Desarrollo de Sistemas de Información

Tema 4. Diseño Físico

Marta Elena Zorrilla Pantaleón

Dpto. de Matemáticas, Estadística y Computación

Objetivos

- Satisfacer los requisitos del sistema optimizando la relación coste/beneficio.
- Se ha de tener en cuenta las características del gestor, del SO y del hardware.
- Esto se concreta en los siguientes objetivos:
  - Disminuir los tiempos de respuesta,
  - Minimizar el espacio de almacenamiento,
  - Evitar las reorganizaciones periódicas,
  - Proporcionar la máxima seguridad, y
  - Optimizar el consumo de recursos.

Criterios

- Integridad
- Disponibilidad
- Escalabilidad
- Facilidad de administración
- Mejora en rendimiento
  - Espacio en memoria y en disco
  - Tiempo en procesador
  - Tiempo en acceso a disco
  - Contención
  - Coste de procesos auxiliares

Dificultades

- Tipo y número de Usuarios
- Recursos limitados
- Imperfecciones en el SGBD (optimizador)
- Comunicaciones (red)
- Criterios contrapuestos

Tareas básicas del diseño físico

- Adaptación al SGBD
  - Organización de ficheros
  - Tipos de datos
  - Tablas/Vistas:
    - Tipo de estructura subyacente (tree, heap, …)
    - Distribución en ficheros
    - Particionamiento del esquema
    - Materialización de vistas
    - Desnormalización y redundacia de datos
  - Definición de Índices:
    - Parámetros y estructura de datos subyacente (hash, tree,…)
  - Comprobación e implementación de restricciones de integridad
  - Procedimientos/Disparadores
    - Abrazos mortales – nivel de aislamiento en transacciones

Tareas básicas del diseño físico

- Pruebas de rendimiento
  - Parámetros del sistema
  - Optimización de consultas
    - Actualización de estadísticas (update statistics).
    - Reorganización y reconstrucción de índices.
    - Reorganización de datos en ficheros.
  - Control de concurrencia
    - Nivel de aislamiento en transacciones
  - Recuperación
    - Configuración del LOG (full, simple, bulk)

Entradas y Salidas del Diseño Físico

- Entradas:
  - Lista de objetivos de diseño físico con sus correspondientes prioridades y cuantificación (a ser posible);
  - Esquema lógico específico;
  - Recursos de máquina disponibles;
  - Recursos de software disponibles (SO, middleware, ...);
  - Información sobre las aplicaciones que utilizarán la BD con objeto de optimizar el tiempo de respuesta.
  - Políticas de seguridad de datos.

- A partir de estas entradas, se producirán las siguientes salidas:
  - Estructura interna (esquema interno, generalmente solo accesible a través de parámetros de sintonización);
  - Especificaciones para el sintonizado (tunning) de la BD; y
  - Normas de seguridad.

Para Finalizar

NO existe un modelo formal general para el diseño físico, depende mucho de cada producto comercial concreto.

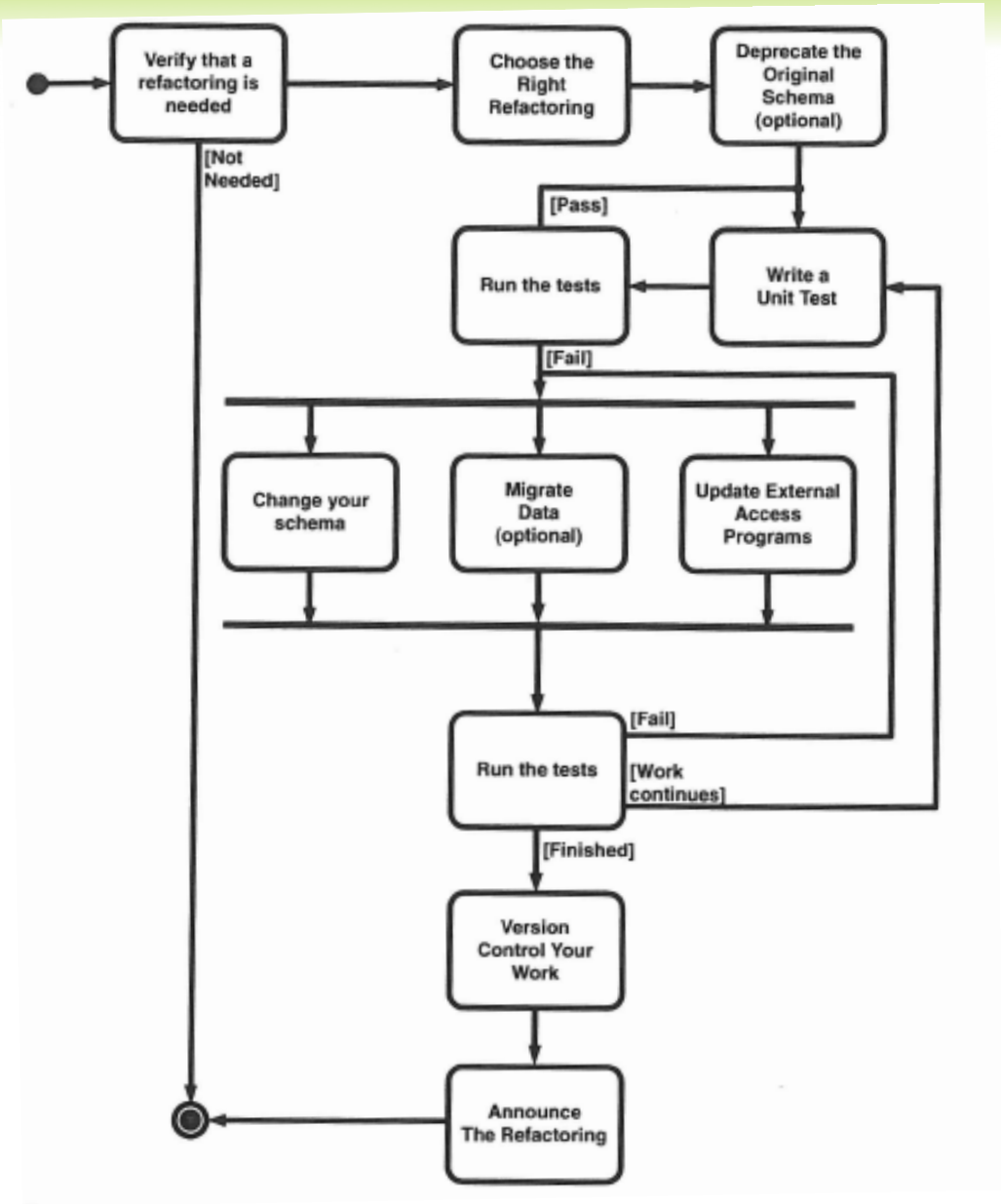
Refactorizado

- Def. (Fowler, 1999; Ambler, 2003) hacer pequeños cambios en el esquema para mejorar su diseño y facilitar su uso, pero sin cambiar la semántica.
  - Ej: AddPersons() por AddPeople()
- Cuestiones que requiere refactorizar:
  - Columnas multipropósito
    - Campos observaciones, fechas que cambian en función del momento dándoles una orientación semántica distinta.
  - Tablas multipropósito
    - Tabla cliente que recoge información de personas y organizaciones. Favorece la inclusión de muchos campos con valores nulos.
  - Datos redundantes
  - Tablas con muchas columnas o con muchas filas.
    - Reduce el rendimiento, mejor particionar
  - Columnas “inteligentes”
    - Identificador empleado dependa del Dpto en el que se encuentra, o del cliente en función del segmento al que pertenezca
  - Miedo a realizar un cambio, síntoma de que el esquema no es mantenible

Database Refactoring Categories (Ambler, 2003)

Database Refactoring Category	Description	Example(s)
Structural	A change to the definition of one or more tables or views.	Moving a column from one table to another or splitting a multipurpose column into several separate columns, one for each purpose.
Data Quality	A change that improves the quality of the information contained within a database.	Making a column non-nullable to ensure that it always contains a value or applying a common format to a column to ensure consistency.
Referential Integrity	A change that ensures that a referenced row exists within another table and/or that ensures that a row that is no longer needed is removed appropriately.	Adding a trigger to enable a cascading delete between two entities, code that was formerly implemented outside of the database.
Architectural	A change that improves the overall manner in which external programs interact with a database.	Replacing an existing Java operation in a shared code library with a stored procedure in the database. Having it as a stored procedure makes it available to non-Java applications.
Method	A change to a method (a stored procedure, stored function, or trigger) that improves its quality. Many code refactorings are applicable to database methods.	Renaming a stored procedure to make it easier to understand.
Non-RefactoringTransformation	A change to your database schema that changes its semantics.	Adding a new column to an existing table.

Proceso de Refactoring (Ambler, 2003)



Para Finalizar

Refactoring se favorece si hay desacople entre la capa de negocio de las aplicaciones y el gestor de bases de datos → Capa de persistencia.