SQL Anywhere対応事例

株式会社バイナル

1.1　Corporate Information　会社概要

社　　名：株式会社バイナル

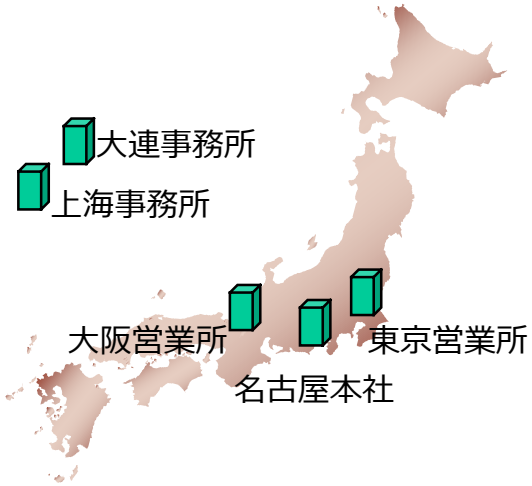
設　　立：1979年　4月

資　本　金：8,000万円

代表取締役　：　岡本　治彦

社　員　数：100名

※バイナル本社



事業内容　：貿易業務管理ｼｽﾃﾑの開発、販売及び保守サポート

■輸出入貿易・販売 購買 在庫管理 業務支援システム

・TOSS-EXPORT System

・TOSS-IMPORT System

・TOSS-J System

■NVOCC通関業者様・港湾物流業者様向システム

・TOSS-LOGIPORT System

■中国版輸出入貿易・販売 購買 在庫管理 業務支援システム

・TOSS-SILKROAD System

事　業　所：本　　社　：名古屋市中区新栄町2-13

東京支店　：東京都中央区八重洲2-4-1

大阪支店　：大阪市中央区北久宝寺町3-5-12

中国事務所：大連、上海

※バイナル東京支店

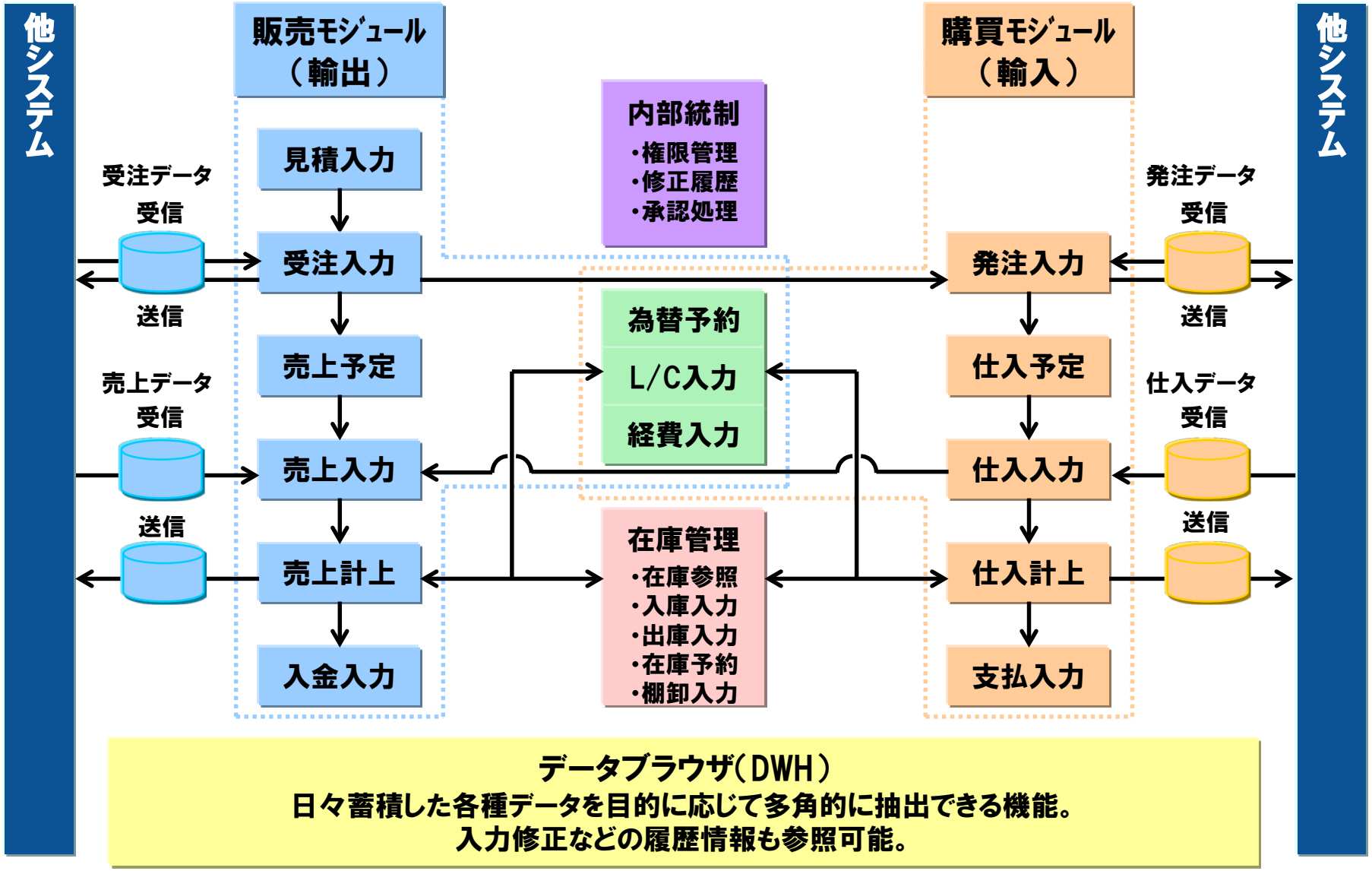
1.2 沿　革

1979年 4月	事務機器販売会社として創業
1982年11月	輸出業務システム「TOSS」の開発に着手
12月	株式会社岡本商会として法人化
1983年12月	「TOSS」の商品化に成功
1986年 1月	東京営業所を開設, 通関業務管理システムを開発（第一号）
1990年 9月	大阪営業所を開設
1991年 7月	株式会社バイナルに社名変更
1996年 3月	「IOMS」リリース開始
1998年 3月	Oracle版通関業務システム「TOSS-DASH」リリース
1999年 4月	東京三菱銀行と業務提携、OEM供給開始
2000年 1月	「TOSS-WEB OPTION」を商品化
2001年 7月	上海・大連に駐在事務所を設置
2002年 4月	TOSSシリーズを中国で本格的に販売スタート
2005年11月	JAVA版TOSSシリーズを商品化
2009年 7月	.NET版 「TOSS-SP・LOGIPORT」を商品化
2010年 7月	資本金を8,000万円に増資
2011年 6月	「TOSS-SP」クラウド版リリース
2013年12月	太陽光ソーラー発電事業を開始
2014年 6月	「TOSS-SP」機能強化版リリース
2015年 7月	SAP社とOEM業務契約締結
2015年 10月	SQL Anywhere版TOSS-SPリリース

※太陽光ソーラー発電 第一号機

※SAP CANADA Waterlooにて

TOSS-SP輸出入貿易業務管理システム機能範囲



データブラウザ（DWH）
日々蓄積した各種データを目的に応じて多角的に抽出できる機能。
入力修正などの履歴情報も参照可能。

SQL Anywhereを標準DBとして採用するまでの経緯

当社が抱えていた課題

最も多い中小規模の導入

当社製品は大規模企業から中堅・中小企業まで非常に幅広く導入いただいておりますが、
その中でも中堅・中小規模への導入が圧倒的に多いという背景があります

そういった中で・・・

規模に適したチューニング作業

各DBのパラメータを初期値のままインストール・稼働させその後チューニングをすることは多くない実態がある。
また状況を監視し適切なチューニングをする場合でも主に拡張を目的としており、
縮小目的でチューニングすることは稀である
結果、規模に対して稼働しているDBがオーバースペックになる可能性もあった

バックアップ、リストアの手順が複雑

バックアップ・リストアの手順がおおげさ(複雑）になりがちで規模に合わない
導入規模に合うシンプルな手法が求められている

システム管理者不在

常にシステム管理者がいる企業ばかりではない。
必然と問合せが増えたり、場合によっては運用の面倒を見てほしいと依頼されることがある
保守コストの増、面倒な作業の増加、結果として双方にメリットが無い

当社が抱えていた課題

価格高騰、稼働環境の多様化

クラウド、仮想化によるライセンスの考え方が複雑（個人の感想）
指名ユーザならまだしもプロセッサライセンスだった場合など、、、

正直難しい！
説明が手間！
手配が手間！

「もっとシンプルにならないの？」
と顧客より指摘されることも

※当社がSQL Anywhereの検討を始めた時期にオラクル社のライセンスの実質値上げ発表
小数ライセンスへの販売が不利になる問題が発生

当社が抱えていた課題

SAP社からの提案

そんな中、SAP社よりSQL Anywhereの提案を受ける

当社が抱えていた課題

SAP社からの提案

規模に適したチューニング作業

実態として各DBのパラメータは初期値のままインストール・稼働させることが多く、
その後チューニングをすることは多くない。
また状況を監視し適切なチューニングをする場合でも主に拡張を目的としており、
縮小目的でチューニングすることは稀である。
結果、規模に対して稼働しているDBがオーバースペックになる可能性もある。

**初期値は最小構成からスタート、稼働後も自動チューニングされるため、
環境に応じた省リソースを実現**

バックアップ、リストアの手順が複雑

バックアップ・リストアの手順がおおげさ(複雑）になりがちで規模に合わない
導入規模に合うシンプルな手法が求められている。

ファイルコピーレベルでのバックアップ・リストアが可能

当社が抱えていた課題

SAP社からの提案

価格高騰、稼働環境の多様化

クラウド、仮想化によるライセンスの考え方が複雑（個人の感想）
指名ユーザならまだしもプロセッサライセンスだった場合など、、、

オンプレミスからクラウド・仮想環境に移行してもライセンスは同じ
SQL Anywhereなら価格面においても優位性があり、規模を問わず幅広いユーザに提案可能

当社が抱えていた課題

SAP社からの提案

そんな中、SAP社よりSQL Anywhereの提案を受ける

実際に評価・開発をしてみよう！
プロジェクト発足を決定

設計・開発から製品リリースまで

開発準備

SAP社のサイトより開発バージョン入手
ハンズオンセミナーへの参加と付属ドキュメントの確認

これだけ

既存の製品開発環境を利用するので、新たに用意をしたのは
SQL Anywhere本体のみでした

設計・開発

前提条件

・SQL Anywhere以外もこれまで通りOracle、SQL Serverと
複数のDBにも対応できること
・外部設定ファイルのみで切り替えができる様シンプルな構造にすること

使用言語、ツール

・.net（SQL Anywhereが提供している.net用のライブラリを採用）
※2016年5月現在JDBCを使った接続も別製品用として開発中
・SQL Anywhereに付属の管理ツールなど

設計・開発

前提条件から想定される構造設計

当社システムは、複数のDBに対応する前提で設計・開発されているので
SQL Anywhereへの接続部と一部個別処理を作るだけで実装できる**はず**

制御イメージ

基本処理部に接続用に必要な共通処理を定義
各DB用に差異がある部分を継承した個別処理で実装

※実際に上図部分の実装のみ行ったあと、次スライド以降の検証・追加修正を実施していきました。

実際に発生した問題、注意した点

DB構築、テーブル作成など

Oracle、SQL Serverで使われているSQL(DDL)との相違検証

↓

当社が生成している他DB用DDLでは動作せず、一部記述変更が必要であることが判明

↓

DB設計書は共通とし各DB用のDDLが発行できるようなツール（自社開発）で
DB間の差異を吸収

※実際にSQL Anywhere用のDDL書式化はSQL Server版をベースに行いました

一部のカラム名においてSQL Anywhereでは予約語になっているものが発生

↓

対象外となるようなスクリプトをセットアップ処理に組み込む様対応

※関数であった場合将来的に使わないものであるかどうか見極めが必要、場合によっては名称変更も検討

実際に発生した問題、注意した点

当社システムから発行する命令部分

システムが発行するSQL命令（SELECT、INSERT、UPDATEなど）に関してはベースが複数のDBに対応するシステムであったためほぼ問題なく動作しました。
それ以外にストアドプロシージャやテーブル上のカラム定義チェックなど各DB毎で構造が異なる部分については、システムのロジック側から呼び出す命令は同じ名称で統一し内部の実体のみ個別に記述することで対応

※個別処理がどれだけあるかで開発工数が大きく変動します。
　当社では開発自体は1ヵ月程度で完了しております。

DBのトランザクション管理の違い

SQL Anyewhere初期値とOracle、SQL Serverの初期値ではトランザクションレベルが異なります。
※当社システムでは、Oracle、SQL Serverの初期値の設定と同等が前提

Oracleと同等のレベルに合わせるスクリプトを用意、セットアップに組込

トランザクションレベル等、既存で利用していたDBとの仕様差異はチェックが必須

実際に発生した問題、注意した点

開発ライブラリについて

SQL Anywhereが提供している.net用ライブラリは、.net Frameworkに標準搭載されているライブラリとは違い、.net Frameworkや、SQL Anywhereのバージョン・リビジョン毎にライブラリがあり、互換性が無い場合もある。

例）実行環境と開発環境とバージョンが異なるとエラーになる

製品リリース後、メジャーバージョンが同じ間はSQL Anywhereのバージョンをある程度固定し出荷する等の運用検討が必要

運用面の検証・評価

バックアップやリカバリについて

最低限でいえばエクスプローラレベルでのファイルコピー・復元をするのみ
また、付属している管理ツールにてジョブ化することが可能
非常に簡単であるため管理者不在の企業でも運用が可能

サーバー間のデータ移行作業

バックアップと同様でファイルを別環境へコピーし、ネットワークサーバー、
パーソナルサーバーのデータベースとして起動すれば移行が完了する
他のDBのように専用の復元コマンド等が無くても良い
※例えば本番環境のデータを別途開発環境で検証する等の作業が短時間で可能

リソース（CPU、メモリ）に対する活動、設定について

当社システムで他のDBと同規模で比較した結果、省リソースで動作することを確認。
省リソースであるため別システムとの共存に対するハードルも低い印象を受けました。
また、環境やデータ規模に合わせて自動チューニングが行われるため、これまでのように
オーバースペックになりにくいのではと期待しています。
※チューニングに関しては自動であるがゆえに大枠の設定に対してはある程度指定が可能でも細かい指定はできない点は注意が必要。

製品リリースまで

要した期間

プロジェクトの発足から開発・検証・製品のリリースまで**半年**と類似開発事例と比較しても非常に短期間で完了することができた。※当社実績

DB特化の機能を利用していなければ、比較的短期間での実装が可能

社内教育・習熟

社員への教育についても数時間の講習を1回、別途個別QA対応のみ
※ある程度Oracle、SQL Serverに対して知識があれば
　置換ながらの説明をしていけば理解しやすいものと思われます。

まとめ

開発準備

・システムのDBリプレース、対応追加のみであれば環境面の準備作業は軽微

設計・開発から出荷まで

・システムが発行しているSQL命令や構築のためのDDL等で独自記述が
どの程度使われているか分析することが必要、開発期間に直結

・他DBとの挙動違いは理解しておく必要があり、設定変更等の検討が必要

・システムとDBバージョンの運用ルールは検討が必要

・ある程度他のRDBMSを利用している開発者であれば短期間の教育で開発可能

運用

・自動設定、自動管理はメリットになることが多いと思われるが、
逆に細かい設定ができないという点は念頭に置いておく。

・バックアップ、リストア等手法が多岐にわたるため社内でベース運用を決める

導入効果

導入効果

価格・運用規模に合わせた提案が可能になった

これまでオーバースペックになりがちだった中堅・中小規模への提案はもちろんのこと

大規模でもコストパフォーマンスに一定のメリットを感じていただける提案が可能になった

稼働後の環境監視、チューニング要否確認等の負荷軽減

完全になくなるわけではないが、監視が必要になる対象案件は減る見込み
顧客側での監視業務の負荷軽減も図れる

バンドル化による個別手配負荷軽減

Oracle、SQL Serverに関してはこれまで別途当社または顧客に個別に手配していただいたが、バンドル化することで手配作業が不要

問合せ対応業務の時間短縮化

Oracle、SQL Serverと違い環境復元作業が短時間且つ容易であるため、作業時間の短縮が図れる

ご清聴ありがとうございました。