Fundamentos de Bases de Datos

Resumen práctico

Diego González
2014
diesgomo@gmail.com

Nerd

1. Bases de Datos

1.1 Definiciones Básicas

- *Dato*:
  Representación simbólica que ante la inexistencia de un contexto, carece de significado
- *Información*:
  Datos interpretados (que poseen una semántica)
- *Sistema de Información (SI)*:
  Conjunto de componentes que interactúan con el objetivo de almacenar, recuperar y procesar datos e información con el fin de crear nueva información
- *Base de Datos (BD)*:
  Conjunto de datos relacionados entre sí
- *Metadatos:*
  Reglas que determinan si una información pertenece a una BD
- *Sistema Manejador de Bases de Datos (DBMS)*:
  SW que administra BDs
- *Modelo*:
  Lenguaje que permite describir:

Aspecto	Modelo	
Estructuras de Datos	De datos	Conceptual
Restricciones de integridad		
Operaciones (sobre la estructura)		

1.2 Bases de Datos

- Ciclo de vida de una BD:

Definición	→	**Construcción**	→	**Manipulación**
(Diseño conceptual)		*(Diseño lógico)*		*(Diseño físico)*
Estructura Restricciones		Carga de datos iniciales		Explotación de datos

- Ventajas de uso de BDs:

Aspecto	Descripción	Ventajas
Definición	De los datos abstractamente	• Centralización • Abstracción (de programas-datos/operaciones) • Múltiples vistas
Construcción	Programación de la BD	• Eficiencia de consultas • Enfoque declarativo de restricciones e integridad • Estandarización[1]
Manipulación	Explotación de la BD	• Datos compartidos • Control de concurrencia • Seguridad • Tolerancia ante fallas • Persistencia de datos consistentes (incluso de objetos)

[1] De modelos y lenguajes que usan la BD

- Uso no adecuado de BDs:

Aspecto	Descripción
Costo de inversión	Alto en HW, SW y capacitación
Costo de administración	De la BD y el DBMS
Naturaleza de los datos	Pocos datos o muy estables en el tiempo
Performance crítica	Sistemas de tiempo real
No hay concurrencia	

- Componentes:

Esquema	Definición de la BD. Totalmente equivalente al Modelo Conceptual
Instancia	Estado de la BD y sus datos en un momento determinado

1.3 Actores

- Primarios:
  Aquellos que interactúan tanto con la DB como con su información:

Actores	Funciones
Administradores	Autorizar, monitorear y coordinar
Diseñadores	Identificar los datos que se almacenarán y sus estructuras
Desarrolladores	Implementar los mecanismos de acceso
Usuarios finales	• Casuales • Paramétricos: Utilizan transacciones enlatadas • Sofisticados • Independientes: Operan mediante interfaces sencillas

- Secundarios:
  Aquellos que no están interesados en el contenido de la BD:

Actores	Funciones
Implementadores	De módulos e interfaces de la DBMS
Operadores de Sistemas	Ejecutan las políticas definidas por los diseñadores

1.4 Modelos de datos

- Modelo de datos:
  Colección de conceptos utilizados para describir la estructura de una BD
- Categorías:

Modelo	Descripción
Conceptual	Describe la relación y características de los datos
Lógico	Representa las características indicadas en el Modelo Conceptual. Ejemplos: Modelo Relacional Modelo Dato-Objeto
Físico	Describe cómo el modelo lógico es implementado en una estructura de archivos

- Arquitectura de tres esquemas:

#	Esquema	Descripción	Lenguaje	
			Acrónimo	Sigla
1	**Interno**	Almacena la estructura física de la BD	DDL	Data Definition Language
2	**Conceptual**	Describe la estructura de toda la BD y sus usuarios	SDL	Storage Definition Language
3	**Vistas de Usuario**	Describe determinada parte de la BD que es de interés para determinado grupo de usuarios	VDL	View Definition Language

- *Mapeo*:
  Acto de transformar y transmitir las consultas entre los esquemas de distintos niveles
- *Independencia de datos*:
  Habilidad de cambiar el esquema de un nivel sin afectar al esquema del nivel superior[2]
- *DML (Data Manipulation Language)*:
  Familia de lenguajes de manipulación de BD:

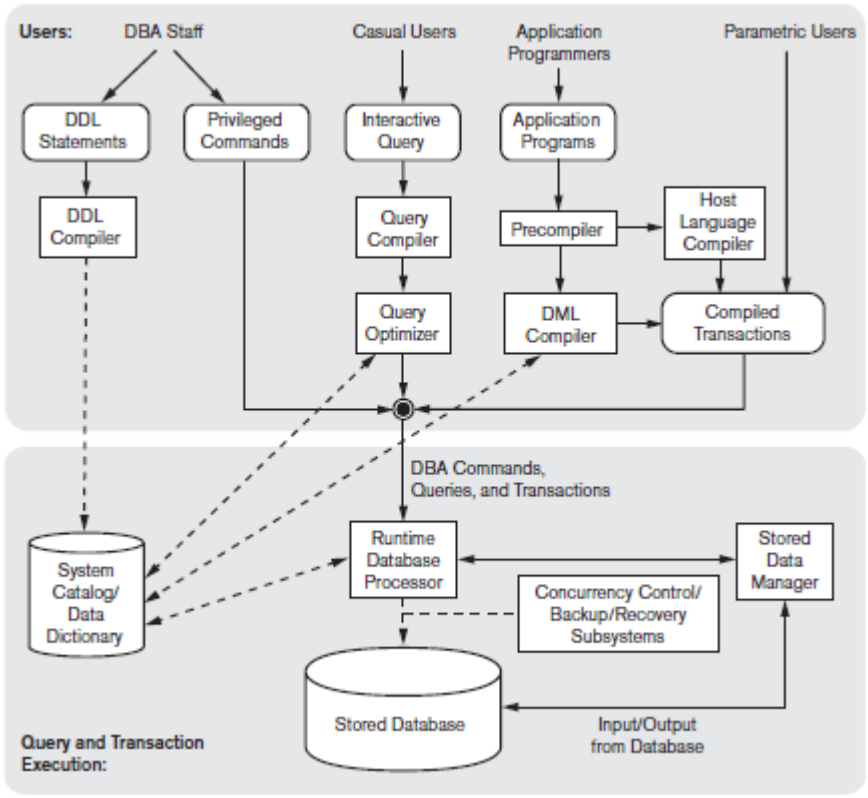
Categoría	Descripción
Procedural	Incrustado en lenguajes de programación de propósito general
Consulta	Utilizado de forma independiente e interactiva

- *SQL (Simply Query Language)*:
  Lenguaje que incluye a las familias DDL, SDL, VDL y DML

1.5 DBMS

- Interfaces de una DBMS:
  Basada en Menúes — Formulario — Gráficas
  Lenguaje Natural — Entrada y Salida de Voz — Usuarios Paramétricos
  Específicas del DBA

- Módulos y Componentes:



- Utilidades del DBMS:
  - Carga de archivos de datos
  - Copia de Seguridad de la BD
  - Organización y almacenamiento de la BD
  - Monitoreo de Performance

[2] Sólo el mapeo se ve afectado por el cambio

- Clasificación:

Criterio	Clasificación
Modelo de Datos	• Relacional • Dato-Objeto • Jerárquica • Legacy • Objeto-Relacional • XML-nativo
Cantidad de Usuarios	• Único • Multiusuario
Distribución	• Centralizado • Distribuido[3]
Tipo de Ruta de Acceso	
Especialización	• Propósito General • Propósito Específico • OLTP (Online Transaction Process)[4]

[3] Esta categoría, a su vez, puede clasificarse en **homogéneo** (utiliza el mismo SW en todas las locaciones) y **heterogéneo**

[4] Soporte a muchas transacciones concurrentes a bajo costo de demoras

2. Diseño Conceptual

2.1 Normas Básicas

- *Diseño conceptual*:
  Etapa en la que se construye un esquema conceptual en un lenguaje de alto nivel, definiendo el dominio del problema

- Construcción:
  1. Estudio de la realidad
  2. Especificación en lenguaje de alto nivel
  3. Validación del resultado

- Componentes:

Origen	Componente	Descripción
Conceptos	Conjuntos	Elementos de interés que por sus características, es conveniente la agrupación
	Relaciones	Entre conjuntos
	Restricciones de integridad	Validan los elementos que pueden pertenecer a una relación
Términos	Atributo	Características comunes de los elementos de un conjunto
	Cardinalidad	De elementos de un conjunto relacionados con el origen: N:1, N:N

- *Totalidad*:
  Una relación es total respecto de un conjunto, si todos sus elementos están en dicha relación

- Principios:

100%	Un esquema conceptual de un problema debe representar todos sus aspectos relevantes
Conceptualización	Un esquema conceptual no debe tener elementos de implementación

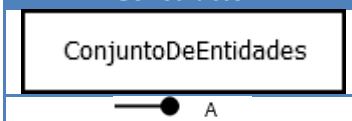
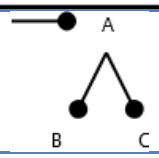
2.2 Modelo Entidad-Relación

- *Modelo Entidad-Relación (ER)*:
  DDL gráfico utilizado para la representación de estructura y restricciones de integridad[5]

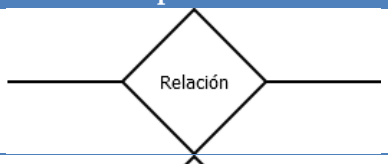
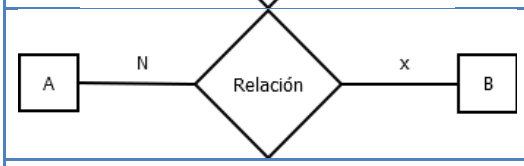
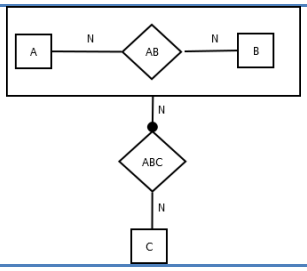
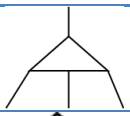
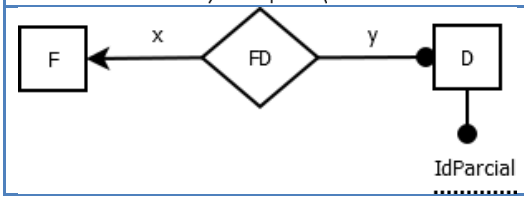
- Construcción:
  1. Identificar elementos del problema
  2. Identificar relaciones entre los elementos
  3. Representar propiedades de los elementos
  4. Especificar restricciones

[5] Las mismas se realizan en lógica de primer orden o teoría de conjuntos

- Constructores:

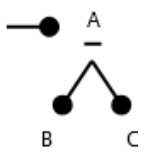
Constructor	Interpretación
	**Conjunto de Entidades**
	**Atributos:** Función entidad-valor[6]: $A: ConjuntoElementos \rightarrow Bs \times Cs$ $B: A = Bs \times Cs \rightarrow Bs$
	**Atributo Multivalorado:** compuesto de muchos valores del mismo dominio
	**Atributo Determinante:** Actúa como "identificador"[7], no permitiendo que dos entidades del conjunto posean el mismo valor en ese atributo[8]

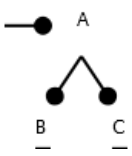
- Operadores:

Operador	Interpretación
	**Relación** entre dos o tres conjuntos de entidades (las relaciones triples pueden verse como la intersección de entidades en un Diagrama de Venn)
	**Cardinalidad** (en el sentido de lectura): N — Cualquier cantidad x — x cantidades como máximo ● — Todo elemento debe estar en la relación
	**Operador de "casting"** que reinterpreta su contenido como un nuevo conjunto de entidades. Ejemplo: 
	**Especialización**: Indica que los "hijos" son subconjuntos[9] de los "padres"
	**Entidad Débil:** Para su identificación ("razón de ser") es necesario viajar hacia la entidad fuerte. Las entidades débiles pueden o no poseer un identificador parcial (que los identifique dentro de todas las relaciones con las entidades fuertes)

[6] Para cada entidad, devuelve un valor. Pueden pensarse como "tipos abstractos"
[7] Estos son necesarios para cada conjunto de entidades: cada entidad debe ser identificable por un atributo determinante
[8] En particular, prestar atención en las combinaciones:

La combinación de B y C conforman un atributo "identificatorio"





No puede repetirse ningún B independientemente de que los C sean distintos y viceversa

[9] En el sentido matemático; por tanto, no se cumple si el cubrimiento del "padre" por sus "hijos", ni que los mismos sean mutuoexcluyentes

- Autorelaciones:
  Deben tener un rol en cada arco. En las restricciones se debe indicar si se desea (+) o no (-):
  - \- Ciclos
  - \- Reflexividad
  - \+ Transitividad

2.3 Calidad de los Esquemas Conceptuales

- Medidores de calidad:

Maximizar	Completitud Correctitud Minimalidad
Balancear	Expresividad Explicitud

- Completitud:
  Representa todas las características del problema

- Correctitud:
  i. Sintáctica
  ii. Semántica

- Minimalidad:
  Cada elemento del problema aparece una sola vez en el esquema

- Expresividad:
  Facilidad de comprensión utilizando semántica del modelo

- Explicitud:
  No utiliza más formalismos que el diagrama ER

3. Modelo Relacional

3.1 Estructuras

- *Dominio:*
  Conjunto de valores atómicos

- *Relación:*

Esquema	$R(A_1, \dots, A_n)$	R nombre de la relación A_i atributo i de dominio D_i
Instancia	$r(R)$	Conjunto[10] de tuplas que cumplen con el esquema de R

- *Tupla*:
  Función elemento de $r(R)$ que, dado un atributo del esquema de R, retorna su valor

- *Esquema Relacional (Tablas)*:
  Conjunto de esquemas de relación

3.2 Restricciones de Integridad (RI)

- Restricciones de Dominio:
  Tipado de cada Dominio

- Restricciones de Claves:

Tipo	Definición
Superclave	Subconjunto de atributos de una relación que es único para cada tupla posible
Clave[11]	Superclave minimal
Clave Foránea (FK)	Atributo en una relación que es clave en otra relación (y cuyos valores se corresponden a ella)

- *RI*:
  Conjunto de Restricciones de Dominio y Restricciones de Claves

- Validez de una BD:
  Una BD es válida sii satisface todas sus RI

- Operaciones de modificación:

Operación	Sintaxis	Conflictos con RI
Inserción	`INSERT` *tupla* `INTO` *R*	
Eliminación	`DELETE FROM` *R* `WHERE` *condición*	Cuando haya una FK sobre *R*

Actualización	`UPDATE` *R* `SET`
	atributo1 = *valor1*
	…
	atributoN = *valorN*
	`WHERE`
	condición
		Al intentar actualizar una clave o FK

[10] Por tanto no hay elementos ni ordenados ni repetidos
[11] Éstas se indican subrayando los atributos en el esquema

3.3 Cálculo Relacional de Tuplas: Sintaxis y Semántica

- Consultas:
  Especificación de un conjunto de tuplas por comprensión sobre el universo de tuplas:

$$\{\langle t_1, \dots, t_n\rangle / \varphi(x_1, \dots, x_n)\} \quad \Big| \quad t_i = \begin{cases} x_i.A_k \quad \text{con} \quad \begin{cases} x_i \text{ variable libre de } \varphi \\ A_k \text{ atributo de la tupla de una tabla} \end{cases} \\ c_i \text{ constante} \end{cases}$$

φ fórmula de lógica de primer orden tal que $FV(\varphi) = \{x_1, \dots, x_n\}$

- Términos de construcción de φ:
  x_i variable c_i constante (de algún Dominio) $x_i.A_j$ con A_j atributo

- Fórmulas de construcción de φ:

Fórmula	Semántica	
$\boldsymbol{t_i\ operador\ t_j}$[12]		
$\boldsymbol{tabla_j(x_i)}$	Verdadera sii $\exists x_i$: $x_i \in tabla_j$ en la BD	
$(\boldsymbol{\varphi\ conectorLógico\ \psi})$	Verdadera si lo es en el momento de la consulta	
$(\neg\boldsymbol{\varphi})$		
$\exists \boldsymbol{x_i}.\boldsymbol{\varphi}$	Equivalente: $\exists t \in P.\varphi$	$\exists t.(P(t) \wedge \varphi)$
$\forall \boldsymbol{x_i}.\boldsymbol{\varphi}$	$\forall t \in P.\varphi$ $(\forall t \in P \wedge t.a = b).\varphi$	$\forall t.(P(t) \rightarrow \varphi)$ $\forall t.(P(t) \wedge t.a = b \rightarrow \varphi) \wedge \exists s.(P(s) \wedge s.a = b)$

3.4 Cálculo Relacional de Tuplas: Seguridad y Pragmática

- *Fórmulas inseguras*:
  Fórmulas que podrían generar infinitos resultados

- *Fórmulas seguras:*
  Una expresión es segura independientemente del dominio sii todos los alores de su resultado pertenecen al dominio de la expresión

- Criterios para fórmulas seguras:

$\bigvee \varphi_i$ Todas las variables libres no negadas deben aparecer en cada φ_i

$\bigwedge \varphi_i$ Cada variable libre debe aparecer no negada en al menos una φ_i

- Pragmática de las consultas:
  Se accede a las claves de un elemento y luego se relacionan a través de ellas

[12] $operador \in \{=, <, \geq, \neq, \dots\}$

3.5 Cáculo Relacional de Dominios y Equivalencias

- Consultas:
  Especificación de un conjunto de valores de dominio por comprensión sobre el universo de todos los dominios:

$$\{t_1, \dots, t_n / \varphi\} \quad \Big| \quad t_i = \begin{cases} x_i \text{ variable libre de } \varphi \\ c_i \text{ constante} \end{cases}$$
$$\varphi \text{ fórmula de lógica de primer orden tal que } FV(\varphi) = \{t_1, \dots, t_n\}$$

- Variables:
  Son atributos y se corresponden entre sí al especificar los esquemas y se utilizan en toda la estructura: se usa "_" para omitir los valores en el mismo. Ejemplo:
  $$\{nom, dir / \exists nf. (FABS(nf, nom, dir) \wedge VENTAS(nf, \_, \_))\}$$

- Equivalencias:

Tipo	Equivalencia	
Cuantificador	$\forall x. P(x)$	$\neg\exists x. \neg P(x)$
	$\exists x. P(x)$	$\neg\forall x. \neg P(x)$
Conector lógico	$\neg(x \wedge y)$	$\neg x \vee \neg y$
	$\neg(x \vee y)$	$\neg x \wedge \neg y$
	$x \rightarrow y$	$\neg x \vee y$

3.6 Álgebra Relacional

- Descripción de los operadores:
  Los operadores retornan siempre una relación. Cada relación –y por tanto composición de operadores- conforma una expresión. Como los resultados son conjuntos, no existen tuplas repetidas

- Definición de los operadores básicos:

Nombre	Notación	Descripción	Ejemplo
Relación	Rel	Expresión que retorna una copia de la relación Rel	$Student$
Selector	$\sigma_{cond} Expr$	Retorna las filas de la relación $Expr$ que cumplen la condición lógica $cond$	$\sigma_{GPA>3,5 \wedge sName='Fulano'} Student$
Proyector	$\pi_{A_{i_1},\dots,A_{i_n}} Expr$	Retorna las columnas de atributos $A_{i_1}, \dots, A_{i_n}$ de la relación $Expr$[13]	$\pi_{sID,dec}(\sigma_{GPA>3,5} Student)$
Producto Cartesiano (Cruz)	$Expr_1 \times Expr_2$	Retorna el producto cartesiano de ambas expresiones[14]	$Student \times Apply$
Unión	$Expr_1 \cup Expr_2$	Retorna el conjunto unión de tuplas de sus operandos (el esquema debe ser el mismo o compatible[15])	$\pi_{cName} College \cup \pi_{sName} Student$
Diferencia	$Expr_1 - Expr_2$	Retorna el conjunto diferencia de tuplas de sus operandos (el esquema debe ser el mismo o compatible)	$\pi_{sID} Student - \pi_{sID} Apply$

[13] Puede utilizarse posiciones de una relación mediante $*posición*. Se deben utilizar todas posiciones o todos nombres de atributos

[14] Puede pensarse como la "unión" de las tablas que representan ambas Expresiones, donde los atributos de igual nombre son diferenciados: $Expr_1.a \neq Expr_2.a$

[15] Por "compatible" se entiende distintos nombres de los atributos pero igual dominio

- Definición de operadores derivados:

Nombre	Notación	Definición	Descripción
Unión Natural	$Expr_1 \bowtie Expr_2 = Expr_1 * Expr_2$	$\pi_{Esquema(Expr_1) \cup Esquema(Expr_2)} \left(\sigma_{\substack{Expr_1.A_1 = Expr_2.A_1 \wedge \\ Expr_1.A_2 = Expr_2.A_2 \wedge \dots}} (Expr_1 \times Expr_2) \right)$	Une las tuplas de sus operandos cuyos atributos de igual nombre adquieran igual valor[16]
Unión Theta	$Expr_1 \bowtie_\theta Expr_2$	$\sigma_\theta(Expr_1 \times Expr_2)$	Une sus operandos en la condición θ
División	$Expr_1 \div Expr_2$	$\pi_X(Expr_1) - \pi_X\big((\pi_X(Expr_1) \times Expr_2) - Expr_1\big)$	Primeros $\lvert X \rvert$ atributos de $Expr_1$ cuyos restantes atributos (coincidientes con $Expr_2$) incluyen todos los valores de $Expr_2$[17]
Intersección	$Expr_1 \cap Expr_2$	$Expr_1 - (Expr_1 - Expr_2)$ Si el esquema coincide: $Expr_1 \bowtie Expr_2$	Intersecta sus operandos
Renombrador	$\rho_{Rel(A_{i_1}, \dots, A_{i_n})}(Expr)$	Retorna un nuevo esquema al computar *Expr* creando una relación *Rel* de atributos $A_{i_1}, \dots, A_{i_n}$[18]	$\rho_{c(name)}(\pi_{cName} College) \cup \rho_{c(name)}(\pi_{sName} Student)$

- Propiedades de los operadores:
  - $A \bowtie B = A \times B \iff \text{esquema}(A) \neq \text{esquema}(B)$
  - $A \times \emptyset = \emptyset$

[16] Por tanto, elimina las columnas con nombres repetidos

[17] Ejemplo: $Q = R \div S$:

$R(A, C)$		$S(C)$		$Q(C)$
$a_1\ c_1$ $a_1\ c_2$ $a_2\ c_1$ $a_2\ c_2$ $a_2\ c_3$ $a_2\ c_4$ $a_3\ c_1$ $a_3\ c_3$	$\div$	c_1 c_2 c_3	$=$	a_2

[18] Su principal utilidad es la de unificar esquemas y así poder aplicar otros operadores

4. Equivalencia entre MER y Modelo Relacional

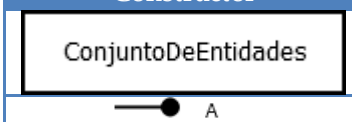
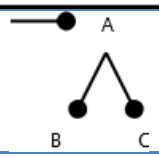
4.1 Dependencias de inclusión

- *Dependencias de Inclusión:*
  Dependencias del modelo relacional que especifica que:

$$\prod_{a_1,\ldots,a_n} A \subseteq \prod_{b_1,\ldots,b_n} B$$

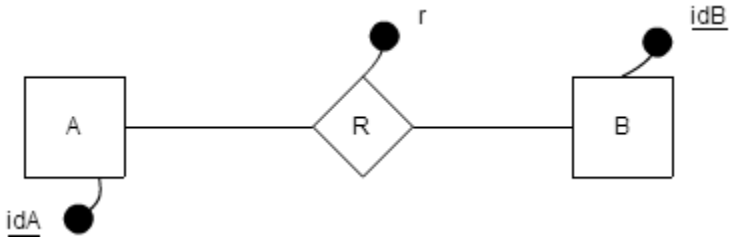
- Claves foráneas y dependencias de inclusión:
  Las claves foráneas son un caso particular de las dependencias de inclusión dado que en éstas no se exige la unicidad de aquella

4.2 Equivalencia de constructores

Constructor	Equivalencia
	ConjuntoDeEntidades(…)
	ConjuntoDeEntidades(…, B, C)
	ConjuntoDeEntidades(*idConjuntoDeEntidades*, …) As(*idConjuntoDeEntidades*, A) Restricción: $\prod_{idConjuntoDeEntidades} As \subseteq \prod_{idConjuntoDeEntidades} ConjuntoDeEntidades$
	ConjuntoDeEntidades(A, …)

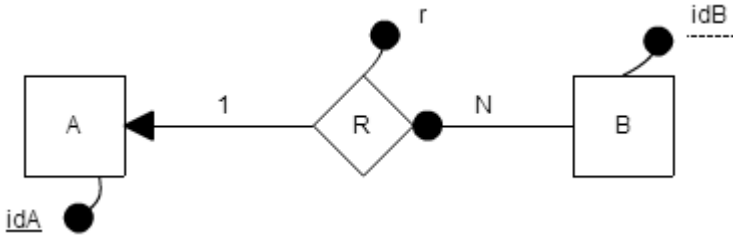
4.3 Equivalencia de relaciones[19]

- Binarias:



Cardinalidad (A : B)	Totalidad (de B a A)	Equivalencia
N : N	×	R(<u>idA, idB</u>, r, …) A(<u>idA</u>, …) B(<u>idB</u>, …) Restricciones: $\prod_{idA} R \subseteq \prod_{idA} A$ $\prod_{idB} R \subseteq \prod_{idB} B$
	✓	Se agrega la restricción: $\prod_{idB} B \subseteq \prod_{idB} R$
1 : N	×	R(idA, <u>idB</u>, r, …) A(<u>idA</u>, …) B(<u>idB</u>, …) Restricciones: $\prod_{idA} R \subseteq \prod_{idA} A$ $\prod_{idB} R \subseteq \prod_{idB} B$
	✓	A(<u>idA</u>, …) B(<u>idB</u>, idA, r, …) Restricciones: $\prod_{idA} B \subseteq \prod_{idA} A$

- Entidades débiles:



Tablas:

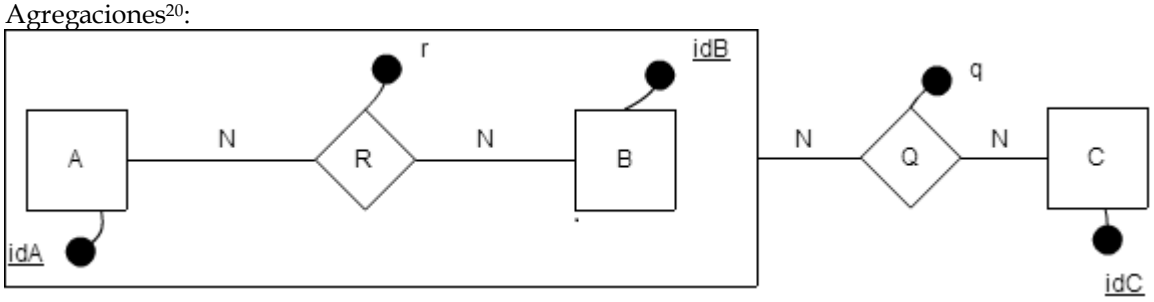
A(<u>idA</u>, …) B(<u>idA, idB</u>, r, …)

Restricción:

$$\prod_{idA} B \subseteq \prod_{idA} A$$

[19] Los atributos son tratados como en 4.2

- Agregaciones[20]:



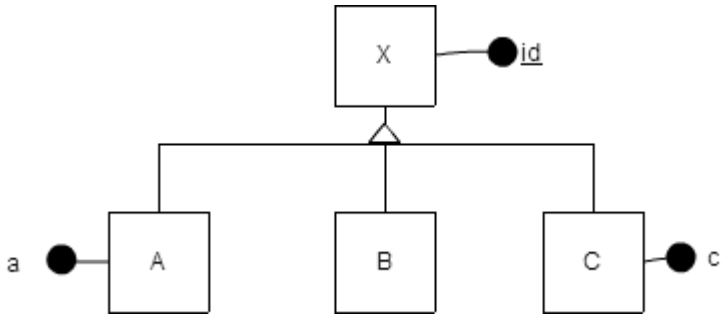
Tablas:

C(idC, …) R(idA, idB, r, …) Q(idA, idB, idC, q, …)

Restricciones:

$$\prod_{idC} Q \subseteq \prod_{idC} C \qquad \prod_{idA,idB} Q \subseteq \prod_{idA,idB} R$$

4.4 Equivalencia de categorizaciones



Restricciones	Equivalencia	
Ninguna[21]	X(id, …)	A(id, a, …) B(id) C(id, c, …)
$\boldsymbol{X = A \cup B \cup C}$	$X = \left\{ \begin{matrix} \langle t.id, \dots \rangle / \\ A(t) \vee B(t) \vee C(t) \end{matrix} \right\}$[22]	A(id, a, …) B(id) C(id, c, …)
Disjuntas dos a dos	X(id, …, a, c, tipo) Donde se incida en *tipo* a qué categoría pertenece	
Disjuntas (no dos a dos)	X(id, …, a, c, esA, esB, esC) Donde *esA, esB, esC* son booleanos que identifican la agregación	

[20] Para otras cardinalidades se debe proceder como se indica anteriormente teniendo en cuenta que la agregación es una operación de casting

[21] Es aplicable a todos los demás casos con las restricciones adecuadas

[22] X se implementa como una vista

5. Diseño Relacional

5.1 Pautas de diseño

Esquema	Nombre	Descripción	Consideraciones
	*Clara semántica de atributos*	Claridad sobre qué representa una tabla y que una tupla esté en ella	No combinar atributos de varios tipos de entidades y vínculos en una sola relación
a b ~~b~~	*Reducción de valores redundantes*	Eliminar la mayor cantidad de valores redundantes	Inserción, eliminación y modificación anómalas
a ~~null~~ b	*Prevención de valores nulos*	Evitar los valores nulos. Si no es posible, que su significado sea único	Que el significado de nulos sea: "no se conoce ese dato para esa tupla"[23]
X(a, b, c, e) ~~Y(c, e)~~ Y(b, c, e)	*Particionamiento de tablas consistente*	Tuplas erróneas: los esquemas deben reunirse por condición de igualdad sobre atributos claves	Sólo "dividir" las tablas sobre atributos claves

5.2 Dependencias

- *Dependencia Funcional (DF)* y *Dependencias Multivaluadas (DMV)*:
  Restricciones de integridad sobre el modelo relacional que implica que:

  Dada una instancia de una relación para la que X *e* Y *son atributos y* Z = R - XY*:*

Dependencia	*Funcional*	*Multivaluada*
Sigla	*DF*	*DMV*
Notación	$X \rightarrow Y$	$X \twoheadrightarrow Y$
Definición	$\forall t_1, t_2 \in r,$ $t_1[X] = t_2[X] \Rightarrow t_1[Y] = t_2[Y]$	$\forall r$, si $\exists t_1, t_2$: $t_1[X] = t_2[X] \Longrightarrow \exists t_3, t_4$ tales que: • $t_1[X] = t_2[X] = t_3[X] = t_4[X]$ • $t_3[Y] = t_1[Y]$ y $t_4[Y] = t_2[Y]$ • $t_3[Z] = t_2[Z]$ y $t_4[Z] = t_1[Z]$
Descripción	X *determina a* Y *para toda tupla de esa instancia*	X *determina para cada* Y *distinto, un conjunto (el mismo y todo) de valores de* Z[24]

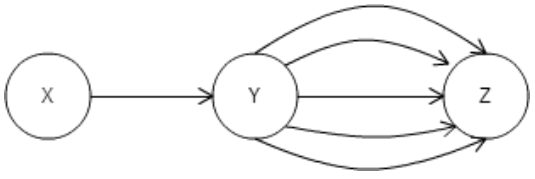
- *DMV Embebida:*
  Una *DMV Embebida* $X \twoheadrightarrow Y \mid Z$ indica que $X \twoheadrightarrow Y$ se cumple para todo esquema $X \cup Y \cup Z$[25]

---

[23] Especial atención sobre categorizaciones

[24] Z es independiente de Y y viceversa, pero se relacionan en R por la existencia de X:
Por cada valor Z_i de X_0Y_0, si existe una tupla X_0Y_1, ésta debe estar relacionada también con Z_i:

Tupla	X	Y	Z
t_1	x_1	y_1	z_1
t_2	x_1	y_2	z_2
t_3	x_1	y_1	z_2
t_4	x_1	y_2	z_1



[25] Por tanto, durante una descomposición, éstas no deben considerarse hasta que se obtenga que algún $R_i = X \cup Y \cup Z$

- Reglas de inferencia[26]:

Naturaleza	Nombre	Antecedente	⊢	Precedente
Básicas (de Armstrong)	*Reflexiva*	$Y \subseteq X$		$X \to Y$
	Aumento	$X \to Y$		$XZ \to YZ$
	Transitiva	$X \to Y, Y \to Z$		$X \to Z$
DF	*Identidad*	X		$X \to X$
	Descomposición	$X \to YZ$		$X \to Y$
	Unión	$X \to Y, Y \to Z$		$X \to YZ$
	Pseudotransitiva	$X \to Y, WY \to Z$		$WX \to Z$
DMV	*Complemento*	$X \twoheadrightarrow Y$		$X \twoheadrightarrow R - X \cup Y$
	Aumento	$X \twoheadrightarrow Y,\ W \supseteq Z$		$WX \twoheadrightarrow YZ$
	Transitiva	$X \twoheadrightarrow Y, Y \twoheadrightarrow Z$		$X \twoheadrightarrow Z - Y$
	Pseudotransitiva	$X \twoheadrightarrow Y, WY \twoheadrightarrow Z$		$WX \twoheadrightarrow Z - WY$
Relacionamiento DF-DMV	*Réplica*	$X \to Y$		$X \twoheadrightarrow Y$
	Combinación	$X \twoheadrightarrow Y, \exists W$: $W \cap Y = \emptyset$, $W \to Z$, $Y \supseteq Z$		$X \to Z$

- *DMV Trivial:*
  $X \twoheadrightarrow Y$ es una *DMV trivial* cuando cumple una de las siguientes condiciones:
  $Y \subset X$ $\qquad$ $X \cup Y = R$

- *Clausura de un conjunto de DFs*:
  F^+ es una *clausura de un conjunto de DFs F* sii contiene todas las DFs inferidas de *F*

- *Clausura de un conjunto de atributos bajo un conjunto de DFs*:
  Para *X* conjunto de atributos y *F* conjunto de DFs, $(X)_F^+$ es la *clausura de* X *bajo* F sii contiene todos los atributos de *X* determinados funcionalmente utilizando las DFs de *F*

- Algoritmo para hallar todas las $(X)_F^+$[27]:
  1. Comenzar con $(X)_F^+ = X$
  2. Para cada $Y \to Z \in F$, si $Y \subseteq X^+ \implies$ agregar Z a $(X)_F^+$

- *Cubrimiento de DFs*:
  Para F y E DFs, *F* cubre a *E* sii para cada $X \to Y \in E \Rightarrow Y \in (X)_F^+$

- *Equivalencia de DFs*:
  E y *F* DFs son equivalentes entre sí sii se cumple al menos una de las siguientes condiciones:
  $E^+ = F^+$ $\qquad$ Todas las DFs de *E* pueden inferirse de *F* y viceversa $\qquad$ *E* cubre a *F* y viceversa

- *Conjunto de DF minimal:*
  F_{min} conjunto de DF es *minimal* sii:
  1. Toda DF tiene un solo atributo a su derecha
  2. No se pueden "achicar" los atributos de la izquierda:
     No es posible reemplazar $X \to A \in F$ por $Y \to A$ con $Y \subset X$
  3. No se pueden eliminar dependencias

- *Cubrimiento minimal:*
  Un *cubrimiento minimal* de *F* conjunto de DFs es un conjunto minimal F_{min} equivalente a *F*

[26] Observar que toda clave primaria define una DF
[27] Observar que podrían haber más elementos que los dados por la unión de clausuras

- Algoritmo de eliminación para encontrar F_{min}:

#	Algoritmo	Aceleración[28]	Descripción
1	$G = F$		Comenzar por dependencias existentes
2	Reemplazar $X \to AB$ ... por $X \to A, X \to B$, ...		"Partir" las DF
3	Para cada $X \to A \in G$, para cada $B \in X$, hallar $(X - B)^+_G$ Si $A \in (X - B)^+_G$, reemplazar $X \to A$ por $X - B \to A$ en G	Atributos que sean compuestos a la izquierda, y se analiza si sacando uno de ellos se infiere el mismo resultado	Eliminar atributos redundantes
4	Para cada $X \to A \in G$, hallar $(X)^+_{G-(X \to A)}$ Si $A \in (X)^+_{G-(X \to A)}$, eliminar $X \to A$ en G	Analizar aquellas DF cuyo atributo a la derecha esté repetido en otra DF	Eliminar dependencias redundantes

5.3 Definiciones sobre Atributos y Dependencias

- Definiciones sobre claves:

Clave	Definición
Superclave	$S \subseteq R$ es una *superclave* sii $\forall r$ instancia de R, $\nexists t_1, t_2 \in r$ tal que $t_1[S] = t_2[S]$
Clave	Una *clave* es una superclave minimal[29]
Clave candidata	Cada una de las claves existentes de una relación es una *clave candidata*
Clave primaria	Clave de una relación asignada arbitrariamente entre las claves candidatas presentes
Clave secundaria	Claves que no son primarias
Atributo primo	Atributo miembro de alguna clave

- Definiciones sobre dependencias:

Dependencia $X \to Y$	Definición
Total	DF para la que no es posible quitar $A \subseteq X$ sin que deje de ser una DF[30]
Parcial	DF no Total: $\exists A \subset X$: $A \to Y$
Transitiva	$\exists Z$ no subconjunto de clave y $X \to Z$ y $Z \to Y$[31]

[28] Criterio para seleccionar rápidamente las DF que deben considerarse para estudiar en cada caso
[29] Esto es, si se le quita alguno de sus atributos, deja de ser una superclave
[30] Los atributos del antecedente ya son "minimales" en el sentido de la existencia de la DF
[31] Por tanto, puede expresarse de forma transitiva usando toras DFs.

- Definiciones sobre descomposiciones:

Descomposición D	Definición
D *de* R	$D = \{R_i\}_{i=1,\dots,m}$ tales que $\bigcup_{i=1}^{m} R_i = R$
***Proyección de* F *sobre* R_i**[32]	$\prod_{R_i} F = \{X \to Y \in F^+ :\ X \cup Y \subseteq R_i\}$
***Condición de preservación de* F *en* D**	$\bigcup_{i=1}^{m} \prod_{R_i} F = F^+$
D *que cumple Join sin pérdida (JSP)*	Si $i = 2$[33] debe cumplirse una de las siguientes condiciones[34]: i. $(R_1 \cap R_2) \to (R_1 - R_2) \in F^+$ ii. $(R_1 \cap R_2) \to (R_2 - R_1) \in F^+$ Si $i > 2$: Para cada instancia r de R se cumple que $\bowtie_{i=1}^{m} \prod_{R_i} (r) = r$

- Cálculo de claves:
  1. Se definen tres conjuntos:

#	Conjunto de Atributos	Propiedad
1	**No determinados por nadie**	Nunca están a la derecha
2	**No forman parte de ninguna clave**	Sólo están a la derecha
3	**Demás atributos**	Están tanto a la derecha como a la izquierda

  2. Se determinan todas las combinaciones de atributos de los conjuntos 1 y 3
  3. Se estudia la clausura de cada elemento de 2 para ver si incluyen a todos los atributos

- Test de JSP para $i > 2$:
  1. Crear $M \in \mathcal{M}_{|D| \times |\{A \text{ atributo de } R\}|}$
  2. Para cada fila i, si:
     $$A_j \in R_i \Longrightarrow m_{ij} = a_j \text{ valor fijo}$$
     $$A_j \notin R_i \Longrightarrow m_{ij} = b_{ij} \text{ variable del algoritmo}$$
  3. Repetir hasta que:
     - No haya modificaciones en M ó
     - Exista una fila con todos los símbolos a (sin importar el subíndice)

     Lo siguiente:

     Para cada $X \to Y \in F$ se "fuerza" la dependencia:

     En las filas donde los atributos de X coincidan

     Igualar los símbolos de los atributos de Y

     (Como los b_{ij} son variables, si un $b_{i_0 j_0}$ adquiere un valor a_k, todas sus referencias deberán hacerlo)
  4. La descomposición es JSP sii existe al menos una fila con todos los símbolos a (sin importar el subíndice)

[32] Dependencias de F^+ donde todos los atributos que aparecen están en R_i

[33] La definición si $i > 2$ incluye si $i = 2$, pero se discrimina en este documento porque la definición para $i = 2$ presenta muchas conveniencias prácticas

[34] Esto es, la intersección de los esquemas de la descomposición determina la diferencia de los esquemas en algún orden. Para las DMV, la definición es análoga reemplazando → por ↠

Ejemplo:

R={NSS, NOMBREE, NÚMEROP, NOMBREPR, LUGARP, HORAS} D={R1, R2, R3}
R1=EMP={NSS, NOMBREE}
R2=PROYECTO={NÚMEROP, NOMBREPR, LUGARP}
R3=TRABAJA_EN={NSS, NÚMEROP, HORAS}

F={NSS →NOMBREE; NÚMEROP →{NOMBREPR, LUGARP}; {NSS, NÚMEROP} →HORAS}

	NSS	NOMBREE	NÚMEROP	NOMBREPR	LUGARP	HORAS
R1	a_1	a_2	b_{13}	b_{14}	b_{15}	b_{16}
R2	b_{21}	b_{22}	a_3	a_4	a_5	b_{26}
R3	a_1	b_{32}	a_3	b_{34}	b_{35}	a_6

(matriz original *S* al principio del algoritmo)

	NSS	NOMBREE	NÚMEROP	NOMBREPR	LUGARP	HORAS
R1	a_1	a_2	b_{13}	b_{14}	b_{15}	b_{16}
R2	b_{21}	b_{22}	a_3	a_4	a_5	b_{26}
R3	a_1	~~b_{32}~~ a_2	a_3	~~b_{34}~~ a_4	~~b_{35}~~ a_5	a_6

(la matriz *S* después de aplicar las dos primeras dependencias funcionales - la última fila sólo tiene símbolos "a", así que nos detenemos)

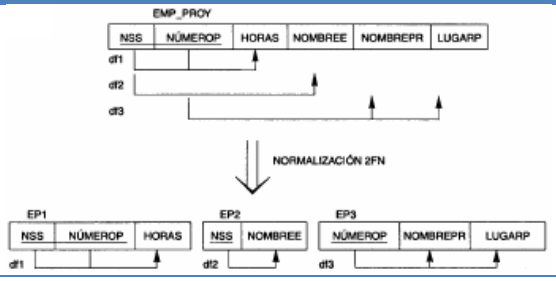
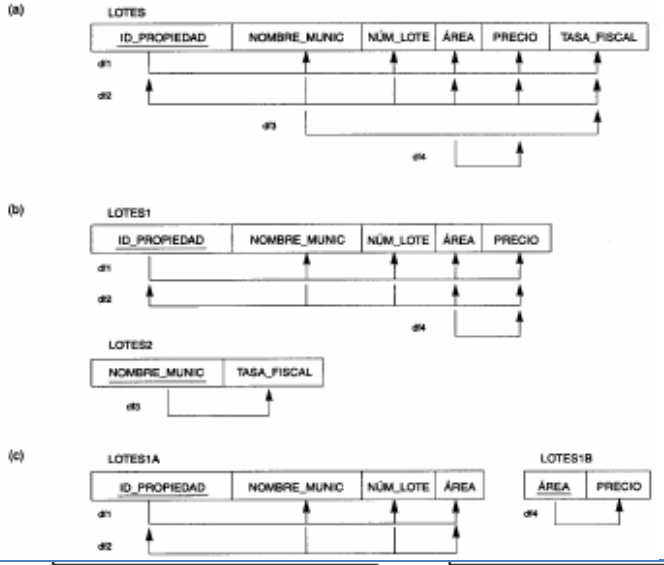
5.4 Formas Normales

- *Forma Normal*:
  Proceso de normalización permite:
  - Catalogar los esquemas
  - Construir un esquema normalizado a partir de uno no normalizado

- Formas Normales:

Abreviación	Forma Normal	Definición	Descripción
1NF	*Primera*	Los dominios de los atributos incluyen sólo valores atómicos	Todo diseño relacional visto en el curso lo cumple
2NF	*Segunda*	Ningún atributo no primo depende parcialmente de una clave	Ningún atributo que está sólo a la derecha depende de una parte de una clave. Por tanto, se podría normalizar creando varias tablas donde esas partes de las que dependen sean las claves de los atributos que dependen de él únicamente
3NF	*Tercera*	$\forall X \rightarrow A \in R$ se tiene que se cumple una de las siguientes propiedades: • X superclave • A atributo primo	Está en 2NF y ningún atributo no primo depende transitivamente de una clave
BCNF	*Boyce-Codd*	$\forall X \rightarrow A \in R$ se tiene que X es una superclave	Todas las DF son una superclave
4NF	*Cuarta*	$\forall X \rightarrow Y$ y $X \twoheadrightarrow Y$ no trivial, X es una superclave	Está en BCNF y todas las DMV no tiriviales son una superclave

- Ejemplos:

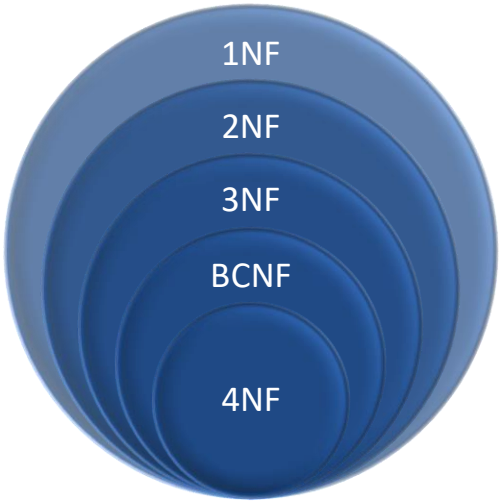
Forma Normal	Ejemplo
2NF	
3NF	
4NF	(a) **EMP** NOMBREE NOMBREPR NOMBRED Silva X Juan Silva Y Ana Silva X Ana Silva Y Juan Bravo W Jaime Bravo X Jaime Bravo Y Jaime Bravo Z Jaime Bravo W Juana Bravo X Juana Bravo Y Juana Bravo Z Juana Bravo W Beto Bravo X Beto Bravo Y Beto Bravo Z Beto (b) **PROYECTOS_EMP** NOMBREE NOMBREPR Silva X Silva Y Bravo W Bravo X Bravo Y Bravo Z **DEPENDIENTES_EMP** NOMBREE NOMBRED Silva Juan Silva Ana Bravo Jaime Bravo Juana Bravo Beto

- Algoritmos de descomposición de Formas Normales con JSP:

Forma Normal	Conserva DFs	Algoritmo
3NF	✓	1. Hallar F_{min} 2. Hacer una tabla para cada $f \in F_{min}$ donde el lado izquierdo sea el mismo: Para cada X lado izquierdo de f Crear $\{X \cup A_1 \cup ... \cup A_m\} \in D$ Donde $X \to A_i \; \forall i = 1,..,m$ sean las únicas f con X como miembro izquierdo 3. Colocar los atributos restantes y X en una sola tabla 4. Si ninguna tabla contiene una clave de R[35]: Crear una tabla adicional que contengan los atributos que forman una clave[36] de R
BCNF[37]	×	1. Hacer $D = \{R\}$ 2. Para cada $d \in D$ que no esté en BCNF: ○ Encontrar $X \to Y \in d$ que, proyectándolo, se determine que viola BCNF ○ Reemplazar d por los dos esquemas siguientes: ▪ $d - Y$ ▪ $X \cup Y$
4NF	×	Análogo a BCNF reemplazando donde diga "BCNF" por "4NF"

- 4NF y MER:
  En general, al pasar de un MER al modelo relacional, se obtendrá que éste estará en 4NF; cuando esto no sucede, existen altas probabilidades de que el MER esté mal construido

- Inclusiones de las Formas Normales:



[35] Para esto, se calcula si desde cada partición, la clausura cubre a F^+
[36] Para ello se aplica el algoritmo de cálculo de claves
[37] Para su ejecución en papel, es útil diagramarlo en forma de árbol e ir descomponiendo los d que violen BCNF. La unión de las hojas será el resultado buscado

6. Procesamiento y Optimización de Consultas

6.1 Índices

- Organización física de los datos por un DBMS:
  Almacenan sus datos en un gran archivo o una partición dedicada, administrándolos en primera persona y utilizando los servicios más básicos del SO o incluso implementando los suyos propios[38].

- *Índice*:
  Estructura de índice que, dada su clave, retorna como valor un puntero a un conjunto de bloques[39] que contiene los datos indizados

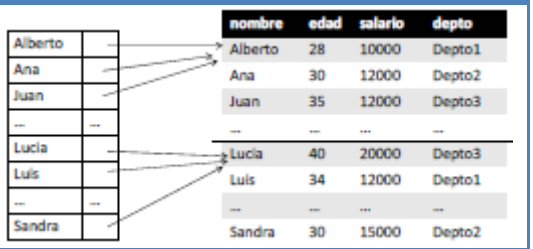
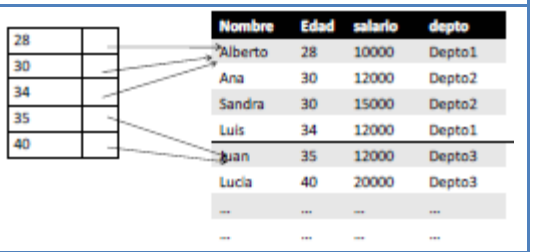
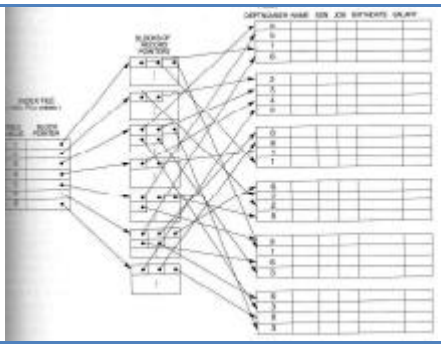
- Tipos de índices:

Relación	Clasificación	Dominio
Abstracción	Físico	Bloque de disco
	Lógico	Otros índices
Orden[40]	Ordenados	Datos ordenados
	No Ordenados	Datos no ordenados
Densidad	Densos	Todas las claves
	No Densos	Algunos valores
Niveles	Simples	
	Multiniveles	
Temporalidad	Permanentes	
	Auxiliares	

- Desventajas en el uso de índices:
  Elevado costo de mantención (para cada inserción, eliminación o actualización de las tuplas).

[38] Con el objetivo de acceder lo más rápido posible a distintas secciones de los archivos almacenados
[39] Este conjunto de todos los bloques conforman el archivo que representa la tabla, un índice apunta a una porción de esos bloques donde justo se encuentran los datos de interés
[40] No se refiere al orden de la estructura de índices, sino de los datos indizados

- Orden de índices:

Clasificación	Índice	Clave	Valor	Ejemplo
Ordenados	Primario	Primaria	Bloque + Desplazamiento	
	Clúster[41]	Atributo que no es clave primaria	Bloque + Desplazamiento del primer registro[42]	
No Ordenados	Secundario	Atributos ordenados	Lista de bloques que contienen el valor	

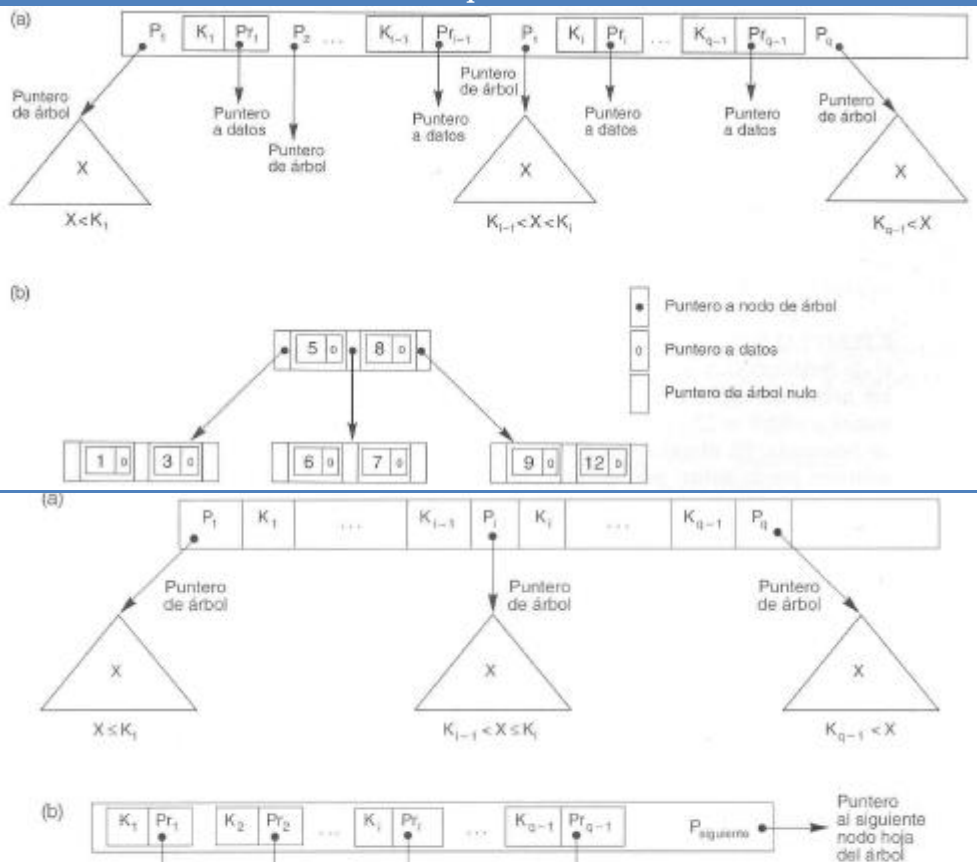
- *Bitmap*:
  Arreglo de bits sobre los valores de un atributo o una condición: para cada posición almacena si la tupla tiene determinado valor o cumple una condición predefinida para esa estructura

- Estructuras implementadoras de índices:

Estructura	Ventajas	Desventajas
Hash	Inserción y recuperación por igualdad	Relaciones de Orden
Árboles	Inserción y recuperación por igualdad y orden	Ocupa más espacio en disco
Bitmap	Atributos numerados	Pocas veces aplicable

[41] De agrupamiento
[42] Por tanto, para este caso debe verificarse que se encontraron todos los valores recorriendo a partir del obtenido por el índice; por este motivo, en general el valor es una lista de punteros

- *Árboles B* y *B+*:

Árbol	Descripción	Esquema
B	Nodo inicial con claves y punteros ordenados	
B+	Punteros a los datos sólo en las hojas, mejorando el balanceo	

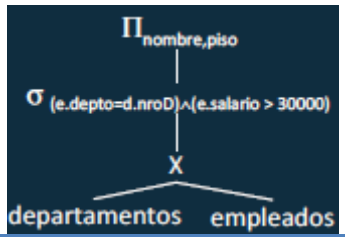
- Comparación entre estructuras:
  Son comparadas utilizando la cantidad de acceso a disco teniendo en cuenta que en un acceso a disco podrían recuperarse más de un nodo de la estructura y que estos podrían buffearse en memoria

6.2 Proceso de Optimización

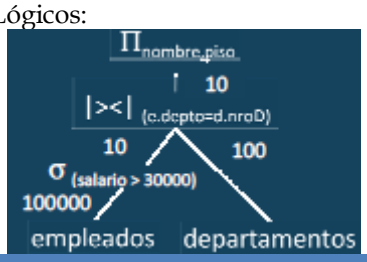
- Tipos de Optimización:

Tipo	Plan	Descripción
Heurística	Lógico	Reduce el árbol de consultas en álgebra relacional
Costos	Físico	A cada operación de los Planes Lógicos le asocia una o más implementaciones según las estructuras disponibles
	Final	Evalúa los Planes Físicos según los accesos de E/S requeridos

- Proceso de Optimización:

<table>
<tr><th>#</th><th>Etapa</th><th>Descripción</th><th>Ejemplo</th></tr>
<tr><td>1</td><td>Álgebra Relacional</td><td>Se construye la consulta en Álgebra Relacional partiendo de SQL como:
$\prod_{select} \sigma_{where}(\times_{from})$
Obteniendo un árbol canónico</td><td><pre>select e.nombre, d.piso
from departamentos d, empleados e
where e.depto = d.nroD
      and e.salario > 30000</pre> 
</td></tr>
<tr><td>2</td><td>Planes Lógicos</td><td>A partir del árbol canónico se reduce el árbol empleando:

- Equivalencias de expresiones
- Catálogo del a DB

En cada arista se indica la cantidad de tuplas estimadas que cumplen con las condiciones de la operación de destino</td><td>Catálogo de la DB:
<table><tr><th>Parámetro</th><th>Valor</th></tr><tr><td>Tamaño de EMPLEADOS (tuplas)</td><td>100.000</td></tr><tr><td>Tamaño de DEPARTAMENTOS (tuplas)</td><td>100</td></tr><tr><td>Selectividad de σ(salario>30000)</td><td>1/10.000</td></tr></table>
Planes Lógicos:
 
</td></tr>
<tr><td>3</td><td>Planes Físicos</td><td>Se consideran las implementaciones de σ y $\bowtie$ para determinar los costos de cada plan</td><td><table><tr><th>Operación</th><th>Implementación</th><th></th><th></th></tr><tr><td>σ</td><td>Búsqueda Lineal</td><td>Búsqueda Binaria</td><td>Índice</td></tr><tr><td>$\bowtie$</td><td>Loop anidado</td><td>Loop único</td><td>Sort Merge</td></tr></table></td></tr>
<tr><td>4</td><td>Plan Final</td><td colspan="2">Calculados los costos de cada plan, se selecciona el de menor valor</td></tr>
</table>

- Índices en el presente modelo teórico:
En el modelo teórico que aquí se construye, asume que los índices sólo pueden ser empleados en el primer nivel de las operaciones (según el Plan Lógico)

6.3 Optimización por Heurísticas

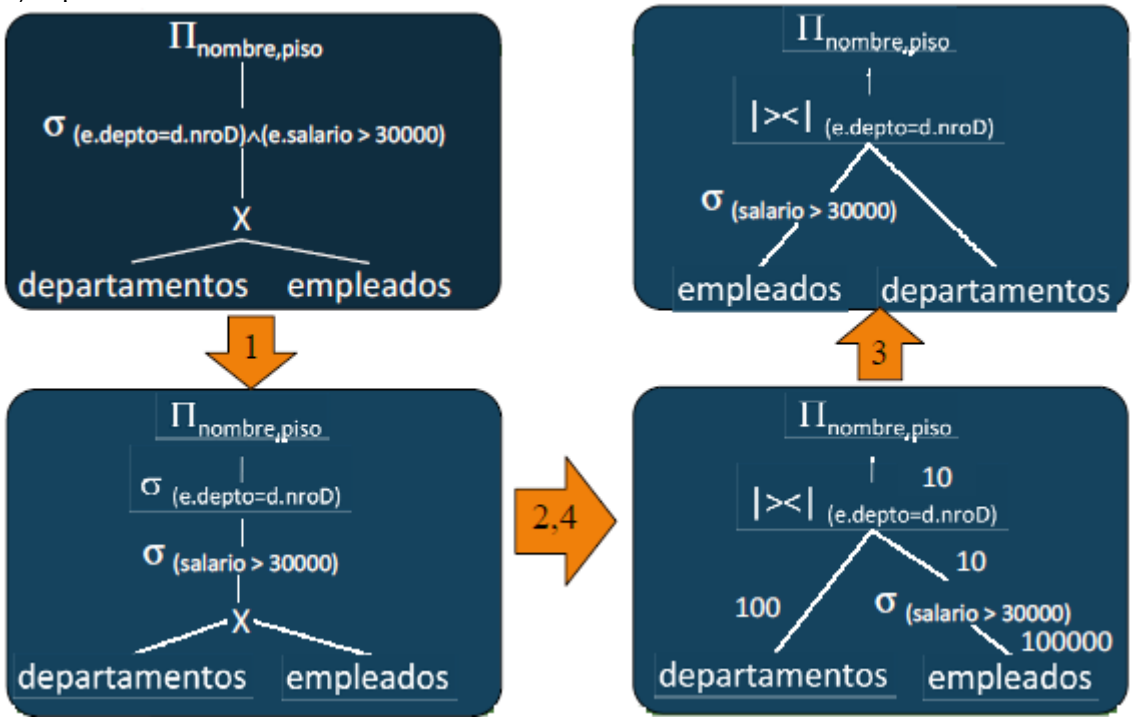
- Criterios:

#	Elemento	Acción
0	En todos los casos y siempre que sea posible, deben aplicarse las Equivalencias de Expresiones a fin de minimizar las operaciones de la consulta	
1	σ	Descomponer en σs más simples
2	σ	Mover lo más abajo posible
3	Mover a la izquierda las ramas con menos tuplas	
4	$(\sigma,\times)$	Reemplazar por $\bowtie_{condición}$
5	π	Mover lo más abajo posible agregando los πs necesarios[43]

- Equivalencia de expresiones:

Categoría	Equivalencia	Descripción/Aclaraciones
$\boldsymbol{\sigma}$	Descomposición	
	Conmutatividad	
$\boldsymbol{\pi}$	$\pi_{a_i}(\pi_{a_1,\dots a_n}R) = \pi_{a_i}R$	Eliminación de redundancia
	$\pi_a\sigma_cR = \sigma_c\pi_aR$	Conmutación con σ si éste contiene atributos de π
$\bowtie$	$\sigma_c(R \times Q) = R \bowtie_c Q$	Equivalencia de $(\sigma,\times)$
	Asociatividad de $\bowtie_c$	
	Conmutatividad de $\bowtie_c$	
$\times$	$\sigma_c(R \times Q) = (\sigma_cR) \times Q$	Si c sólo contiene atributos de R
	$\pi_{a\cup b}(R \times Q) = \pi_aR \times \pi_bQ$	Si a es de R y b es de Q
$\cup/\cap$	Reglas de conjuntos	
	$\sigma_c(R \cup Q) = \sigma_cR \cup \sigma_cQ$	Análogo para $\cap$, $-$ y π en lugar de σ

- Ejemplo:



[43] En general, no se aplica debido a la pérdida de índices y la necesidad de mantener varios atributos de proyección en los niveles inferiores porque serán utilizados en los niveles superiores

6.4 Optimización por Costos

- Parámetros:

Notación	Unidad	Definición	Fórmula
$\boldsymbol{n_T}$	tuplas	…de la tabla T	
$\boldsymbol{R_T}$	bytes	…de una tupla de T	
$\boldsymbol{b_T}$	bloques	…necesarios para almacenar las tuplas de T	$\left\lceil \frac{n_T}{bfr_T} \right\rceil$
$\boldsymbol{bf_T}$	tuplas	…en un bloque de T	$\left\lfloor \frac{bytes\ del\ bloque}{bytes\ de\ la\ tupla} \right\rfloor$
$\boldsymbol{x_T}$	niveles	…de un índice de T	$\log_k n_T + 1$[44]
$\boldsymbol{V(A,T)}$		…valores distintos que tiene un atributo A sobre una tabla T	n_T[45]
$\boldsymbol{sl(\sigma_c T)}$	tuplas	Fracción de tuplas que cumplen c en T original	$\frac{1}{V(A,T)}$[46]
$\boldsymbol{js(R \bowtie_c S)}$	tuplas	…que deben seleccionarse respecto del producto de $\times$	$\frac{1}{min\{V(A,R),V(A,S)\}}$[47]
$\boldsymbol{T(\sigma_c R)}$	tuplas	…que cumplen con c en T	$n_R \times sl(\sigma_c R)$
$\boldsymbol{T(R \bowtie_c S)}$	tuplas	…que cumplen con $\bowtie_c$	$n_R \times n_S \times js(R \bowtie_c S)$

- Estrategia de implementación de operadores:

Pipelined	Ejecución paralela de operadores	Generación de resultados	Aplicación
✓	✓	Son comunicados entre operadores sin grabarse a disco[48]	π
✗	✗	Son comunicados entre operadores grabando resultados intermedios	σ $\bowtie$

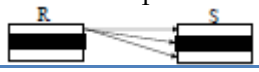
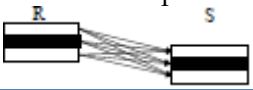
- Estimaciones de costos:
  Los costos son calculados para cada bloque de disco[49] en las acciones de:

  Lectura Depende de la organización de los datos

  Escritura Costo de grabar *todo* el resultado R: $\left\lceil \frac{n_R}{bf_R} \right\rceil$

[44] Para un B+ con k punteros por nodo sobre clave
[45] Para un atributo clave
[46] Si la condición es una igualdad sobre A y se asume distribución uniforme
[47] Si es $R \bowtie S$ natural sobre A
[48] No hay costo de lectura ya que los datos están en el buffer
[49] Que por tanto pueden contener varios registros o índices

- Implementación de operadores:

Operador	Estrategia	Restricción	Descripción	Costo de lectura[50]
$\sigma_c R$	Búsqueda Lineal		…de los registros que cumplen *c*	Caso promedio: $\frac{b_R}{2}$ Peor caso: b_R
	Búsqueda Binaria	Ordenación	Se divide la cantidad de registro y se lee en el sentido de la ordenación	$\log_2 b_R + \left\lceil \frac{s}{b_R} \right\rceil - 1$
	Índice Primario	Ordenación	Se consulta el índice	$x + 1$
	Índice de Cluster	Ordenación	Se consulta el índice	$x + \left\lceil \frac{s}{bf_R} \right\rceil$
	Índice Hash	*c* relación de igualdad	Lectura sobre el índice	1 o 2
	Índice secundario con B+		Se sigue la estructura B+	Peor caso: $x + s$
$R \bowtie_c S$	Loop anidado por registros		Leída una tupla de *R*, se leen todas las tuplas de *S* 	$b_R + n_R \times b_S$
	Loop anidado por bloques		Leído un bloque de *R*, se leen todos los bloques de *S* 	$b_R + \left\lceil \frac{b_R}{M-2} \right\rceil \times b_S$[51]
	Sort-Merge Join	Ordenación en *R* y *S*	Recorrer *S* y *R* en paralelo: por cada valor ordenado de *R* se recorren los valores de *S* que le correspondan según *c*[52] (y éstos serán un conjunto ordenado de *S*)	Costo de lectura: $b_R + b_S$ Costo de ordenación: $2 \times b \times (1 + \log_2 b)$
	Index Join (Single Loop)	Índice sobre *S*	Recorrer *R* accediendo por índice a *S*	$b_R + (n_R \times Z)$ Donde *Z*:

Índice	Z
Secundario	$x + s_S$
Cluster	$x + \left\lceil \frac{s_S}{bf_S} \right\rceil$
Primario	$x + 1$
Hash	h

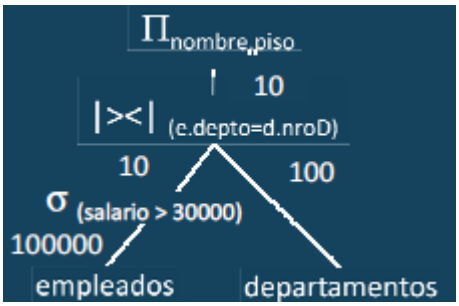
[50] *x* representa la cantidad de niveles del índice. *s* la cantidad de tuplas resultado de la operación

[51] *M* indica el tamaño del buffer. Es -2 porque uno se usa para *R* y otro para *S*. El factor $\left\lceil \frac{b_R}{M-2} \right\rceil$ indica la cantidad de bloques de *S* que se pueden leer a la vez

[52] Y éstos serán un conjunto ordenado de *S*. Por ejemplo, si *c* fuera de igualdad, para cada registro de *R* se recorren los *S* que le sean iguales, cuando deje de cumplirlo, es a partir de allí que debe analizarse para el siguiente registro de *R*

- Ejemplo:

Plan Lógico



Valores del Catálogo

Parámetro	Valor
Tamaño de EMPLEADOS (tuplas)	100.000
Tamaño de DEPARTAMENTOS	100
Selectividad de $\sigma_{(salario>3000)}$	1/10.000
$Bf_{empleados}, bf_{departamentos}$	10
$bf_{empleados \mid X \mid deparmentos}$	5
Índices sobre EMPLEADOS	B+ en salario, x=5
Todas las tablas tienen índice primario con x=1	
Asumimos que hay 3 buffers (M=3)	

Planes Físicos

	implementación	Costo leer	Costo grabar
$\sigma_{(salario>3000)}$ Empleados	Búsqueda lineal	10.000	1
	Búsqueda binaria		No es posible
	Índice secundario	15	1
|><|	Loop anidado reg.	101	2
	Loop anidado bloq.	11	2
	Sort Merge	11 o 12	2
Costo total mínimo		15+11 = 26	2+1=3

El tipo de índice con el que se cuenta no garantiza datos ordenados

Acceder a todos los niveles del índice (5) más uno por cada tupla (hay 10 tuplas; surge de la estimación 1/10.000) = 10 + 5

10×10 + 1

Es 12 si la ordenación hace una lectura más

Mejor plan

Selección con Índice Secundario y Join con Loop Anidado por Bloques, dando como resultado, 26 + 3 = 29 operaciones

7. Control de Concurrencia y Recuperación

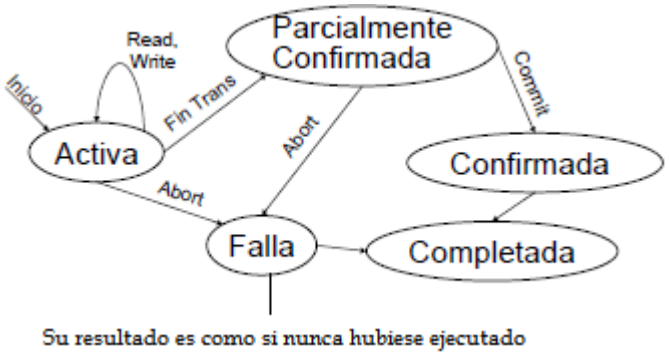
7.1 Transacciones e Historias

- Propiedad *ACID*:
  Un proceso cumple con la propiedad ACID si cumple con las siguientes propiedades:

Sigla	Propiedad	Descripción
A	Atomicidad	La ejecución es considerada una unidad de trabajo
C	Consistencia	Mantiene la consistencia de la DB
I	Aislamiento	Su ejecución no interfiere con otra
D	Durabilidad	Sus modificaciones son persistentes en la DB

- *Transacción*:
  Procesos concurrentes que ejecutan sobre datos compartidos y cumplen la propiedad ACID

- Estados de una Transacción:



- Operaciones de una Transacción[53]:

Acción	Notación completa	Notación compacta
Leer	$read_i(X)$	$r_i(X)$
Escribir	$write_i(X)$	$w_i(X)$
Confirmar	$commit_i$	c_i
Abortar	$abort_i$	a_i

- *Rollback*:
  Acción de recuperación de la DB a un estado previo a la ejecución de a_i en una Transacción

- *Grafo de Seriabilidad:*
  Grafo de precedencia de las Transacciones que componen una *Historia*. Construido como:

Paso	Componente[54]	Acción
1	T_i	Nodo T_i
2	$w_i \dots r_j$	$T_i \rightarrow T_j$
3	$r_i \dots w_j$	$T_i \rightarrow T_j$
4	$w_i \dots w_j$	$T_i \rightarrow T_j$

[53] i representa el número de la Transacción y X el lugar de la DB sobre el que se opera.
La secuencia de estas operaciones, identifica una Transacción

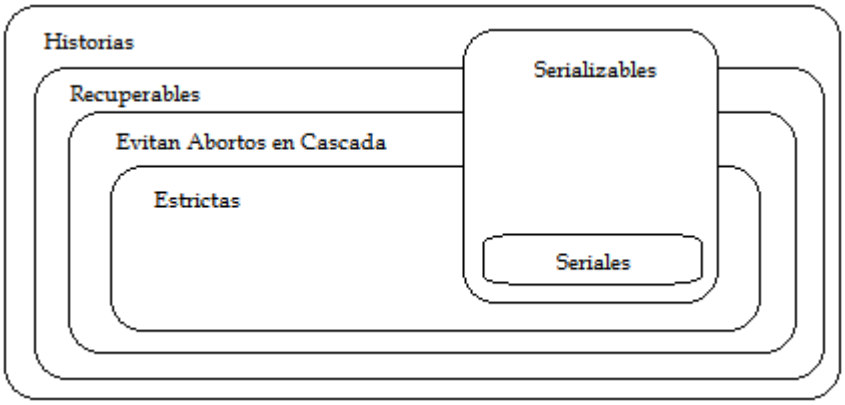
[54] Los r y w de esta columna tienen el mismo parámetro

- *Historia:*
  Ordenación de las operaciones de un conjunto determinado de transacciones que aparecen de forma lineal respecto de ésta[55]

- Clasificación de Historias:

Historia	Definición	Descripción
Serializable	Su Grafo de Seriabilidad es acíclico[56]	
Recuperable	Ninguna transacción confirma si no han confirmado aquellas desde las que se leyó datos previamente modificados	Los commits aparecen en el orden de flujo de datos: Se satisfacen las dependencias de confirmaciones[57]
Evitan Abortos en Cascada	Ninguna transacción lee valores escritos por otras no confirmadas	Que una transacción grabe y haga un commit antes de que otra lea
Estricta	Ninguna transacción lee o escribe hasta que todas las que escribieron sobre el mismo ítem fueron confirmadas	No hay un w ni un r si no existió sobre ese ítem modificado un c

- Relaciones entre las Historias:



- *Transaction Manager (TM):*
  Módulo del DBMS encargado de construir las transacciones y ordenarlas en historias serializables, *recuperables* y que no tengan *abortos en cascada*. Puede cambiar un c_i por un a_i

7.2 Control de Concurrencia

- *Lock* y *Unlock binarios*:
  Operación con comportamiento análogo a los semáforos de SOs cuyo dominio es una porción de la DB y su codominio la transacción en la que se ejecuta:

Operación	Notación completa	Notación compacta	Acción de bloqueo/desbloqueo sobre X
Lock	$lock_i(X)$	$l_i(X)$	Todas las operaciones
Read Lock	$read\_lock_i(X)$	$rl_i(X)$	Escrituras
Write Lock	$write\_lock_i(X)$	$wl_i(X)$	Todas las operaciones
Unlock	$unlock_i(X)$	$u_i(X)$	Todas las operaciones

[55] Esto es, se "intercalan" operaciones de todas las transacciones pero nunca se cambia el orden relativo dentro de la transacción al a que pertenecen
[56] Si posee ciclos, sufre de deadlock
[57] Podría pensarse como qué pasaría con transacciones que aún no confirmaron respecto de una que ya lo hizo: si el aborto de aquella impacta sobre estas, entonces, no es recuperable

- *Protocolo de Locking 2PL:*
  Reglas de locking que garanticen la seriabilidad:

Variante	Descripción	Características	Ejemplo
Básico	A lo largo de una Historia, primero están todos los *l* (fase de expansión) y luego y luego todos los *u* (fase de contracción)	↑ Fácil ↓ Deadlock[58]	$\underbrace{rl_i(X) \dots wl_j(Y)}_{Expansión} \dots \underbrace{u_i(X) \dots u_j(Y)}_{Contracción}$
Conservador	Declarar primero todos los *l*	↑ No hay Deadlock ↓ Exige declarar todo lo que se utilizará	$rl_i(X)wl_j(Y) \dots u_i(X)u_j(Y)$
Estricto	No se liberan *wl* hasta confirmar la *T*	↑ Historias *Estrictas* ↓ Deadlock	$\dots wl_i(X) \dots c_i u_i(X)$
Riguroso	No se liberan ni *wl* ni *rl* hasta confirmar la *T*	↑ Fácil (respecto del Estricto) ↓ Deadlock	$\dots rl_i(X)wl_j(Y) \dots c_i u_i(X)u_j(Y)$

- *TimeStamp (TS)*:
  Para cada ítem *X* se define un *Read_TS(X)* y un *Write_TS(X)* que informa el TS de la transacción más joven que leyó y escribió sobre él respectivamente

- Control de Concurrencia basado en TS:

Acción que pretende *T*	Condición para su ejecución	*TS(T)*	Acción
w(X)	Nadie leyó o escribió posteriormente a *T*	< *Read_TS(X)*	*T* es reiniciado con un TS actualizado
		< *Write_TS(X)*	
		Caso contrario	Se cambia *Write_TS(X)* por *TS(X)*
r(X)	Nadie escribió posteriormente a *T*	< *Write_TS(X)*	*T* es reiniciado con un TS actualizado
		Caso contrario	$Read\_TS(X) = max\{TS(T), Read\_TS(X)\}$

- *Multiversión*:
  Se mantienen varias versiones de cada ítem y se selecciona la adecuada para cada transacción:

$$X = \{(\ X_i,\ Read_{TS(X_i)},\ Write_{TS}(X_i)\ )\}_{i=1,\dots,n}$$

- Control de Concurrencia basado en Multiversión:

Acción que pretende *T*	Condición para su ejecución	*TS(T)*	Acción
w(X)	Nadie leyó posteriormente a *T*	$Write\_TS(X_i) \leq TS(T) < Read\_TS(X_i)$	*T* es reiniciado con un TS actualizado
		Caso contrario	Se crea $\left(X_j, TS(T), TS(T)\right)$
r(X)	Ninguna: Se utiliza la última escritura más cercana a *T*	< *Write_TS(X)*	*T* es reiniciado con un TS actualizado
		Caso contrario	$Read\_TS(X) = max\{TS(T), Read\_TS(X)\}$

7.3 Otros aspectos de Concurrencia

- *Granularidad*:
  Refiere al detalle con el que se define un ítem *X*: tupla, registro, tabla, bloque de disco…

- Granularidad y performance:

Granularidad	Impacto
Mayor	Menor concurrencia
Menor	Mayor overhead

[58] En esta columna, esto significa "Susceptible a…"

- *Registros Fantasmas*:
  Es un registro que, a la espera de un desbloqueo, no es insertado pese a que es de interés para la transacción bloqueante

- Detección de Deadlocks:
  Mantener un grafo de espera que detecte la existencia de deadlocks

- Solución a Deadlocks:

Timeout Si una transacción espera por mucho tiempo, reiniciarla

Basados en TS

Estrategia	Descripción
Wait-Die	La transacción más nueva del deadlock debe abortar y recomenzar con el mismo TS
Wound-Wait	La transacción más nueva del deadlock debe esperar por la más vieja

- Starvation:
  Wait-Die y Wound-Wait evitan el problema que, generalmente, pueden originarse por malos mecanismos de "selección de víctima" (puede suceder por su reinicio indefinido)

7.4 Recuperación

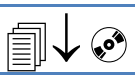
- *Log*:
  Registro de actividad sobre la DB

- Administración de logs:
  Deben respaldarse frecuentemente y almacenarse en lugares distintos a la DB

- Valores posibles de un ítem:

Sigla	Valor	Valor del ítem respecto de la actualización	Log de[59]
BFIM	BeFore IMage	Antes	Undo
AFIM	AFter IMage	Después	Redo

- Ciclo de vida de una instrucción de una transacción:

Paso	Descripción	Esquema
1	Se hacen los cambios en un buffer de memoria	
2	Se registran los cambios en el log	
3	La transacción confirma	✓
4	Se baja el log al disco	
5	Se bajan los cambios al disco	

[59] Clasificación de un log que contiene una operación y el registro BFIM/AFIM

- Técnicas de recuperación:

Actualización	Operativa	Ante un abort
Diferida	Cada transacción trabaja en una sandbox que es enviada a disco luego del commit	No debe realizarse acción alguna
Inmediata	La base es actualizada antes del commit	Se deben deshacer las operaciones de la transacción valiéndose del log
Shadow Paging[60]	Cada transacción mantiene en memoria un conjunto de punteros hacia los valores originales y los va cambiando a nuevas páginas de memoria a medida que surjan modificaciones	Recupera el conjunto de punteros original

- *Write-Ahead Loggin (WAL):*
  Protocolo que garantiza que el log con el BFIM está en el disco antes de grabar la operación en la DB

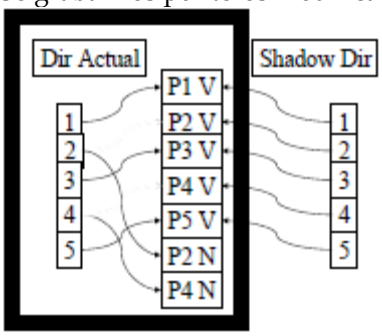
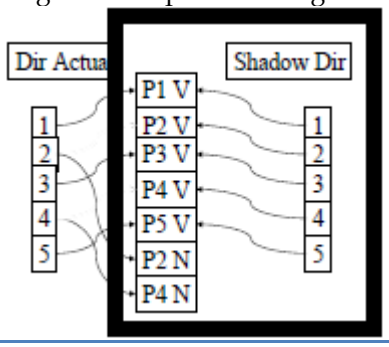
- *Checkpoint:*
  Registro del log que indica que todos los buffers modificados de la DB están en disco:

Paso	Descripción	Esquema
1	Suspender transacciones en curso	
2	Bajar buffers modificados	
3	Registrar checkpoint	
4	Bajar log	
5	Reanudar transacciones	

- *Rollback:*
  Se debe realizar cuando una transacción aborta. Si otras transacciones se basaron en ella para continuar, éstas también deben ser abortadas, provocando un aborto en cascada

[60] Útil para DB de pequeño porte debido al overhead del sistema

- Algoritmos de recuperación[61]:

Algoritmo	Acciones de *T* confirmada	Acciones de *T* no confirmada
Basado en Actualización Diferida	Se reconstruye el resultado utilizando el log	Se relanza al finalizar la recuperación
Basada en Recuperación Inmediata	Se reconstruye el resultado utilizando el log	Se revierten los cambios empleando los valores anteriores en el log. Se relanza la transacción
Shadow Paging	Se graban los punteros modificados 	Se graban los punteros originales 

[61] Todos ellos asumen la generación de historias estrictas y serializables