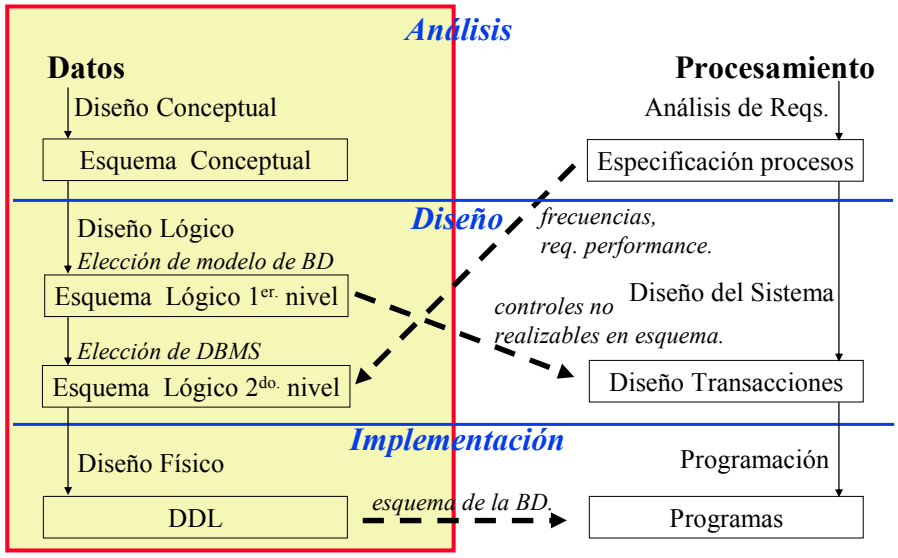
Diseño Avanzado de BDs

Diseño Lógico y Físico de Bases de Datos

Federico Piedrabuena
Instituto de Computación – Facultad de Ingeniería
Universidad De La República

Proceso de Diseño de BDs



Diseño Lógico

- **En que consiste ?**
  - Diseño de la base de datos en términos de un modelo lógico (p. Ej. Modelo Relacional).
    - » Traducir el esquema de datos conceptual en un esquema de datos lógico para un DBMS específico.

- **Objetivo:**
  - Obtener una representación del modelo conceptual que use de forma eficiente las facilidades de estructuración de datos y modelado de restricciones, disponibles en el modelo.

- **Problemas planteados:**
  - Mapeo desde Modelos Conceptuales.
  - Aplicación de información sobre transacciones y requerimientos de performance.
  - Buen diseño en el Modelo Lógico (ej. Normalización).
  - Conocimiento sobre procesamiento de operaciones.
  - Optimización del esquema lógico.

Diseño Lógico

- **Técnicas a aplicar:**
  - Pasaje de Mod. Conceptuales → Mod. Lógicos.
  - Técnicas de Diseño Relacional.
  - Técnicas de optimización del esquema relacional basado en:
    - » Información sobre volúmenes de datos.
    - » Procesamiento de operaciones en el DBMS específico.

Diseño Físico

- **En que consiste ?**
  - Diseñar el esquema físico a través del refinamiento del esquema lógico, para su implementación en un DBMS específico.
- **Objetivo:**
  - Implementar el modelo lógico empleando de forma eficiente las estructuras físicas del DBMS, de forma de obtener optimizar la performance del sistema.
- **Problemas planteados:**
  - Aplicación de información sobre transacciones y requerimientos de performance.
  - Conocimiento sobre procesamiento de operaciones.
  - Elección de estructuras físicas adecuadas para el DBMS.
  - Configuración de la BD.

Diseño Físico

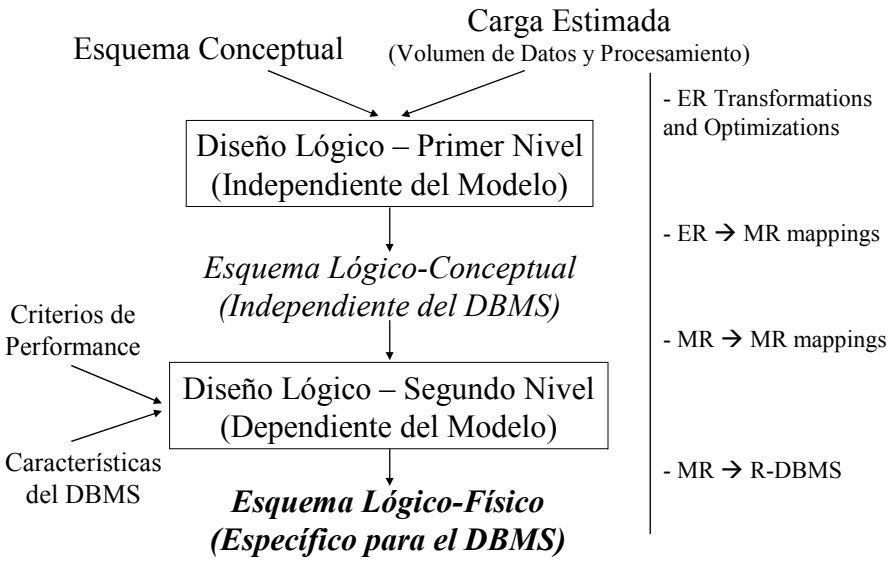
- **Técnicas a aplicar:**
  - Pasaje de Mod. Lógicos → Mod. Físico.
  - Refinamiento del almacenamiento basado en:
    - » Información sobre volúmenes de datos
    - » Estructuras físicas disponibles en el DBMS específico.
    - » Estrategias de procesamiento de operaciones en el DMBS específico.

Diseño Lógico

- **Temas:**
  - » Introducción.
  - » Diseño Independiente del Modelo.
  - » Optimizaciones.
  - » Diseño Dependiente del Modelo.
  - » Pasaje ER → MR.

- **Entrada:**
  - Esquema Conceptual
  - Descripción del modelo lógico objetivo y sus restricciones
  - Estimaciones de carga (volumen de datos), consultas y transacciones involucradas y su frecuencia
  - Criterios de performance
    - » Tiempo de repuesta
    - » Espacio ocupado por la BD
    - » Utilización de CPU e I/O
- **Técnica a aplicar:**
  - Se divide esta actividad en dos fases:
    - » Diseño lógico independiente del modelo (primer nivel)
    - » Diseño lógico dependiente del modelo (segundo nivel)

Diseño Lógico del Esquema BD



Diseño Lógico – Primer Nivel

- **Objetivo:**
  - Realizar transformaciones y optimizaciones al modelo conceptual.
- **Resultado:**
  - Esquema intermedio llamado conceptual-to-logical.
  - Implica elegir claves, realizar particionamiento de entidades, etc.

Carga de la BD

- **Modelar:**
  - Sobre cada Entidad/Relación:
    - » Promedio de datos y características de estos.
    - » Promedio de cardinalidades.
  - Sobre cada Transacción.
    - » Frecuencia de aplicación.
    - » Tipo: On-Line, Batch, ad-hoc.
    - » Para cada Entidad/Relación recorrida.
      - Tipo de operación: read/write.
      - Promedio de cada una de dichas operaciones.
      - Promedio de instancias implicadas por operación.

Carga de la BD

- **Criterios de aplicación:**
  - Esta no es toda la info. utilizable:
    - » Falta más info. sobre los atributos involucrados.
  - Transacciones On-Line más críticas y costosas que las Batch.
  - Regla de 20-80:
    - » El estudio de carga de la BD es complejo.
    - » El 20% de las Transacciones representa el 80% de la Carga.
    - » Poner el énfasis en ese 20% crítico.
- **Dificultades:**
  - Es difícil obtener toda esta información.

Constructores ER no Mapeables

- **El M-ER tiene constructores no fácilmente mapeables a los Modelos Lógico:**
  - Jerarquías de generalización.
    - » Varias entidades son sub-tipos de otra.
  - Atributos multivaluados.
    - » Son atributos que toman valores tipo conjunto.
  - Atributos calculados.
    - » Son atributos que toman valores según una fórmula.

Eliminación de Generalización

- **Problema:**
  - Los modelos lógicos no permiten representar generalizaciones y subconjuntos.
- **Tipos de jerarquías:**
  - Totales vs. Parciales.
  - Exclusivas vs. Solapadas.
- **Tres alternativas:**
  - Colapsar la jerarquía de generalización en una entidad.
  - Dejar sólo las sub-entidades.
  - Modelar la jerarquía con Relaciones.

Colapsar Jerarquía en Entidad

- **Técnica:**
  - Se toma la unión de todos los atributos de las sub-entidades y se los agrega a la super-entidad.
  - Se debería incluir un atributo de discriminación.
- **Ventajas:**
  - Genera pocas tablas (simpleza y menos joins).
  - Es fácilmente aplicable a los diferentes tipos de generalización.
- **Desventajas:**
  - Valores nulos.
  - Aumenta la cantidad de datos involucrados en operaciones.

Dejar Solo Sub-Entidades

- **Técnica:**
  - Se propagan los atributos de la super-entidad.
- **Ventajas:**
  - Cuando las operaciones:
    - » se aplican sobre sub-entidades.
    - » no requieren visión sobre super-entidad.
- **Desventajas:**
  - No aplicable cuando la generalización es:
    - » parcial: quedan objetos afuera.
    - » solapadas: se repiten objetos.
  - Se diluye la existencia de super-entidad y con ello la de subconjunto.
  - Se complican las operaciones que originalmente se aplicaban a la super-entidad.

Jerarquía Vía Relaciones

- **Técnica:**
  - Se mantienen todas las estructuras y se establecen relaciones entre la super-entidad y las sub-entidades explícitamente.
- **Ventajas:**
  - Mecanismo siempre válido.
  - Adecuado si unas operaciones se realizan sobre la super-entidad y otras sobre las subs-entidades.
- **Desventajas:**
  - Esquema sobrecargado.
  - Esquema poco claro.
  - Multiplica la cantidad de tablas ➔ Joins.
  - Agrega redundancia.

Atributos Multivaluados

- **Problema:**
  - Los modelos lógicos no permiten representar atributos multivaluados.
- **Alternativas:**
  - Una nueva entidad.
  - Varios atributos
    - » Uno por valor posible.
    - » Donde se registre cada valor.
  - Un único atributo.

Nueva Entidad

- **Técnica:**
  - Se representa el concepto como entidad y no como atributo.
- **Ventajas:**
  - Mecanismo mas flexible.
  - Adecuado si los valores posibles son acotados.
- **Desventajas:**
  - Esquema sobrecargado.

Varios atributos – Uno por valor posible

- **Técnica:**
  - Se representa el concepto como un atributo por valor posible.
- **Ventajas:**
  - Se evitan los joins.
- **Desventajas:**
  - Pueden introducirse varios nulos.
  - Si la cantidad de valores no es acotada, no se puede aplicar.
  - Las operaciones sobre este concepto deben considerar varios atributos.

Varios atributos – Uno por valor que se tome

- **Técnica:**
  - Se representa el concepto como un atributo por cada valor que pueda tomar la entidad (dueña del atributo).
- **Ventajas:**
  - Apropiada si los valores posibles no son acotados y hay operaciones que trabajan con el atributo.
- **Desventajas:**
  - Pueden introducirse varios nulos.
  - Como la cantidad de atributos es finita. No se le puede asociar a la entidad mas de “esa cantidad” de valores

Un atributo

- **Técnica:**
  - Se representa el concepto como un atributo que almacena todos los valores.
- **Ventajas:**
  - Técnica siempre aplicable.
- **Desventajas:**
  - Las operaciones se tornan mas complejas
  - Puede afectarse la performance

Atributos Derivados

- **Opciones:**
  - Mantenerlo calculado (materializado).
    - » Ventajas: no cálculos en consultas.
    - » Desventajas: Procesamiento adicional para mantener integridad con datos relacionados. Requiere mas espacio.
  - Mantener una función.
    - » Ventajas: La integridad ante updates es trivial.
    - » Desventajas: mayor calculo en consultas.

- **Elementos a tener en cuenta:**
  - » Operación realizada:
    - Dentro de la tupla
    - Varias tuplas (p.ej. totalización).
    - Varias tablas.
  - » Relación consultas/updates.
  - » Espacio necesario.

Optimizaciones en Modelo Lógico

- **Objetivo**
  - Reacomodar la distribución de instancias o atributos en base a las operaciones involucradas.

- **Técnicas:**
  - Particionamiento
    - » Separar instancias o atributos de una misma entidad.
  - Fusión
    - » Colapsar varias entidades en una sola.
  - Desnormalización
    - » Cambiar la organización del esquema.

Optimizaciones en Modelo Lógico

- **Particionamiento de tablas.**
  - Particionamiento Horizontal:
    - » Los datos se almacenan en varias tablas con igual esquema.
    - » La Unión de las tablas particionadas debe ser igual al conjunto original.
  - Particionamiento Vertical:
    - » Los datos se almacenan en varias tablas con conjuntos complementarios de atributos.
    - » Todas las tablas deben tener una misma clave.
    - » El Join de las tablas particionadas debe ser igual al conjunto original.
  - Aplicables:
    - » Por performance
    - » En base de datos distribuidas.
    - » Por temas de seguridad.

Optimizaciones en Modelo Lógico

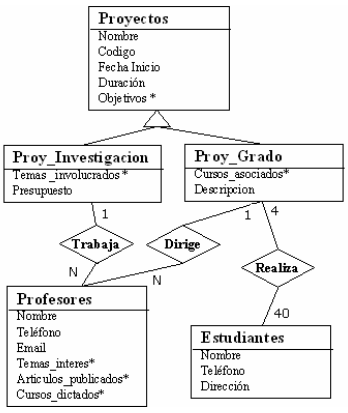
- **Fusión de entidades y relaciones.**
  - Datos de varias entidades se representan en una sola.
  - Se obtiene por Join o Unión de las entidades originales.
  - Permite reducir las operaciones de Join y por lo tanto aumentar la performance en transacciones con join.
  - Se generan diseños poco claros y/o con valores nulos.
  - Las operaciones de Update pueden ser mas complejas.

Optimizaciones en Modelo Lógico

- **Desnormalización**
  - Se reduce la cantidad de tablas por medio de tener tablas no normalizadas.
  - Permite reducir las operaciones de Join entre tablas.
  - Se introduce redundancia y posibles inconsistencias en los datos.
  - Se debe distinguir entre los problemas introducidos en cada caso.

Ejercicio

- **Información**
  - La generalización de Proyectos es total y exclusiva.
  - $1x10^{20}$ estudiantes, $8x10^{10}$ profesores y $8x10^{5}$ proyectos.
- **Operaciones:**
  - Consultas sobre Estudiantes que realizan proyectos de grado por Nombre.
  - Consultas sobre Profesores que trabajan en proyectos de investigación por Nombre, y temas de interés. Lo mismo sobre profesores que dirigen proyectos de grado.
  - Consultas sobre proyectos de investigación por cualquiera de sus atributos.
  - Consultas sobre proyectos de grado por cualquiera de sus atributos.
  - Consultas sobre datos completos de profesores.



Diseño Lógico – Segundo Nivel

- **Objetivo:**
  - Realizar transformaciones y optimizaciones dependientes del modelo.
- **Resultado:**
  - Esquema lógico dependiente del modelo.
  - Implica características y operadores específicos del DBMS.

Diseño Lógico Para el Modelo Relacional

- **ER vs. MR:**
  - ER no es muy diferente del MR.
    - » Pero deben realizarse mas simplificaciones:
      - Eliminar atributos compuestos y multivaluados.
      - Modelar los identificadores externos como internos.
      - Eliminar las relaciones (no existe el concepto de link).
- **Modelo Relacional:**
  - Propuesto por Codd en 1970.
  - Elementos básicos → Relación y Atributo
  - Los RDBMS son los mas populares del mercado (MS SQLServer, IBM DB2, Oracle, PostgresSQL, etc.).
  - Se ha adoptado al SQL como “El” lenguaje relacional.

Encare del Pasaje ER → MR

- **Alternativas:**
  - "Transformar ER → ER" + "Mapear ER → MR"
    - » Si el mapeo ER → MR es demasiado complejo.
  - "Mapear ER → MR" + "Transformar MR → MR"
    - » Evitar pasaje por ER no justificado.
- **Elementos a tener en cuenta en el pasaje:**
  - Criterios de performance.
  - Reglas y estrategias de pasaje ER → MR
    - » Constructores que no existen en MR.
  - Optimizaciones en diseño lógico.
    - » Particionamiento/fusión de estructuras.

Reglas y Estrategias ER → MR

- **Estrategias / Metodologías habituales:**
  - Pasaje esquema ER a MR.
    - » Se aplican reglas y eventualmente optimizaciones.
    - » Se verifica que la integridad del esquema Relacional
  - Ajustes al Esquema Relacional.
    - » Optimizaciones:
      - Particionamientos
      - Merge
      - Desnormalización.

Reglas y Estrategias ER → MR

- **Reglas generales:**
  - Entidad → Tabla
  - Atributo Monovaluado → Atributo
  - Atributo Multivaluado → Tabla / Atributo(s)
  - Atributo Estructurado → Tabla
  - Relación → Tabla
    - » Incluyendo claves de Entidades participantes
- **Optimizaciones:**
  - Reducir tablas en relaciones con cardinalidad 1.
  - "Achatar" atributos estructurados.

ER → ER" + "Mapear ER → MR

- **Esquema ER → Conjunto de Entidades y Relaciones**
  - Primer Paso
    - » Compuesto de dos actividades:
      - Eliminar los identificadores externos
      - Eliminar atributos compuestos y multivaluados
  - Segundo Paso
    - » Entidad → Relación
    - » Relación (many-to-many) → Relación
    - » Relación (one-to-one/one-to-many) → Agregar atributo a las relaciones existentes.
- **Pasaje de ER a MR es directo**

Identificadores

- **Eliminación de identificador externo**:
  - Se importa el identificador a la relación subordinada.
- **ER: Identificador → RM: Clave**
- **Clave:**
  - Similar al concepto de identificador en ER, pero en RM solo se aceptan identificadores internos.
  - Conjunto de atributos de una relación.
  - Identifica de forma única cada tupla de cada instancia de la relación.
  - Una relación puede tener mas de una clave
    - » Cada clave se llama **Clave Candidata**

Atributos Multivaluados

- **Opciones:**
  - Atributo Múltiple → Tabla (Diferente)
    - » Ventajas:
      - Se mantiene normalización.
      - Facilita el Join y selección por el campo.
    - » Desventajas:
      - Multiplicidad de tablas.
  - Atributo Múltiple → Multiples Atributos
    - » Ventajas:
      - Evita manejar multiples tablas.
    - » Desventajas:
      - Acotado en cantidad de ocurrencias.
      - Complica el Join y selección por el campo.

Atributos Multivaluados

- **Opciones:**
  - Atributo Múltiple → Atributo no en 1NF.
    - » Ventajas:
      - Unico atributo.
      - No hay limite (práctico) en la cardinalidad.
    - » Desventajas:
      - El Join y selección son muy complicados de realizar.

- **Elementos a tener en cuenta:**
  - Uso del atributo:
    - » Cardinalidad aproximada del atributo.
    - » Se realizan selecciones ?
    - » Se hacen Joins por el atributo ?
    - » Tiene restricciones de algún tipo ?

Relaciones

- **ER: Relación → RM: Operador *equi-join* entre atributos de tablas**
- **Join:**
  - Operador de Algebra Relacional para relacionar dos tablas.
  - Intervienen atributos comparables (mismos dominios).

Constraint

- **Integrity Constraint**
  - Condición especificada en el esquema que restringe los datos a ser almacenados en la instancia.
- **Variantes en RM:**
  - Domain Constraint
    - » Especifica el dominio de cada campo.
  - Key Constraint
    - » Especifica el conjunto de campos que identifica cada registros.
    - » Los valores de las **Primary Key** no pueden ser nulos.
  - Referential Integrity Constraint
    - » Se especifica entre pares de relaciones.
    - » Mantienen la consistencia entre tuplas de distintas relaciones.
    - » Se especifica mediante **Foreign Key**

Casos Problemáticos en ER → MR

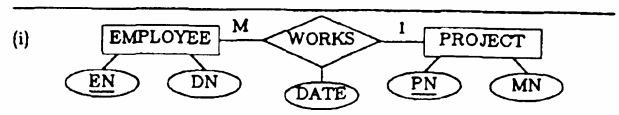
- **Casos límite:**

  [Markowitz & Shoshani – SIGMOD'89]

  - Relaciones N:1 no totales.
    - » Se tiende a no representar la relación explícitamente.
    - » Se pueden introducir valores nulos.
  - Caminos alternativos entre pares de Entidades.
    - » Dos Entidades están relacionadas de diferente forma mediante diferentes cardinalidades.
    - » No resulta claro como son las dependencias funcionales.

ER → MR : Relaciones N:1 no Totales

- **Ejemplo:**

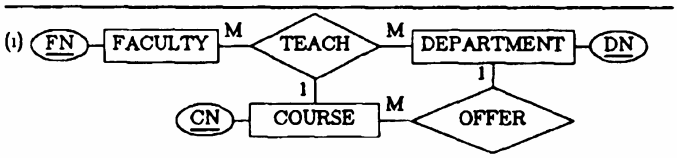


Abbreviations: EN=EMPLOYEE NAME, DN=DEPARTMENT NUMBER, MN=MANAGER NAME, PN=PROJECT NUMBER

Fig 1 Relational Representations for an ER Structure

ER → MR : Caminos Entre Tablas

- **Ejemplo:**



(ii)

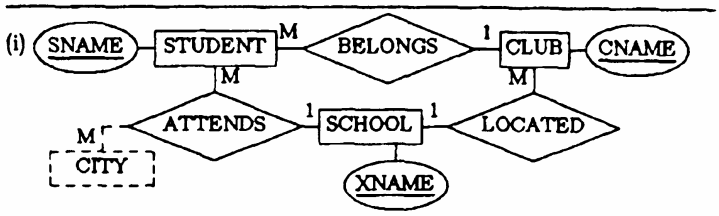
FACULTY	(FN)		
COURSE	(CN)	OFFER	(CN, DN)
DEPARTMENT	(DN)	TEACH	(FN, DN, CN)

Abbreviations CN=COURSE NUMBER, DN=DEPARTMENT NAME, FN=FACULTY NAME

Fig 2 Relational Representations for ER Structures

ER → MR : Caminos Entre Tablas

- **Ejemplo:**



(ii)

STUDENT	(SNAME)	BELONGS	(SNAME, CNAME)
CLUB	(CNAME)	ATTENDS	(SNAME, XNAME)
SCHOOL	(XNAME)	LOCATED	(CNAME, XNAME)

Fig 3 Relational Representation for an ER Structure

Diseño Avanzado de BDs

Diseño Físico

- **Temas:**
  - » Tratamiento de consultas por el DBMS.
  - » Estructuras de Diseño Físico.
  - » Particiones de la BD.
  - » Administración del almacenamiento.

Diseño Físico del Esquema BD

- **Entrada:**
  - Esquema Lógico.
  - Características del DBMS.
  - Listado de consultas, transacciones y procesos involucrados y su frecuencia.
  - Requerimientos de performance.
  - Recursos computacionales.
- **Técnica a aplicar:**
  - Especificar:
    - » Estructuras de datos.
    - » Estructuras de índices.
    - » Organización física general.
    - » Políticas de seguridad.
    - » Estrategia de administración.

Algoritmos de Ejecución de Consultas

- **Los DBMS implementan los operadores relacionales:**
  - Selección, Proyección, Unión, Resta, Join.
  - Se tienen diferentes tipos de algoritmos.
- **Uso de los diferentes algoritmos:**
  - Se ejecutan según el que se evalúe como más eficiente.
    - » Esto depende de como estén almacenados los datos (p. ej., con o sin índices).

Selección – Implementación

- **Tipos de algoritmos:**
  - ***Table scan :***
    - » Se recorre una tabla verificándose la condición.
    - » Son los menos eficientes.
  - ***Binary search :***
    - » Se aplica si la selección es una igualdad sobre un atributo clave por el cual el archivo esta ordenado. Mejora sobre el Table Scan.
  - ***Index scan :***
    - » algoritmos que buscan valores en un Índice.
    - » La eficiencia depende de la estructura del índice (árbol-B, hash).
    - » Hay variantes según el tipo de búsqueda.

Selección – Implementación

- **Variantes en Index Scan:**
  - **Busq. en índice primario o índice por hash para recuperar 1 registro.**
    - » Se aplica en condiciones de igualdad por campo clave.
  - **Busq. en índice clustereado para múltiples registros**.
    - Un *índice clustereado (clustering index)* agrupa en mismos bloques de disco todos los registros que tienen igual valor para el atributo indexado.
    - Se aplica si se recuperan varios registros con un mismo valor en un campo no clave.
  - **Busq. en árbol B.**
    - » Se aplica en condiciones de (>,<,<=,>=) para recuperar registros múltiples o únicos.

Selección – Condición

- **Variantes de Condiciones:**
  - Conjuntivas (AND)
  - Disyuntivas (OR)
- **Selectividad de una condición (siendo T la tabla):**
  - cant_tuplas(Cond(T))/cant_tuplas(T)
  - En el caso de ser por **igualdad de clave**:
    - » 1/cant_tuplas(T)
- **Interesa aplicar primero las condiciones con menor selectividad.**

Selección – Condición Conjuntiva

- **Algoritmos para condiciones conjuntivas:**
  - Una condición conjuntiva es el **AND** de condiciones simples.
  - **Selección conjuntiva.**
    - » Si un atributo involucrado en una única condición simple (en una cond. conjuntiva) permite la aplicación de alguno de los métodos anteriores, entonces se utiliza esta última para recuperar los registros y luego se chequea si satisfacen el resto de la condición.

Selección – Condición Conjuntiva

- Selección conjuntiva usando:
  - » Índice compuesto:
    - u Un *Índice compuesto (composite index)* es un índice por clave compuesta (p.ej., <pais,nro_pasaporte>).
    - u Si dos o mas atributos en la condición conjuntiva son atributos indexados en un índice compuesto, entonces se puede usar el índice directamente.
  - » Intersección de punteros a registros.
    - u Si existen índices para los atributos involucrados en las condiciones simples, se pueden recuperar los conjuntos de *punteros a registro* que satisfacen las condiciones simples y luego intersecarlos.
    - u Si habían otros atributos en la condición que no tienen índices, se deberá verificar la condición para los registros resultantes de la intersección.

Selección – Condición Disyuntiva

- u **Algoritmos para condiciones disyuntivas:**
  - Una condición disyuntiva es el **OR** de condiciones simples.
  - **Selección disyuntiva.**
    - » Son mucho mas complejas de procesar y optimizar.
    - » Solo se pueden optimizar:
      - u Dividiendo la consulta por las condiciones simples y luego uniendo los resultados.
      - u En las condiciones que no hay índices estamos obligados a hacer un TS.

Access Path

- ***Access Paths* de una consulta**
  - Caminos entre las estructuras (tablas y/o índices) que debe seguirse para recuperar los registros especificados.
  - Ejemplo:
    - » Si se tiene una tabla indexada por un atributo, los *Access Path* de una consulta por igualdad en el atributo son:
      - Recorrer la tabla.
      - Buscar en el índice, y luego acceder al registro en la tabla.

Elección del *Access Path*

- **La "Optimización" se basa en elegir un buen *Access Path* .**
- **En Selección con:**
  - Una condición simple:
    - » Se busca un *access path* a través de índices. Si no se encuentra se aplica TS.
  - Múltiples condiciones:
    - » Interesa estudiar la ***selectividad*** de las condiciones.
    - » Se debe utilizar primero el *access path* con menor selectividad (que devuelve la menor cantidad de registros).

Join – Implementación

- **Tipos de Join:**
  - **Two-Way Join:** Join entre dos tablas.
  - **Multiple-Way Join:** Join entre más de dos.

- **Variantes del Join:**
  - **Join:** Se retornan las tuplas de R y S en las que hay *matching*.
  - **Left Outer Join:** Se retornan todas las tuplas de R y de S solo en las que hay *matching*.
  - **Right Outer Join:** Se retornan solo las tuplas de R en las que hay *matching* y todas las de S.
  - **Full Outer Join:** Se retornan todas las tuplas de R y todas las de S.
  - En el Left, Right y Full cuando no hay *matching* se devuelven campos nulos.

- **Nos concentraremos en:**
  - Algoritmos para *Two-Way Joins* para *Joins.*
    - » Consulta tipo = JOIN <R.A=S.B> (R,S)

Join – Algoritmos

- **J1: Nested (inner-outer) loop. (fuerza bruta)**
  - » Para cada registro en R (outer loop), se recuperan los registros de S (inner loop) testeando si se satisface la condición de Join.

- **J2: Index access.**
  - » Si existe un índice (o hash) por uno de los atributos de join (p.ej., S.B), entonces recuperar cada registro en R y acceder a los de S a través del índice según el valor R.A.

Join – Algoritmos

- **J3: Sort-Merge.**
  - » Si los registros de R y S están físicamente ordenados por valores del atributo de Join, se pueden *aparear* los dos archivos macheando los registros que tienen igual en ese atributo.
- **J4: Hash-Join.**
  - » Los registros de R y S se *hashean* al mismo *hash-file* usando la misma función de hash. Se hace una pasada por la tabla con menos registros (R) para crear las *entries* en el hash file. Luego se hace una pasada por la otra tabla (S) agregando los registros si estos caen en las *entries* ya creadas.

Join – Algoritmos

- **Observaciones:**
  - – En la práctica se implementan accediendo a bloques y no a registros individuales.
  - – También importa el espacio de memoria real disponible para almacenar datos intermedios (p.ej., tablas de hash).

Proyección – Implementación

- **Permite extraer columnas de una tabla.**
- **Casos:**
  - En algebra relacional:
    - » Las relaciones se definen como conjuntos.
    - » El resultado de la proyección no contiene repetidos.
  - En la práctica:
    - » Los RDBMS omiten el costos paso de eliminar repetidos.
- **Resultado en la proyección de una tabla:**
  - Si los atributos proyectados incluyen la clave:
    - » Las tuplas resultado no contienen repetidos, son similares a las de la tabla origen.
  - Si los atributos proyectados no incluyen la clave:
    - » Se deben eliminar las tuplas repetidas.
      - Para esto se ordena (utilizando una función de hash) el resultado.

Set Operations – Implementación

- ***Operaciones*:**
  - Unión (**U**)
    - » Retorna las tuplas que están en al menos una de las tablas involucradas.
  - Intersección (∩)
    - » Retorna las tupas que están en todas las tablas involucradas.
  - Diferencia (**–**)
    - » Retorna las tuplas que están en R pero no en S, siendo la operación R – S.
  - Producto Cartesiano (**X**)
    - » Retorna una tupla por cada par de tulas de R y S, siendo la operación R X S.
- **Restricciones:**
  - Unión, Intersección, Diferencia
    - » Las tablas involucradas deben ser Union-Compatible.
      - Igual número de campos
      - Los campos tomados de izquierda a derecha deben coincidir en dominio.
  - Producto Cartesiano
    - » Si las tablas involucradas tienen campos con el mismo nombre se produce conflicto.

Set Operations – Implementación

- **Son costosas de ejecutar.**
  - X es especialmente costosa:
    - » Se debe evitar llevándolo a otras expresiones.
- **Las operaciones U, ∩ y – se ejecutan:**
  - Ordenando por los mismos atributos, y haciendo una recorrida lineal en la que se va efectuando la operación.

Optimización – Técnicas Avanzadas

- **Calculo de Costos.**
  - Basada en el cálculo del costo de ejecutar una operación.
  - Para esto se dispone de *funciones de costo* asociadas a cada algoritmo, que dan el costo de su aplicación en términos de cantidad de registros o bloques involucrados.
- **Tipos:**
  - Basado en estructura:
    - » Se emplea el catálogo para obtener la cantidad de tuplas por tabla, los índices, cluster tables, etc.
  - Basado en estadística:
    - » Se genera información en base a costos anteriores de operaciones o sub-operaciones.
  - Basado en semántica:
    - » Conocimiento del dominio que representa el esquema de base de datos.

Optimización – Técnicas Avanzadas

- **Optimización Semántica.**
  - Utilizar Reglas de Integridad de la base para optimizar las consultas.
    - » Por ejemplo:
      - Se piden los empleados que ganan mas que su supervisor, y hay una RI que dice que nadie puede ganar más que su supervisor.
      - Se deduce que la respuesta debe ser vacía por lo que no se ejecuta.
  - Problema:
    - » Búsqueda y análisis de las RI muy costosa.

Optimización – Decisiones

- **Decisiones dependientes del DBMS:**
  - Estructuras:
    - » Indices, Clusters.
  - Almacenamiento:
    - » Porcentajes de ocupación de bloques.
    - » Ubicación de tablas e índices.
  - Tunning del DBMS.
- **Considerar:**
  - Operaciones a ejecutar:
    - » Tipo, frecuencia, tablas involucradas, etc.
  - Restricciones:
    - » Performance, espacio.

Optimización – Factores Determinantes

- **Análisis de consultas:**
  1. tablas accedidas.
  2. campos con condiciones de selección.
  3. campos con condiciones de join.
  4. campos proyectados.

- **Acciones:**
  - Los campos tipo 2 son candidatos a indexar.
  - Los campos tipo 3 son candidatos a indexar :
    - » Si es accedido directamente en el join.
    - » Idem, si se trata de un Merge-Join.

Optimización – Factores Determinantes

- **Análisis de transacciones-updates:**
  1. tablas modificadas.
  2. tipo de operación: *update*, *insert, delete*.
  3. campos con condiciones de selección.
  4. campos modificados.

- **Acciones:**
  - Los campos tipo 4 son candidatos a NO indexar.
    - » Ya que su modificación implica modificar índices.
  - Los campos tipo 3 son candidatos a indexar.

Optimización – Factores Determinantes

- **Frecuencia esperada de invocación:**
  - Consultas.
  - Transacciones-update.
- **Restricciones de performance:**
  - Tiempo absoluto que demora una transacción.
  - Por ejemplo:
    - » Una transacción debe tomar menos de 5s en el 95 % de casos.

Optimización – Ejemplo

- **Tablas:**
  - Session
    - » 3.000 tuplas
  - Log
    - » 18.000 tuplas
- **Selección del join por session_id:**
  - Sin índice: 17 transacciones por segundo (t/s)
  - Con índices:
    - » B-Tree → 301 t/s
    - » Hash → 303 t/s

Optimización – Estructuras

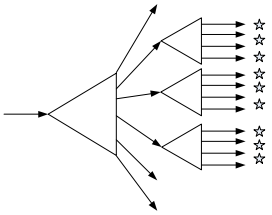
- **Objetivo: Acelerar Equi-Joins**
  - Almacenar contiguamente las tuplas que joinenan.
    - » Ejemplo: CLUSTER de *ORACLE*.
  - Crear índices por campos de join.
    - » Típicamente PK con FKs.
  - Reducir la cantidad de tablas involucradas:
    - » Fusionar tablas:
      - OJO con los valores nulos!
    - » Desnormalizar.
      - OJO con las anomalías!!!

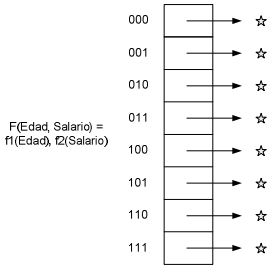
Optimización – Estructuras

- **Elección de índices:**
  - Los índices tienen puntos fuertes y débiles.
    - » Conocer sus características.
    - » Conocer características y uso de los datos.
  - Crear índices:
    - » Por atributos frecuentemente usados en proyección y condición.
    - » Si el tiempo de respuesta es un factor crítico.
  - Evitar índices:
    - » Por atributos que se modifican.
    - » Si el espacio es un factor crítico.

Optimización – Estructuras

- **Variantes de índices:**
  - Basadas en árbol
    - » B-tree → árbol
    - » B+-tree → árbol balanceado
    - » R-tree → similar al B-tree pero para información multidimensional



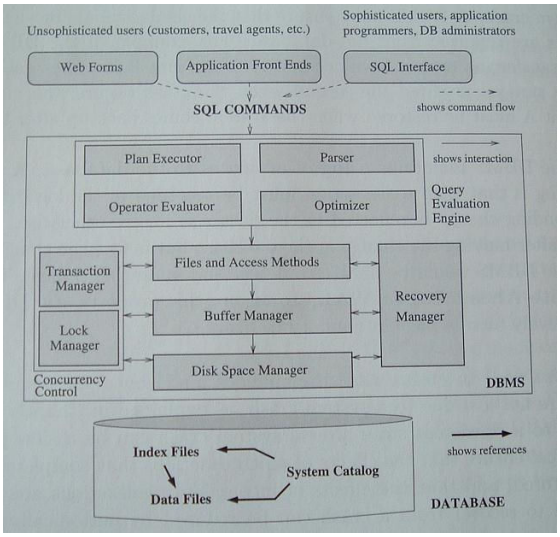


  - Basadas en hash
    - » Static Hashing → Número de bucket fijo
    - » Extendible Hashing → Número variable de bucket

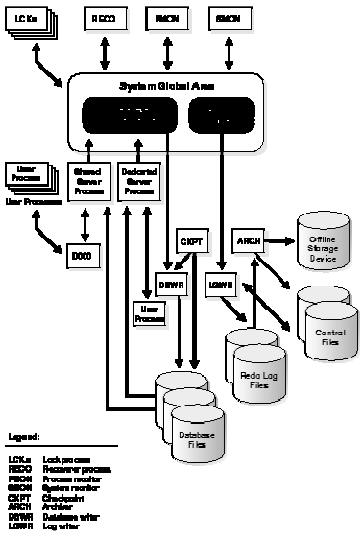
Optimización – Configuración

- **Factores:**
  - Equilibrio: Performance vs. Espacio utilizado.
    - » Disco y memoria.
  - Maximizar eficiencia de acceso a disco:
    - » Definir adecuadamente los tamaños de bloque.
    - » Definir estructuras adecuadas de disco a bajo nivel.
      - Striping, Mirroring, RAID, SAN, etc.
    - » Minimizar contención de disco.
  - Maximizar el uso de memoria:
    - » El tamaño del bloque de las estructuras de memoria deben ser múltiplo de la de disco.
      - Buffer: Espacio temporal para las lecturas de disco.
  - Locks
    - » Row level, Page level, Table level.

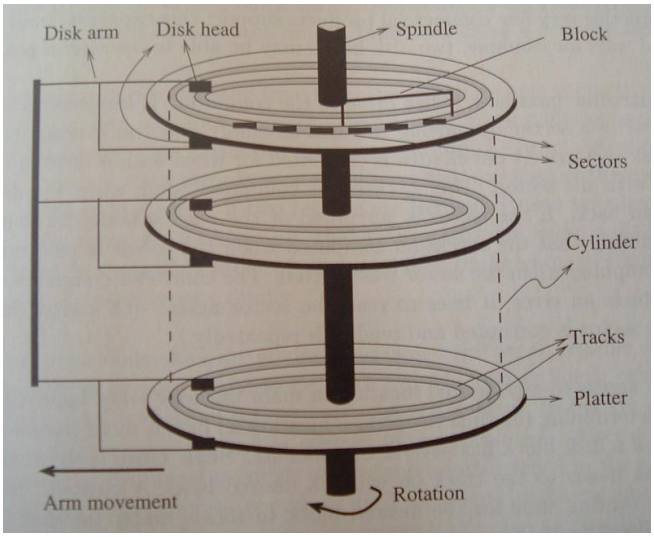
Arquitectura de un DBMS



Arquitectura de Oracle



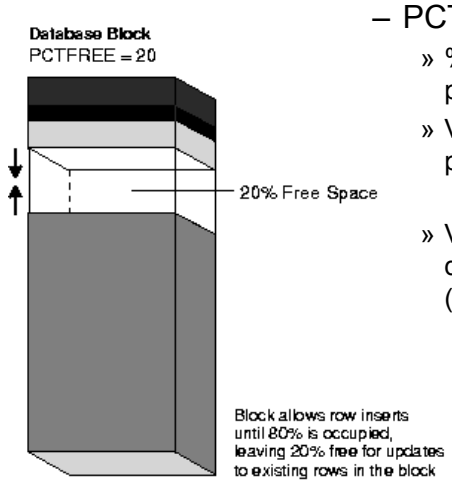
Estructura de Disco



Almacenamiento

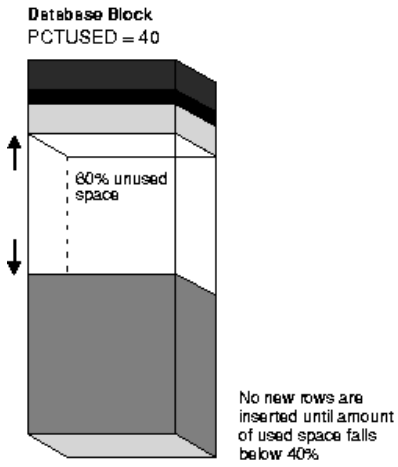
- **Ejemplo:**
  - PCTFREE & PCTUSED de Oracle:
    - » Determinan cuando los bloques de un segmento son tenidos en cuenta para inserciones.
      - Determinan si el bloque es parte de la Free List y la posición en ella.
    - » Los valores de estos parámetros se dan en la creación y alteración de tablas y clusters.
  - Free Lists de Oracle:
    - » Lista de bloques que contienen espacio libre.
    - » Se emplea para registrar los bloques que tienen suficiente espacio libre para otra tupla.
    - » Cada tabla tiene una Free List asociada.

Almacenamiento

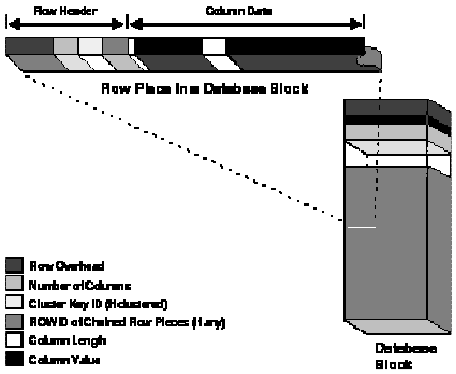


- PCTFREE:
  - » % del bloque que debe reservarse para futuras modificaciones.
  - » Valores bajos mejora la performance de fullscan.
    - u Mas tuplas por bloques.
  - » Valores altos reduce la probabilidad de fragmentación de registros (chaining).
    - u Espacio libre mas grandes por bloque.

Almacenamiento



- PCTUSED
  - » % por debajo del cual se considera el bloque suficientemente libre.
  - » Valores bajos reducen la probabilidad de fragmentación de registros.
    - u Generalmente hay espacio para todo el registro.
  - » Valores altos reducen la cantidad de espacio libre.
    - u El espacio libre es desaprovechado ya que no permite inserciones.



- http://www-rohan.sdsu.edu/doc/oracle/server803/A54643_01/ch5.htm#2680

Almacenamiento

- **Factores Determinantes**
  - Tipo de Registro:
    - » Tamaño fijo o variable
      - Ejemplo: Tamaño fijo → PCTFREE=0 (No es necesario reservar espacio para updates)
    - » Dominio de los campos
      - Ejemplo: Si la representación subyacente de los tipos de dato de los campos no admite crecimiento → PCTFREE=0
  - Operación sobre la tabla:
    - » Modificaciones o consultas
      - Ejemplo: Si las inserciones son poco frecuentes → PCTUSED=100%

Almacenamiento

- **Particiones de datos:**
  - Existen particiones lógicas y físicas.
    - » Striping de tablas para accesos concurrentes.
    - » Separar índices de tablas de datos.
- **Criterios de diseño:**
  - Maximizar acceso concurrente:
    - » Distribución en diferentes discos.
  - Seguridad.
  - Maximizar localización de datos.
    - » Minimizar fragmentación en diferentes particiones.
  - Flexibilidad:
    - » En creación/borrado de estructuras.

Almacenamiento – Archivos de Datos

- **Heap File**
  - Registros almacenados en forma aleatoria.
- **Sorted File**
  - Registros almacenados en forma ordenada según una secuencia de campos.
- **Hashed File**
  - Registros almacenados en base a una función de hash según una secuencia de campos.
- **Mejoras**
  - Compresión a bajo nivel.

Almacenamiento – Archivos de Datos

- **Costos:**

Tipo Archivo	Scan	Equality Search	Range Search	Insert	Delete
Heap	B.D	0,5.B.D	B.D	2.D	Search+D
Sorted	B.D	$D.\log_2 B$	$D.\log_2 B$+# matches	Search+ B.D	Search+B.D
Hashed	1,25.B.D	D	1,25.B.D	2.D	Search+D

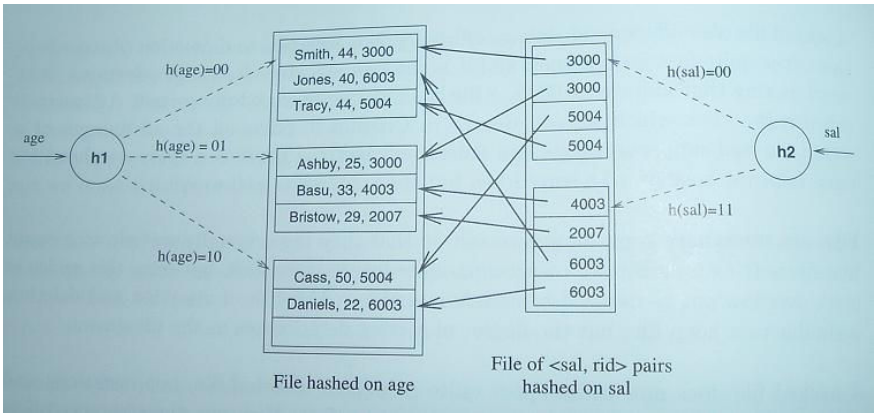
B → Número total de páginas.

D → Tiempo promedio de lectura o escritura de una página de disco.

Almacenamiento – Archivo de Indices

- **Data Entry en un Indice**
  - Un Data Entry *k** permite obtener uno o mas registros con valor de clave *k*.
- **Variantes:**
  - Un Data Entry *k** es un registro de datos (con clave de búsqueda *k*).
  - Un Data Entry es la pareja (*k*, rid).
  - Un Data Entry es la pareja (*k*, rid-list).

Almacenamiento – Archivo de Indices



Almacenamiento – Archivo de Indices

- ***k**** **(Registro de Datos)**
  - No hay necesidad de almacenar por separado el índice de los datos.
  - Se puede ver el índice como una forma especial de organización del archivo de datos.
- **(*k*, rid) y (*k*, rid-list)**
  - Son independiente de la organización del archivo de datos.
  - (*k*, rid-list) Ofrece mayor eficiencia en la utilización del espacio pero es una estructura de tamaño variable.

Diseño Avanzado

- **Temas:**
  - » DDL, DML
  - » Catálogo
  - » Seguridad
  - » Backup & Recovery

- **Data Definition Language (DDL):**
  - – Este subconjunto de SQL permite la creación, borrado y modificación de tablas y vistas.
    - » Si bien el estándar no discute otras estructuras como índices o secuencias, las implementaciones las consideran.
  - – Permite especificar permisos de acceso o privilegios.
- **Data Manipulation Language (DML):**
  - – Este subconjunto de SQL permite realizar consultas, inserciones, borrados y modificaciones.

Aspectos del SQL

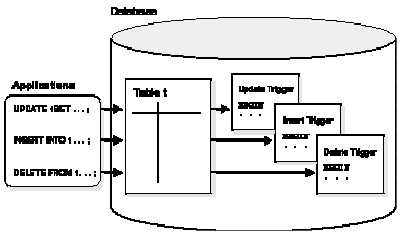
- **Triggers:**
  - SQL:1999 incluye soporte para triggers.
  - Son acciones ejecutadas por el DBMS cuando cambios en los datos cumplen las condiciones especificadas.
- **Seguridad:**
  - SQL provee mecanismos para el control de acceso a los objetos de la base de datos.
- **Manejo de Transacciones:**
  - Varios comandos permiten al usuario manejar explícitamente aspectos de ejecución de una transacción.

DDL y DML – Catálogo

- **Por Tabla:**
  - Nombre de tabla, Nombre de archivo, Estructura de archivo
  - Nombre de campo, Tipo de dato
  - Nombre de cada índice
  - Restricción de integridad (ejemplo PK, FK)
- **Por Indice:**
  - Nombre de índice, Estructura de índice
  - Atributos clave de búsqueda
- **Por Vista:**
  - Nombre de vista y definición
- **Estadística (por Tabla e Indice):**
  - Cardinalidad
  - Tamaño
  - Altura del índice
  - Rango del índice

Trigger

- **Descripción**
  - Es un procedimiento invocado automáticamente por el DBMS en respuesta a ciertos cambios.
  - Está asociado a una tabla y puede ser activado por inserciones, modificaciones y borrados.
  - Comprende la misma transacción que la operación que lo activó
- **Consiste en:**
  - **Evento:** Cambio en la base de datos que activa el trigger.
  - **Condición:** Consulta o test que corre cuando el trigger se activa.
  - **Acción:** Procedimiento que se ejecuta cuando un trigger es activado y la condición es verdadera.



Trigger

- **Uso:**
  - Generar automáticamente valores derivados de columnas
  - Prevenir transacciones inválidas
  - Políticas complejas de seguridad
  - Integridad referencial in a bases de datos distribuidas
  - Especificar reglas de negocio complejas
  - Registro de eventos
  - Auditoria
  - Replicación de tablas
  - Obtener estadística del uso de tablas

Seguridad

- **Objetivos:**
  - Privacidad, Integridad, Disponibilidad
  - Controlar el uso de recursos
    - » Disco, CPU, memoria
- **Access Control**
  - Discretionary
    - » Basado en privilegios.
      - Insert, Update, Delete y Select sobre objetos.
      - Grant y Revoke.
  - Mandatory
    - » Basado en etiquetas
      - Cada objeto tiene una *Clase de Seguridad* asignada y los usuarios tienen *Niveles de Permisos*.
      - El acceso se determina en función de reglas en base a las Clases de Seguridad y los Niveles de Permisos

Seguridad

- **Mecanismos:**
  - Vistas
  - Roles
  - Encriptación
    - » Datos, comunicación, archivos, etc.
  - Auditoria
    - » Pueden implementarse con Triggers

Backup & Recovery

- **Tipo de Errores:**
  - De usuario
  - De procesos o sentencias
    - » Error en el manejo de sentencias o procesos
  - De instancia
  - De disco

Backup & Recovery

- **Backup:**
  - Política acorde a las necesidades del sistema que utiliza la base de datos.
    - » Disponibilidad
    - » Tiempo de caída
  - Frecuencia
  - Tipo de Backup
    - » Online, Offline
- **Recovery:**
  - Recuperar físicamente la base de datos
  - Roll-forward: Rehacer transacciones terminadas que no se persistieron)
  - Roll-back: Deshacer transacciones no culminadas.

Proceso de Diseño de Base de Datos

- **Pasos:**
  - Recolección y análisis de requerimientos
  - Diseño conceptual
  - Selección del DBMS
  - Diseño lógico
  - Diseño físico
  - Implementación de la base de datos