Oracle Direct Seminar

ORACLE

VB6から最新.NETへ
～DBアプリ移行方法を徹底解説～

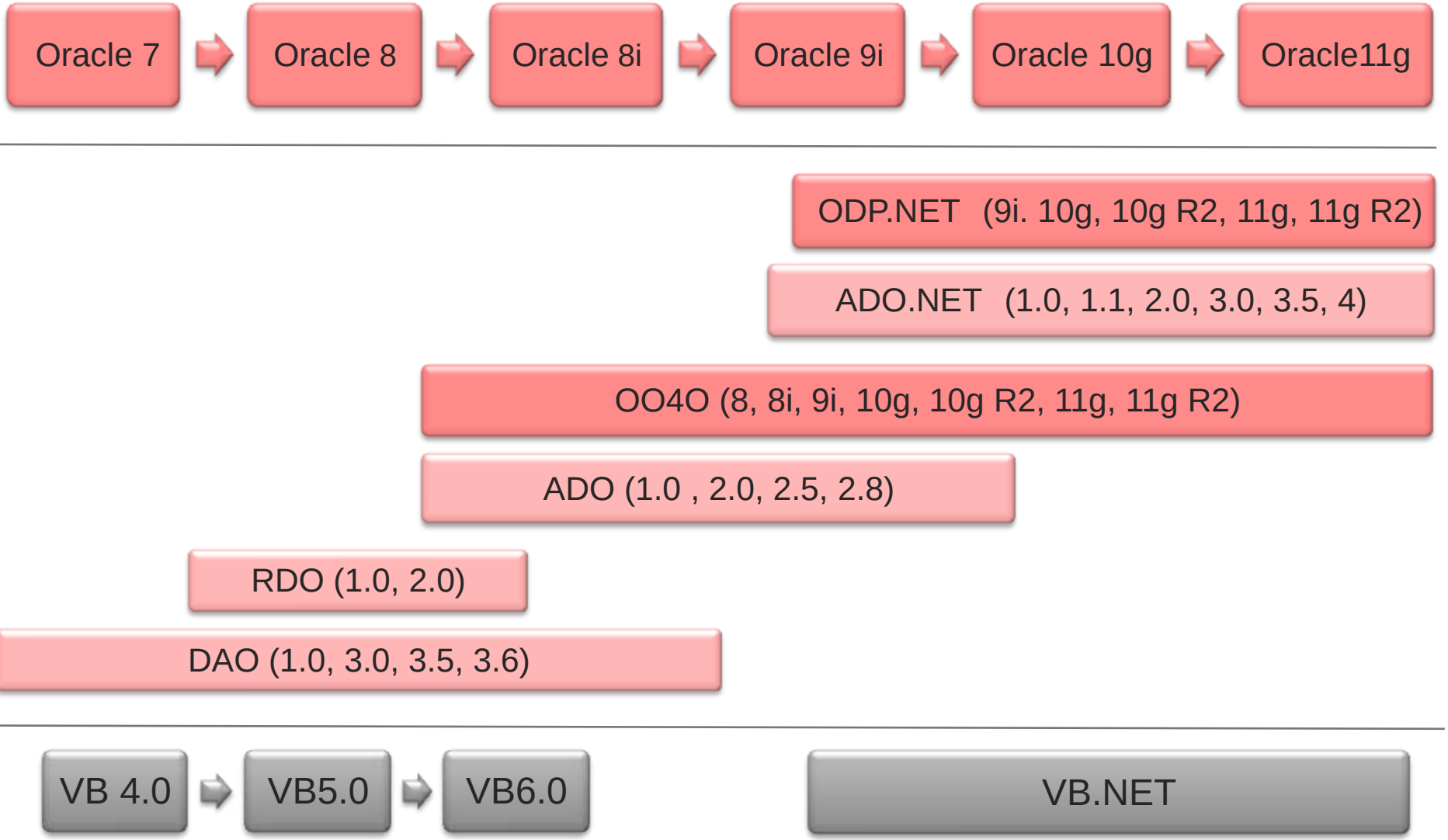
日本オラクル株式会社

以下の事項は、弊社の一般的な製品の方向性に関する概要を説明するものです。また、情報提供を唯一の目的とするものであり、いかなる契約にも組み込むことはできません。以下の事項は、マテリアルやコード、機能を提供することをコミットメント（確約）するものではないため、購買決定を行う際の判断材料になさらないで下さい。オラクル製品に関して記載されている機能の開発、リリースおよび時期については、弊社の裁量により決定されます。

Agenda

- はじめに
- 移行するメリット
- 移行方法の選定
- 移行についての具体的な手順
- 移行後の注意点
- まとめ

データアクセス技術の移り変わり



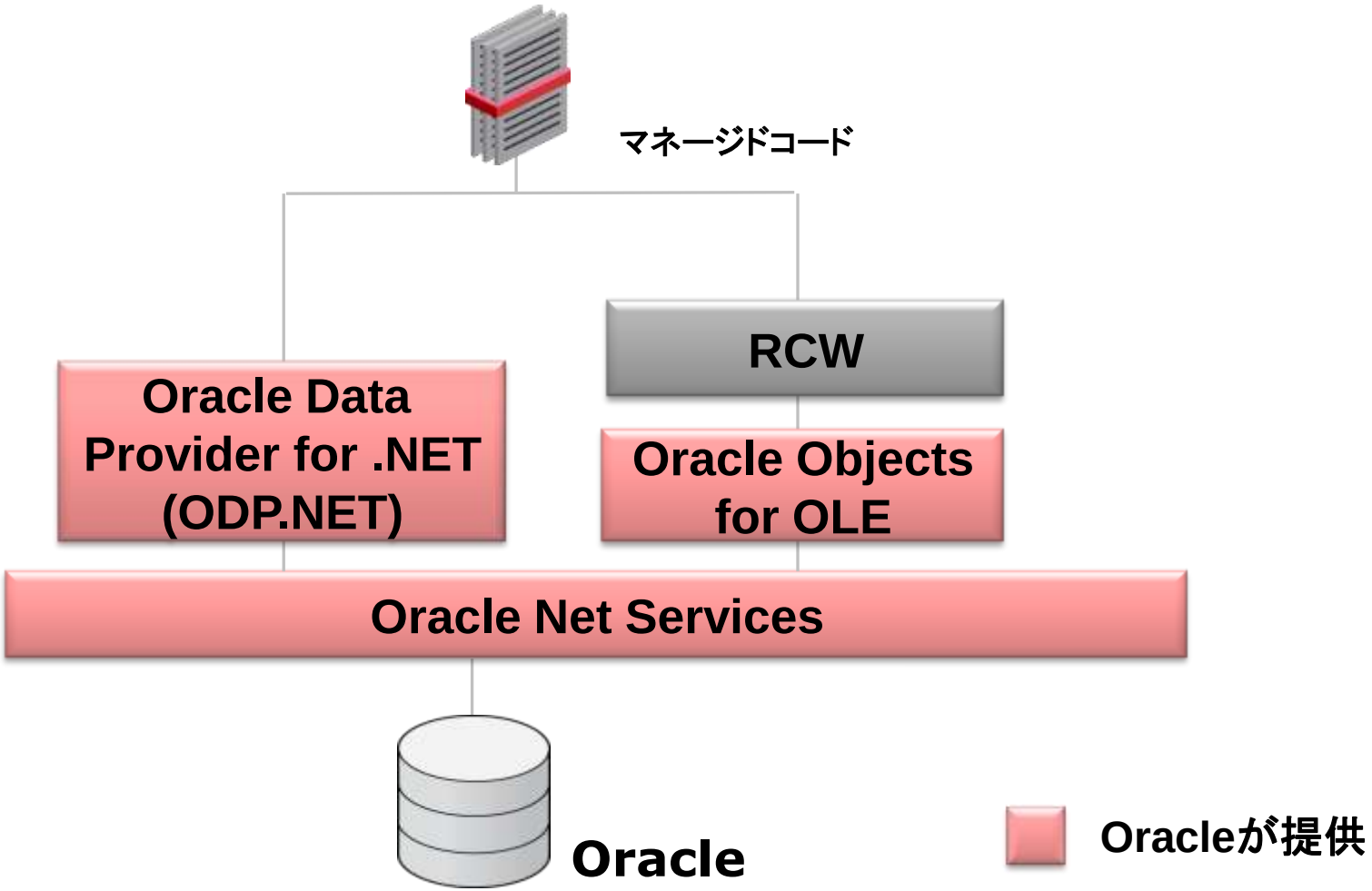
Agenda

- はじめに
- 移行するメリット
- 移行方法の選定
- 移行についての具体的な手順
- 移行後の注意点
- まとめ

ODP.NETにするメリット

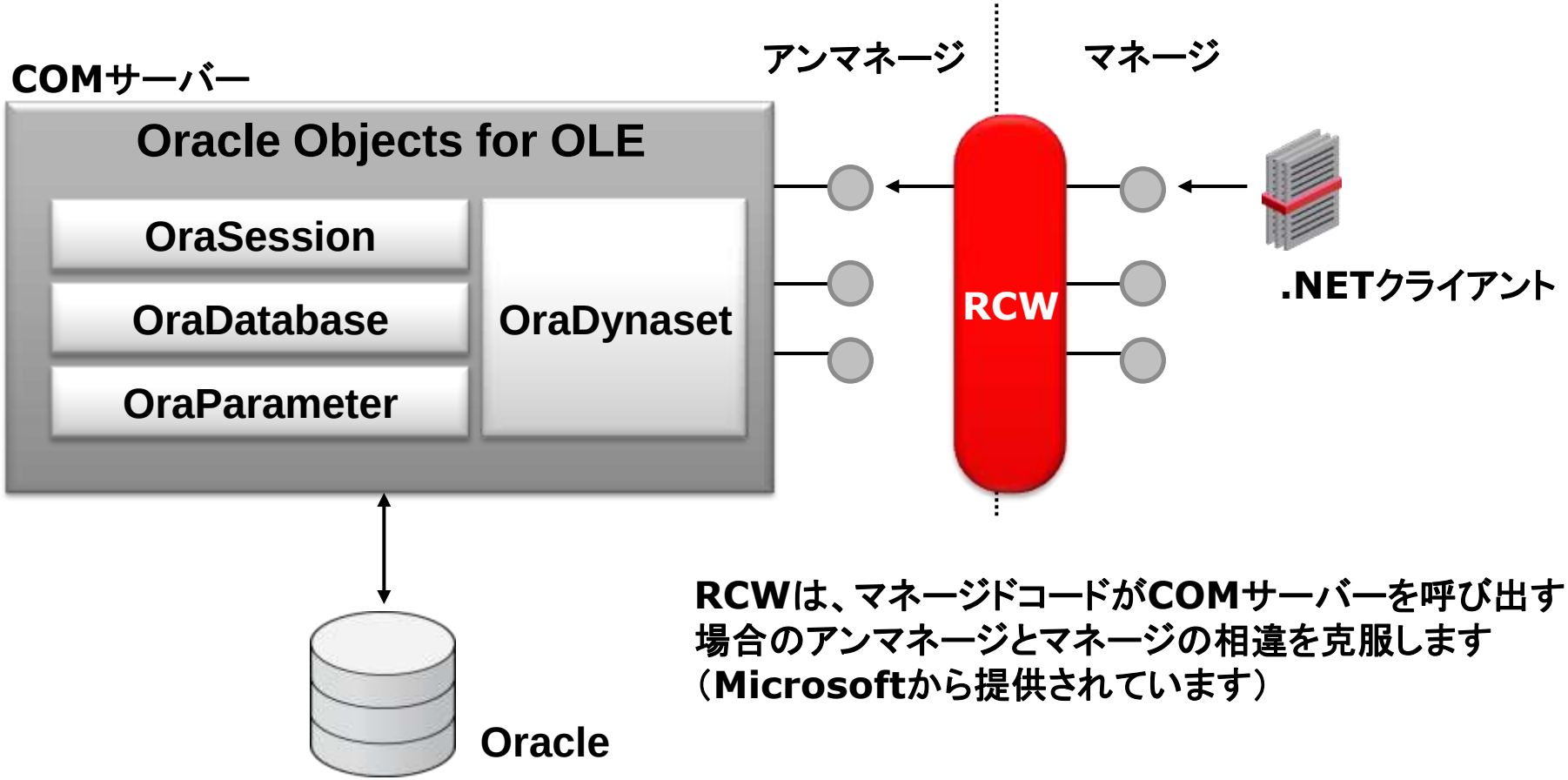
- ODP.NETはADO.NET準拠の高速プロバイダー
- Oracle固有の機能により、RDBMS機能を最大限利用可能
  - XML
  - PL/SQL
  - RAC (Real Application Cluster)
- Microsoft .NETネイティブなAPIとのシームレスな連携（XML）でOO4Oより開発生産性が向上

Oracleへの接続



RCWについて

- RCW(Runtime Callable Wrapper)



RCWは、マネージドコードがCOMサーバーを呼び出す場合のアンマネージとマネージの相違を克服します（Microsoftから提供されています）

OO4OとODP.NETの違い

- ODP.NET
  - Oracleネイティブなドライバー
  - データ・アクセスにブリッジが入らない
  - Oracle固有の機能のサポート
- OO4O
  - COMサーバーである
  - RCWによるデータ変換等のオーバヘッドが発生する
  - Oracle固有の機能のサポート
- ODP.NETの方が高速で最適なアクセスを提供

OO4Oには無い概念

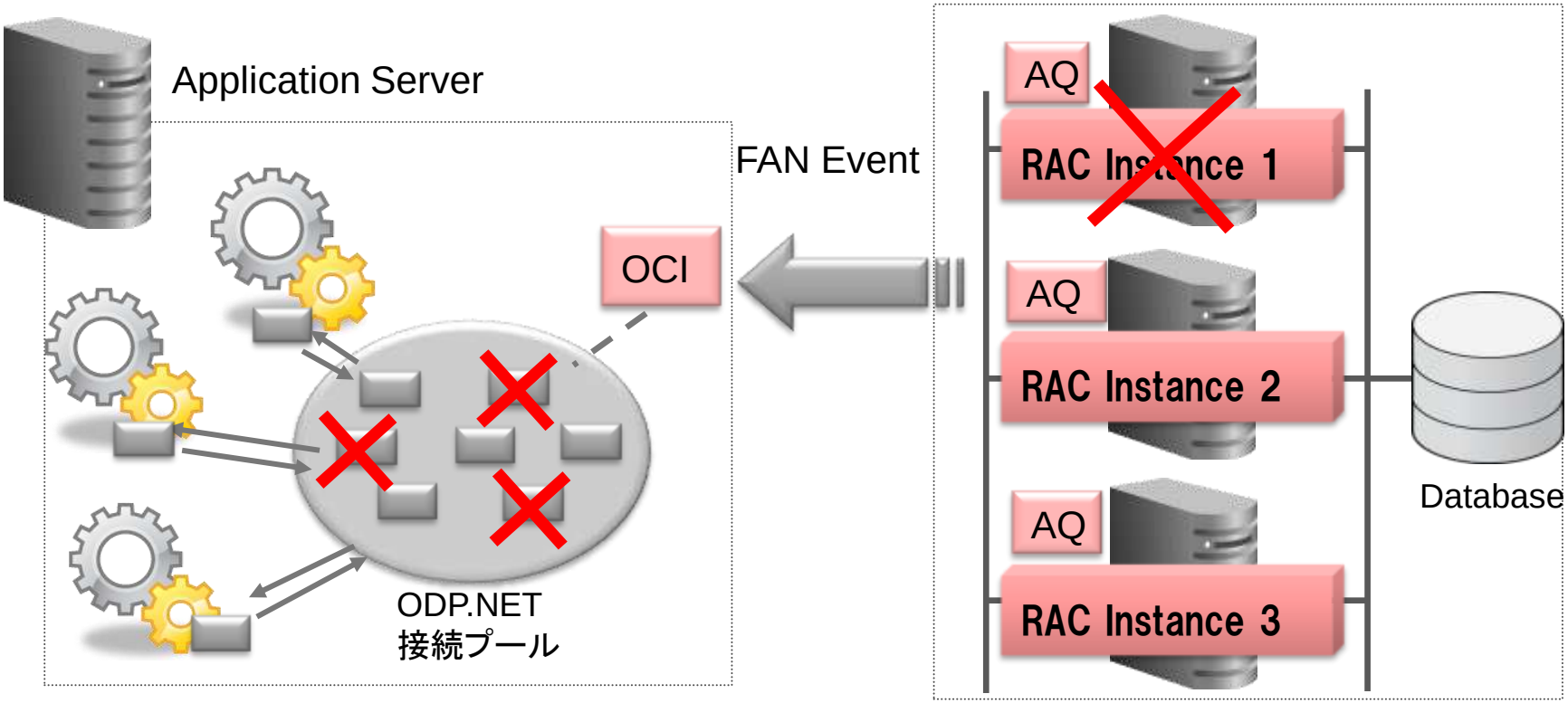
- ODP.NET には、OO4Oが扱わない二つの重要なニーズが取り込まれています

- 1. Web 環境で重要となる、総合的な非接続型のデータアクセスモデル
- 2.「.NET フレームワーク」とのシームレスな統合
  （例：基本クラスライブラリのタイプシステムとの互換）

- Microsoft Visual Basic .NET (VB .NET)でもOO4Oを使用することは可能。

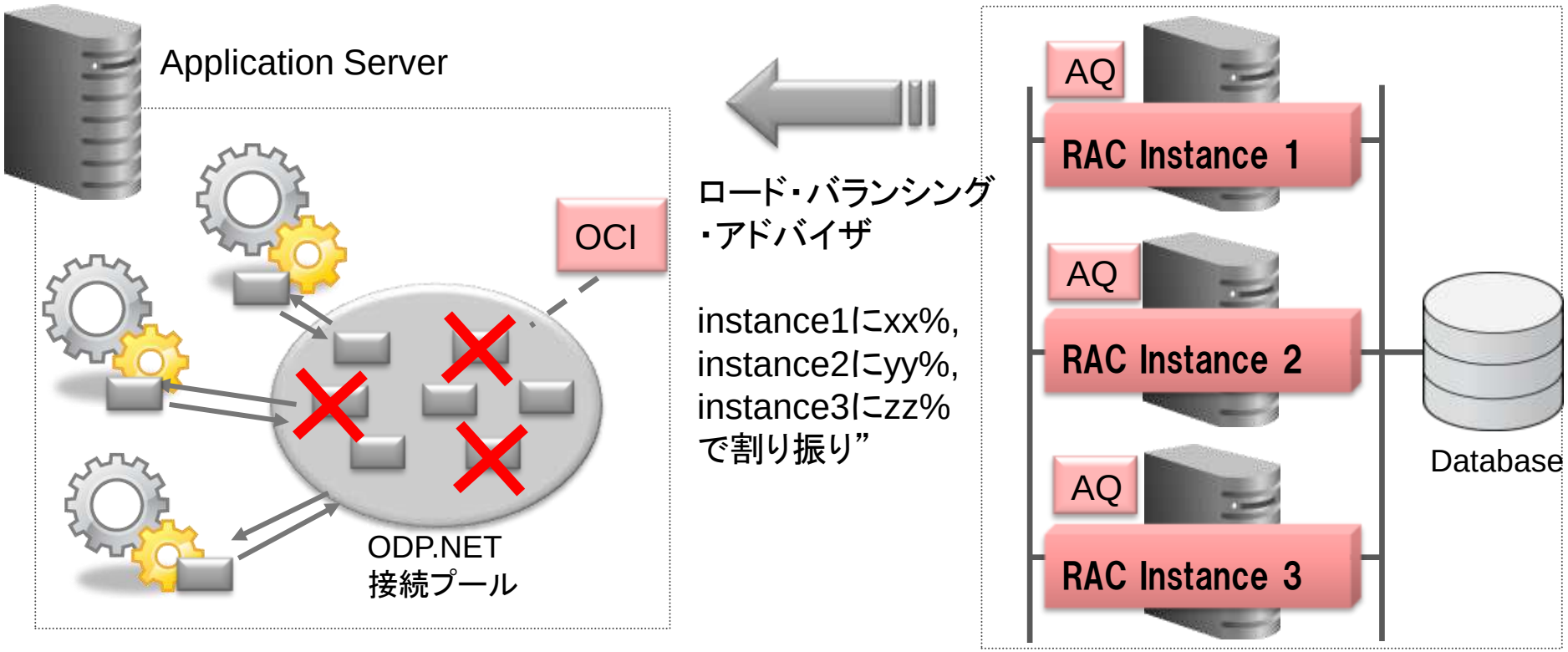
ODP.NETの新機能
高速接続フェイルオーバー

- 障害発生時に失効した接続の除去、接続要求のフェイルオーバー、トランザクションの迅速なロールバックを実現。



ODP.NETの新機能
ランタイム接続ロードバランシング

- クライアント側のコネクション・キャッシュは、FANイベントで指示された割合でアプリケーション・スレッドにコネクションを渡します。
- ODP.NET 10.2以上で使用可能



Agenda

- はじめに
- 移行するメリット
- 移行方法の選定
- 移行についての具体的な手順
- 移行後の注意点
- まとめ

移行方法の選定

- Visual Studio に付属しているアップグレードツールを利用する
  - アップグレードツール：Visual Basic 6.0 プロジェクトを Visual Basic .NET プロジェクトにアップグレードするツール
  - Visual Studio 2010に上記ツールは付属されません。
    - Visual Studio 2008などを経由してVisual Studio 2010に移行

- 移行に際して、コードを.NET用に変更する
  - ツールを利用するのに比較して移行工数が掛るが、メリットは大きい。

アップグレードツールを利用する場合

- Visual Studio に付属しているアップグレードツールを利用する

アップグレードツールを利用する場合

- 移行前のVB6で記述されたコード

```
Dim conn As ADODB.Connection
Dim rs As ADODB.Recordset


Set conn = New ADODB.Connection
conn.CursorLocation = adUseClient
conn.ConnectionString = _
      "Provider=OraOLEDB.Oracle.1;User ID=scott;Password=tiger;Data
Source=orcl;"
conn.Open
Set rs = conn.Execute("SELECT * FROM emp", , adCmdText)
Set Me.DataGrid1.DataSource = rs
```

アップグレードツールを利用する場合

- 移行後のVB.NETコード

```
Dim conn As ADODB.Connection
Dim rs As ADODB.Recordset


conn = New ADODB.Connection
conn.CursorLocation = ADODB.CursorLocationEnum.adUseClient
conn.ConnectionString = _
     Provider=OraOLEDB.Oracle.1;User ID=scott;Password=tiger;Data
Source=orcl;"
conn.Open()
rs = conn.Execute("SELECT * FROM emp", , ADODB.CommandTypeEnum.adCmdText)
Me.DataGrid1.DataSource = rs
```

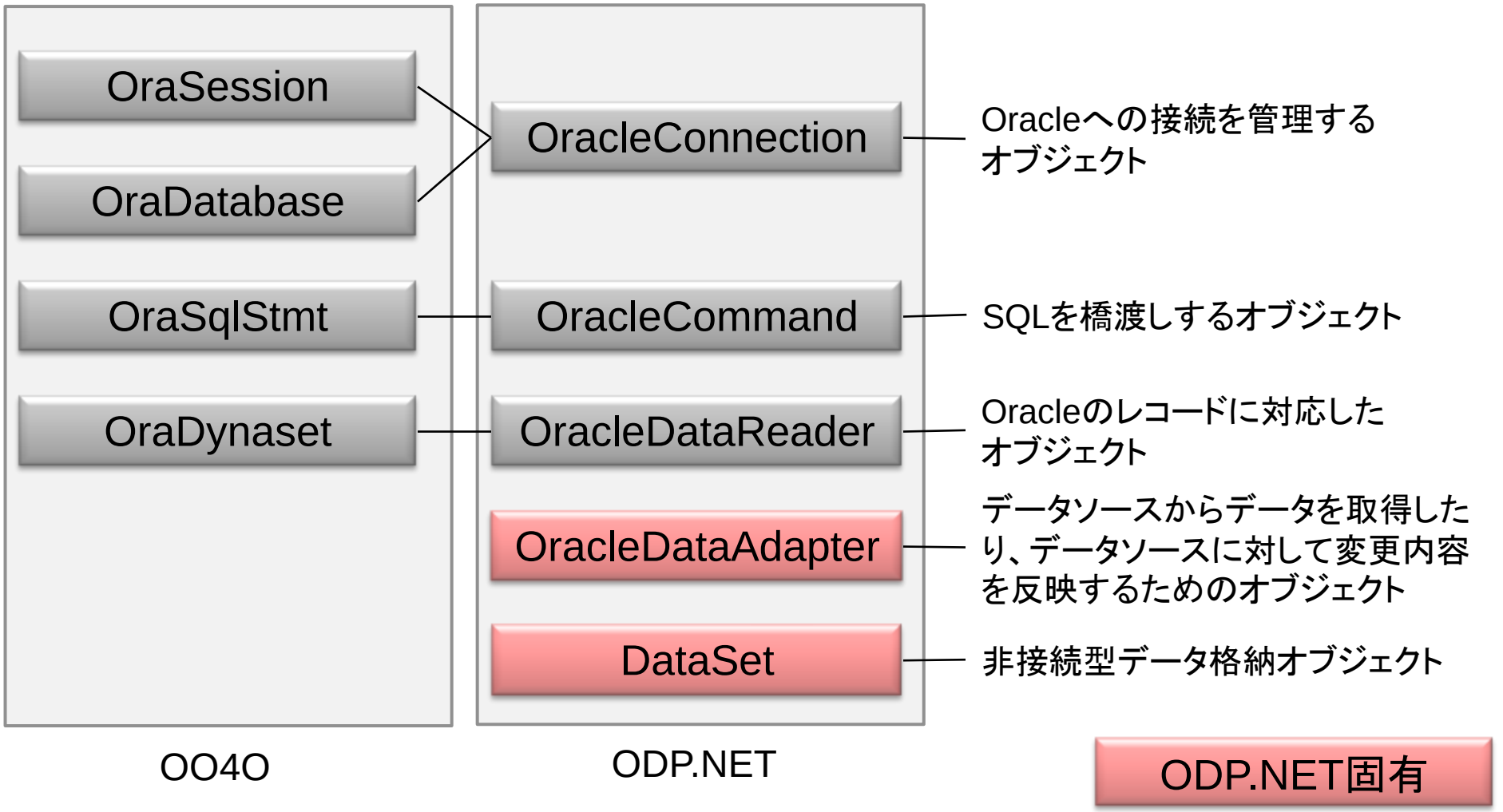
移行方法の選定

- Visual Studio に付属しているアップグレードツールを利用する
  - データベースへのアクセスは、ADO.NET に変換されるわけではない
  - ADO のラッパークラスを使って ADO のまま動作するように変換される
  - ラッパークラスを返して ADO を使うことにより、オーバーヘッドが発生
  - 従来の Visual Basic 6.0 で作成したアプリケーションよりパフォーマンスが低下する可能性がある
- 移行に際して、コードを.NET用に変更する
  - .NET Framework が提供する ADO.NET を使用可能
  - パフォーマンスの向上や、非接続型のデータアクセスと言った .NET のメリットを十分に発揮できる

Agenda

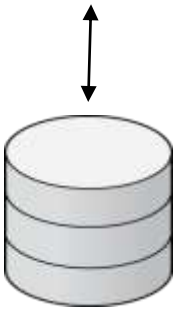
- はじめに
- 移行するメリット
- 移行方法の選定
- 移行についての具体的な手順
- 移行後の注意点
- まとめ

OO4OとODP.NET　使用オブジェクトの比較



OO4OとODP.NET　使用オブジェクトの比較

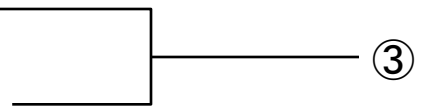
OraSession	OO4O自体との接続を管理するオブジェクト
OraDatabase	Oracle Databaseとの接続情報を管理するオブジェクト
OraParameter	ホスト変数を使用するためのオブジェクト
OraDynaset	Oracleのレコードに対応したオブジェクト



Oracle 9i/10g/11g

OO4Oでのコネクション

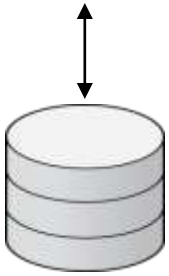
```
Dim OraSession As New OraSessionClass ―――――――――――――――――――――――――― ①
Dim OraDatabase As OraDatabase ―――――――――――――――――――――――――――― ②
Set OraDatabase = OraSession.OpenDatabase( _
    "orcl", "scott/tiger", dbOption.ORADB_DEFAULT)
```



1. OraSessionClassをオブジェクト化して、OracleInProcServerとの接続を確立する。
2. Databaseとの接続を管理するOraDatabaseオブジェクト
3. サービス名、ユーザーID、パスワードを指定してOpenDatabase　メソッドを実行し、OraDatabaseオブジェクトを生成する。

ODP.NETで使用するオブジェクト

OracleConnection	Oracle Databaseとの接続情報を管理するオブジェクト
OracleCommand	SQLを橋渡しするオブジェクト
OracleDataReader	接続型データ格納オブジェクト
OracleDataAdapter	非接続型で使用。データソースからデータを取得したり、データソースに対して変更内容を反映するためのオブジェクト
DataSet	非接続型データ格納オブジェクト



Oracle 9i/10g/11g

ODP.NET固有のオブジェクト

ODP.NETでのコネクション

```
Imports Oracle.DataAccess.Client                                          ①
Imports Oracle.DataAccess.Types

Public Class fmMainMenu
    Inherits System.Windows.Forms.Form
     (略)
    Private Sub DbConnect()
        Dim conn As New OracleConnection                                  ②
        conn.ConnectionString = "Data Source=orcl;User id=scott;Password=tiger"  ③
        conn.Open()                                                       ④
    End Sub
```

①ODP.NETを修飾を行わずに使用できるようにImportsステートメントを実行
②OracleConnectionインスタンスの生成
③サービス名、ユーザーID、パスワードをConnectionプロパティに設定
④Openメソッドを実行して、接続を確立する

tnsname.oraを使用しない接続

ODP.NET ONLY

```
Dim cnn As New OracleConnection
Dim sb As New System.Text.StringBuilder

sb.Append("User Id=scott; Password=tiger;")
sb.Append("Data Source=(DESCRIPTION = (ADDRESS_LIST = ")
sb.Append("(ADDRESS = (PROTOCOL = TCP)(HOST = localhost)")
sb.Append("(PORT = 1521)))(CONNECT_DATA = (SERVER = DEDICATED)")
sb.Append("(SERVICE_NAME = orcl)));")

cnn.ConnectionString = sb.ToString
cnn.Open()

MsgBox("Connect OK!!")
cnn.Close()
```

tnsnams.oraの内容をそのまま記述

オペレーティング・システム認証と特権接続

ODP.NET ONLY

```
Dim cnn As New OracleConnection

cnn.ConnectionString = "User Id=/;Data Source=orcl;DBA Privilege=SYSDBA" ――― ①
cnn.Open()

MsgBox("Connect OK!!")
cnn.Close()
```

- ConnectionString属性のUser Idを / に設定することにより、データベース・ユーザーの認証にWindowsユーザー・ログイン資格証明を使用できます。また、DBA Privilege属性を介してSYSDBA権限またはSYSOPER権限のいずれかを使用してOracleデータベースに接続できます。
- ODP.NET 11.1.0.6.20からOS認証を使用してログインした場合でもコネクションプールが有効

パスワードの期限切れ

ODP.NET ONLY

```
Dim cnn As New OracleConnection


cnn.ConnectionString = "User Id=scott/tiger;Data Source=orcl"
Try
    cnn.Open()
Catch ex As Exception
    cnn.OpenWithNewPassword("panther")
End Try

MsgBox("Connect OK!!")
cnn.Close()
```

新規パスワードpantherで接続するために、OracleConnectionのOpenWithNewPasswordメソッドを使用しています

OO4Oでのコネクションプール

```
Set OraSession = CreateObject("OracleInProcServer.XOraSession")
OraSession.CreateDatabasePool 2, 40, 200, "ORCL", "SCOTT/TIGER", 0 ―――――― ①
Set OraDatabase = OraSession.GetDatabaseFromPool(100) ―――――――――――――― ②
```

- ①CreateDatabasePoolメソッドを使用して、OracleDatabaseとの　コネクションをプール

（初期プールサイズ 2, 最大プールサイズ 40, タイムアウト 200秒で設定）

- ②　作成したコネクションプールからOracleDatabaseへ接続（Wait Time100秒で設定）

ODP.NETでのコネクションプール

```
Dim sConn As String = _
    "User Id=scott;password=tiger;Data Source=orcl;pooling=true;" & _
    "min pool size=2;max pool size=40;connection timeout=200"
Dim cnn As New OracleConnection
cnn.ConnectionString = sConn
cnn.Open()
```

①

- ①コネクション文字列にpooling=trueと記述

（初期プールサイズ 2, 最大プールサイズ 40,タイムアウト 200秒で設定）

- ODP.NETではコネクションプールはデフォルトで有効

OO4OとODP.NET　コネクションの違い

	OO4O	ODP.NET
コネクションプール	コーディングが必要	デフォルトで有効
オープンモード	読み取り専用モードの設定などの設定が可能	使用するオブジェクトで接続型／非接続型を切り分け

OO4Oでのデータアクセス

```
Dim OraDynaset As OraDynaset
  Set OraDynaset = _                                                    ①
    OraDatabase.CreateDynaset("SELECT empno,ename FROM emp", ORADYN_READONLY)
  Do Until OraDynaset.EOF
    Debug.Print OraDynaset.Fields("empno").Value + " / " + _            ②
      OraDynaset.Fields("ename").Value


    OraDynaset.MoveNext
  Loop
```

①OraDatabaseオブジェクトのCreateDynasetメソッドを実行して、OraDynasetオブジェクトを生成

②OraFieldオブジェクトからフィールドの値を取得

ODP.NET 接続型データアクセス

```
Dim conn As New OracleConnection
conn.ConnectionString = _
  "User ID=scott;Password=tiger;Data Source=orcl"
conn.Open()
Dim cmd As New OracleCommand                                  ①
cmd.Connection = conn                                         ①
cmd.CommandText = "select empno, ename from emp"              ①
Dim rdr As OracleDataReader                                   ②
rdr = cmd.ExecuteReader                                       ②
Do While rdr.Read                                             ③
  Debug.WriteLine(CStr(rdr("empno")) + " / " + _              ③
    rdr("ename"))                                             ③
Loop                                                          ③
rdr.Close()                                                   ④
cmd.Dispose()
```

①OracleConnectionオブジェクトを OracleCommandオブジェクトの Connectionプロパティに設定し、CommandTextにSQLを設定する。

②OracleCommandオブジェクトの ExecuteReaderメソッドを実行して、OracleDataReaderオブジェクトを生成

③OracleDataReaderオブジェクトの Readメソッドの実行結果がTrueの時には、レコード取得が出来たということなので、デバックウィンドウに値を表示する。

④OracleDataReaderをCloseします。

Multiple Active Result Sets "MARS"

```
Dim cnn As New OracleConnection
Dim cmd As New OracleCommand

cnn.ConnectionString = "user id=scott;password=tiger;data source=orcl"
cnn.Open()
cmd.Connection = cnn

cmd.CommandText = "Select * from emp"
Dim rdr1 As OracleDataReader = cmd.ExecuteReader

cmd.CommandText = "Select * from dept"
Dim rdr2 As OracleDataReader = cmd.ExecuteReader

rdr1.Close()
rdr2.Close()
cnn.Close()
```

複数のOracleDataReaderを同時に開く事が可能

OO4OとODP.NET　データアクセスの違い（接続型）

	OO4O	ODP.NET
更新	**更新可能**	**読み取りのみ。更新不可**
読み取り	**前方・後方移動が可能**	**前方のみ**

ODP.NET 非接続型データアクセス

```
Dim cnn As New OracleConnection
Dim cmd As New OracleCommand
Dim dsList As New DataSet
Dim iCnt As Integer

cnn.ConnectionString = _
   "user id=scott;password=tiger;data source=orcl"
cmd.Connection = cnn

cmd.CommandText = "Select * from emp"
Dim adp As New OracleDataAdapter(cmd)
adp.Fill(dsList, "EmpList")

With dsList.Tables("EmpList")
   For iCnt = 0 To .Rows.Count - 1
      Debug.WriteLine(CStr(.Rows(iCnt).Item("empno")) + " / " + _
        .Rows(iCnt).Item("ename"))
   Next
End With
```

OracleDataAdapterのFillメソッドを使用しDataSetオブジェクトへデータを格納します

OO4OとODP.NETのコードの比較 (SQLの発行)

OO4O

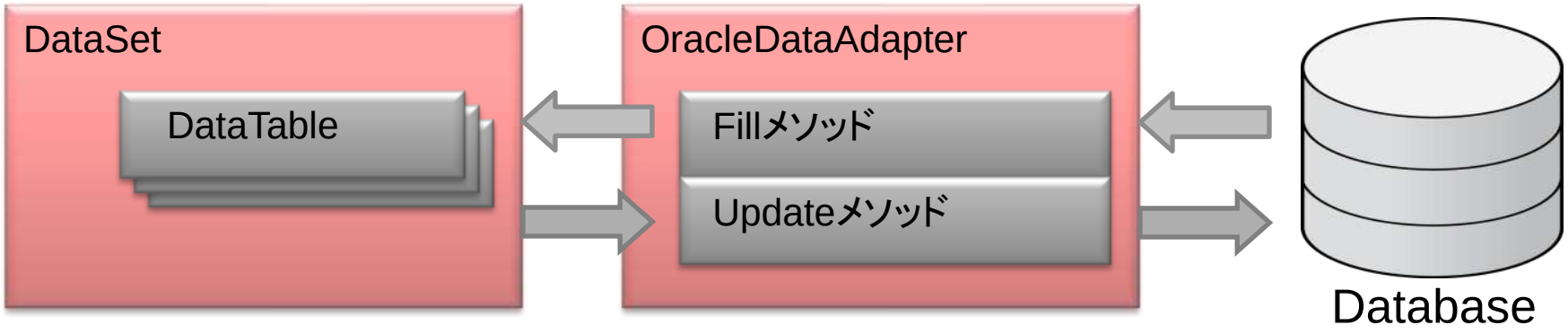
```
Dim OraSession  As Object      "セッション・オブジェクト
Dim OraDatabase  As Object      "データベース・オブジェクト
Set OraSession = CreateObject("OracleInProcServer.XOraSession")
Set OraDatabase = OraSession.OpenDatabase("orcl", "scott/tiger", &H2&)
OraDatabase.ExecuteSQL "update emp set ename='scott' where empno=6"
```

OO4Oでのデータの更新（フィールド指定）

```
Set OraDatabase = OraSession.OpenDatabase( _
    "orcl", "scott/tiger", dbOption.ORADB_DEFAULT)
sSql = "SELECT * FROM MstCustomer Where CustomerId='" + TextCustomerId.Text + "'"
Set OraDynaset = _
    OraDatabase.CreateDynaset(sSql, ORADYN_DEFAULT)
OraDynaset.Edit
OraDynaset("CustomerID") = TextCustomerId.Text
OraDynaset("CustomerNAME") = TextCustomerName.Text
OraDynaset("EmployeeNAME") = TextEmployeeName.Text
OraDynaset("kana") = TextKana.Text
OraDynaset("job") = TextJob.Text
OraDynaset("postcode") = TextPostCode.Text
OraDynaset("prefectures") = ComboPrefectures.Text
OraDynaset("address") = TextAddress.Text
OraDynaset("tel") = TextTel.Text
OraDynaset("fax") = TextFax.Text
OraDynaset("note") = TextNote.Text
OraDynaset.Update
```

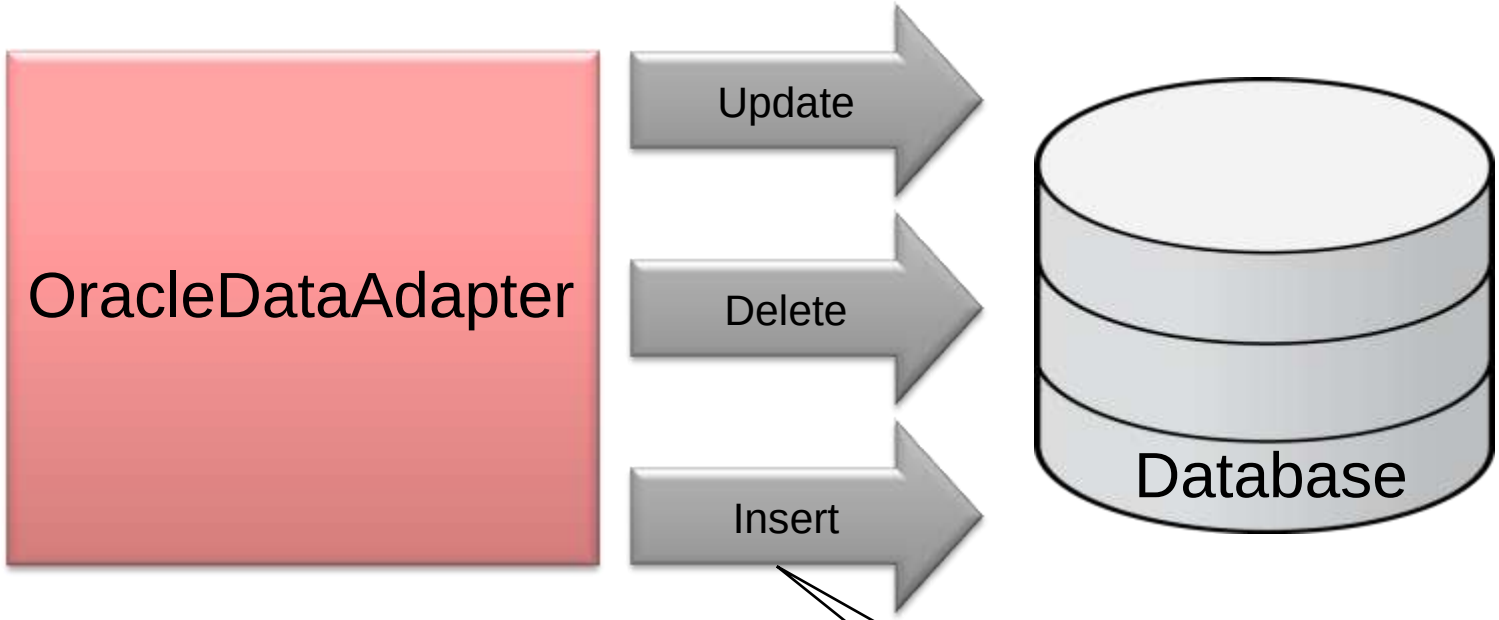
OraDynasetオブジェクトのOraFieldオブジェクトに画面上の値を設定している。

DataSetを使用した更新



DataAdapterはDataSet(DataTable)とデータベースとの間で、データを仲介します。

DataSetからデータベースへの反映



それぞれのSQLは個別に設定することも可能ですが、
OracleCommandBuilderを使用し、
SQLを自動生成することが可能。

Databaseに対して、
Update・Delete・
Insert SQLを発行

OracleCommandBuilderの使用

```
cmd.CommandText = "Select * From MstCustomer Where CustomerId=:CustomerId"
If cmd.Parameters.Contains("CustomerId") Then
    cmd.Parameters("CustomerId").Value = TextBoxID.Text
Else
    cmd.Parameters.Add("CustomerId", TextBoxID.Text)
End If
da.SelectCommand = cmd
dsCustomer.Clear()
da.Fill(dsCustomer, "MstCustomer")

Dim cb As New OracleCommandBuilder(da)
da.UpdateCommand = cb.GetUpdateCommand
da.InsertCommand = cb.GetInsertCommand
da.DeleteCommand = cb.GetDeleteCommand
```

OracleDataAdapterのSelectCommandプロパティがInsertCommand, UpdateCommand, DeleteCommandプロパティに設定可能なSqlCommandを自動的に生成

OracleCommandBuilderの使用

```
Dim CustRow As DataRow
If dsCustomer.Tables("MstCustomer").Rows.Count = 0 Then
    CustRow = dsCustomer.Tables("MstCustomer").NewRow()          ①
Else
    CustRow = dsCustomer.Tables("MstCustomer").Rows(0)           ②
End If
CustRow.Item("CustomerId") = TextBoxID.Text                      ③
CustRow.Item("CustomerNAME") = TextBoxCompanyName.Text
      ～ 以下略 ～
If CustRow.RowState = DataRowState.Detached Then _               ④
    dsCustomer.Tables("MstCustomer").Rows.Add(CustRow)
da.Update(dsCustomer.Tables("MstCustomer"))                      ⑤
```

①新規行の場合はNewRowメソッドで新規行を生成

②既にデータが存在する場合は、既存行を指定

③DataSetにフォームの値を代入

④新規行の場合はDataSetに行を追加

⑤OracleDataAdapterのUpdateメソッドで変更を反映

型付DataSetを使用した更新

形なしデータセットの場合

```
Dim strCustName As String  = _
  dsCustomer.Tables("MstCustomer").Rows(0).Item(“CustomerName”)
```

カラム名を文字列で指定しており、データ型の復元もキャストにより行われているため、実行時エラーが発生する可能性がある。

型付データセットの場合

```
Dim strCustName As String = _
  dsCustomer.MstCustomer(0).CustomerName
```

定義情報をすでに取り込んでいるため、本来のデータ型での取り扱いが可能で、前途のような実行時エラーがない。

コレクションベースのメソッドを使う代わりに、名前を使ってテーブルや列にアクセスできる。

- コードの信頼性が向上
- コード補完の利用

OO4OでのRefCursorの使用

Package

```
CREATE OR REPLACE PACKAGE SCOTT.pkg_ref AS
  CURSOR c1 IS SELECT ename FROM emp;
  TYPE empCur IS REF CURSOR RETURN c1%ROWTYPE;
  PROCEDURE GetEmpData(
    indeptno IN NUMBER,
    EmpCursor in out empCur );
END;
```

Package Body

```
CREATE OR REPLACE PACKAGE BODY SCOTT.pkg_ref AS
  PROCEDURE GetEmpData(indeptno IN NUMBER,
    EmpCursor in out empCur ) IS
      BEGIN
        OPEN EmpCursor FOR
          SELECT ename FROM emp WHERE deptno=indeptno;
      END GetEmpData;
END pkg_ref;
```

ODP.NETでのRefCursorの使用

Package

```
CREATE OR REPLACE PACKAGE SCOTT.pkg_ref2 AS
  CURSOR c1 IS SELECT * FROM emp;
  CURSOR c2 IS SELECT * FROM dept;
  TYPE empCur IS REF CURSOR RETURN c1%ROWTYPE;
  TYPE deptCur IS REF CURSOR RETURN c2%ROWTYPE;
  PROCEDURE GetEmpDeptData(
    EmpCursor in out empCur,
    DeptCursor in out deptCur
   );
END;
```

Package Body

```
CREATE OR REPLACE PACKAGE BODY SCOTT.pkg_ref2 AS
  PROCEDURE GetEmpDeptData(
    EmpCursor in out empCur,
    DeptCursor in out deptCur) IS
      BEGIN
        OPEN EmpCursor FOR SELECT * FROM emp;
        OPEN DeptCursor FOR SELECT * FROM dept;
      END GetEmpDeptData;
END pkg_ref2;
```

ODP.NETでのRefCursorの使用(非接続型)

```
Dim cmd As New OracleCommand("pkg_ref2.GetEmpDeptData", cnn)
cmd.CommandType = CommandType.StoredProcedure

'REF  CURSORパラメータのバインド
cmd.Parameters.Add("EmpCursor", OracleDbType.RefCursor, ParameterDirection.Output)
cmd.Parameters.Add("DeptCursor", OracleDbType.RefCursor, ParameterDirection.Output)

'SQL文の実行とRef Cursorの使用
Dim dsData As New DataSet
Dim da As New OracleDataAdapter(cmd)
da.Fill(dsData, "data")

'Gridへ表示
DataGridEmp.SetDataBinding(dsData, "data")
DataGridDept.SetDataBinding(dsData, "data1")
```

①

②

③

④

①発行するストアドプロシージャを定義。

②RefCursor型のパラメータを定義

③RefCursorの内容をデータセットに格納

④DataSetの内容をグリッドに表示

ODP.NETでのRefCursorの使用(接続型）

```
Dim cmd As New OracleCommand("pkg_ref2.GetEmpDeptData", cnn)
cmd.CommandType = CommandType.StoredProcedure

'REF  CURSORパラメータのバインド
cmd.Parameters.Add("EmpCursor", OracleDbType.RefCursor, ParameterDirection.Output)
cmd.Parameters.Add("DeptCursor", OracleDbType.RefCursor, ParameterDirection.Output)
cmd.ExecuteNonQuery()

'SQL文の実行とRef Cursorの使用
Dim dr1 As OracleDataReader = CType(cmd.Parameters(0).Value, OracleRefCursor).GetDataReader
Dim dr2 As OracleDataReader = CType(cmd.Parameters(1).Value, OracleRefCursor).GetDataReader

'RefCursorの内容をデバックウィンドウに表示
Do While dr1.Read
  Debug.WriteLine(dr1("ename"))
Loop
Do While dr2.Read
  Debug.WriteLine(dr2("dname"))
Loop
```

①

①RefCursorの値をGetDataReaderで取得

OO4OとODP.NETのコードの比較（トランザクション）

OO4O

```
'---データベースとの接続を開く
Dim OraSession As New OraSessionClass
 Dim OraDatabase As OraDatabase
 Set OraDatabase = OraSession.OpenDatabase( _
        "orcl", "scott/tiger", dbOption.ORADB_DEFAULT)
'トランザクションの開始
 OraSession.BeginTrans
 OraDatabase.ExecuteSQL "insert into emp(empno,ename) values(6,'Michel')"
 OraSession.CommitTrans
```

OO4OとODP.NETのコードの比較（トランザクション）

ODP.NET

```
'---データベースとの接続を開く
cnn.Open()
Dim cmd As New OracleCommand
cmd.Connection = cnn
'トランザクションの開始
Dim txn As OracleTransaction = cnn.BeginTransaction()
cmd.CommandText = "insert into emp(empno) values(9999)"
cmd.CommandType = CommandType.Text
cmd.ExecuteNonQuery()
txn.Commit()
```

OO4OとODP.NETのコードの比較（トランザクション・Save Point）

ODP.NET

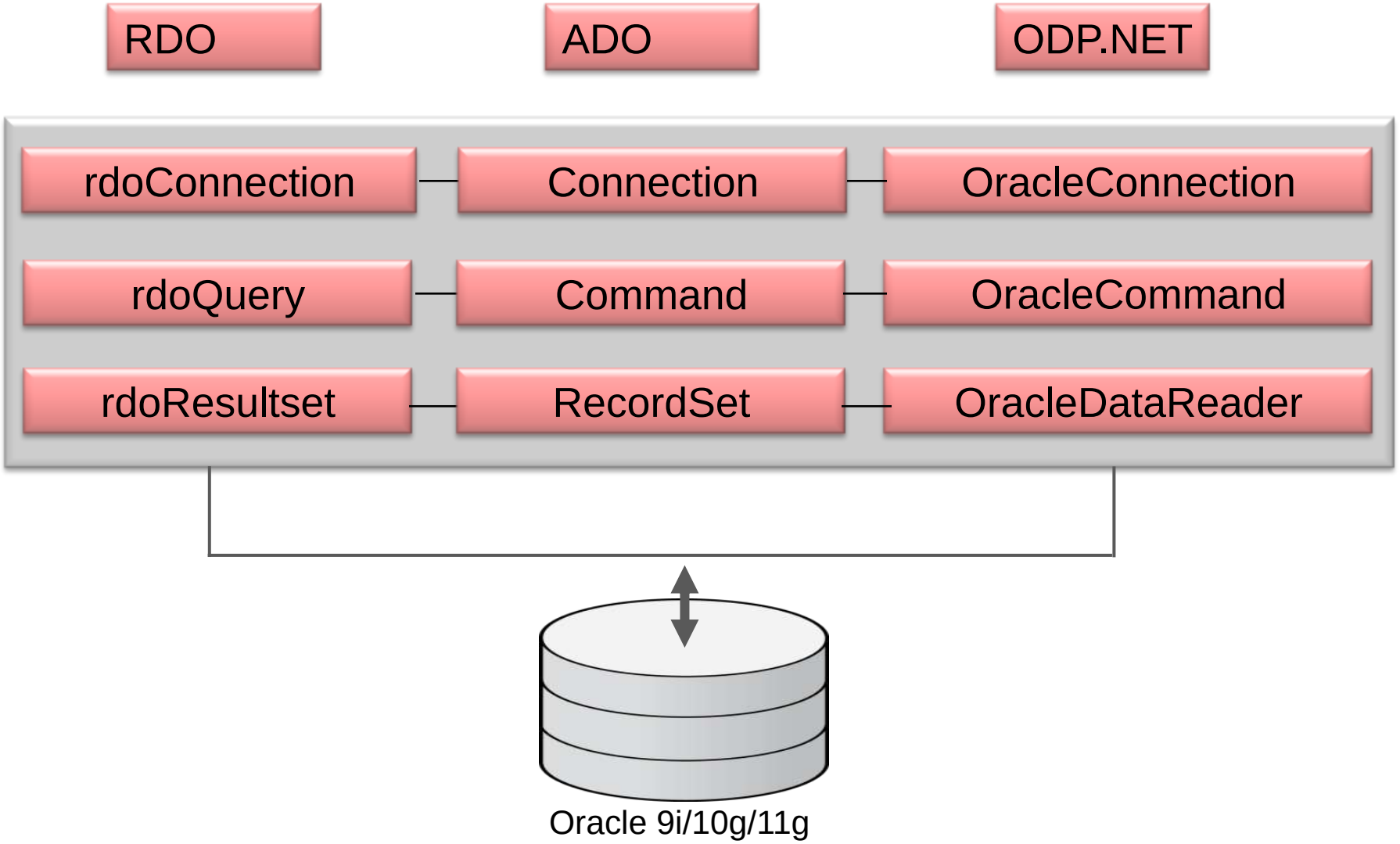
```
Dim cnn As New OracleConnection

cnn.ConnectionString = "user id=scott;password=tiger;data source=orcl"
cnn.Open()
'トランザクションの開始
Dim txn As OracleTransaction = cnn.BeginTransaction()
Dim strSQL1 As String = "INSERT INTO emp (empno, ename) VALUES (1,'Employee1')"
Dim myCmd As New OracleCommand(strSQL1, cnn)
Dim res As Integer = myCmd.ExecuteNonQuery()
‘SavePoint ‘a’ でトランザクションを保存
txn.Save("a")

Dim strSQL2 As String = "INSERT INTO emp (empno, ename) VALUES (2,'Employee2')"
Dim myCmd2 As New OracleCommand(strSQL2, cnn)
Dim res2 As Integer = myCmd2.ExecuteNonQuery()
‘SavePoint ‘b’ でトランザクションを保存
txn.Save("b")

‘SavePoint ‘a’までロールバックしコミット
txn.Rollback("a")
txn.Commit()
```

ADO/RDOで使用するオブジェクト



ADO/RDOでのデータアクセスモデル

rdoResultset

RecordSet

```
Do Until RecordSet.Eof
  ～　データアクセス　～
  RecordSet.Movenext
Loop
```

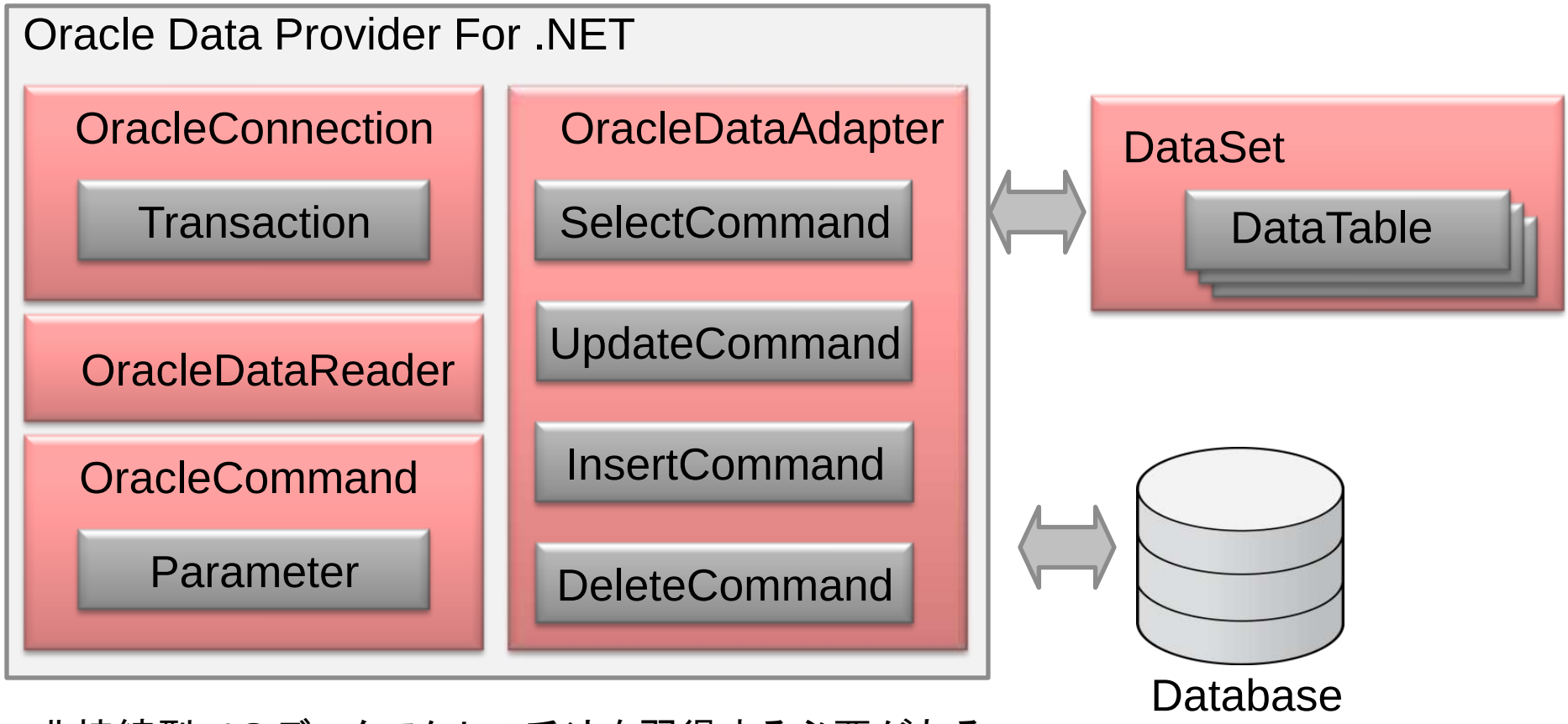
このようにRecordSetを開いて、Movenextメソッドでシーケンシャルに読み込みながら処理を行うプログラミングスタイルが多い。

ループ処理の間、データベースと接続された状態

OracleDataReader

OracleDataReaderは、RecordSetのReadOnly , FowardOnlyオプションで開いた状態に似ている。

ODP.NETでのデータアクセスモデル



非接続型でのデータアクセス手法を習得する必要がある。

- しかし、高負荷となる接続状態でのデータアクセス処理を回避することが可能

Agenda

- はじめに
- 移行するメリット
- 移行方法の選定
- 移行についての具体的な手順
- 移行後の注意点
- まとめ

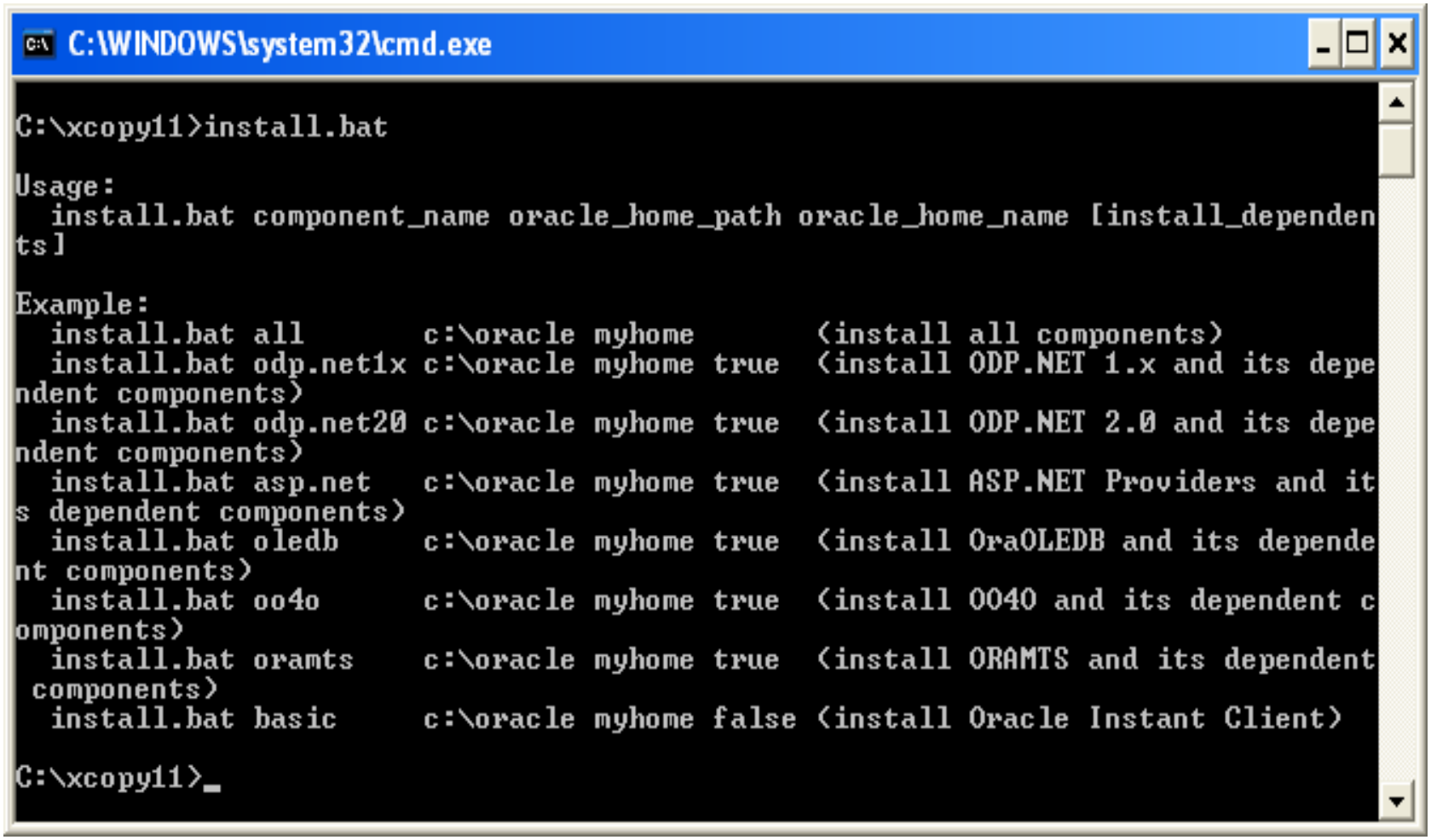
ODP.NETの効率的な配布

- OUI
  - Oracle GUIベースの インストール
  - DB Server, Client共、GUIベースで容易にインストール
- Silent install
  - インストール作業の自動化
  - Microsoft SMS もしくは、他のインストール製品を利用して配布可能
  - 大量クライアントの配布に最適
- Instant Client XCopy
  - ファイルコピーとバッチの実行でインストール可能
  - 大量クライアントの配布に最適

Instant Client XCopy での配布

- Install の手順
  - Step 1: Unzip and copy IC files to disk
  - Step 2: install.batの実行
    - 配布したいコンポーネント、Oracle Home の名前、配布先の3つを指定。
  - Step 3: Windows PATHにインストールパスを追加
  - Step 4: NLS_LANG の設定

XCopy Install.bat



Instant Client XCopy コンポーネント

- Instant Client xcopy でインストール可能なコンポーネント
  - SQL*Plus
  - OCI
  - OCCI
  - ODBC
  - JDBC-OCI
- Instant Client xcopy で追加されたコンポーネント
  - **ODP.NET**
  - Oracle Providers for ASP.NET
  - Oracle Provider for OLE DB
  - OO4O
  - Oracle Services for MTS

Agenda

- はじめに
- 移行するメリット
- 移行方法の選定
- 移行についての具体的な手順
- 移行後の注意点
- まとめ

まとめ

- ODP.NETにするメリットは大
  - ODP.NETはADO.NET準拠の高速プロバイダー
  - OO4Oより開発生産性が向上
  - Oracle固有の機能によりRDBMS機能を最大限
  - 利用可能
    - XML、PL/SQL、配列パラメータ
- .NETネイティブなAPIとのシームレスな連携（XML）

あなたにいちばん近いオラクル

Oracle Direct

まずはお問合せください

Oracle Direct　検索

システムの検討・構築から運用まで、ITプロジェクト全般の相談窓口としてご支援いたします。
システム構成やライセンス/購入方法などお気軽にお問い合わせ下さい。

Web問い合わせフォーム

専用お問い合わせフォームにてご相談内容を承ります。
http://www.oracle.co.jp/inq_pl/INQUIRY/quest?rid=28

※フォームの入力には、Oracle Direct Seminar申込時と同じ
　ログインが必要となります。
※こちらから詳細確認のお電話を差し上げる場合がありますので、ご登録されている連絡先が最新のものになっているか、ご確認下さい。

フリーダイヤル

0120－155－096

※月曜~金曜　9:00~12:00、13:00~18:00
（祝日および年末年始除く）

以上の事項は、弊社の一般的な製品の方向性に関する概要を説明するものです。また、情報提供を唯一の目的とするものであり、いかなる契約にも組み込むことはできません。以下の事項は、マテリアルやコード、機能を提供することをコミットメント（確約）するものではないため、購買決定を行う際の判断材料になさらないで下さい。オラクル製品に関して記載されている機能の開発、リリースおよび時期については、弊社の裁量により決定されます。