Desarrollo de Sistemas de Información

Tema 7. Pruebas

Marta Elena Zorrilla Pantaleón

Dpto. de Matemáticas, Estadística y Computación

Referencias bibliográficas

- S. Brass, C. Goldberg. Semantic errors in SQL queries: A quite complete list. *Journal of Systems and Software* (79). 2006
- W.E. Lewis. *Software Testing and Continuous Quality Improvement.* 3rd Edition. Auerbach Publications. 2009
- *Ley Orgánica 15/1999 de 13 de diciembre de Protección de Datos de Carácter Personal*, (LOPD)
- G. J. Myers. *The art of software testing*. 2nd Edition. Wiley. 2004.
- International Standards Organisation, *Information technology – Database languages – SQL:2008*

Introducción

- Este tema se centra en el proceso de **pruebas**, en concreto lo relativo a la estructura de BD y las reglas de integridad definidas
- Objetivos
  - Diseñar y generar los casos de prueba
  - Determinar necesidades para llevar a cabo las pruebas
  - Establecer tipos de pruebas, elementos sobre los que actuar y recomendaciones en cada caso
  - Mostrar alternativas software para realizarlas

¿Por qué hacer pruebas a la BD?

- Recoge funcionalidad crítica del negocio
- Ayuda a la ampliación y refactorizado de la BD
- Es un artefacto más de la aplicación software
- Sirven como documentación

Tipos de Pruebas en BD

- **Pruebas sobre el esquema** (estática)
  - Cómputo de campos calculados, borrados en cascada, valores nulos, conversión en tipos de datos, vistas que devuelven los datos que se esperan, referenciabilidad, etc.
- **Pruebas sobre la parte programática** (dinámica)
  - Procedimientos almacenados, funciones, disparadores, atención especial a las transacciones, cumplir requisitos funcionales.
- **Pruebas de seguridad**
  - Qué usuarios-roles existen, qué pueden hacer, garantizar LOPD, creación y restauración de copias de seguridad, etc.
- **Pruebas de rendimiento**
  - Carga de usuarios, procesos (concurrencia y deadlocks), situaciones límites, etc.
- **Pruebas sobre datos**
  - Comprobar que datos de tablas maestras, tipo CP, provincia, etc están presentes en la BD. Comunicaciones con aplicaciones externas

Técnicas comunes

- BD:
  - Combinación de sentencias PRINT y pruebas ad-hoc
  - Uso de T-SQL debugger (en SQL Server) para inspeccionar el valor de las variables
  - En ambos se requiere la intervención del programador , no está automatizado
  - Las pruebas son ad-hoc y, por tanto, no exactamente reproducibles
  - Solución: **Pruebas unitarias**

- Aplicaciones de BD:
  - Pruebas de caja negra: pruebas que velan por el cumplimiento de los requisitos funcionales. Vistas en Ing. Software II
  - CRUD testing [Lewis, 2009]: matriz donde se recogen todos los objetos y se comprueba la inserción, actualización, borrado y recuperación de datos

Pruebas unitarias

- Prueba unitaria, procedimiento que permite validar si una unidad de código fuente realiza la funcionalidad especificada
- Cada prueba, idealmente, es independiente del resto
- Las pruebas unitarias son responsabilidad de los desarrolladores
- Deben ser rápidas de ejecutar y deben ejecutarse frecuentemente (cada refactorizado)
- Pueden programarse para hacerse en la noche de forma automatizada

Esquema para definir Pruebas unitarias

- E8. Test Case [Lewis, 2009]

Date: Tested by:
System: Environment:
Objetive: TestID Req.ID:
Function: Screen:
Versión: Test Type (unit, integr., system, accept.):

Condition to test:

Data/Steps to perform:

Expected results:

Actual results (pass/fail):

Frameworks para Pruebas

- TSQLUnit http://sourceforge.net/apps/trac/tsqlunit/
  - framework para escribir pruebas para aplicaciones escritas en Transact-SQL.
- DataDude incluido en MS Visual Studio 2010, soporta todo el ciclo de vida de la BD
- tSQLt http://tsqlt.org/
  - gratuita para SQL Server 2005 y posteriores, definición y ejecución de pruebas en SQL Management Studio. Similar a TSQLUnit pero con más procedimientos que aprender para su uso.
- AnyDbTest http://www.anydbtest.com
  - Gratuita, independiente del gestor, casos de test escritos en xml, java,c,..
- Spawner http://sourceforge.net/projects/spawner/
  - Generación de datos manual (no lee del esquema de la BD)

Frameworks para Pruebas

Basadas en Java http://www.java2s.com/Product/Java/Testing/Database-Testing.htm

- SQLUnit http://sqlunit.sourceforge.net
  - Framework que se apoya en JUnit , convierte pruebas escritas en XML a llamadas JDBC y compara los resultados con los suministrados al efecto.
- DbUnit http://dbunit.sourceforge.net/
  - Extensión de JUnit orientada a proyectos de BD. Versión Net en www.ndbunit.org
- dbMonster http://dbmonster.kernelpanic.pl/
  - Herramienta para sintonizar la BD y probar su rendimiento. Genera datos de test aleatorios. Escrita en Java

Gestionar el Estado de la BD

- ¿Cómo garantizar que los resultados de las pruebas sean correctos?
  - Se ha de garantizar que la BD se encuentra en un estado inicial determinado, antes de ejecutar las pruebas
  - Se ha de asegurar que la base de datos tiene el estado apropiado entre cada prueba

Técnicas para Gestionar el Estado de una BD

- Estado inicial:
  - Utilizar una herramienta de generación de datos antes de ejecutar las pruebas
  - Restaurar un Backup
  - Diseñar pruebas que establezcan el estado como punto de comienzo del test
- Entre pruebas:
  - Deshacer los cambios realizados por una prueba, después de su ejecución
  - Utilizar "**Transaction Rollback"** para envolver cada prueba.

Opciones para generar datos de Prueba

- Una copia de datos de producción
  - La más representativa pero generalmente problemas de privacidad
- Generar datos desde cero
  - P.ej. en aplicativos nuevos. Costoso
- Utilizar generadores de código configurables
  - Aplican datos adecuados al tipo de dato de la columna, tienen en cuenta las restricciones, se pueden modelar relaciones entre tablas (p.ej. Por cada pedido, 10 líneas)
  - Generación repetible, útil para probar la BD
  - Los buenos de pago

Pruebas sobre Esquema

- Cubrir para todas las tablas:
  - Tipos de datos, Checks, Dominios, Valores requeridos, PK, FK, operaciones en cascada, Disparadores, índices
- Comprobaciones:
  - Inserción y actualización de valores únicos; rechazo de repetidos
  - Inserción y actualización de valores en claves ajenas; rechazo de valores sin referencia
  - Inserción de valores nulos; rechazo cuando no sea posible
  - Las cuestiones relativas a restricciones también se pueden comprobar leyendo del catálogo de la BD
  - Eliminación y actualización de valores referenciados en maestros con la opción de «en cascada»; rechazo en caso contrario
  - Inserción de valores según restricciones de dominio; rechazo en caso contrario;
  - Inserción, actualización y borrado de valores para ejecutar los correspondientes disparadores

Pruebas sobre Esquema

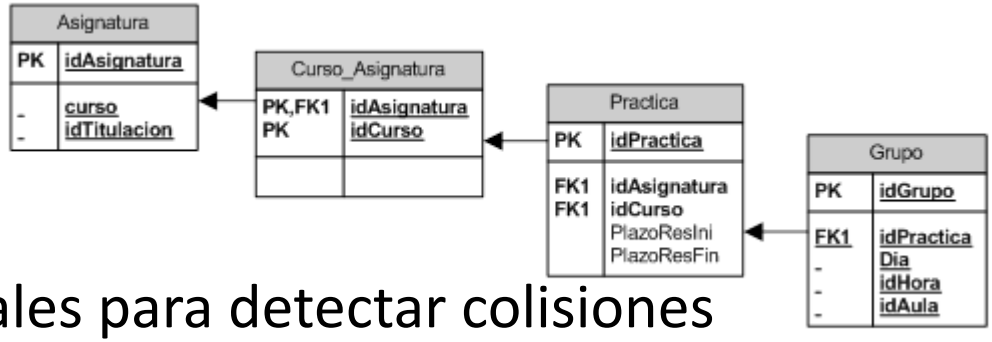
- Vistas, Procedimientos y funciones :
  - Comprobar errores semánticos [Brass y Goldberg, 2006] , SQL que no hacen lo que deben aunque sintácticamente estén bien escritos
    - SELECT: constantes, atributos duplicados, DISTINCT
    - FROM: joins innecesarias, uso de joins definidas en SQL92, condiciones olvidadas
    - WHERE/HAVING: condiciones inconsistentes, condiciones siempre falsas, condiciones entre valores de diferentes dominios, valores NULL
    - GROUP BY: grupos de una fila, grupo único, atributos innecesarios
  - Evitar uso de cursores
  - Vigilar el nivel de aislamiento de transacciones: Acceso concurrente a la información

Pruebas sobre transacciones

- Procedimientos almacenados:
  - Transacciones definidas
- Comprobaciones:
  - Ejecución de secuencias de caminos posibles (atención a transacciones anidadas)
  - Consistencia de la información
  - Validez de los datos
  - Situaciones de éxito con cero, una o más filas resultado
  - Situaciones de fallo con restauración del estado

Pruebas "Regla de Negocio": caso práctico*

- Realizar reservas de espacios para impartir clases prácticas de un centro docente
- Modelo de datos



- Requisitos funcionales para detectar colisiones

ID	DESCRIPCION
1	Requisitos funcionales
1.1	Proceso de reserva (no es objeto de pruebas, pero sirve como introducción al sistema)
1.1.1	(otros requisitos eliminados)
1.2	El sistema debe avisar al profesor antes de confirmar una reserva si esta tiene algún tipo de colisión con otras.
1.2.1	Una reserva está en colisión con otra asignatura si en algún caso la hora y día coinciden en un curso académico con otras de su misma titulación y curso (curso<>curso académico)
1.2.2	Si una reserva no indica fecha de inicio o fin se supone vigente desde el inicio hasta el fin de curso respectivamente. Si no se incluye ninguno se considera provisional y no debe provocar colisiones
1.2.3	Cuando se detecta una colisión el sistema mostrará todas las reservas que la causan
1.2.4	El profesor puede aceptar o cancelar la reserva después de obtener el informe de colisiones

* Ejemplo extraído de la ponencia Pruebas de Aplicaciones en Bases de datos impartida por Mª José Suárez-Cabal de la U. de Oviedo en el curso de verano Gestión Avanzada de Datos

Diseño de pruebas

- Definir «requisitos de prueba»: Determinar aspectos del sistema a probar y refinar hasta llegar a un nivel de detalle suficiente (caso de aplicación en disparadores)

- En este caso de estudio:
  - Requisito 1.2.1: «Una reserva está en colisión con otra asignatura si en algún caso la hora y el día coinciden en un curso académico de su misma titulación y curso»
  - Requisito 1.2.2: «Si una reserva no indica fecha de inicio o fin se supone vigente desde el inicio hasta el fin de curso, respectivamente. Si no se incluye ninguno se considera provisional y no debe provocar colisiones»

Requisitos de prueba (preliminares): requisito 1.2.1

- Técnica: Decisión/condición modificada (MC/DC)
  - Cada condición determina el valor de salida, sin que el resto cambie
- El requisito presenta condiciones separadas por Y (hora, día, titulación, curso y curso académico) → se deben dar las siguientes situaciones o requisitos de prueba:
  - Todas las condiciones son ciertas
  - Para cada condición, sólo esa es falsa

Definición de casos de prueba

- ¿Cuántos casos de prueba? Tantos como combinaciones de condiciones
- ¿Por qué están formados los casos de prueba?
  - Entrada:
    - Parámetros (nueva reserva)
    - Base de datos (reservas existentes)
  - Salida:
    - Filas con datos de reservas que producen colisión

	Asignatura			Curso_asigna		Practica					Grupo					
	idAsignatura	curso	idTitulacion	idCurso	idAsignatura	idPractica	idAsignatura	idCurso	plazoResIni	plazoResFin	idGrupo	idPractica	Dia	idHora	idAula	Colisión?
Parámet.	9	4	2	2004	9	99	9	2004	5	11	199	99	J	17	3	
1.1.1.1	10	4	2	2004	10	100	10	2004	7	9	200	100	J	17	3	SI
1.1.1.2.1	10	4	2	2003	10	101	10	2003	7	9	201	101	J	17	3	NO
1.1.1.2.2	12	5	2	2004	12	102	12	2004	7	9	202	102	J	17	3	NO
1.1.1.2.3	13	4	1	2004	13	103	13	2004	7	9	203	103	J	17	3	NO
1.1.1.2.4	10	4	2	2004	10	100	10	2004	7	9	204	100	V	17	3	NO
1.1.1.2.5	10	4	2	2004	10	100	10	2004	7	9	205	100	J	18	3	NO

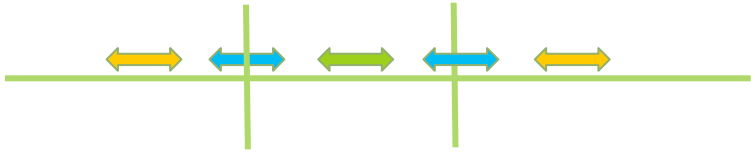
Requisitos de prueba (refinamiento): requisito 1.2.1

- Planteamiento: ¿están los requisitos del sistema suficientemente especificados? ¿falta o sobra algo?
- Necesidad de incluir pruebas para condiciones omitidas o que sobran
- Si el comportamiento del sistema no está claro, incluyendo este tipo de situaciones se debe determinar cómo debería ser
- En el requisito 1.2.1:
  - ¿Qué sucede si es la misma asignatura? ¿hay colisión?
  - El aula para la reserva ¿no se tiene en cuenta?

	Asignatura			Curso_asigna		Practica					Grupo					
	idAsignatura	curso	idTitulacion	idCurso	idAsignatura	idPractica	idAsignatura	idCurso	plazoResIni	plazoResFin	idGrupo	idPractica	Dia	idHora	idAula	Colisión?
1.1.1.3.2	9	4	2	2004	9	99	9	2004	7	9	198	99	J	17	3	NO
1.1.1.3.1	9	4	2	2004	9	97	9	2004	7	9	197	97	J	17	3	NO
1.1.1.4.1	10	4	2	2004	10	100	10	2004	7	9	210	100	J	17	4	SI

Requisitos de prueba: requisito 1.2.2 (I)

- Técnica: Partición en clases de equivalencia
- El requisito presenta condiciones sobre fechas: inicio y fin (intervalos) de la nueva reserva y de las existentes. Se darán las siguientes situaciones en función de las clases de equivalencia:
  - Intervalos cerrados (fechas de inicio y fin establecidas):
    - Inclusión total de un intervalo en otro
      - La nueva reserva (NR) incluida en alguna reserva de la base de datos (BD)
    - Inclusión parcial
      - NR a la derecha o izquierda del periodo establecido de BD
    - Los intervalos no coinciden
      - NR a la derecha o izquierda del periodo establecido en BD



Requisitos de prueba: requisito 1.2.2 (II)

- Más situaciones en función de las clases de equivalencia:
  - Intervalos abiertos (alguna fecha no establecida):
    - Intervalo inicio abierto
      - NR anterior a fecha fin BD
      - NR contiene fecha fin BD
      - NR posterior a fecha fin BD
    - Intervalo fin abierto
      - NR anterior a fecha inicio BD
      - NR contiene fecha inicio BD
      - BD posterior a fecha inicio BD
    - Ambos intervalos abiertos

Requisitos de prueba: requisito 1.2.2 (III)

- Técnica: Análisis de valores límite
- El análisis de clases de equivalencia se completa con análisis de valores límite en relación con los intervalos
- En lugar de realizar la prueba con cualquier elemento de la partición equivalente, se escogen los valores en los bordes de la clase.

Con estos casos de prueba, ¿podríamos localizar defectos?

- Más casos a probar:
  - No considerar el curso académico.
  - Sólo colisiones con asignaturas de la misma titulación.
  - Una práctica de una asignatura nunca colisiona con otra de la misma asignatura.
  - Problemas en los límites de los intervalos de fechas (cerrados): si coincide el día de fin con el de inicio se produce colisión.
  - Comprobación de los intervalos abiertos por el inicio o por el final.
  - Las reservas que no especifican ni plazo de inicio ni de fin nunca producen colisiones.
  - Coincidencia del aula propuesta con la de otra reserva.

Pruebas de seguridad

- Elementos del esquema a evaluar (por roles):
  - Validación de credenciales de usuario
  - Permisos sobre Consultas SQL
  - Datos de entrada de usuario e información de salida de usuario
    - Identificar datos de carácter personal [LOPD, 1999]
    - Anotar en log datos sensibles nivel 3 [LOPD, 1999]
- Copias de seguridad
  - Tareas de backup y restauración

Pruebas de rendimiento

- Rendimiento de las aplicaciones bajo diferentes cargas de trabajo:
  - Volumen de datos almacenados previsto
  - Número de usuarios potenciales concurrentes
  - Número de accesos diarios
  - Picos de carga previstos
  - Tiempo medio de respuesta por transacción
  - Uso de recursos (ancho de banda, memoria, cursores abiertos, etc.)

- Recomendaciones de pruebas a realizar
  - Estrés: rendimiento del sistema en el peor escenario durante cortos periodos de tiempo
  - Resistencia: rendimiento del sistema en condiciones continuas de carga elevada
  - Monitorización del peso de las consultas (duración o uso de recursos) y de cursores y conexiones abiertos