バッチ処理の最適化を考えておられる皆様

2023.10.13

実録！
当社のバッチ処理は、
SQLの順次実行で行っています。

SQLSequencer 概要説明

株式会社オンネット・システムズ
https://www.onnet.ne.jp

ON NET SYSTEMS

本書で述べたいこと

バッチ処理を効率的に開発するツール「SQLSequencer」のご紹介

- バッチ処理をSQL文の順次実行で行う仕組み

バッチ処理は、汎用機的、COBOL的発想の方が効率的と考える

- Java、C#などに代表されるオブジェクト指向言語より、汎用機のCOBOLの手続き言語の方が効率的だった。
- COBOLのデータ処理は、SQL文で代替できる。
- バッチ処理を有効に機能させるためには、JOB管理機能が必要。

COBOL、レガシー移行に最適

- 既存言語（COBOL、VBなど）のソース解読（無理だと思い込んでいる）が重要。
- SQLSでSQL記述。JOB単位で移行、検証が可能。

弊社での利用実績は、10年以上

- 汎用機の生産管理（COBOL800本）の移行、全国規模スポーツジム通販サイトのバック業務。
- 医療機器販売POS、一般受注、在庫管理業務のバック業務。
- 弊社「オンネット統合業務」のバッチは、すべてSQLSで記述（一部、例外あり。後述）

目次

第1章　バッチ処理をどう見るか

- 1.1　バッチ処理を「レガシーシステム移行」の観点で考える
- 1.2　レガシーというネガティブ言葉に、こんな疑問があります
- 1.3　当社は、現状をどう見ているか

1.1　バッチ処理を「レガシーシステム移行」の観点で考える

■バッチ処理とは、画面の無いプログラムです

・画面処理（オンラインと呼ぶ）は、メニューで起動します。
・バッチ処理は、予めスケジュールされた日時で起動します。

■バッチ処理は、大量のデータを効率よく一括処理します

・「オンライン処理は、応答性能重視」に対し「バッチ処理は大量データの効率的処理」です。
・例えば、オンラインが「勤務データの登録」、バッチが「給与計算」となります。

ココの観点で
説明します。

■バッチ処理は、レガシーシステムの移行で重要な考慮ポイントです

・レガシーシステムは汎用機とCOBOLの組み合わせで、システムが作られています。
・これをどう、「安価になった、現代のコンピュータ環境に適用させるか」が社会課題です。

ON NET SYSTEMS

1.2　レガシーというネガティブ言葉に、こんな疑問があります

レガシーシステム（汎用機とCOBOL）だから、新しい時代に対応不能になっている。ホント！？
どんな言語で作られたシステムでも、「そうなる、可能性がある」では？

主な迷信 **（「野球をやったことのない人がダブルプレーを語る」と同じに見える）**	**弊社の考え** **（過去の経験は重要です。有用な考え方を継承すべきです！）**
JavaやC++(オブジェクト指向言語)は、これまでのCOBOL(古典的、手続き言語）に比べて開発生産性が高い。	画面プログラム、通信プログラムなどに、オブジェクト指向言語は有効と考えます。 しかし、事務業務機能は、READ、WRITE、MOVE、COMPUTE程度で書く、COBOL言語の方が生産性が高いのでは？
COBOLは、廃れてしまった。「誰もCOBOLを使っていない」。	現在、世界中のトランザクションの7割は、今も、COBOLで処理されているという分析もあります。 証券、銀行のプログラムの多くは、COBOL、PL/1で記述されていると思います。標準規格化は今も行われています。
汎用機＋COBOLシステムは、個別開発である。だから業務標準である、最新の業務パッケージに、会社業務を合わせて、システム利用することが最適である。	筆者は、40年のシステム構築経験があります。各社の業務機能が同じといういい経験をしていません。何らかのカスタマイズをしています。システムは、業態、規模、統治風土で異なります。「電卓」の様に、同じに使えるわけではありません。
「世界的なシステムパッケージには、開発者が沢山いる」ので、運用保守が、安価で楽。	関与する開発者（プログラマー）は沢山居ると思います。でも、重要なのは、業務機能を理解している人です（でも、商品固有言語の利用ですからC、Javaなどに比べれば少ないでしょう）。 カスタマイズがあれば属人化します。これをどう和らげるかを導入企業が考える必要があります。「カスタマイズ内容」は、利用者側で、把握、理解する必要があります。
COBOLや汎用機を使っているから、システムが複雑で、分からなくなる。	そうではありません。システム維持を導入会社が「丸投げ」していたからです。オブジェクト指向の今風言語でも同じことが起きます。現に起きています（もっとブラックボックス化します）。 再構築の際、COBOLの方が、再構築のための分析が容易です。

1.3　当社は、現状をどう見ているか

■汎用機時代に比べて、システム構築が非効率になったのでは？

（1）JOBという概念が無くなり、画面に機能を集中させている。
　→業務分担が難しくなっている。
　→進捗把握が困難。
（2）業務設計よりも、プログラミングに負荷が掛っている。
　→方法論の複雑さと非継続性（オブジェクト指向の高まり）。
　→「どう絵を描くか」より「どう鉛筆を持つか」に重点。

プログラマー管理が困難

クラスの継続性は大丈夫？

（1）バッチの分離による単純化（オンライン、バッチ分離）

（2）バッチの場合SQL文だけで、業務システムは構築可能

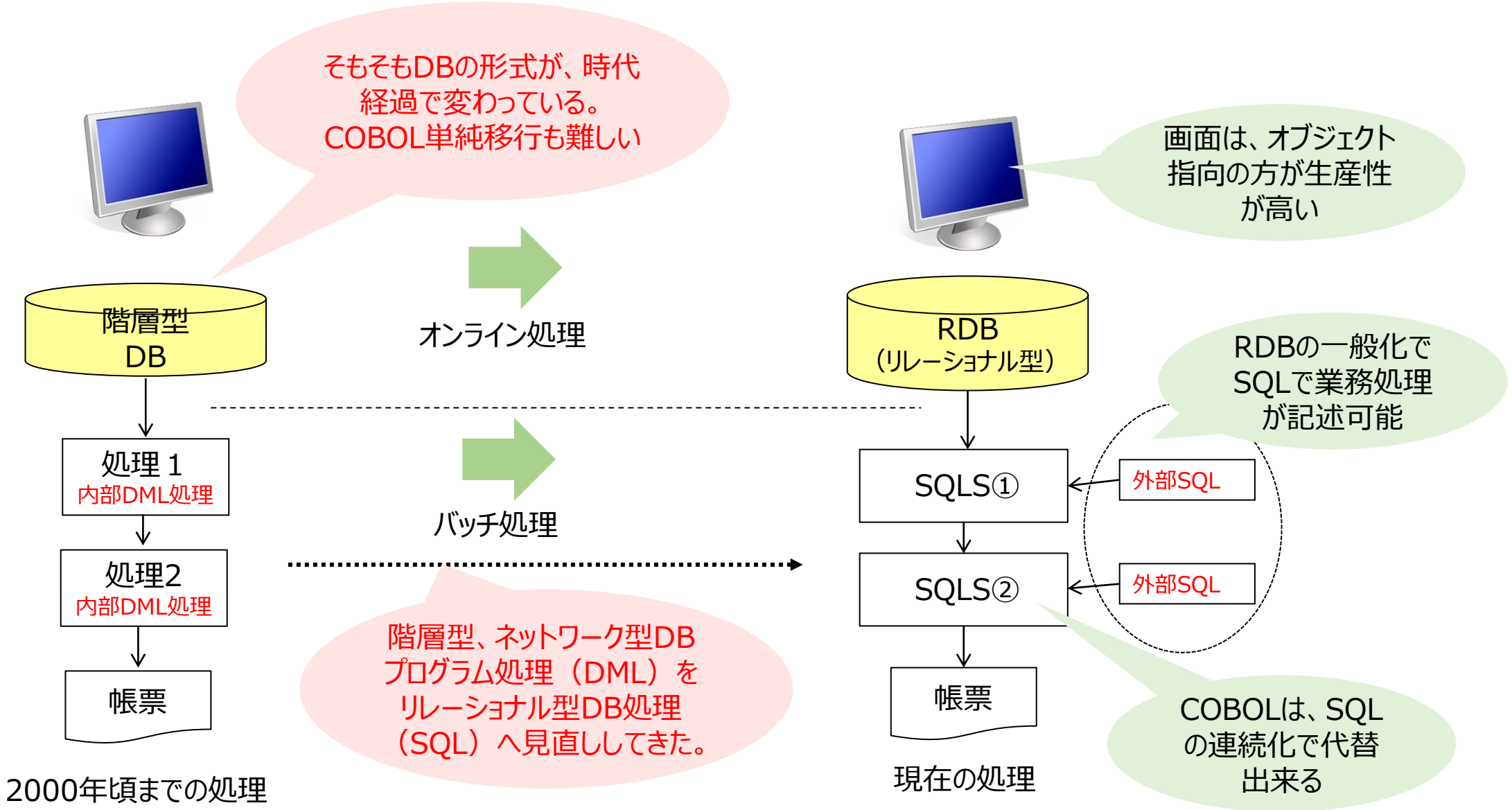
（3）DB製品が替わっても移行が簡単（長期の継続性）

COBOL的発想に戻そう！

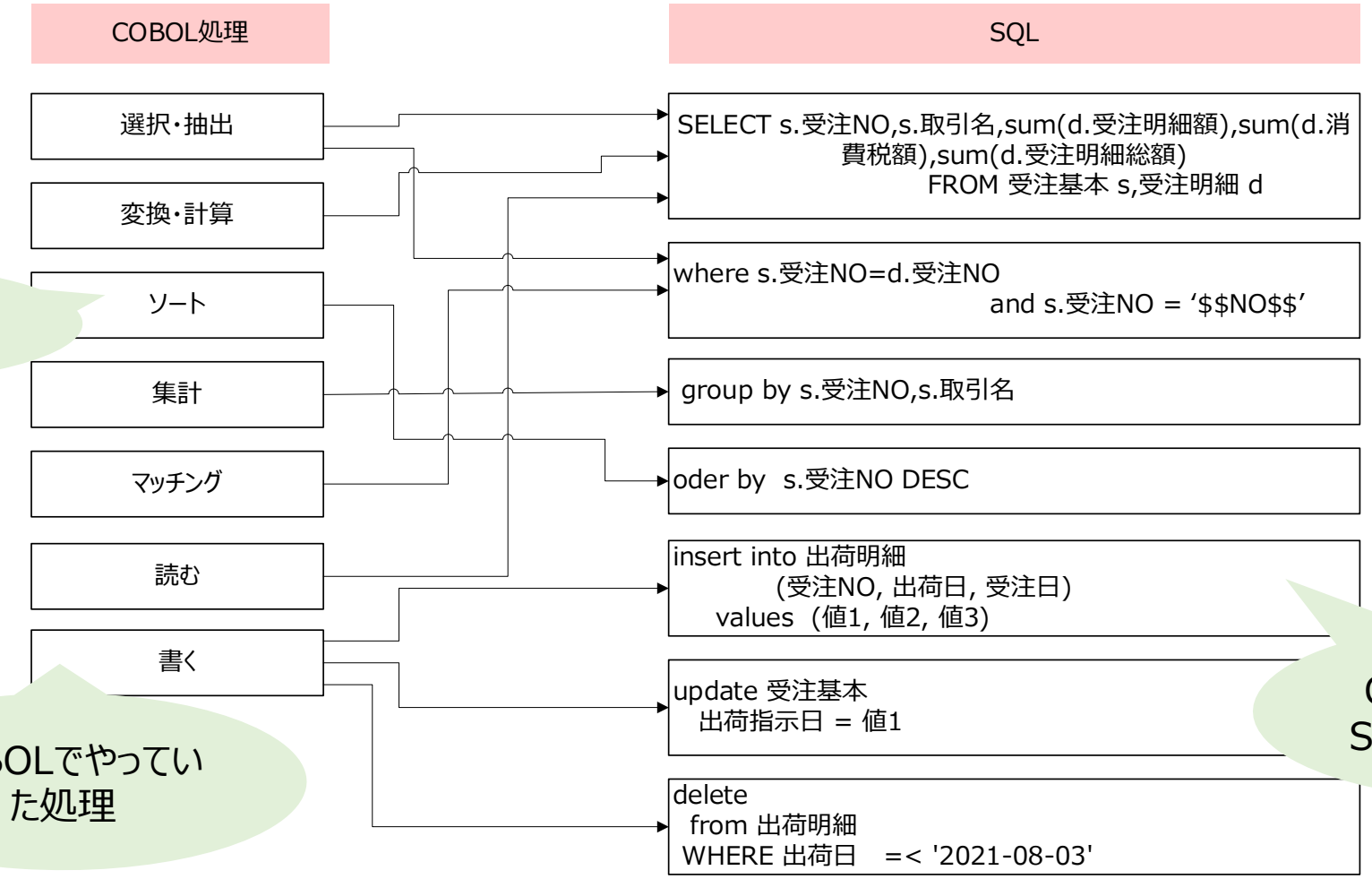
第2章　レガシーシステム移行をSQLSで

- 2.1　経験談！レガシーシステムにSQLSを利用する
- 2.2　経験談！COBOL処理は、SQL文で代替できる
- 2.3　COBOL処理をSQLに代替するための仕組み（これがSQLSだ！）
- 2.4　動作定義XML（COBOL処理は、これで記述できる！）
- 2.5　これまで、SQLSで出来なかったこと
- 2.6　JOB（業務単位）として考える
- 2.7　JOB管理で自動処理させる
- 2.8　関連機能と疎結合すれば、ココまで出来る！
- 2.9　これまで、開発した周辺ツール

2.1 経験談！レガシーシステムにSQLSを利用する



2.2 経験談！COBOL処理は、SQL文で代替できる



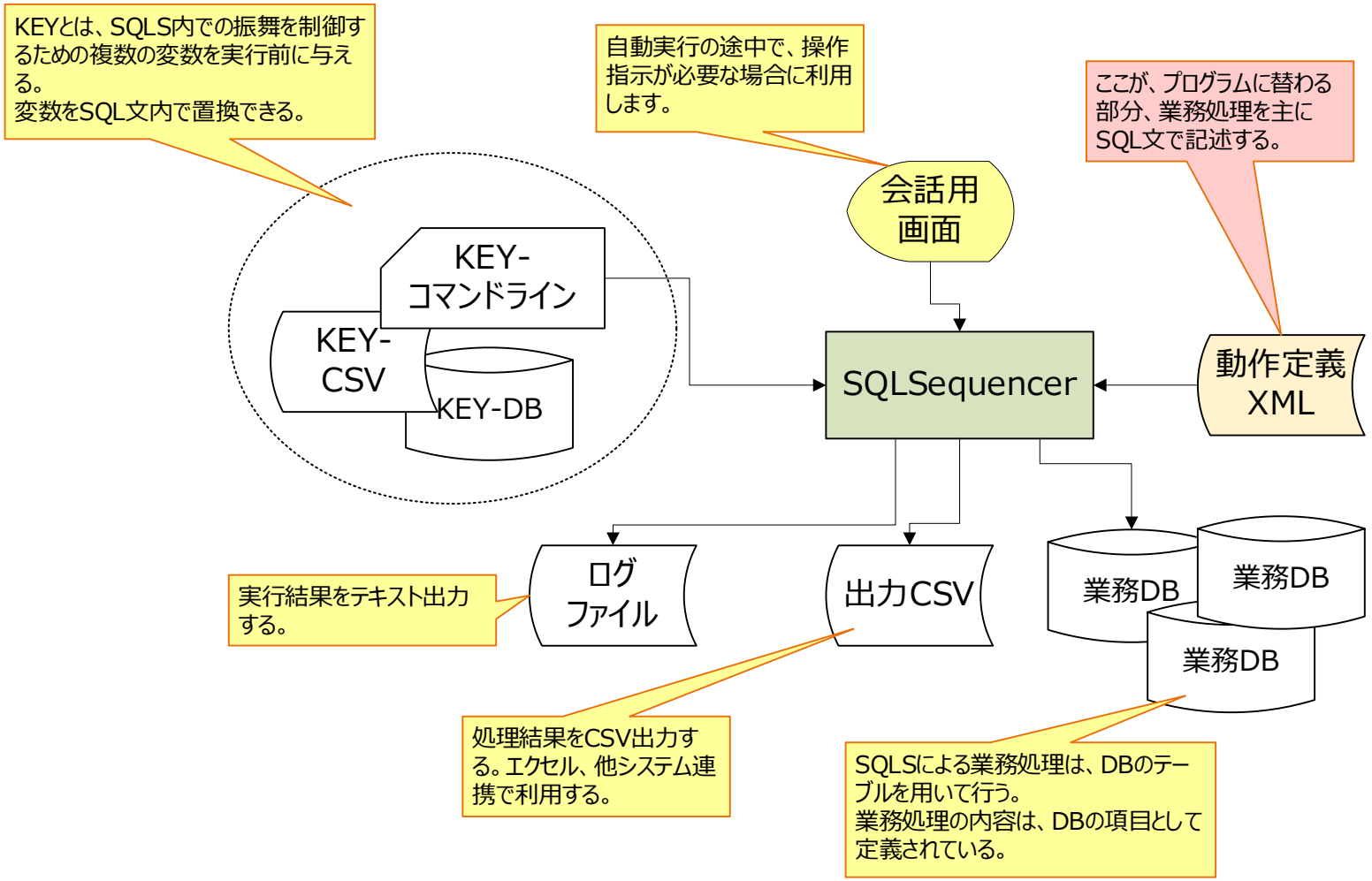
多くは、ユーティリティで行っていた。

COBOLでやっていた処理

COBOL処理は、SQL文に変更可能

COBOL(手続き言語）は、SQL（非手続き言語）に置換できる。

2.3 COBOL処理をSQLに代替するための仕組み（これがSQLSだ！）



2.4 動作定義XML（COBOL処理は、これで記述できる！）

```
<!-- 受注明細に処理対象レコードが存在するかのチェック -->
<!-- 受注伝票NO($$NO$$)を外部から与える。出力はCSVファイルとなる -->
<SQL情報リスト>
  <SQL情報>
  <SQLNO>01</SQLNO>
  <SQLタイトル>受注明細の件数を確認</SQLタイトル>
    <SQL>
    SELECT count(*) 件数,  受注NO          <--$$は置換対象文字列です
           from 受注明細
           where 受注NO = '$$NO$$'
    </SQL>
  </SQL情報>
<!--メイン処理 -->
  <SQL情報>
  <SQLNO>02</SQLNO>
  <SQLタイトル>受注合計のログ表示</SQLタイトル>
    <SQL>
    SELECT s.受注NO,s.取引名,sum(d.受注明細額),sum(d.消費税額),sum(d.受注明細総額)
           FROM 受注基本 s,受注明細 d
           where s.受注NO=d.受注NO
                 and s.受注NO = '$$NO$$'
           group by s.受注NO,s.取引名
    </SQL>
   <SQL実行結果表示>WithHeaderFILE</SQL実行結果表示>
    <SQL実行結果FILE>sample4.csv</SQL実行結果FILE>  <--CSV出力も簡単です。
    <エラー時無視>N</エラー時無視>
    </SQL情報>
</SQL情報リスト>
```

COBOLプログラムの単位

COBOLプログラムの単位

$$文字列は、外部KEYとして与えた内容に置換される。

CSV出力する場合

条件分岐も可能

T-SQLも記述可能
手続き記述も可能

2.5 これまで、SQLSで出来なかったこと

1.WEBサービス、RESTサービスの利用

・JSON利用の売掛債権譲渡サービスとの接続

・JSON利用のSMS送信及びURLフックにるTeams接続など。

【解決策】

・C＃などでプログラム作成し、DBテーブルを介して接続

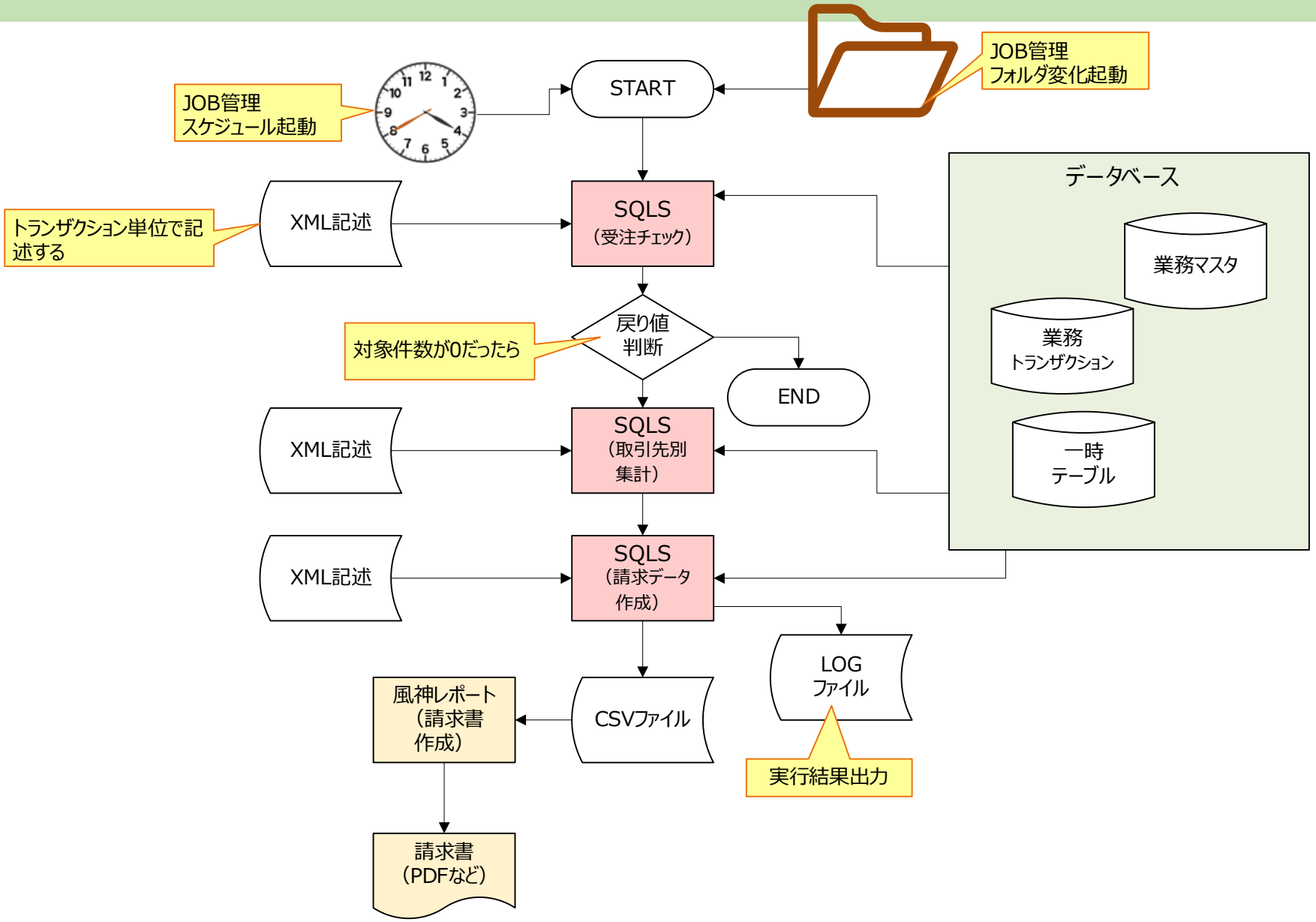
２.複雑な、データ構造での入出力（ヘッダー、トレイラーレコードなどEDIでの利用型式）

・他社、食材受発注サービスへの受注データ接続

【解決策】

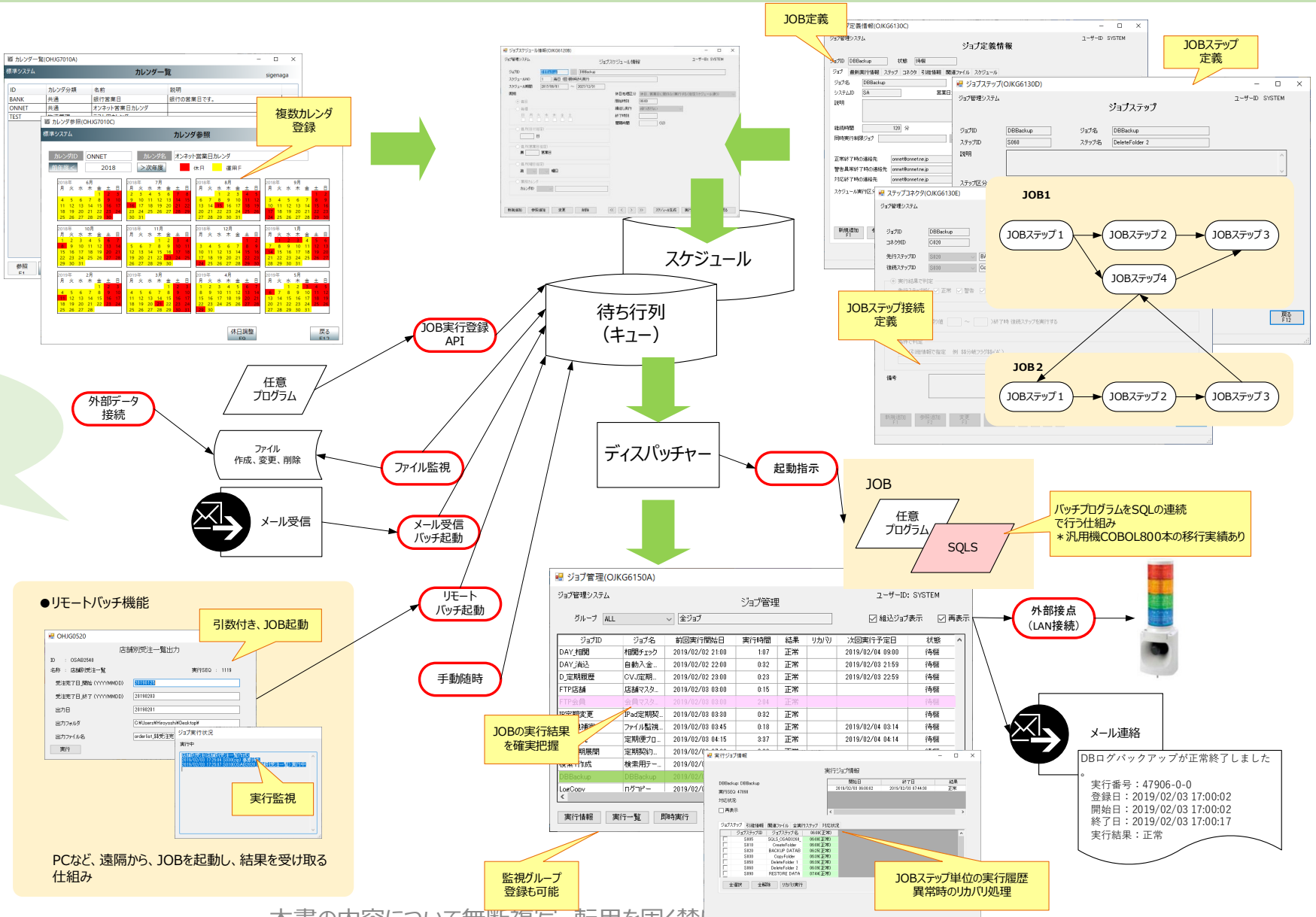
・C#などでプログラム作成し、DBテーブルを介して接続

2.6 JOB（業務単位）として考える



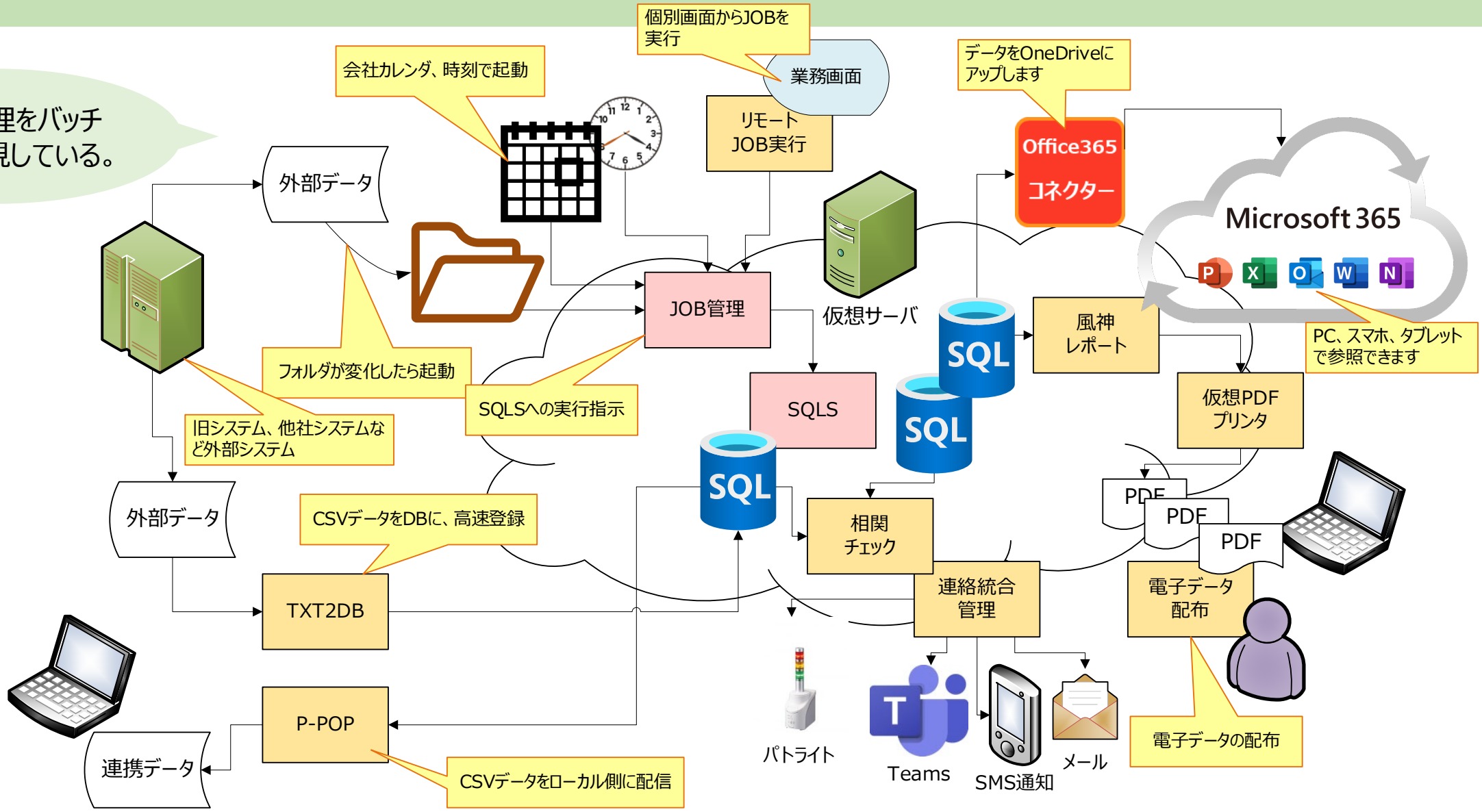
2.7 JOB管理で自動処理させる

大規模利用にはJOB管理が必要、昔のJCL。



2.8 関連機能と疎結合すれば、ココまで出来る！

高度な処理をバッチ処理で実現している。



2.9 これまで、開発した周辺ツール

周辺機能	説　明
TXT2DB	CSVデータをDBテーブルに高速に格納します。データ登録時の会話画面を標準化しています。
P-POP	サーバ側で処理したデータをCSV化してローカル側に転送する仕組みです。
相関チェック	処理した、DBテーブル間の整合性をSQLSでチェックする仕組みです。 例えば、受注データにデータが存在しているが、出荷データに無いなどです。
連絡統合管理	連絡用TBLに、連絡要求を登録すると、メール、Teams、SMS、パトライトなどに通知出力するものです。システムからの連絡・通知をプログラム個別でなく、統合して管理するものです。
風神レポート （他社商品）	作成したCSVを帳票にして出力するものです。紙だけでなく、PDF化してデータ配布する起点としても利用しています。
仮想プリンター	Azure内にコンテナーとして作成しています。帳票データを一括してPDF化します。
電子データ配布	電子ファイル（データ、PDF）を取引先に電子配布する仕組みです。
Microsoft365連携	システム出力された基幹データをMS365（OneDrive）接続すると、スマホ、PC、タブレットなどのモバイル機器と連携出来ます。 ファイルの履歴管理は、電子帳簿保存法にも対応できます。

第3章　SQLS利用のまとめ

- 3.1　当社の利用観
- 3.2　これまでの利用規模観
- 3.3　今、大事と思う事

3.1 当社の利用観

1.バッチ処理は、SQLSで記述（手続き言語を利用していない）

・バッチ処理を手続き言語で、作成していない。

2.画面（オンライン）、バッチ（SQLS）、帳票分離による生産性向上と品質確保

・機能分離を明確に行う事で、設計工程、製造工程が単純化する。
・その結果、単体テスト、組み合わせテストが容易となる。
・機能分離による、処理速度の低下は感じていない。
・業務のモデル化（DB構造）とSQL（非手続き言語）による処理記述は、新入社員の即戦力化に役立った。
・バッチ記述（SQLS）の個人差は、SQLの記述差のみになり、品質が安定する。

3.維持保守作業の容易性

・保守作業はSQLS内の動作定義XMLのテキスト変更で対応できる。
・クラウドと親和性が高い（業務負荷変動への対応（経済性）、DB、サーババックアップ利点（高度なデータ保全））。

4.システム利用の継続性

・業務をDBとSQLで定義することになる。これらの技術は、1970年代より標準化により時代変化が無い。

5.その他

・Windows環境、DBの準備で、開発・運用が可能（非常に安価）

3.2 これまでの利用規模観

1.汎用機、生産管理COBOL（800本）のバッチ処理コンバート

・COBOLプログラム目視分析（2年程度）と移行作業・運用
・「オンネットJOB管理」により、日次で100JOBを運用
・小型クラス汎用機（1990年代、月額レンタル100万円以上）の撤去
・汎用機データとのブリッジシステム構築後、段階的移行
・処理低下の懸念があったが、汎用機1.5時間処理が20分に短縮

結構、
大規模システムでもイケる！

2.全国規模、医療機器販売

・2000年代、初旬の旧ＶＢシステムパッケージ移行
・ＰＯＳ、受注ＰＣで、200台規模
・ＰＯＳ、一般受注の画面部は、Ｃ＃とＷＥＢサービスで実現。後続するバッチ処理は、すべてＳＱＬＳ利用
・上位コンピュータとの連携処理もＳＱＬＳ利用

3.全国規模、スポーツジム商品販売

・全国2000店舗
・電話オペレータでの受注（画面はＣ＃とＷＥＢサービスで実現）をＳＱＬＳでバッチ処理
・出荷、入金（外部コンビニ収納データの連携）、督促などの処理をＳＱＬＳ化
・契約販売、キャンペーン販売の対応をＳＱＬＳで迅速化
・他社タブレットPOSとのREST連携でも利用
・上位コンピュータとの連携処理もＳＱＬＳ利用

4.当社「オンネット統合業務」開発

・画面、バッチ、帳票処理の内、バッチ、帳票出力をＳＱＬＳで記述
・提供先ごとのカスタマイズは、ＳＱＬＳ対応も多い（画面修正の回避）。

ON NET SYSTEMS

3.3 今、大事と思う事

1.基幹業務は、DB構造でモデル化できるということ

・2000年前後、データ指向（ＤＯＡ）という考え方はあった。
・でも、製造業などでの汎用機利用では、階層型ＤＢ利用が主流であった。
・今、関係型ＤＢが主流なんだから、「今こそＤＯＡ」と思う。

SQLSを使って自社業務を
定義、効率化しよう！

2.オブジェクト指向は、バッチ業務処理に有効なのだろうか？

・2000頃から、オブジェクト指向言語が一般化した。
・確かに、画面やネットワークプログラムの作成は便利になった。
・しかし、販売、購買などの業務処理で、継承、カプセル化などを意識する？
・例えば、「入出力データと入出庫処理でクラス化すれば、現金出納にも再利用できそうだけど、だれか考えている？」。

3.DXと言うけれど、自社システムの情報処理化が「前提でショ！」

・日本の経営者は、「情報処理（ＩＴとかデジタル化と言っているけど）を顧客視点（顧客経験）で変化を捉えていない」と言うけれど。
・40年間、情報処理に携わってきたけど、業務システムとの関連で「具体性が分からない」。
・自社業務で扱っているデータの整理（マスタ化）、業務処理（機能）の自動化が重要と考えている。これって、1960年代から同じこと。
・「これからは「マイクロプログラミング」でサービスが、企業の枠を超えて相互利用するようになる」、「10年後、ホントにそうなる？だれが処理の責任を持つ？」。一部はなるでしょけど。
・ＡＩによって時代が変わったと言うけれど、「画像解析、音声解析、音声合成」の分野が殆どだった（情報通信白書）。視覚、聴覚の自動化が業務処理と関係するためには、「汚い手書きの発注メモを自動解析すること」などを連想するが、出来る？もし、実現できたとしても、自社業務が自動化出来ていないと、活用できない。

4.COBOLプログラムは、多く現存している。まずは、「中身を解析する事でショ！」

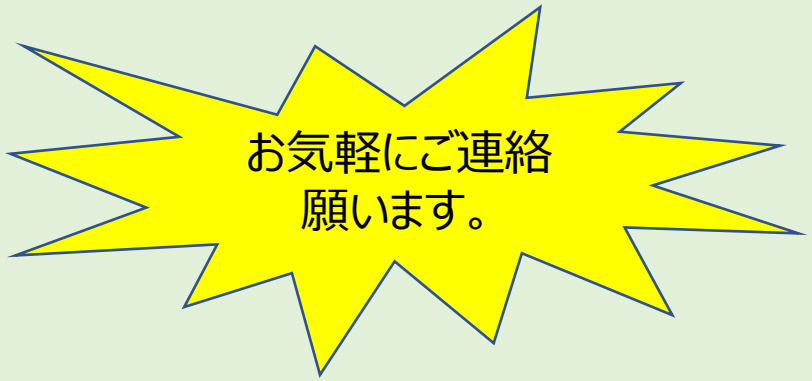
・「旧システムは複雑で解読不能」と言うけれど、じゃ、「分からないま出力結果見てるの？」
・面倒とは思うけど、ＣＯＢＯＬプログラムを読むことは、非常に意味のあることと思う。これが出来なきゃ、新システムへの移行は出来ない。

第4章　最後に

・これまで、「SQLS」について、ご説明しました。

・記述内容は、20年間、弊社がレガシー移行の実施とその運用経験に基づいています。

・ レガシーシステム移行につきまして、お気軽にお問い合わせください。実際画面でデモも可能です。



株式会社オンネット・システムズ
03-5807-5081
onnet@onnet.ne.jp
https://www.onnet.ne.jp