Resumen Teórico para Parcial de Bases de Datos I

🌎 Unidad 1: Fundamentos de Bases de Datos

Conceptos Básicos

- **Base de Datos (BD):** Es un conjunto de **datos persistentes** y relacionados entre sí, guardados en un medio de almacenamiento masivo.
- **SGBD (Sistema de Gestión de Bases de Datos):** Es el **software** (conjunto de programas) que permite a los usuarios crear, mantener y consultar una base de datos. Ejemplos: SQL Server, Oracle, MySQL, PostgreSQL.
- **Dato vs. Información:** El dato es el valor almacenado (Ej: "9"). La información es el significado de ese dato (Ej: "El alumno aprobó").
- **Metadatos:** Son los "datos acerca de los datos". Describen la estructura de la base de datos (nombres de campos, tipos de datos, etc.) y se almacenan en el catálogo.
- **Esquema vs. Estado:**
    - **Esquema:** Es la **descripción** de la base de datos (la estructura, los metadatos). No cambia frecuentemente.
    - **Estado:** Son los **datos almacenados** en la base de datos en un momento determinado. Cambia constantemente con las operaciones de actualización.

Arquitectura e Independencia

El objetivo de la arquitectura de 3 niveles es **separar las aplicaciones del usuario de la base de datos física**.

1. **Nivel Interno (Físico):** Describe la **estructura física de almacenamiento**.
2. **Nivel Conceptual:** Describe la base de datos completa en términos de su estructura lógica (registros, campos, tipos de datos, relaciones), ocultando los detalles físicos.
3. **Nivel Externo (Vistas):** Describe la **parte de la base de datos que le interesa a un usuario** o aplicación en particular, ocultando el resto.

Esto da lugar a la **Independencia de Datos**:

- **Independencia Lógica:** Es la capacidad de **modificar el esquema conceptual** (ej: añadir un nuevo campo) sin tener que alterar los esquemas externos o las aplicaciones.
- **Independencia Física:** Es la capacidad de **modificar el esquema interno** (ej: cambiar cómo se almacenan los datos en disco) sin tener que alterar el esquema conceptual.

Lenguajes de Bases de Datos

- **DDL (Lenguaje de Definición de Datos):** Permite definir el esquema conceptual (crear tablas, definir campos).
- **DML (Lenguaje de Manipulación de Datos):** Permite insertar, modificar, eliminar y consultar los datos.
- **SQL (Structured Query Language):** Es el lenguaje estándar que combina funciones de DDL y DML.

🏛️ Unidad 2: El Modelo Relacional

Fue introducido por **Edgar Frank Codd** en 1970 y se basa en el concepto matemático de "relación".

Terminología Clave

Existe una equivalencia entre los términos formales del modelo y su representación común en un SGBD:

Modelo Relacional (Formal)	**Representación SGBD (Común)**	**Definición**
Relación	Tabla	Estructura de datos simple y uniforme.
Tupla	Fila o Registro	Una fila que representa un conjunto de valores relacionados.

Modelo Relacional (Formal)	Representación SGBD (Común)	Definición
Atributo	Columna o Campo	Una cabecera de columna que describe un dato.
Dominio	Tipo de datos	Conjunto de **valores atómicos** (indivisibles) permitidos para un atributo.

- **Esquema de Relación:** Es la "cabecera" o definición de la tabla. Se denota como $R(A1, A2, ..., An)$.
- **Estado de Relación:** Es el "cuerpo" o contenido de la tabla en un momento dado; un conjunto de tuplas.
- **Valor Nulo (NULL):** Un valor especial usado cuando un dato es **desconocido** o **no aplicable**.

🔑 Unidad 3: Restricciones de Integridad y Claves (¡Muy Importante!)

Son las reglas que garantizan la consistencia y validez de los datos.

Conceptos de Claves

- **Superclave:** Un subconjunto de atributos que **garantiza que una tupla es única**.
- **Clave Candidata (CC):** Una **superclave mínima**; es decir, única y sin atributos redundantes. Una relación puede tener varias claves candidatas.
- **Clave Primaria (CP o PK):** Es la Clave Candidata que **se elige** para identificar unívocamente a las tuplas en la relación.
- **Clave Alternativa (CA o AK):** Es una Clave Candidata que **no fue elegida** como Clave Primaria.

Restricciones Fundamentales

1. **Restricción de Integridad de Entidades:** Establece que **ningún valor de Clave Primaria puede ser NULO**. Esto es esencial porque la CP se usa para identificar las tuplas; si fuera nula, no podríamos identificar algunas de ellas.
2. **Restricción de Integridad Referencial (Claves Externas):** Es una restricción que se especifica **entre dos relaciones** (tablas) para mantener el vínculo entre ellas.
    - **Clave Externa (CE o FK):** Es un conjunto de atributos en una relación (R1) que hace referencia a la Clave Primaria (CP) de otra relación (R2).
    - **La Regla:** El valor de la Clave Externa en una tupla de R1 debe **coincidir con un valor de CP existente en R2** o, si se permite, **ser NULO**.

Vínculos entre Entidades

- **1:N (Uno a N):** Una tupla de R1 se vincula con *varias* tuplas de R2. (Ej: Un DEPARTAMENTO tiene muchos EMPLEADOS).
- **1:1 (Uno a Uno):** Una tupla de R1 se vincula con *una única* tupla de R2. (Ej: Un EMPLEADO ocupa un CARGO).
- **M:N (Muchos a N):** Varias tuplas de R1 se vinculan con varias de R2. Este vínculo **requiere una nueva relación (tabla) intermedia** para implementarse. (Ej: EMPLEADOS y PROYECTOS se vinculan mediante la tabla HORAS).

✨ Unidad 4: Normalización (¡Muy Importante!)

Es un **método formal** para analizar y descomponer esquemas de relaciones con el objetivo de **minimizar la redundancia de datos** y **evitar anomalías**.

- **Anomalías:** Problemas que surgen en esquemas mal diseñados:
    - **De Inserción:** No poder insertar un dato porque falta otro (ej: no poder agregar un médico nuevo si aún no tiene pacientes asignados en una tabla única).
    - **De Eliminación:** Perder datos de forma no intencionada (ej: si se elimina la última consulta de un paciente, se borran todos los datos del paciente).

    - **De Actualización:** Tener que actualizar el mismo dato en múltiples filas (ej: cambiar la descripción de una obra social) y correr el riesgo de inconsistencias.

Dependencia Funcional (DF)

Es el concepto central de la normalización. Se denota como $X \rightarrow Y$.

- **Definición:** "X determina funcionalmente a Y".
- **La Regla:** Para dos tuplas t_1 y t_2, si sus valores en los atributos X son iguales ($t_1[X] = t_2[X]$), entonces sus valores en los atributos Y también **deben ser iguales** ($t_1[Y] = t_2[Y]$).

Tipos de Atributos y Dependencias

- **Atributo Primo:** Es un atributo que forma parte de **alguna** Clave Candidata.
- **Atributo No Primo:** No forma parte de ninguna Clave Candidata.
- **Dependencia Parcial:** Un atributo no primo depende de **sólo una parte** de una clave primaria compuesta. (Viola la 2FN).
- **Dependencia Total:** Un atributo no primo depende funcionalmente de la clave primaria **completa**, y no solo de una parte de ella.
- **Dependencia Transitiva:** Ocurre cuando $X \rightarrow Z$ y $Z \rightarrow Y$, donde X es la clave primaria, pero Z **no es parte de ninguna clave**. (Viola la 3FN).

Las Formas Normales (1FN, 2FN, 3FN)

1. **Primera Forma Normal (1FN):**
    - **Definición:** Una relación está en 1FN si y sólo si todos los dominios de sus atributos contienen únicamente **valores atómicos (indivisibles)**.
    - **En corto:** No se permiten atributos multivaluados (como una lista de teléfonos en un solo campo) ni atributos compuestos (como "Consulta" agrupando CodMed, NomMed, Fecha).
2. **Segunda Forma Normal (2FN):**
    - **Definición:** Una relación está en 2FN si está en 1FN y, además, **todos los atributos no clave (no primos) dependen de forma total** de la clave primaria.

    - **En corto:** No se permiten **dependencias parciales**. Si un atributo depende solo de *parte* de la clave compuesta, debe sacarse a otra tabla.

3. **Tercera Forma Normal (3FN):**
    - **Definición:** Una relación está en 3FN si está en 2FN y, además, **ningún atributo no clave depende de forma transitiva** de la clave primaria.
    - **En corto:** No se permiten **dependencias transitivas**. Si un atributo no-clave depende de otro atributo no-clave, debe sacarse a otra tabla.

➗ Unidad 5: Álgebra Relacional (Teoría de Consultas)

Es un conjunto de operaciones formales que sirven para **manipular relaciones (tablas)**. Cada operación toma una o dos relaciones y **produce una nueva relación** como resultado.

Operadores Unarios (sobre 1 tabla)

- **Selección (σ):** Selecciona un subconjunto de **tuplas (filas)** que cumplen una condición.
    - *Sintaxis:* $\sigma_{\text{condición}}(\text{Relación})$ (Ej: $\sigma_{\text{NroDep}=4}(\text{EMPLEADO})$).
- **Proyección (Π):** Selecciona un subconjunto de **atributos (columnas)**.
    - *Sintaxis:* $\Pi_{\text{lista\_de\_atributos}}(\text{Relación})$ (Ej: $\Pi_{\text{Nombre, Sueldo}}(\text{EMPLEADO})$).
    - *Nota:* En la teoría formal, la proyección **elimina tuplas duplicadas**.

Operadores Binarios (sobre 2 tablas)

- **Producto Cartesiano ($\times$):** Combina *cada* tupla de una relación R con *cada* tupla de una relación S. El resultado tiene $nr \times ns$ tuplas.
- **Reunión (Join - $\bowtie$):** Es la operación más importante. Es un **Producto Cartesiano seguido de una Selección**.
    - *Sintaxis:* $R \bowtie_{\text{condición\_de\_reunión}} S$.
    - **Equireunión:** Es una reunión donde la condición usa solo la igualdad (=).

- **Reunión Natural (Natural Join - $*$):** Es una Equireunión que se hace automáticamente sobre los atributos que tienen el **mismo nombre** en ambas tablas, y **elimina la columna duplicada** en el resultado.
- **Reunión Externa (Outer Join):** Se usa para conservar tuplas que **no tienen una coincidencia** en la otra relación (se rellenan con NULL).
    - **Externa Izquierda ($\bowtie\kern-0.5em\mid$):** Conserva todas las tuplas de la relación de la **izquierda**.
    - **Externa Derecha ($\mid\kern-0.5em\bowtie$):** Conserva todas las de la **derecha**.
    - **Externa Completa ($|\bowtie|$):** Conserva todas las tuplas de **ambas** relaciones.

Operadores de Teoría de Conjuntos

Requieren que las relaciones sean **compatibles con la unión** (mismo número de atributos y mismos dominios).

- **Unión ($\cup$):** Incluye todas las tuplas que están en R, o en S, o en ambas (elimina duplicados).
- **Intersección ($\cap$):** Incluye solo las tuplas que están **tanto en R como en S**.
- **Diferencia ($-$):** Incluye las tuplas que están en R **pero no están en S**.

💻 Unidad 6: Lenguaje SQL (Práctica de Consultas)

SQL (Structured Query Language) se divide en sublenguajes:

- **DDL (Data Definition Language):** Para definir la estructura.
    - CREATE TABLE: Crea una nueva tabla.
    - ALTER TABLE: Modifica una tabla existente (ej: ADD column_name).
    - DROP TABLE: Elimina una tabla.
- **DML (Data Manipulation Language):** Para leer y modificar los datos.
    - INSERT INTO ... VALUES ...: Agrega nuevas filas.
    - UPDATE ... SET ... WHERE ...: Modifica filas existentes.

    - DELETE FROM ... WHERE ...: Elimina filas.
- **DCL (Data Control Language):** Para administrar permisos.

Definición de Restricciones en DDL

- PRIMARY KEY: Especifica la Clave Primaria.
- FOREIGN KEY ... REFERENCES ...: Especifica una Clave Externa y la tabla a la que referencia.
- UNIQUE: Asegura que todos los valores en la columna son únicos (define una Clave Alternativa).
- NOT NULL: El atributo no puede contener valores nulos.
- CHECK: Valida que el valor cumpla una condición (ej: CHECK (Amount > 0)).

La Sentencia SELECT

Es el comando principal para la **recuperación de datos**.

- **Sintaxis y Orden Lógico (simplificado):** El orden en que se *escribe* no es el orden en que se *procesa*.
    1. FROM / JOIN: Obtiene las tablas y las une.
    2. WHERE: **Filtra las filas** individuales.
    3. GROUP BY: Agrupa las filas.
    4. HAVING: **Filtra los grupos** ya creados.
    5. SELECT: Elige las columnas a mostrar.
    6. DISTINCT: Elimina duplicados del resultado final.
    7. ORDER BY: Ordena el resultado final.
- **Cláusulas Clave:**
    - SELECT: Especifica las columnas. SELECT * trae todas. SELECT DISTINCT elimina duplicados.
    - FROM: Especifica las tablas.
    - WHERE: Filtra filas usando predicados.
        - LIKE: Para búsqueda de patrones (ej: LIKE 'F%').

        - BETWEEN: Para rangos (ej: BETWEEN 10 AND 20).
        - IN: Para listas de valores (ej: IN (1, 6, 8)).
        - IS NULL / IS NOT NULL: Para buscar nulos.
    - ORDER BY: Ordena los resultados (por defecto ASC - ascendente; DESC - descendente).
    - GROUP BY: Agrupa filas que tienen el mismo valor en una columna (para usar funciones de agregación como COUNT, SUM, AVG).
    - HAVING: Filtra el resultado *después* de GROUP BY (a diferencia de WHERE que filtra *antes*).

Uniones (JOINS) en SQL

Es la forma estándar (ANSI) de combinar tablas, usada en la cláusula FROM.

- INNER JOIN ... ON ...: (Reunión interna). Devuelve **solo las filas que tienen coincidencia** en ambas tablas, según la condición ON.
- LEFT JOIN ... ON ...: (Externa izquierda). Devuelve **todas las filas de la tabla izquierda** y las filas coincidentes de la tabla derecha. Si no hay coincidencia, rellena con NULL.
- RIGHT JOIN ... ON ...: (Externa derecha). Igual que LEFT pero devuelve **todas las filas de la tabla derecha**.
- FULL JOIN ... ON ...: (Externa completa). Devuelve **todas las filas de ambas tablas**, rellenando con NULL donde no haya coincidencia.