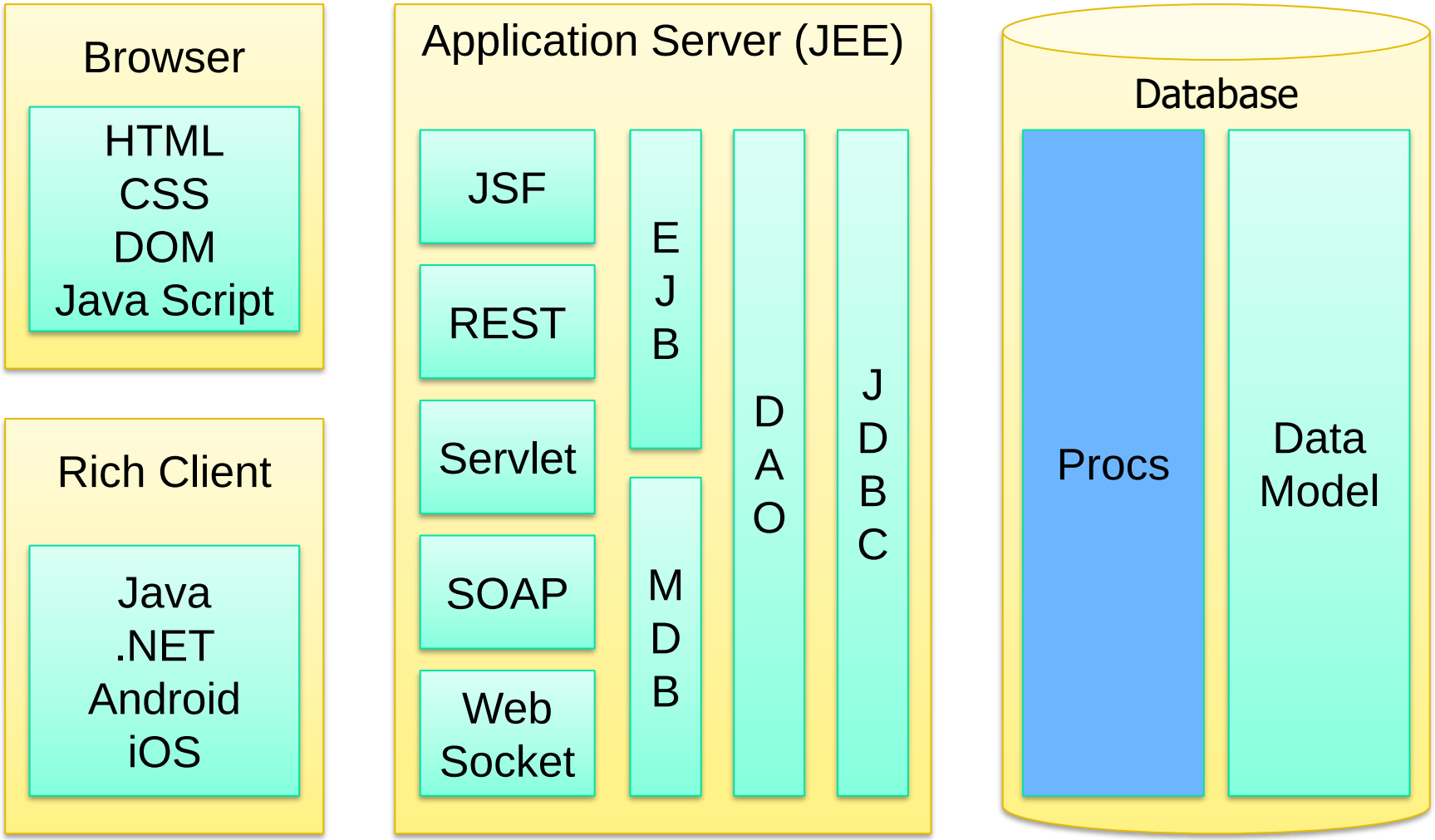
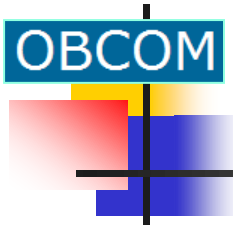
Aplicaciones Web (parte 3)

Eduardo Ostertag Jenkins, Ph.D.
OBCOM INGENIERIA S.A. (Chile)
Eduardo.Ostertag@obcom.cl

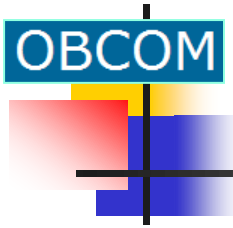
Procedimientos almacenados





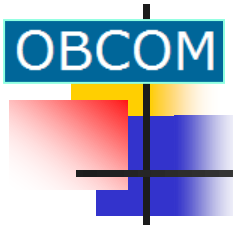
Bajo acoplamiento

- Encapsulación: principio básico de OO: sólo se conoce la interfaz, y no la implementación
  - Interfaz: nombres de métodos, parámetros
  - Implementación: lógica, estructuras de datos
- Tanto la lógica de los procedimientos, como el modelo de datos se pueden cambiar sin afectar a los clientes (capa intermedia)
- Estos cambios se pueden hacer sin modificar la capa intermedia, ni la capa cliente



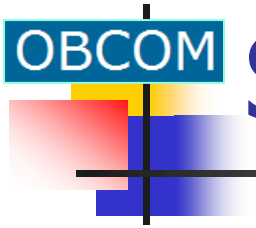
Alto rendimiento

- Los procedimientos almacenados se compilan y optimizan una vez al cargarlos
  - OJO: se deben cargar con tablas con datos!!
- Todo el acceso a los datos se hace dentro de la base de datos, y no a través de la red
- Se reduce al mínimo el número de llamados y la cantidad de datos que fluye en la red
- La red es muy lenta comparada con accesar la memoria o cache de la base de datos



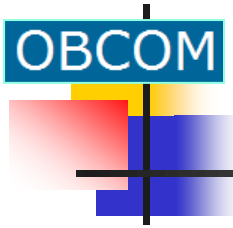
Acceso universal

- Los procedimientos almacenados pueden ser utilizados por todo tipo de clientes: COBOL, C++, Java, .NET, Visual Basic, Reports, etc.
- Existen bibliotecas de comunicación para todo tipo de clientes y en todos los sistemas operativos: ADO, ODBC, JDBC, etc.
- La reglas del negocio están implementadas en un solo lugar (el procedimiento). No es necesario reimplementarlas en cada cliente



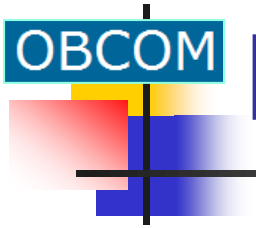
Seguridad

- Los clientes no tienen acceso directo a las tablas del modelo de datos, y sólo pueden invocar a procedimientos almacenados
- Los clientes no pueden hacer SELECT, INSERT, UPDATE o DELETE de las tablas
- La lógica de los procedimientos controla con exactitud el tipo de acceso, y la información a la cual tienen acceso los clientes
- ¿Puede hacer esto usando sólo un GRANT?



Lenguaje propietario

- Oracle se programa con PL/SQL, MS-SQL se programa con Transac-SQL, MySQL …
- Estos lenguajes son distintos, y la migración de una base de datos a otra no es directa
- Existen herramientas que traducen de un lenguaje a otro. La efectividad es baja, y el código generado es malito (mantención)
- Se puede traducir manualmente con mucha productividad (20-30 procs. día-hombre)



Procedimientos públicos (1)

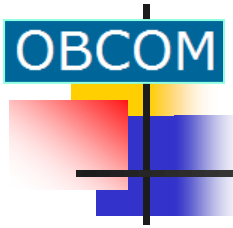
- Diseñados para llamarlos vía JDBC
  - Parámetros de entrada JDBC (input)
  - Parámetros de salida JDBC (output)
  - Tablas (ResultSet) con columnas JDBC (output)
- Tipos de datos Java versus JDBC
  - Integer ↔ INTEGER, DECIMAL(p,0) [p ≤ 9]
  - Long ↔ BIGINT, DECIMAL(p,0) [p ≤ 18]
  - String ↔ VARCHAR, CHAR, TEXT, CLOB, LONGTEXT
  - Date ↔ DATETIME, TIMESTAMP
  - BigDecimal ↔ DECIMAL, NUMERIC
  - byte[] ↔ VARBINARY, BLOB, BYTEA, LONGBLOB

Procedimientos públicos (2)

- El carácter "**$**" se puede usar en nombres en todas las bases de datos
- Tienen nombres con prefijo "**modulo$**", donde "**modulo**" es el nombre de un conjunto de procedimientos relacionados
- No usar **PACKAGE** de Oracle, porque los llamados JDBC requieren sintaxis propietaria

Procedimientos públicos (3)

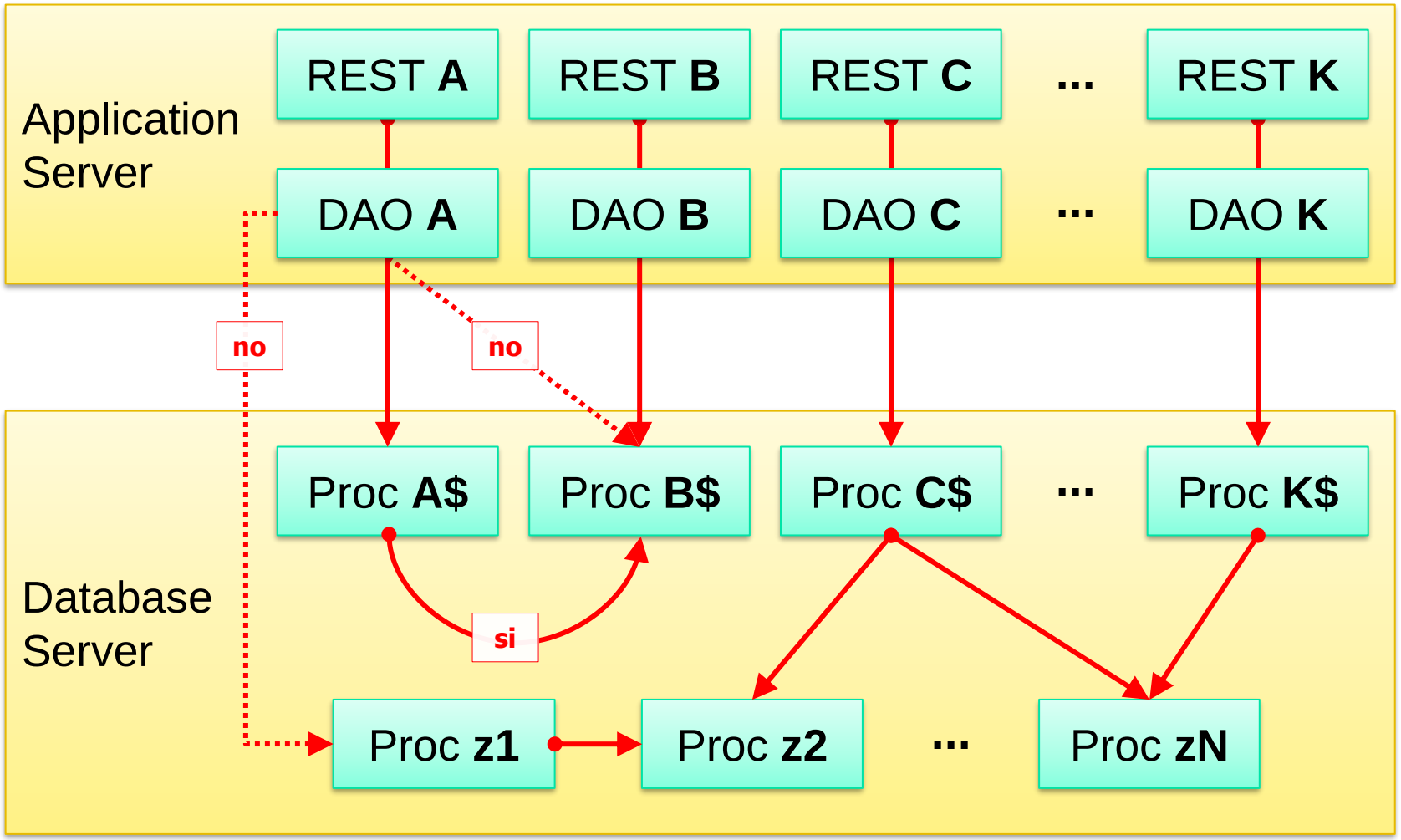
- Formato de nombres de procedimientos:
  - **module$object_action**
- Ejemplos de nombres de procedimientos:
  - DESKTOPFX$USER_BY_RUT
  - STORAGE$PRODUCT_SKU_UPDATE
  - PAYCORE$RECYCLE_ID_CHECK (SP_CheckRecycleId)
  - PAYCORE$TRX_ENQUEUE (SP_EnqueueTrx)
  - PAYCORE$TRX_STATUS_UPDATE (SP_UpdateStatusTrx)
  - PAYCORE$ORDER_CREATE (SP_CreateOrder)



Procedimientos privados

- Diseñados para llamarlos desde otros procedimientos dentro de la base de datos
- Se pueden usar todas las tecnologías de la base de datos, sin restricción
- Se usan para factorizar la lógica común, reduciendo costos de mantención
- Tienen nombres con prefijo "**z**" para no confundirlos con procedimientos públicos
  - zUF_Obtener, zInteres_Calcular, …

Llamados a procedimientos



Muchas gracias

Muchas gracias