UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE CIENCIAS

Examen 3

PROFESOR

Gerardo Avilés Rosas

AYUDANTES

Gerardo Uriel Soto Miranda
Valeria Fernanda Manjarrez Angeles
Ricardo Badillo Macías
Jerónimo Almeida Rodríguez

ALUMNOS:

Castañon Maldonado Carlos Emilio
Navarro Santana Pablo César
Nepomuceno Escarcega Arizdelcy Lizbeth

ASIGNATURA

Fundamentos de Bases de Datos

1 **(20 puntos) Conceptos básicos**

- ¿Cuál es el objetivo de la normalización?
  Su objetivo es el de utilizar una serie de reglas (formas normales) que permiten identificar el conjunto adecuado de relaciones que respalden los requerimientos de datos de una organización, esto con el fin de tener un número mínimo de atributos necesarios para respaldar los requerimientos de datos, atributos de estrecha relación lógica y sobre todo una redundancia mínima en cada atributo representado una sola vez, exceptuando claro a los atributos que forman parte o son la llaves foránea.
- ¿Define llave en términos de dependencias funcionales?
  Una llave será el conjunto de atributos $A_1, A_2, A_3, ..., A_n$ tales que:
  - Determinan funcionalmente cualquier otro atributo de R(dos tuplas distintas no pueden coincidir en todos los atributos $A_1, A_2, A_3, ..., A_n$)
  - Ningún subconjunto propio de $A_1, A_2, A_3, ..., A_n$ determina funcionalmente a otros atributos de R, es decir, debe ser mínimo.
- ¿Qué es un JOIN? ¿En qué se diferencia un NATURAL LEFT/RIGHT JOIN de un NATURAL JOIN? ¿En qué casos conviene utilizar estos tipos de JOIN?
  Un JOIN es una operación que combina filas de dos o más tablas basándose en una condición relacionada entre ellas. La condición se especifica mediante una cláusula JOIN en una consulta SQL. La operación JOIN es fundamental para relacionar datos almacenados en tablas diferentes.
  La principal diferencia entre NATURAL JOIN y los JOINS LEFT/RIGHT es que NATURAL JOIN utiliza automáticamente las columnas con el mismo nombre y tipo, mientras que en los JOINS LEFT y RIGHT se debe especificar explícitamente la condición de coincidencia con la cláusula ON. Además, el NATURAL JOIN no permite especificar qué columnas se deben utilizar para la combinación, lo cual puede ser un riesgo si las tablas tienen muchas columnas con el mismo nombre pero no todas deben ser utilizadas en la combinación.
  - **NATURAL JOIN:**
    - Cuando las columnas coincidentes tienen el mismo significado en ambas tablas: El NATURAL JOIN es útil cuando las tablas comparten columnas con el mismo nombre y tipo, y estas columnas representan la misma información en ambas tablas.
    - Para simplificar consultas: Si se tienen muchas columnas coincidentes con el mismo nombre en ambas tablas y queremos simplificar la consulta, el NATURAL JOIN puede ser útil.
  - **LEFT JOIN:**
    - Cuando necesitamos todas las filas de la tabla izquierda: Si queremos obtener todas las filas de la tabla izquierda, incluso si no hay coincidencias en la tabla derecha, podemos utilizar un LEFT JOIN.
    - Para consultas jerárquicas: Si estamos trabajando con datos jerárquicos, como una tabla de empleados y una tabla de supervisores, un LEFT JOIN puede ayudarnos a obtener una lista completa de empleados; incluso si algunos no tienen supervisores asignados.
  - **RIGHT JOIN:**
    - Cuando necesitamos todas las filas de la tabla derecha: Similar al LEFT JOIN, el RIGHT JOIN se utiliza cuando deseamos obtener todas las filas de la tabla derecha, incluso si no hay coincidencias en la tabla izquierda.
    - En casos específicos: En la práctica, el RIGHT JOIN se utiliza menos comúnmente que el LEFT JOIN. En muchos casos, podemos expresar la misma lógica de consulta utilizando un LEFT JOIN.

- Indica la estructura de una consulta en SQL.

```
SELECT a1,a2,a3,a4,a5, ... ,an
FROM R1,R2,R3,R4,R5, ... ,Rm
WHERE condicion
```

La cláusula **FROM** indica las relaciones que serán consultadas.
La cláusula **WHERE** especifica la condición que deben satisfacer las tuplas para ser seleccionadas.
La cláusula **SELECT** se utiliza para describir los atributos que se desea formen parte de la respuesta.
Condición:

- **Operandos:** Constantes y atributos de las relaciones mencionadas en la cláusula FROM.
- **Operadores:** $=, <>, >, <, <=, >=, AND, OR, NOT$

- En qué se diferencia una consulta que se realiza con INTERSECT de una que se realiza con NATURAL JOIN.
INTERSECT se utiliza para obtener la intersección de conjuntos de resultados, devolviendo filas comunes entre las consultas, mientras que NATURAL JOIN se utiliza para combinar tablas basándose en columnas con el mismo nombre, eliminando la necesidad de especificar explícitamente las columnas de unión, por ende estas operaciones tienen propósitos distintos y se aplican en contextos diferentes.

2 **(15 puntos) Join sin pérdida**
Se tiene el esquema $R(A, B, C, D, E, F)$.
Se sabe que A es llave primaria, F es una llave candidata y se cumplen las dependencias:
$\{BD \to E, CD \to A, E \to C, B \to D\}$:

a) ¿Qué puedes decir de $A+$ y $F+$?
Tenemos las dependencias funcionales dadas: $\{BD \to E, CD \to A, E \to C, B \to D\}$
Para $A+$:

- $A \in A+$ (Reflexividad)
- $CD \to A$: $C, D \in A+$
- $E \to C$: $E \in A+$ (Transitividad)
- $BD \to E$: $B, D \in A+$ (Transitividad)
- F es una llave candidata, por lo que $F \in A+$.

Entonces: $A^+ = \{A, B, C, D, E, F\}$.
Para $F+$:
Dado que F es una llave candidata y contiene todos los atributos de R, $F+ = \{A, B, C, D, E, F\}$.

b) Sí se divide R en las relaciones $R_1(A, B, C, D, F)$ y $R_2(C, E)$, ¿esta división de R tiene **join sin pérdida**?

Tenemos las dependencias funcionales dadas: $\{BD \to E, CD \to A, E \to C, B \to D\}$
Sabemos que la llave primaria de R es A, y la única dependencia funcional que involucra a A es $CD \to A$. Al dividir R en $R_1(A, B, C, D, F)$ y $R_2(C, E)$, estamos manteniendo la información de CD en ambas relaciones.

- En R_1, se mantienen las dependencias $CD \to A$ y $B \to D$. Sin embargo, falta la dependencia $BD \to E$ en R_1.
- En R_2, se mantiene la dependencia $E \to C$. Sin embargo, faltan las dependencia $BD \to E$ en R_2.

Por lo tanto, al dividir la relación R en R_1 y R_2, no se mantienen todas las dependencias funcionales, entonces, podemos decir que no tiene join sin perdida.

3 **(20 puntos) Tercera Forma Normal**
Dada la siguiente relación $R(A, B, C, D, E, F)$ con las dependencias $F = \{B \to D, B \to E, D \to F, AB \to C\}$. Indica alguna llave para la relación R y las **violaciones** a la **3NF**. Encuentra la **descomposición** según la **Tercera Forma Normal**.

Aplicamos la regla de la unión. Calculamos la llave, para eso calculamos la cerradura:

$$\{B\}+ = \{BDEF\}, \text{ no es llave}$$
$$\{D\}+ = \{DF\}, \text{ no es llave}$$
$$\{AB\}+ = \{ABCDEF\}, AB \text{ es una llave para } R$$

Violaciones para la Tercera Forma Normal:

$$B \to DE$$
$$D \to F$$

Superfluos por la izquierda:

① $AB \to C$
- Verificamos si A es superfluo: $B \to C$; $\{B\}+ = \{BDEF\}$, A no es superfluo.
- Verificamos si B es superfluo: $A \to C$; $\{A\}+ = \{ABCDEF\}$, B no es superfluo.

Superfluos por la derecha:

② $B \to DE$
- Verificamos si D es superfluo: $B \to E$; $\{B\}+ = \{BE\}$. D no es superfluo.
- Verificamos si E es superfluo: $B \to D$; $\{B\}+ = \{BDF\}$. E no es superfluo.

La descomposición en 3NF es correcta:

$$R_1(A, B, C) \text{ con } AB \to C$$
$$R_2(D, F) \text{ con } D \to F$$
$$R_3(B, D, E) \text{ con } B \to DE$$

Además, las dependencias funcionales minimizadas son $F_{\min} = \{AB \to C, D \to F, B \to DE\}$.