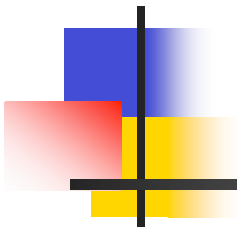


Fundamentos Lógicos de Bases de Datos





Agenda

- Lógica Proposicional y Lógica de Predicados
- Álgebra Relacional
- SQL



Sistema Formal o Cálculo

Un sistema formal es una 4-tupla:

$S=(\Sigma, F, A, R)$, donde

Σ : es un alfabeto.

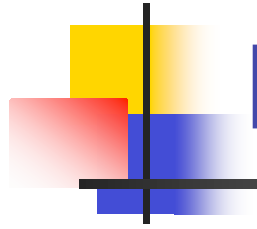
F : es un conjunto recursivo Σ^* , llamado conjunto de fórmulas, $F \subseteq \Sigma^*$,

A : es un conjunto recursivo Σ^* , llamado conjunto de axiomas, $A \subseteq \Sigma^*$,

R : es un conjunto finito de reglas de inferencia:

$R_i(x_1, x_2, \dots, x_m; y)$ cuyas variables toman valores en F .

Indica que la fórmula **y** se obtiene a partir de la fórmulas **x_1, x_2, \dots, x_m** aplicando la regla **R_i**



Lógica-Cálculo Proposicional

- Cálculo Proposicional-Definición Informal:
 - Unidad mínima de formalización: proposición o afirmación.
 - Cada proposición básica o atómica se modela como una variable.
 - Propositiones compuestas se modelan como conexiones de proposiciones atómicas. Las conexiones se realizan a través de conectores que modelan preposiciones del lenguaje natural.



Lógica-Cálculo Proposicional

- Toda variable p, q, r, \dots , con o sin subíndice es una fórmula del cálculo proposicional.
- Las constantes: true y false, son fórmulas del cálculo proposicional.
- Si $B1$ y $B2$ son fórmulas del cálculo proposicional, entonces:
 - $\neg B1$ es una fórmula del cálculo proposicional.
 - $B1 * B2$ es una fórmula del cálculo proposicional, donde,
 $* \in \{ \vee, \&, \rightarrow, \leftarrow, \leftrightarrow \}$
 - $(B1)$ es una fórmula del cálculo proposicional.
- No existe ninguna otra expresión que sea una fórmula del cálculo proposicional.



Lógica-Cálculo Proposicional

Dado un conjunto de variables del cálculo proposicional, una asignación ρ es una función, $\rho: V \rightarrow \{\text{true}, \text{false}\}$.

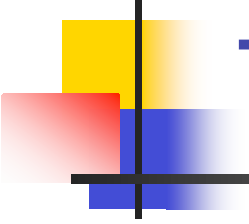
El valor de verdad de una fórmula del cálculo proposicional φ bajo la asignación ρ ($\varphi[\rho]$) para las variables que ocurren en φ se define como sigue:

- $\text{true}[\rho] = \text{true}$; $\text{false}[\rho] = \text{false}$
- Si $\varphi = p$ para alguna variable p , entonces $\varphi[\rho] = \rho(p)$
- Si $\varphi = \neg \phi$, entonces $\varphi[\rho] = \text{true}$ ssi $\phi[\rho] = \text{false}$
- Si $\varphi = (\phi_1 \vee \phi_2)$ y $(\phi_1 \vee \phi_2)[\rho] = \text{true}$ ssi $\phi_1[\rho] = \text{true}$ o $\phi_2[\rho] = \text{true}$



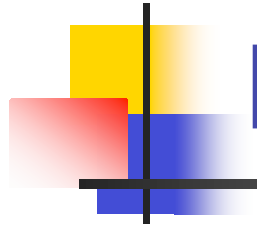
Lógica-Cálculo Proposicional

- Una fórmula φ es satisfactible ssi existe al menos una asignación ρ y $\varphi[\rho]=\text{true}$.
- Una fórmula φ es válida ssi para cualquier asignación ρ , $\varphi[\rho]=\text{true}$.
- Una fórmula φ_1 implica lógicamente a la fórmula φ_2 , $\varphi_1 \models \varphi_2$, si para cualquier asignación ρ donde $\varphi_1[\rho]=\text{true}$, entonces $\varphi_2[\rho]=\text{true}$.
- Una fórmula del cálculo proposicional φ está en forma normal conjuntiva (CNF) si tiene la forma $\varphi_1 \& \varphi_2 \& \varphi_3 \& \dots \& \varphi_n$, donde cada φ_i es una disjunción de literales.
- Determinar si una fórmula del cálculo proposicional φ en forma normal conjuntiva, en la cual cada conjuntor tiene a los más tres literales, (3-SAT) es un problema NP-completo.



Teoría de Modelos versus Teoría de Prueba.

- Sea ϕ un conjunto de fórmulas y φ una fórmula de cálculo proposicional:
 - $\phi \vdash \varphi$, si haciendo uso de los axiomas de cálculo y las reglas de inferencia, se puede derivar o probar φ a partir de ϕ . (Teoría de Prueba).
 - $\phi \models \varphi$, si toda asignación que sea satisfaga ϕ también satisface φ .
 - Un cálculo es sólido y completo ssi:
 - $\phi \models \varphi$ ssi $\phi \vdash \varphi$, intuitivamente, todo lo que se puede derivar es verdad, y todo lo que es verdad, se puede derivar.



Lógica-Cálculo de Predicados

- Cálculo Predicados-Definición Informal:
 - Unidad mínima de formalización: predicados que describen el sujeto de una oración o afirmación.
 - Cada predicado se modela como relación sobre un conjunto de dominios en el universo de discurso.
 - Variables en los predicados representan a cualquier elemento en el universo de discurso. Variables pueden estar cuantificadas.
 - Afirmaciones compuestas se modelan como conexiones de afirmaciones atómicas. Las conexiones se realizan a través de conectores que modelan preposiciones del lenguaje natural.



Lógica-Cálculo de Predicados

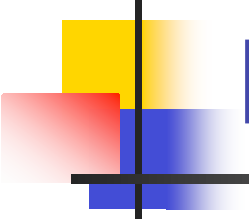
Sea P un símbolo de predicado de aridad n , y sean t_1, \dots, t_n términos, entonces, $P(t_1, \dots, t_n)$ es una fórmula del cálculo de predicados.

- Las constantes true y false son fórmulas del cálculo de predicados.
- Sean φ_1 y φ_2 fórmulas del cálculo de predicados, entonces:
 - $\neg \varphi_1$ es una fórmula del cálculo de predicados.
 - $\varphi_1 * \varphi_2$ es una fórmula del cálculo de predicados, donde $*$ $\in \{\vee, \&, \rightarrow, \leftarrow, \leftrightarrow\}$.
 - (φ_1) es una fórmula del cálculo de predicados.
 - $(\forall i | \varphi_1: \varphi_2)$ es una fórmula del cálculo de predicados.
 - $(\exists i | \varphi_1: \varphi_2)$ es una fórmula del cálculo de predicados.

Axiomas básicos:

$$(\forall i | \varphi_1: \varphi_2) \equiv (\forall i | : \varphi_1 \rightarrow \varphi_2)$$

$$(\exists i | \varphi_1: \varphi_2) \equiv (\exists i | : \varphi_1 \& \varphi_2)$$



Lógica-Cálculo de Predicados-Términos

- Si c es una constante, entonces c es un término.
- Si x es una variable, entonces x es un término.
- Si F es un símbolo funcional de aridad n y t_1, \dots, t_n son términos, entonces, $F(t_1, \dots, t_n)$ es un término.



Lógica-Cálculo de Predicados-Variables Libres

- Variable libre:
 - Si P es un predicado entonces la variable i ocurre libre en P .
 - La variable i ocurre libre en true y false.
 - Sea i una variable que ocurre libre en E . Entonces, i es libre en:
 - (E)
 - $\neg E$
 - $E * E1$
 - $(\forall x | \varphi1: \varphi2)$ o $(\exists x | \varphi1: \varphi2)$ ssi i no pertenece a las variables en x

Lógica-Cálculo de Predicados-Variables Limitadas

- Variable limitada:
 - Sea i una variable que ocurre libre en E . Entonces, i está limitada en las expresiones:
 - $(\forall x | E: F)$ o $(\forall x | F: E)$ o $(\exists x | E: F)$ o $(\exists x | F: E)$ ssi i pertenece a las variables en x .
 - Si i está limitada en $E1$, entonces, i está limitada en:
 - $(E1)$
 - $(\forall y | E1: F)$ o $(\forall y | F: E1)$ o $(\exists y | E1: F)$ o $(\exists y | F: E1)$
- Una fórmula es cerrada si no tiene ocurrencias de variables libres.



Lógica-Cálculo de Predicados

- Una interpretación de un lenguaje del cálculo de Predicados es una 4-tupla:

$I=(U,C,P,F)$, donde:

U : es un conjunto no vacío de elementos llamado universo de discurso.

C : es una función de las constantes a elementos en U .

P : hace corresponder cada símbolo de predicado p de aridad n a un subconjunto U^n .

F : hace corresponder cada símbolo de funcional p de aridad n de subconjunto U^n a un conjunto U .



Lógica-Cálculo de Predicados

Asignación/Satisfacción

- Dado un lenguaje L , φ una fórmula sobre L ,
e I una interpretación de L sobre el universo U
 - Una asignación de φ es una función parcial de las variables de L a U . I



Satisfacibilidad de una fórmula en una asignación

- Sea ϕ una fórmula y φ una asignación de las variables libres de ϕ en la estructura de interpretación I :
 - ϕ es satisfacible en φ en la interpretación I , ssi:
 - Si ϕ es un predicado P , entonces $\phi[\varphi]$ pertenece a la interpretación de P en I .
 - Si $\phi = \text{not}(\phi_1)$, entonces ϕ_1 no es satisfacible en φ en la interpretación I .
 - Si $\phi = \phi_1 \vee \phi_2$, entonces ϕ_1 es satisfacible en φ en la interpretación I o ϕ_2 es satisfacible en φ en la interpretación I .
 - Si $\phi = \phi_1 \& \phi_2$, entonces ϕ_1 es satisfacible en φ en la interpretación I y ϕ_2 es satisfacible en φ en la interpretación I .
 - Si $\phi = \phi_1 \Rightarrow \phi_2$, entonces ϕ_1 no es satisfacible en φ en la interpretación I o ϕ_2 es satisfacible en φ en la interpretación I .



Satisfacibilidad de una fórmula en una asignación

- Si $\phi = \phi_1 \iff \phi_2$, entonces:
 - ϕ_1 no es satisfacible en φ en la interpretación I y ϕ_2 no es satisfacible en φ en la interpretación I , o
 - ϕ_1 es satisfacible en φ en la interpretación I y ϕ_2 es satisfacible en φ en la interpretación I
- Si $\phi = (\forall i | \phi_1 : \phi_2)$, entonces:
 - Para cualquier asignación φ en la interpretación I , ϕ_1 no es satisfacible en φ en la interpretación I o ϕ_2 es satisfacible en φ en la interpretación I .
- Si $\phi = (\exists i | \phi_1 : \phi_2)$, entonces:
 - Si existe al menos asignación φ' en la interpretación I , ϕ_1 es satisfacible en φ' en la interpretación I o ϕ_2 es satisfacible en φ en la interpretación I .



Satisfacibilidad en una Estructura de Interpretación

- Sea ϕ una fórmula y una estructura de interpretación I :
 - ϕ es satisfacible en la interpretación I , ssi, para cualquier asignación φ de las variables libres de ϕ en la estructura de interpretación I , ϕ es satisfacible en φ .



Modelo de una teoría

- Una interpretación I es modelo de un conjunto de fórmulas ϕ , ssi I satisface cada fórmula en ϕ .
- El problema de decidir si $\phi \models \varphi$ es un problema no decidable.



Modelo Relacional

- Estructura básica:
 - Relación:
 - Se define sobre un conjunto de dominios D_1, \dots, D_n .
 - Corresponde a un subconjunto sobre el producto cartesiano de D_1, \dots, D_n
 - Propiedades:
 - No existen tuplas repetidas.
 - El orden de los elementos es irrelevante.
 - Si las columnas se nombran, en orden de las columnas es irrelevante.
 - Restricciones:
 - Integridad Referencial
 - Clave Primaria



Modelo Relacional-Terminología

- Sea R una relación con los atributos A_1, \dots, A_n sobre los dominios D_1, \dots, D_n , respectivamente, entonces
$$R \subseteq D_1 \times \dots \times D_n$$
 - No hay tuplas repetidas.
 - El orden de las tuplas es irrelevante.
 - El orden de las columnas es irrelevante.



Esquema del Modelo Relacional

- Un Esquema Relacional es una teoría T del lenguaje de la Lógica de Primer Orden formada por:
- Por cada relación R en el esquema, existe un predicado R en la teoría.
- Por cada clave (primaria o alterna) A de una relación R , existe una fórmula en T que modela que A es una clave.
- Por cada clave foránea $A1$ que en la relación $R1$ referencia a la relación $R2$, existe una regla que modela que $A1$ es una clave foránea.



Instancia de un esquema relacional

- Sea T un teoría que representa a un esquema relacional EsR, una instancia de EsR es una estructura de interpretación I que es modelo de T .



Cálculo de Predicados y Bases de Datos-Limitaciones

- Interpretaciones Finitas.
- Closed World Assumption.
- Lenguajes basados en Lógica de Predicados pero con restricciones:
 - Consultas Conjuntivas:
 - Cálculo y Algebra Relacional sin diferencia o negación.
Datalog sin recursión y negación.
 - Consultas Conjuntivas con Negación.
 - Consultas Conjuntivas con clausura transitiva.



Álgebra Relacional

■ Álgebra cerrada:

- El resultado de la aplicación de cualquier operador del álgebra relacional a una o más relaciones es también una relación.

■ Operadores Básicos:

- Selección
- Proyección
- Producto Cartesiano
- Unión
- Intersección

■ Operadores No Básicos:

- Join: Theta, Natural.



Álgebra Relacional

- Selección: selecciona las tuplas de una relación R que satisface la condición F:

$$\sigma(R, F) = \{t / t \in R \text{ y } t \text{ satisface } F\}$$

- Proyección: permite identificar los valores de los campos identificados, descartando el resto de los campos de la relación.

$$\pi(R, A_1, \dots, A_m) = \{ \langle t.A_1, \dots, t.A_m \rangle / t \in R \}$$



Álgebra Relacional

- Producto Cartesiano:

$$R1 \times R2 = \{ \langle t.A1, \dots, t.An, k.B1, \dots, k.Bm \rangle / t \in R1 \text{ y } k \in R2 \}$$

- Theta Join:

$$R1 \text{ Join Cond } R2 = \sigma(R1 \times R2, \text{Cond}).$$

- Join Natural:

$$R1 \text{ Join } R2 =$$

$$\pi(R1 \text{ Join Cond } R2, A1..An, B1, \dots, Bm-n)$$



Álgebra Relacional

- Dar todas la compañías que producen productos de la categoría “chocolate”.
 - T1: $\sigma_{\text{Categoria}=\text{“chocolate”}}(\text{Producto})$
 - T2: $\pi_{\text{Fabricante}}(\text{T1})$



Álgebra Relacional

T1: $\sigma_{(\text{Categoria}=\text{"chocolate"})}(\text{Producto})$

Nombre	Precio	Categoria	Fabricante
Carton	1000	chocolate	Savoy
Toronto	6000	chocolate	Savoy



Álgebra Relacional

- T2: $\pi_{\text{Fabricante}}(\text{T1})$

Fabricante

Savoy



Álgebra Relacional

- **Producto(Nombre,Precio,Categoria,Fabricante)**
- **Fabricante(Fabricante,RazonSocial,NumeroEmpleados)**

“El número de empleados de las empresas que fabrican productos de la categoría *chocolate*”

- **T1: (Producto Join Fabricante)**
- **T2: $\sigma_{\text{Categoria}=\text{“chocolate”}}(\text{T1})$**
- **T3: $\pi_{\text{NumeroEmpleados}}(\text{T2})$**

Join= Join Natural



Álgebra Relacional

- **Estadísticas:**
 - **Producto**
 - Cuatro millones de tuplas
 - Diez mil valores diferentes en el atributo **Categoria**.
 - Tuplas uniformemente distribuidas en los valores del atributo **Categoria**.
 - Cada categoría es producida por un único fabricante.
 - **Fabricante**
 - Dos mil tuplas.



Álgebra Relacional

- Tamaño en tuplas de T1:
 - 4.000.000 tuplas
- Tamaño en tuplas de T2:
 - 400 tuplas
- Tamaño en tuplas de T3:
 - 1 tuplas
- Se puede hacer algo mejor?



Álgebra Relacional

- **Producto(Nombre,Precio,Categoria,Fabricante)**
- **Fabricante(Fabricante,RazonSocial,NumeroEmpleados)**

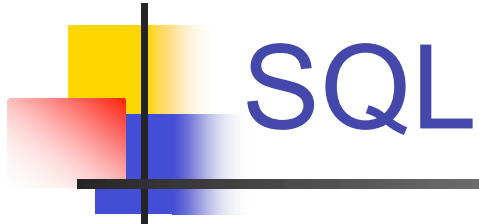
“El número de empleados de las empresas que fabrican productos de la categoría *chocolate*”

- **T'1: $\sigma_{\text{Categoria}=\text{"chocolate"}}(\text{Producto})$**
- **T'2: $\pi_{\text{Fabricante}}(\text{T'1})$**
- **T'3: $\pi_{\text{NumeroEmpleados}}(\text{T'2 Join Fabricante})$**



Álgebra Relacional

- Tamaño en tuplas de T'1:
 - 400 tuplas
- Tamaño en tuplas de T'2:
 - 1 tuplas
- Tamaño en tuplas de T'3:
 - 1 tuplas



- Estructura de una consulta SQL:

Select ListAtt

From T1,...,Tn

Where Cond

Equivale a la expresión del álgebra relacional:

$$\pi_{ListAtt}(\sigma_{Cond} (T1 \times T2 \times \dots \times Tn))$$



Complejidad de Lenguajes Lógicos

- Poder expresivo de un lenguaje L: es el conjunto de funciones que pueden ser escritas en L.
- Requerimiento lenguaje de consultas para ser *relacionalmente completo*:
 - Ser capaz de expresar todas las consultas expresables en álgebra relacional.
 - Cálculo Relacional, SQL sin agregación son *relacionalmente completos*. Se conocen también como lenguajes FO.



Complejidad de Lenguajes Lógicos- Complejidad de los Datos

Sea q una consulta acíclica, DB una instancia de un esquema relacional R y A el conjunto de las respuestas de q en DB ,

$q: \text{PowerSet}(DB) \rightarrow \text{PowerSet}(A)$,

Es decir, q es una correspondencia desde el conjunto de instancias de R al conjunto de posibles respuestas. La medida de complejidad determina la cantidad de veces que instancias de R deben ser consultadas para producir A .



Complejidad de Lenguajes Lógicos- Complejidad de los Datos

Formalmente, el modelo de computación puede ser una máquina de Turing y una instancia BD de un esquema R de tamaño n , se codifica en una cinta $O(n)$. Todas las consultas en la BD pueden ser vistas como máquinas de Turing. En caso que q sea evaluado en tiempo polinomial, se requerirán un número polinomial de pasos en la cinta (BD) para encontrar la respuesta, es decir, $O(n^k)$, donde k es un número positivo.

El conjunto de máquinas que pueden encontrar la respuesta en un número polinomial de pasos se denominan funciones DB-PTIME.



Complejidad de Lenguajes Lógicos- Complejidad de los Datos

- Se dice que un lenguaje L es DB-PTIME si cada función que pueden expresar, se computa en tiempo polinomial en función de n .
- Se dice que un lenguaje L es DB-PTIME completo, si L es DB-PTIME y L puede expresar todas las funciones que son DB-PTIME computables.
- Los Lenguajes FO que representan consultas acíclicas, son DB-PTIME computables. Técnicas sofisticadas de optimización y evaluación son usadas por los DBMS para mantener los exponentes y coeficientes de $O(n^k)$ bajos.
- Consultas FO cíclicas pueden requerir DB-EXPTIME.
- Los Lenguajes FO no son DB-PTIME completo porque existen funciones, por ejemplo, la clausura transitiva, que son DB-PTIME, que no pueden ser expresadas en lenguajes FO.

Ejemplo de una consulta Cíclica-Tablas

R_i

A _i	A _{i+1}
0	A
0	B
1	A
1	B
A	0
A	1
B	0
B	1

R_n

A _n	A ₁
0	A
0	B
1	A
1	B
A	0
A	1
B	0
B	1

Consulta:

R₁ Join R₂ Join...Join R_n