良いでしょう。

SQL データベースが外部に置かれている場合には、ネットワーク経路上のセキュリティにも注意が必要です。特に設定を行なわない限りは、ODBC ドライバから先の経路では、クエリーの内容も、結果の応答もプレーンなテキストになって流れています。経路上の盗聴でデータが漏洩しないよう、ODBC ドライバの暗号化オプションを使用するようにしましょう。

ケース４（Web アプリケーションの管理・帳票ツールとしての FileMaker）



背景

ケース４は、SQL データベースの Web アプリケーションに対して、管理用のコンソール代わりに、または、帳票作成用のツールとして FileMaker を利用するパターンです。

データベースと連動する Web アプリケーションは、EC サイト、CMS（コンテンツマネジメントシステム）、ブログなど多岐にわたりますが、コストの節減や対応環境の広さなどのため、現在広く普及している Web アプリケーションの多くでは、データベースにオープンソースの MySQL が使用されています。

多くの Web アプリケーションでは、管理者向けの機能も Web システムとして実装されていますので、マスタの登録、データの検索、集計や帳票の作成なども Web ブラウザで操作を行なうことになります。

ここで問題となるのは、Web アプリケーションと Web ブラウザの組み合わせは、大量のデータを操作する操作や帳票作成などの場面では、必ずしも使い勝手の良い環境ではないということです。

例えば、FileMaker のデータベースであれば、今見ている画面でレコードを絞り込みたいと思えば、検索モードに切り替えて、検索を実行すれば簡単に望む結果を得ることができます。画面に追加したいフィールドがあれば、レイアウトモードに切り替えて、フィールドを追加すればすぐに使い始めることができます。

Web アプリケーションでは、所定の操作の範囲ではそれなりの利便性を実現できても、それを超えた操作をしたい時には限界になってしまうことが多いのです。

印刷帳票については、FileMaker の利便性がさらに生きてきます。印刷帳票が必要な場合、Web アプリケーションでは、ブラウザの画面を印刷をするか、PDF ファイルを出力してそのイメージを印刷するような手法がよく用いられます。

ブラウザ画面の印刷では、スタイルシートを駆使することである程度のデザイン性、柔軟性をもつ印刷帳票を作ることもできますが、罫線やページ送りといった日本で重視される要素を十分に表現することが困難です。

PDF を出力する場合でも、細かい罫線制御など自由度の高いレイアウトを再現することは難しく、またプログラム言語を要するために、初期の要件を満たしたとしても、現場レベルで随時、追加や変更を行なうような運用はほぼ不可能です。

FileMaker と ESS の組み合わせであれば、エンドユーザが、現場の必要に応じて、管理機能を使いやすいように修正したり、印刷帳票を新しく作ったり、カスタマイズすることができるようになります。

ESS の導入手順

設定手順は、基本的なパターンと同様になります。

□ SQL データベース側の準備

1. ESS として接続するための、テーブルやビュー、アクセス権限、アカウントを準備。

□ FileMaker 側の設定手順

1. 接続先データベースに対応した ODBC ドライバのインストール
2. DSN（データソース名）の設定
3. FileMaker 上での外部データソースの定義
4. リレーションシップグラフへの「テーブルの別の名前」の追加
5. 管理画面用レイアウト、印刷用レイアウトの作成

手順に大きな違いはありませんが、ケース４では、FileMaker 内にあるテーブルの大半がシャドウテーブルになること、リレーションもシャドウフィールド同士になる点が特徴的な部分になります。

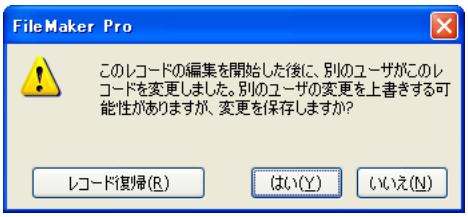
ESS 利用上の注意点

管理画面に、複数のシャドウテーブルにまたがる、大量のフィールドを配置したり、大量の関連レコードを参照する統計関数を含む計算フィールドを配置すると、パフォーマンスの問題が発生しやすくなります。画面ごとに、その画面での操作に必要なフィールドに絞り込むと良いでしょう。

シャドウテーブル上の同じレコードに対して、Web と FileMaker の両方から更新を行うような運用は、できるだけ避けるのが賢明です。例えば、EC サイトでお客様が自分の住所を変更できるようになっていて、FileMaker からも更新を行える環境では下記のような問題が起こる場合があります。

1. Web 上でお客様 A が、自分の住所の変更画面に移動。
2. FileMaker 上で管理者 B が、お客様マスタ画面に移動。
3. Web 上でお客様 A が、お客様の A のレコードの住所を変更。
4. FileMaker 上で管理者 B が、お客様 A のレコードの備考欄を変更。

ここで、図のように、編集の競合があったことは FileMaker から通知されますが、上書きを実行する「はい」ボタンがデフォルトになっているため、レコードの上書きでお客様が変更した住所の変更が失われてしまうおそれがあります。



FileMaker 側からも変更を行いたい場合は、フィールドに直接変更をかけることを許可するのではなく、変更用のフィールドからスクリプトを使って書き写すようにしておくことで、スクリプト内でエラーを検出して、望ましい操作フローに誘導することができます。

他のケースでも FileMaker 以外のインターフェースから行なわれる更新と競合が起こる可能性があるシステム構成では、同じような対処が有効です。

外部に SQL データベースを置く場合は、ケース３と同様、ODBC ドライバの暗号化オプションを有効にするようにしてください。

■ ESS に適していないシステム例

ケース１からケース４までは、ESS 導入によるメリットがあるパターンとして、導入の手順と注意点を説明しましたが、ここでは、あえて ESS には向いていない用途の例も挙げてみたいと思います。

1. 高度なパフォーマンス・チューニングが必要なシステム
2. トランザクション制御が必要なシステム

「ESS の位置づけ」の章で触れたように、ESS が開発された目的は「FileMaker によるシステムに SQL データベースをシームレスに統合すること」です。SQL の知識が不要で、FileMaker 上の操作を自動的に SQL に翻訳するという ESS の特徴は、見方を変えれば「パフォーマンス・チューニングの余地が少ない」ことになります。

大きな効果が期待できるチューニングは、索引設定、ビュー作成といった SQL データベース上での作業になりますので、まずはそれを試してみます。また、必要以上に SQL データベースへの問い合わせが発生しないよう、リレーションの階層、画面構成や操作フローを目的別に絞り込んでみましょう。

それで期待した速度がえられない場合には、ESS ではなく、自分の好きな SQL クエリー文を生成することができる ODBC インポート、［SQL を実行］スクリプトで代替できるかを検討するような順序になるでしょう。

また、FileMaker は［レコードを開く］スクリプトによって、１レコード単位でロックする機能はありますが、commit、rollback による、いわゆるトランザクションの機能はありません。FileMaker テーブルでは、処理速度を犠牲にすれば擬似的なトランザクション制御も可能ですが、シャドウテーブル上では［レコードを開く］スクリプトが効果を持たないので、一括更新の処理で一貫性を保つことは極めて困難です。

FileMaker から SQL データベースに対する一括更新が必要な場合は、ESS ではなく、［SQL を実行］スクリプトでの実現を検討することになります。

ODBC インポート、［SQL を実行］スクリプトは、開発者が SQL を組み立てる必要があります。最適化の余地が大きい代わりに、要求されるスキルレベル、開発の手間は ESS とかなりの違いがあるので、上記２点の要件が不可欠で、ESS だけでは難しそうな場合には、IT 部署や開発会社に相談すると良いでしょう。

■まとめ

高信頼性でも、小回りが利きにくい SQL データベースに対して、エンドユーザでも柔軟に機能の追加や変更が可能な FileMaker。以前は、データ連携の制約もあって、相乗効果を得ることは簡単ではありませんでした。ESS が登場したことにより、データ連携の問題は大きく改善され、基幹システムと FileMaker はシームレスな連携が可能になりました。

IT 部署は、必要なデータと権限を提供すれば、非定型の業務や細かいことはユーザ部署に任せて、より上位のシステム要件に取り組むことができます。ユーザ部署は、IT 部署の意向やスケジュールにやきもきせずに、必要な機能は自分たちで用意することができるようになります。

Web アプリケーションも、不得手な部分は FileMaker でカバーして、より柔軟なシステムにすることが可能になります。Web テクノロジーは便利なものですが、何もかも Web 化しようとするとかえって効率が悪くなる場合もあるのです。

ESS には多くのメリットがありますが、“ 打ち出の小槌 ” ではありません。その特徴と長所／短所を理解して、適切な用途で活用すれば、大きな効果を得ることができるでしょう。

■リファレンス

最後に ESS に関する技術情報を記載している資料をあげておきたいと思います。

□ SQL リソースキット

http://www.filemakertrial.com/ja-JP/sqlkit/default.aspx

□ FileMaker TechNet Resource Center（要 FileMaker TechNet 登録）

http://developer.filemaker.com/technet/resources/

FileMaker TechNet メンバーでない方は、下記からプログラムの詳細やお申込みが可能

http://www.filemaker.co.jp/technet

□ FileMaker データベース開発テクニック 改訂版（書籍）

http://www.key-planning.co.jp/books/239-filemaker-development-technique-r2

■筆者紹介

木下　雄一朗（きのした　ゆういちろう）

株式会社キー・プランニング代表取締役

http://www.key-planning.co.jp/

FBA メンバー、FileMaker TechNet メンバー。コンサルティング会社勤務を経て、ソフトウェア開発会社キー・プランニングを設立。もと尺八演奏家という異色の開発者。クライアントの本当のニーズを見極めたコンサルティング、システム開発を提供することをポリシーに日々奮戦中。

FileMaker 9 Certified Developer

FileMaker 10 Certified Developer

著作

FileMaker データベース開発テクニック 改訂版（アスキー・メディアワークス）

FileMaker データベース開発テクニック（アスキー）

ファイルメーカー Pro7 エキスパートテクニック（アスキー）

ファイルメーカー Pro によるシステム構築（アスキー）

ファイルメーカー Pro アドバンステクニック（アスキー）

プロフェッショナル Web データベースプロデュース（ＳＣＣ）

www.filemaker.co.jp

2010.4