EXAMEN ABD (Arquitecturas de Sistemas de Bases de Datos)
Escola Tècnica Superior d'Informàtica Aplicada

8 de septiembre de 2004

NOTA_TOTAL (de 0 a 4) = NOTA_OBTENIDA_TEST + NOTA_CUESTIONES.

CUESTIONES TIPO TEST (máximo 3,25 puntos)
(A RELLENAR EN LA TABLILLA APARTE)

NOTA_OBTENIDA_TEST = (Bien_contestadas – Mal_contestadas/3) × 0,1625.

1) La arquitectura ANSI/SPARC para los SGBD con los niveles conceptual, interno y externo:
   a. Obliga a una arquitectura de tres capas: servidor de datos, servidor de aplicaciones y cliente.
   b. Obliga a una arquitectura de cuatro capas (separando presentación de la lógica de la paliación), porque hay que considerar el esquema lógico.
   c. Permiten cualquier tipo de arquitectura para la lógica de la aplicación y la localización de los datos: una, dos, tres o más capas.
   d. Ya no se utiliza en los sistemas distribuidos.

2) ¿Qué es el "middleware"?:
   a. Un concepto intermedio entre software y hardware.
   b. Un recubrimiento objetual a las bases de datos.
   c. Un término vago para referirse al software y protocolos existentes entre el servidor y el cliente.
   d. Un tipo de shareware en la que el consumidor sólo paga la mitad.

3) ¿Cuáles son los objetivos principales de la distribución?:
   a. Mayor disponibilidad, flexibilidad y eficiencia.
   b. Mayor abstracción y eficiencia.
   c. Aumentar el número de capas y el número de usuarios.
   d. Mayor portabilidad y abstracción.

4) Los tipos ROW (fila) en SQL3:
   a. Permiten incluir varias filas en una tabla.
   b. Permiten incluir multiplicidad en una fila.
   c. Permiten claves ajenas múltiples.
   d. Son equivalentes a los ARRAY.

5) Data una tabla PERSONA(nombre: VARCHAR(20), nombre_padre: VARCHAR(20), nombre_madre: VARCHAR(20)), ¿cómo recuperaríamos todos los ancestros de una persona?
   a. Mediante una consulta SELECT con OUTER JOIN.
   b. Mediante una consulta SELECT con INNER JOIN.
   c. Mediante una consulta.SELECT con subconsultas.
   d. Mediante una consulta SELECT recursiva (WITH RECURSIVE).

6) En una tabla de pacientes tenemos un atributo “Fecha_de_Nacimiento” y queremos hacer múltiples informes y consultas que obtengan la edad de la paciente en el momento de la consulta. Señala de las siguientes, cuál **NO** sería una forma válida de hacerlo.
    a. Añadiríamos un atributo “Edad” que se debería introducir cuando se da de alta un paciente.
    b. Añadiríamos un atributo “Edad” y lo mantendríamos actualizado mediante un disparador que se lanzaría todas las noches, para mantener congruentes la “Fecha_de_Nacimiento”, la Edad y la fecha actual.
    c. Haríamos una vista sobre la tabla paciente con un SELECT de todos los atributos, añadiendo además a la SELECT un valor calculado con la edad en el momento de ejecución.
    d. Utilizaríamos un método de SQL3 para añadir a la tabla de pacientes un atributo virtual derivado.

7) Señala cuál de los siguientes **NO** es un inconveniente de la arquitectura monolítica o centralizada:
    a. Suele aparecer cuello de botella en el sistema de control de terminales.
    b. Poca fiabilidad / disponibilidad. Si el sistema central falla, el servicio se para.
    c. Si la visualización es gráfica se requieren refrescos inmediatos, saturando la red.
    d. Los terminales son caros.

8) Indica, de los diferentes niveles de granularidad a la hora de comunicarse datos un cliente y un servidor, cuál es el de mayor nivel de abstracción:
    a. Servidores de páginas.
    b. Servidores de ficheros sin paginación.
    c. Servidores de ficheros con paginación.
    d. Servidores de consultas.

9) Respecto a la caché de datos en el cliente, indica cuál de las siguientes afirmaciones es **FALSA**:
    a. Reduce el tráfico por la red y aumenta el rendimiento.
    b. Puede generar problemas de consistencia si los datos se actualizan por otras aplicaciones.
    c. Permite enviar datos de unos clientes a otros directamente.
    d. Genera una dificultad en la gestión de la caché.

10) Si en una arquitectura de tres capas para la web (Capa 1: servidor de datos, Capa 2: servidor web, Capa 3: navegadores), cae la red que conecta la Capa 1 y la Capa 2, pero no cae la red que conecta la Capa 2 y la Capa 3, indica cuál de las siguientes afirmaciones es **CIERTA**:
    a. Las aplicaciones se quedarán completamente paralizadas.
    b. Funcionarán aquellas partes de la aplicación que no accedan a los datos.
    c. Funcionarán aquellas partes de la aplicación que lean pero no actualicen los datos
    d. Se pueden quedar datos inconsistentes en el servidor.

11) Las variables indicador, sirven para:
    a. Indicar en la fila en la que está posicionado un CURSOR.
    b. Indicar el tipo de datos de una variable.
    c. Indicar si una variable toma valor nulo o fuera de rango.
    d. Indicar en la dirección en la que avanza un CURSOR.

12) Si usamos SQL embebido sobre lenguaje C++.
    a. Podemos usar cualquier compilador de C++ después del preprocesador.
    b. Sólo podemos usar el compilador de C++ proporcionado por el SGBD.
    c. Podemos usar los compiladores de C++ después del preprocesador que acepten directivas SQL.
    d. No hay SQL embebido sobre C++, por ser éste un lenguaje orientado a objetos.

13) Selecciona cuál **NO** es un ejemplo de SQL embebido:
    a. SQL3 Host Language Binding (Part 5 – Bindings).
    b. SQLJ.
    c. ECPG de PostgreSQL.
    d. JDBC.

14) En SQL dinámico, para pasar los parámetros de entrada y recoger los resultados de salida:
    a. Se utilizan variables indicadores.
    b. Se utilizan "descriptores" u otras estructuras de datos, para el intercambio de datos.
    c. Se deben utilizar listas u otras estructuras dinámicas de datos en disco, para recoger múltiples filas.
    d. Se reserva memoria en la pila para tal fin.

15) Indicar, de las siguientes propuestas y protocolos, cuál NO sirve para el acceso a datos entre cliente y servidor:
    a. DAO.
    b. ADO.
    c. JDBC
    d. Swing.

16) En una arquitectura de tres capas (aplicación, negocio y datos), ¿qué solución es más apropiada cuando las distintas aplicaciones comparten mucha lógica común?
    a. Construir una API con todas las funcionalidades comunes.
    b. Integrar toda la funcionalidad común en la capa de datos.
    c. Repartir toda la funcionalidad común entre la capa de datos y la capa de negocio.
    d. Repetir las funcionalidades y adaptarlas a cada aplicación.

17) ¿Cuál de estos cuatro casos favorece que bajemos la lógica de la aplicación a capas inferiores (capa de negocio o capa de datos)?
    a. Si las aplicaciones diferentes tienen mucha lógica común y se desean aplicaciones más modificables.
    b. La capa de negocio y de datos están saturadas.
    c. Si las aplicaciones no comparten lógica común, son de alta disponibilidad y tienen gran componente no transaccional.
    d. Necesitamos alta disponibilidad de las aplicaciones.

18) ¿Qué tipos de fragmentación podemos tener en un sistema de bases de datos distribuidos?
    a. Remota y local.
    b. De memoria compartida, de disco compartido y sin compartir nada.
    c. Horizontal, vertical y mixta.
    d. En anillo y en tubería.

19) Respecto a la ejecución de las consultas en un sistema de bases de datos distribuido:
    a. Se ejecutan igual que en los no distribuidos, ya que todos los sitios tienen copia de todos los datos.
    b. Da igual como se desarrollen pues cada sitio es transparente y ve todos los datos de los demás sitios, siendo igual de rápido en cualquier caso.
    c. Es importante utilizar alguna estrategia para minimizar los datos que se transmiten entre los sitios para realizar la consulta compleja.
    d. Sólo se pueden realizar si el sitio local cumple la condición de "mantenimiento de la clave", igual que en la actualización de vistas.

20) Indica cuál de las siguientes afirmaciones es **FALSA** respecto a las bases de datos en memoria principal:
    a. Se utilizan cuando se requieren grandes rendimientos.
    b. Si se tiene un SAI (mantenedor de corriente) se puede prescindir del disco duro.
    c. El fichero de diario (log) va al disco duro.
    d. La utilidad de los índices se ha de replantear completamente.

CUESTIONES (A CONTESTAR EN LA MISMA HOJA APARTE, POR DETRÁS. NO MÁS DE 20 LÍNEAS POR CUESTIÓN)

1) Destaca las ventajas e inconvenientes de una arquitectura cliente/servidor sobre una arquitectura monolítica. (0,25 puntos)

2) Compara los sistemas de bases de datos paralelos con los distribuidos. (0,25 puntos)

3) Enumera cinco aspectos que afecten a la colocación de la lógica de la aplicación, explicando brevemente de qué modo afecta. (0,25 puntos)

NOMBRE Y APELLIDOS:_____

EXAMEN ABD (Arquitecturas de Sistemas de Bases de Datos). 8 de septiembre de 2004

Respuestas TEST:

Num. Pregunta	Respuesta
1	C
2	C
3	A
4	B
5	D
6	A
7	D
8	D
9	C
10	B
11	C
12	A
13	D
14	B
15	D
16	C
17	A
18	C
19	C
20	B

Respuestas Cuestiones:

CUESTIÓN 1:

VENTAJAS:

- Permite la interconexión de distintos sistemas cliente.
- Libera de carga el sistema central.
- Permite repartir la lógica entre el cliente y el servidor.
- Permite combinar distintos servicios en una misma red.
- Permite una gran cantidad de aplicaciones diversas.
- Facilita las aplicaciones con interfaz gráfico y refrescos inmediatos.
- Mayor fiabilidad y disponibilidad al no depender (para todo) del sistema central.

DESVENTAJAS:

- Las aplicaciones mezclan interfaz con la lógica de la aplicación/negocio.
- Las aplicaciones se vuelven "pesadas": clientes gruesos.

CUESTIÓN 2:

Bases de datos paralelas	Bases de datos distribuidas
Un sistema de gestión de bases de datos incorpora todos los datos.	Los datos se distribuyen entre diferentes sistemas de gestión de bases de datos.
El sistema se encuentra en la misma localización, con un catálogo único.	Se trata de un conjunto de sistemas en diferentes localizaciones, con un catálogo distribuido o unificado a partir de cada sitio.
Tiene una filosofía jerarquizada. Un control central determina cómo repartir los procesos y datos.	Tiene una filosofía colaborativa. Cada sitio pide y sirve datos a los demás.
Basan la mejora del rendimiento en el uso de varias CPUs, o varias memorias, o varios discos.	Basan la mejora del rendimiento en que son capaces de servir consultas separadamente, según los datos que tiene cada sitio.
Es transparente al usuario.	Es transparente al usuario.
Puede haber replicación y partición (fragmentación).	Puede haber replicación y partición (fragmentación).
La partición de datos es generalmente vertical u horizontal (mixta) y raramente por los tipos de tablas.	La partición es generalmente distribuyendo las tablas según las diferentes áreas.
La arquitectura está muy prefijada por el fabricante. La extensibilidad es limitada.	Pueden ser homogéneos o heterogéneos, y puede construirse, por tanto, a partir de piezas que funcionan independientemente. La extensibilidad es más alta.

CUESTIÓN 3: (era suficiente con nombrar y explicar brevemente 5 de los 20 aspectos siguientes)

Aspectos organizacionales:

- Un solo repositorio o varios: si hay varios repositorios es necesario integrar más arriba (negocio o en el cliente)
- Hay lógica común: si hay lógica común conviene bajarla a la capa de negocio o de datos.
- Un solo departamento o varios: si hay varios es más difícil centralizar, y más difícil la opción de la API común y tiene más sentido integrar en la capa de negocio.
- Clientes finales fuera o dentro de la organización: De nuevo la idea de la API pierde fuerza y tiene más sentido integrar en la capa de negocio.

Aspectos funcionales:

- Funcionalidad transaccional (lectura y escritura): obliga a hacer bloqueos frecuentes. Tendencia: liberar al cliente de esta tarea.
- Acceso concurrente frecuente: obliga a hacer bloqueos frecuentes. Tendencia: liberar al cliente de esta tarea.
- Transacciones con bloqueos "masivos": preferible bajar estas operaciones a negocio o datos.
- Acceso a los mismos datos reiteradamente: uso de cachés, que son más fáciles de manejar si se baja la funcionalidad.
- Información calculada o derivada: si es derivada debe ir cuanto más abajo mejor.
- Restricciones (aplicación, organización o datos): las de aplicación al cliente y las otras más abajo.
- Gestión de errores: a veces se obliga a gestionarlo doblemente, para que tenga una apariencia más cómoda al usuario.

Aspectos de seguridad e integridad:

- ¿Pueden bloquear las aplicaciones fiablemente? Si las aplicaciones bloquean y luego pueden no desbloquear o caer, es preferible que no bloquen o bajar la lógica a capas inferiores.
- La aplicación ha de agregar valores que no debe conocer: hacer procedimientos almacenados que lo calculen en capas inferiores.
- Gestión de permisos: en la capa de datos mediante permisos, en la de aplicaciones ha de ser mediante otros mecanismos.
- Hay transmisión segura por la red: delicado si se envía información confidencial o contraseñas. A veces es necesario bajar la funcionalidad para no correr el riesgo.
- ¿La aplicación ha de seguir funcionando si el SGBD cae? En ese caso no se han de bajar funcionalidades que no accedan a datos, aunque sean comunes a otras aplicaciones.

Aspectos de portabilidad, modificabilidad y extensibilidad:

- ¿Se requieren aplicaciones portables? Generalmente es más difícil migrar la funcionalidad de los clientes que de los servidores, con lo que si está en capas inferiores será más portable. Esto también cambia si se utilizan protocolos estándares, como SGBD o JDBC.
- ¿Se requieren aplicaciones modificables y extensibles? Todo más fácil si las funcionalidades están integradas en capas inferiores.
- Más independencia de las modificaciones. Siguiendo ANSI/SPARC, usando procedimientos almacenados para vistas no modificables.
- Aplicaciones modificando el esquema: Mala práctica. Esta "funcionalidad" siempre al servidor de datos.