# Resumen Exámen DB

## 1. Álgebra relacional

Existen los siguientes operadores:

- $\sigma$ Selecciona filas de una relación. (Es como el WHERE de SQL)
- $\pi$ Hace una proyección de las columnas de la relación. (Es como el SELECT de SQL)
- $\cup$ Une relaciones, para esto el dominio de las relaciones tiene que ser el mismo. El esquema generado es idéntico al del primer conjunto.
- $\cap$ Intersecta relaciones, tienen que ser union-compatible. El esquema generado es idéntico al del primer conjunto.
- $-$ Diferencia de relaciones, tienen que ser union-compatible. El esquema generado es idéntico al del primer conjunto.
- $\times$ Producto cruz, el esquema generado tiene todos los campos del primer conjunto seguido de todos los del segundo. Contiene todas las combinaciones posibles de los primeros con cada uno de los segundos.
- $\rho$ Renombramiento de relaciones, tiene la siguiente forma $\rho(C(1 \to a, 2 \to b), R1 \times R2$ lo que hace es que a $R1 \times R2$ lo llama $C$ y las columnas en la posición $1$ y $2$ se llamarán $a$ y $b$ respectivamente en el esquema de $C$
- $\bowtie$ Join de relaciones, es un producto cruz seguido de selecciones y proyecciones.
  Se cumple $R \bowtie_c S = \sigma_c(R \times S)$ donde $R, S$ son relaciones y $c$ es una condición. Cuando la condición es una igualdad se llama Equi-Join y cuando la condicion es una igualdad donde ambas columnas tienen el mismo nombre en las relaciones, esta se omite y se llama Natural-Join
- / Division, se mostrará como ejemplo.

Tomemos las siguientes instancias con el fin de realizar ejemplos:

| sid | sname | rating | age |
|---|---|---|---|
| 28 | yuppy | 9 | 35.0 |
| 31 | lubber | 8 | 55.5 |
| 44 | guppy | 5 | 35.0 |
| 58 | rusty | 10 | 35.0 |

T1

| age |
|---|
| 35.0 |

T2

| sname | sex |
|---|---|
| yuppy | male |
| lubber | female |
| guppy | female |
| rusty | male |

T3

1. Seleccionar las filas con rating mayor a 8 en T1: $\sigma_{rating>8}(T1)$
2. Seleccionar sname y rating en T1: $\pi_{sname,rating}(T1)$
3. Seleccionar sname y rating de las filas con rating mayor a 8 en T1: $\pi_{sname,rating}(\sigma_{rating>8}(T1))$

4. Seleccionar sname de las filas con edad 35.0, lo haremos con renombramiento. $\rho(Edad35, \sigma_{age=35,0}(T1)$ y luego $\pi_{sname}(Edad35)$
5. Seleccionar sname, age y sex de las tablas T1 y T3 uniendo por sname. $\pi_{sname,age,sex}(T1 \bowtie T3)$
6. Seleccionar las filas con age 35.0: $T1/T2$

# 2. Cálculo relacional

Ejemplo: Encontrar todas las filas con rating mayor a 8 $\{R|R \in T1 \wedge R.rating > 8\}$

Definición de fórmula atómica: Sean $Rel$ una relación, $R, S$ tuplas y $c$ una constante, entonces una formula atómica es:

- $R \in Rel$
- $R.a$ op $S.b$
- $R.a$ op $c$ ó $c$ op $R.a$

Definición de una formula: Sean $p, q$ fórmulas:

- Una fórmula atómica cualquiera
- $\neg p$, $p \wedge q$, $p \vee q$ y $p \Rightarrow q$
- $\exists R(p(R))$ donde R es una tupla de variables.
- $\forall R(p(R))$ donde R es una tupla de variables.

# 3. Dependencias y Normalización

## 3.1. Dependencia Funcional

**Definición :** Sea $R$ un equema de relación y $X, Y$ conjuntos no vacíos de atributos en $R$. Decimos que una instancia $r$ de $R$ satisface la dependencia funcional $X \rightarrow Y$ si lo siguiente se cumple para cualquier par de tuplas $t1, t2$ en $r$: Si $t1.X = t2.X$, entonces $t1.Y = t2.Y$

Ejemplo de dependencia funcional: $AB \rightarrow C$

| A | B | C | D |
|---|---|---|---|
| a1 | b1 | c1 | d1 |
| a1 | b1 | c1 | d1 |
| a1 | b2 | c2 | d2 |
| a2 | b1 | c3 | d1 |

Una llave primaria es un caso especial de dependencia funcional, donde todos las columnas de la relación dependen de la llave.

Si se cumple que $X \rightarrow Y$, donde $Y$ es el conjunto de todos los atributos, y existe $V \subset X$ tq $V \rightarrow Y$ se cumple, entonces $X$ es una superllave.

### 3.1.1. Axiomas de Armstrong, determinan el conjunto de todas las FDs (Su clausura)

- **Reflexividad** Si $X \supseteq Y$, entonces $X \to Y$
- **Aumentativa** Si $X \to Y$, entonces $XZ \to YZ$ para cualquier $Z$
- **Transitiva** Si $X \to Y$ y $Y \to Z$, entonces $X \to Z$

Es conveniente usar las siguiente reglas adicionales cuando se piensa en la clausura de las dependencias funcionales.

- **Union** Si $X \to Y$ y $X \to Z$, entonces $X \to YZ$
- **Descomposición** Si $X \to YZ$, entonces $X \to Y$ y $X \to Z$,

## 3.2. Dependencia Multivaluada MVD

Un atributo implica que otro atributo este dentro de un grupo de opciones
**Ejemplo:**

| ramo | profesor | libro |
|---|---|---|
| Bases de Datos | Perez | Libro1 |
| Bases de Datos | Perez | Libro2 |
| Bases de Datos | Gonzales | Libro1 |
| Bases de Datos | Gonzales | Libro2 |
| Teoria | Hevia | Libro1 |
| Teoria | Hevia | Libro3 |
| Teoria | Hevia | Libro4 |

Hay 3 puntos que notar aca:

- El esquema esta en BCNF. Por lo tanto no pensaríamos descomponerlo si solo nos preocupamos de las FDs.
- Hay redundancia. El dato que Perez enseña Bases de datos esta repetido por cada libro que el ocupa. Ademas el dato de que se puede enseñar Bases de Datos con el Libro1 esta repetido por cada profesor que lo ocupa.
- La redundancia podría eliminarse descomponiendo el esquema RPL (Ramo,Profesor,Libro) en RP (Ramo,Profesor) y RL (Ramo,Libro)

La redundancia en este ejemplo es debido a la restricción de que los libros por cursos son independientes de los profesores, lo que no puede ser expresado en términos de FDs.
Esta restricción es un ejemplo de dependencias multivaluadas.

**Definición:**
Sea $R$ una relación y $X, Y$ subconjuntos de atributos de $R$. La dependencia multivaluada $X \to\to Y$ pertenece a $R$ si $\forall$ instancia de $R$ cada valor de $X$ es asociado a un conjunto de valores de $Y$ y ese conjunto es independiente de los valores de los otros atributos.

**Formalmente:**
Si la MVD $X \twoheadrightarrow Y$ pertence a $R$ y $Z = (R - XY)$, entonces $\forall r$ instancia de $R$ si $t_1 \in r$, $t_2 \in r$ y $t_1.X = t_2.X$ entonces debe existir algún $t_3 \in r$ tal que $t_1.X = t_3.X$ y $t_2.Z = t_3.Z$

**Ejemplo: MVD $X \twoheadrightarrow Y$**

| $X$ | $Y$ | $Z$ | |
|---|---|---|---|
| $a$ | $b_1$ | $c_1$ | -tupla $t_1$ |
| $a$ | $b_2$ | $c_2$ | -tupla $t_2$ |
| $a$ | $b_1$ | $c_2$ | -tupla $t_3$ |
| $a$ | $b_2$ | $c_1$ | -tupla $t_4$ |

### 3.2.1. Axiomas de Armstrong para MVD

Hay tres reglas adicionales que sólo involucran a las MVDs:

- **Complemento** Si $X \twoheadrightarrow Y$, entonces $X \twoheadrightarrow R - XY$
- **Aumento** Si $X \twoheadrightarrow Y$ y $W \supseteq Z$, entonces $WX \twoheadrightarrow YZ$
- **Transitividad** Si $X \twoheadrightarrow Y$ e $Y \twoheadrightarrow Z$, entonces $X \twoheadrightarrow (Z - Y)$
- **Replicación** Si $X \to Y$, entonces $X \twoheadrightarrow Y$
- **Unión** Si $X \twoheadrightarrow Y$ y existe $W$ tq $W \cap Y$ es vacío, $W \to Z$, y $Y \supseteq Z$, entonces $X \to Z$

## 3.3. Formas Normales

Toda relación en BCNF esta en 3NF. Toda relación en 3NF esta en 2NF y toda relación en 2NF esta en 1NF.

### 3.3.1. 1NF

Cada campo contiene solo valores atómicos, es decir cada campo contiene solo 1 valor.

### 3.3.2. 2NF

Las dependencias parciales no están permitidas.

### 3.3.3. 3NF

Sean $R$ una relación, $X$ un conjunto de atributos de $R$ y $A$ un atributo de $R$. $R$ esta en 3NF si $\forall$ FD (Dependencia Funcional) $X \to A$ en $R$, se cumple uno de los siguientes:

- $A \in X$ (Es decir es una FD trivial)
- $X$ es superllave

- $A$ es parte de una llave de $R$

Sólo hay 3 opciones de FD (Triviales, que una key implique algo, o que algo implique una parte de una key)

Si una FD $X \to A$ viola 3NF, hay solo 2 posibles casos:

- $X$ es un subconjunto de una llave $K$. (En este caso la FD se llama dependencia parcial, y se esta guardando repetidamente el par $(X, A)$)
- $X$ no es subconjunto de ninguna llave. (En este caso la FD se llama dependencia transitiva, debido a que tenemos una cadena de dependencias $K \to X \to A$)

### 3.3.4. BCNF (Boyce-Codd Normal Form)

Libre de redundancia Sean $R$ una relación, $X$ un conjunto de atributos de $R$ y $A$ un atributo de $R$. $R$ esta en BCNF si $\forall$ FD (Dependencia Funcional) $X \to A$ en $R$, se cumple uno de los siguientes:

- $A \in X$ (Es decir es una FD trivial)
- $X$ es superllave

Ejemplo ilustrativo:

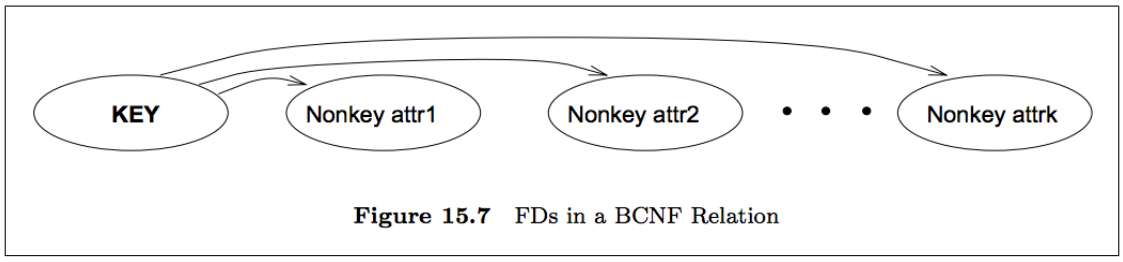


**Figure 15.7** FDs in a BCNF Relation

### 3.3.5. Dependencias parciales (S: subconjunto de una llave, X: conjunto de atributos)

Es de la siguiente forma: $S \to X$.
Cumple 3NF pero no BCNF

### 3.3.6. Dependencias transitivas (K: llave, X: no es parte de una llave, A: atributo)

Es de la siguiente forma: $K \to X \to A$.
No cumple 3NF ni BCNF

## 3.4. Descomposición

Es una técnica para eliminar redundancia

### 3.4.1. Test de descomposición sin perdida

Sea $R$ una relación y $F$ un conjunto de FD en $R$. La descomposición de $R$ en los conjuntos de atributos $R_1$ y $R_2$ es sin perdida $\Leftrightarrow$ $F$ contiene la FD $(R_1 \cap R_2) \rightarrow R_1$ o contiene la FD $(R_1 \cap R_2) \rightarrow R_2$

Si una FD $X \rightarrow Y$ en una relación $R$ tal que $(X \cap Y)$ es vacío, entonces la descomposición de $R$ en $(R - Y)$ y $XY$ es sin perdida.

En BCNF se elimina la redundancia pero se pueden perder FDs
En 3NF se elimina parte de la redundancia pero se puede asegurar no perder FDs

## 3.5. Normalización

### 3.5.1. Descomposición a BCNF (podrían perderse FDs)

1. Suponemos que $R$ no está en BCNF. Sea $X \subset R$, $A \in R$, y $X \rightarrow A$ una FD que viola BCNF. Descomponemos $R$ en $R - A$ y $XA$.
2. Si $R - A$ o $XA$ no está en BCNF, descomponemos nuevamente haciendo recursión con el paso $1$

Ejemplo ilustrativo descomposición a BCNF:

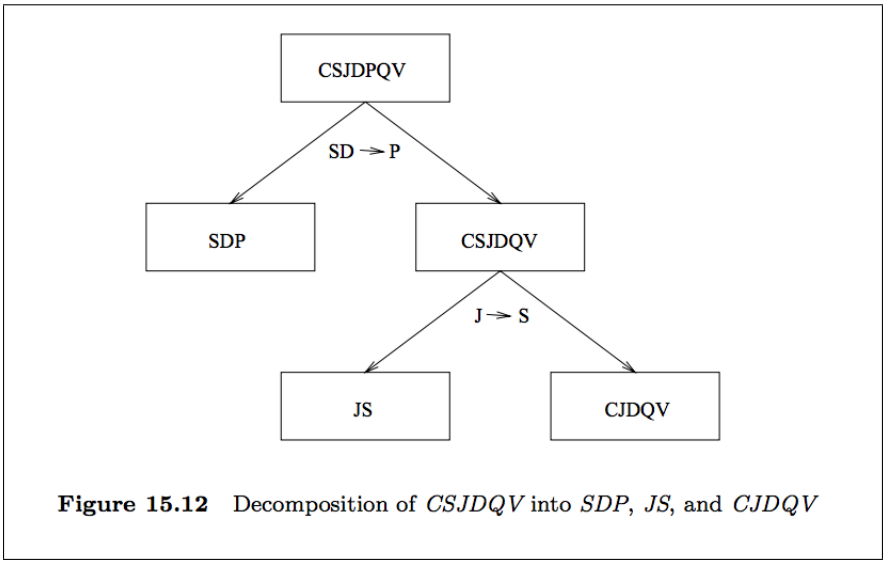


**Figure 15.12** Decomposition of *CSJDQV* into *SDP*, *JS*, and *CJDQV*

### 3.5.2. Descomposición a 3NF

**Minimal Cover (Mínima cantidad de FDs tq la clausura sea la misma que la del conjunto de FDs original** Un minimal cover para un conjunto $F$ de FDs es un conjunto $G$ de FDs tal que:

1. Toda dependencia en $G$ es de la forma $X \rightarrow A$, donde $A$ es un solo atributo.
2. La Clausura $F+$ es igual a la clausura $G+$
3. Si obtenemos un conjunto $H$ de FDs borrando una o más dependencias del conjunto $G$, o borrando atributos de una dependencia de $G$, entonces $F+ \neq H+$

Intuitivamente, un minimal cover de un conjunto $F$ de FDs es un conjunto de dependencias que es mínimo en dos aspectos:

1. Cada dependencia es tan pequeña como es posible; esto es que cada atributo en el lado izquierdo es necesario y que el lado derecho tiene sólo un atributo.
2. Cada dependencia es necesaria para que la clausura sea igual a $F+$

**Algoritmo para encontrar minimal cover de $F$**

1. Obtener un conjunto $G$ equivalente a $F$ donde todas las FDs tienen sólo un atributo al lado derecho (Usando el axioma de descomposición)
2. Minimizar el lado izquierdo de cada FD: Para cada FD $g \in G$, revizamos cada atributo del lado izquierdo y vemos si puede ser borrado manteniendo la equivalencia con $F+$.
3. Borrar FDs redundantes: Revisar cada FD restante en $G$, se borra si se mantiene la equivalencia con $F+$

**Algoritmo para transformar a 3NF**

- Identificar el conjunto $N$ de dependencias en $F$ que no cumplen con 3NF
- Para cada FD $X \to A$ en $N$, crear una relación $XA$ y sumarla a la descomposición de $R$

**Método:** Sea $F$ el conjunto de FDs de una relación $R$

1. Obtener el minimal basis (El conjunto $N$ del paso anterior) de $F$, llamémoslo $G$.
2. Para cada dependencia funcional $X \to A$ en $G$, usar la relacion $XA$ como parte de la descomposición de $R$.
3. Si ni uno de los esquemas del paso 2 es una superllave de $R$, añadir una relación cuyo esquema sea una llave de $R$.

# 4. SQL

Forma típica de una query en SQL:

```
SELECT [DISTINCT] [column_1] ... [column_n]
FROM [table_1] ... [table_n]
[WHERE condition_1 ... [AND|OR condition_n]]
[GROUP BY column_1
[HAVING condition_1 ... [AND|OR condition_n]]]
[ORDER BY column_i ASC|DESC]
```

## 4.1. Create Tables

Crear una tabla de departamentos

```
CREATE TABLE departamento (
    dep_nom VARCHAR(50),
    fecha_creacion DATE,
    telefono INT,
    direccion VARCHAR(100),
    comuna VARCHAR(20),

    PRIMARY KEY (dep_nom)
);
```

Crear una tabla empleado que referencia a la de departamentos

```
CREATE TABLE empleado (
    nombre VARCHAR(50),
    depto VARCHAR(50),
    sueldo MONEY DEFAULT 500000,
    fecha_ing DATE NOT NULL,

    PRIMARY KEY (nombre),

    FOREIGN KEY (depto) REFERENCES departamento(dep_nom) ON DELETE CASCADE,

    CHECK (sueldo > 0)
);
```

Las opciones para ON DELETE son:

- **CASCADE** Elimina si se elimina el padre
- **RESTRICT** Detiene la operación de delete en el padre
- **NO ACTION** Lo mismo que RESTRICT
- **SET NULL** Elimina el padre y pone en NULL en la foreign key

## 4.2. ALTER TABLE

Se ocupa para modificar una tabla:

```
ALTER TABLE departamento
    ADD COLUMN jefe VARCHAR(50);
ALTER TABLE departamento
    ADD FOREIGN KEY (jefe) REFERENCES empleado(nombre);
ALTER TABLE departamento
    DROP COLUMN jefe;
```

## 4.3. CREATE INDEX

Se usa para agregar un INDEX a una tabla

```
CREATE INDEX direccionIndex ON departamento
WITH STRUCTURE = [BTREE|HASH],
KEY = (direccion)
```

## 4.4. Queries

### 4.4.1. JOIN

- **JOIN** Es el producto cruz de las dos tablas, se puede especificar un join condition con la cláusula ON.

```
SELECT e.nombre, d.dep_nom, d.telefono
FROM empleado e JOIN departamento d ON d.dep_nom = e.depto
```

- **LEFT OUTER JOIN** Es el join entre las tablas, las filas de la primera tabla que no coinciden con alguna de la seguna aparecen igual mostrando NULL en los valores correspondiente a la segunda tabla.
- **RIGHT OUTER JOIN** Análogo al anterior
- **FULL OUTER JOIN** Muestra NULL en todos los que no coinciden en ambos lados.

### 4.4.2. GROUP BY/HAVING

Muestra los departamentos que gastan mas de 5.000.000 en sueldos de sus empleados y el total que gastan.

```
SELECT d.dep_nom, SUM(e.sueldo)
FROM empleado e JOIN departamento d ON d.dep_nom = e.depto
GROUP BY d.dep_nom
HAVING SUM(e.sueldo) > 5000000
```

### 4.4.3. UNION/EXCEPT/INTERSECT

- **UNION** Q1 UNION Q2 entrega los resultados que aparecen en Q1 o en Q2, si hay resultados repetidos sólo se muestran una vez.
- **UNION ALL** Q1 UNION ALL Q2 entrega los resultados que aparece en Q1 o en Q2, si hay resultados repetidos estos se muestran varias veces
- **EXCEPT** Q1 EXCEPT Q2 entrega los resultados que están en Q1 y no en Q2.
- **INTERSECT** Q1 INTERSECT Q2 entrega los resultados que están en Q1 y en Q2

### 4.4.4. NESTED QUERIES

- **IN** La condición es que la columna tenga uno de los valores que devuelve Q2

  **SELECT** ...
  **FROM** ...
  **WHERE column IN** (Q2)

- **NOT IN** La condición es que la columna **no** tenga uno de los valores que devuelve Q2

  **SELECT** ...
  **FROM** ...
  **WHERE column NOT IN** (Q2)

- **EXISTS** La condición es que Q2 devuelva al menos un resultado

  **SELECT** ...
  **FROM** ...
  **WHERE EXISTS** (Q2)

- **ANY/SOME** Que haya algún valor de los resultados de Q2 que cumpla la condición

  **SELECT** ...
  **FROM** ...
  **WHERE column** [=|>|<>|<] **ANY** (Q2)

- **ALL** Que todos los valores de los resultados de Q2 cumplan la condición

  **SELECT** ...
  **FROM** ...
  **WHERE column** [=|>|<>|<] **ALL** (Q2)

  También se pueden usar queries anidadas dentro del FROM. Para el HAVING se pueden usar los operadores ANY y EVERY.

### 4.4.5. LIKE

Se usa para comparar con strings, tiene los siguientes caracteres especiales:

- **%** 0 o más caracteres arbitrarios.
- _ Exactamente 1 caracter arbitrario.

### 4.4.6. AGREGATE OPERATORS

- **COUNT [DISTINCT]** cantidad de valores [únicos]
- **SUM [DISTINCT]** suma de todos los valores [únicos]
- **AVG [DISTINCT]** promedio de los valores [únicos]
- **MAX** máximo entre los valores
- **MIN** mínimo entre los valores

## 4.5. VIEWS

Son pseudo-tablas, muestran información de una o más tablas según la query asignada al momento de crearla

```
CREATE VIEW empleadosRicos (nombre, sueldo)
AS SELECT e.nombre, e.sueldo
   FROM empleado e
   WHERE e.sueldo >= 1000000;
```

## 4.6. TRIGGERS

Son acciones que se ejecutan frente a un evento cuando se cumple una condición. El siguiente trigger crea una variable y la inicializa en 0 antes de un insert

```
CREATE TRIGGER declararVariable
BEFORE INSERT ON departamentos
DECLARE
   count INTEGER;
BEGIN
   count := 0;
END
```

El siguiente trigger hace que todos los empleados que se ingresen con sueldo menor a 100.000, le asigna 100.000

```
CREATE TRIGGER sueldoMinimo
BEFORE INSERT ON empleado
FOR EACH ROW
BEGIN
   IF NEW.sueldo < 100000 THEN
      NEW.sueldo = 100000;
   END IF;
END
```

El siguiente trigger inserta en employee_tools las tuplas (employee.id, tool.name) donde tool.division = new.division, es decir todas las herramientas correspondientes a la división del empleado.

```
CREATE TRIGGER employee_INSERT
AFTER INSERT ON employee
FOR EACH ROW
BEGIN
    INSERT INTO employee_tools (Id, Tool)
        SELECT NEW.Id, tools.Tool_Name
        FROM tools
        WHERE tools.Division = NEW.Division;
END
```

## 4.7. TRANSACTIONS

Una transacción es la ejecución de un programa, ya sea de lectura o escritura. Se usa el acronimo ACID (Atomicity Consistency Isolation Durabilty) para nombrar las cuatro propiedades de una trasacción.

- **Atomicity** Los usuarios no deben preocuparse de los efectos de transacciones incompletas, cada transacción debería ejecutarse sin interferir en las otras, esta labor es del DBMS.
- **Consistency** Cada transacción debería ocurrir de manera no concurrente, el DBMS asume que las transacciones mantienen la consistencia. Por la tanto mantenerla es responsabilidad del usuario.
- **Isolation** El usuario no debe preocuparse de problemas que puedan ocurrir de manera simultanea, el DBMS se encarga de esto.
- **Durability** Cuando el DBMS avisa al usuario que la transacción se realizó adecuadamente, la información debería persistir aunque el sistema se caiga. Claramente el DBMS es responsable de esto.

SQL permite al usuario determinar 3 características para una transacción:

- **Access Mode** puede ser READ ONLY que significa que la transacción no puede modificar la DB, o READ WRITE que permite leer y escribir en la DB
- **Diagnostics Size** Determina la cantidad de errores que pueden ser guardados para mostrarlos.
- **Isolation level** Hay cuatro niveles distintos de aislamiento para controlar las transacciones concurrentes. Estos existen para permitir mayor concurrencia aumentando el riesgo de errores. Se especifican abajo.

Los niveles de aislamiento, ordenados de mayor aislamiento a menor, son:

- **Serializable** Asegura que la transacción sólo va a leer cambios realizados por otras transacciones commiteadas. Ningún valor que la transacción lea o escriba va a ser cambiado por otra transacción hasta que esta transacción termine. Si la transacción lee un conjunto de valores dependientes de una condición de búsqueda, este conjunto no puede ser modificado por ninguna otra transacción hasta que esta termine.
- **Repeatable Read** Asegura que la transacción sólo va a leer cambios realizados por otras transacciones commiteadas. Ningún valor que la transacción lea o escriba va a ser cambiado por otra transacción hasta que esta transacción termine.

- **Read Commited** Asegura que la transacción sólo va a leer cambios realizados por otras transacciones commiteadas. Ningún valor que la transaccion escriba va a ser cambiado por otra transacción hasta que esta transacción termine. Los valores que se leen podrían ser modificados por otra transacción antes de que esta termine. Está expuesta a phantom read (lecturas fantasmas).
- **Read Uncommited** Los valores leídos pueden ser cambiados por otras transacciones antes de que esta termine. Está expuesta a phantom read. Para usar esta transacción se tiene que estar en ACCESS MODE READ ONLY.

# 5. XML

# 6. SPARQL