CONCURRENCIA DE DATOS Y CONSISTENCIA

JUAN MANUEL MANGA HURTADO

JOSÉ LUIS DÍAZ PAVÓN

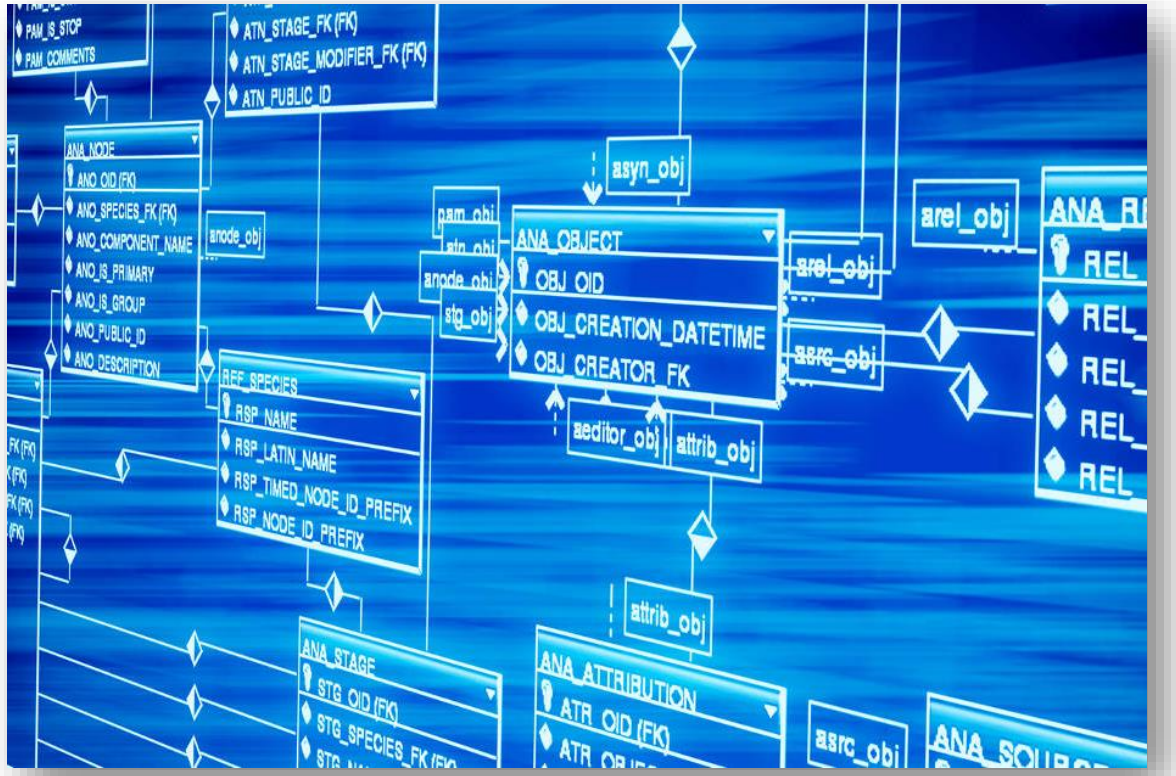
ÍNDICE

1. INTRODUCCIÓN A LA CONSISTENCIA DE DATOS Y CONCURRENCIA

- ¿QUÉ ES LA **CONSISTENCIA DE DATOS**?
  - Concepto fundamental en la gestión de bases de datos.
  - Propiedad de una base de datos de mantener una información correcta y coherente en todo momento = datos precisos y libres de contradicciones.
- ¿CÓMO PODEMOS GARANTIZAR LA CONSISTENCIA DE DATOS EN UNA BD?
  - Haciendo uso de restricciones de integridad (claves primarias, foráneas, restricciones de unicidad, etc.
  - Gracias a estas restricciones podemos garantizar que los datos almacenados en la base de datos cumplen con ciertas condiciones y que se respeten las relaciones entre las distintas tablas.

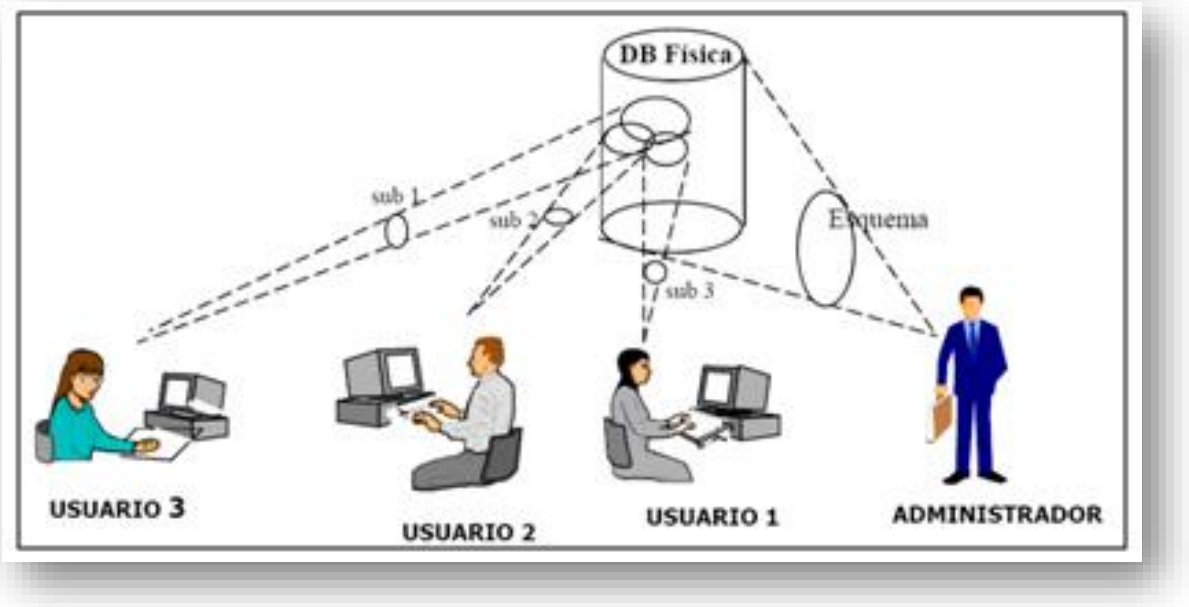
1. INTRODUCCIÓN A LA CONSISTENCIA DE DATOS Y CONCURRENCIA

- ¿CÓMO DE IMPORTANTE ES MANTENER LA CONSISTENCIA EN UNA BD?
  - Es crucial para asegurar la calidad de la información almacenada y la confiabilidad de los resultados obtenidos a partir de ella.
  - Cualquier base de datos que no cumpla con la propiedad de consistencia puede llevar a resultados incorrectos = consecuencias graves para los usuarios y organizaciones que dependen de la información almacenada en la BD.



1. INTRODUCCIÓN A LA CONSISTENCIA DE DATOS Y CONCURRENCIA

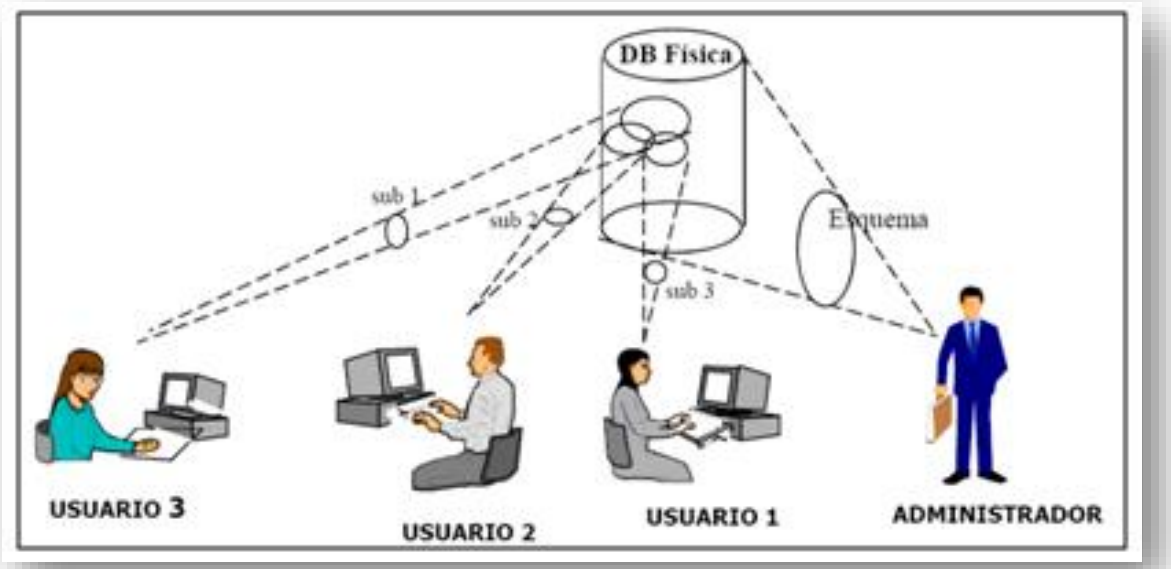
- ¿QUÉ ES LA **CONCURRENCIA** EN UNA BASE DE DATOS?
  - Capacidad que tiene una BD de que múltiples usuarios o aplicaciones puedan acceder y modificar los datos almacenados en ella, de manera simultánea.
- ¿QUÉ *VENTAJAS* TIENE LA CONCURRENCIA EN LAS BASES DE DATOS?
  1. **Mejor rendimiento**. Permite que varios usuarios o procesos accedan simultáneamente.
  2. **Aumenta la escalabilidad**. Soporta un mayor número de usuarios y transacciones.
  3. **Aumenta la disponibilidad**. Los usuarios pueden seguir trabajando incluso si un proceso o transacción se bloquea o falla.
  4. **Optimiza el uso de recursos.** Los recursos no se desperdician mientras esperan a que un usuario o proceso termine su trabajo.
  5. **Mejora la eficiencia**. Se reduce el tiempo de espera.



1. INTRODUCCIÓN A LA CONSISTENCIA DE DATOS Y CONCURRENCIA

- ¿QUÉ *DESVENTAJAS* TIENE LA CONCURRENCIA EN LAS BBDDs?

1. **Conflictos**. Como varios usuarios o procesos acceden de forma simultánea, hay riesgo de que se produzcan conflictos y como consecuencia se pueden generar resultados inesperados y erróneos.
2. **Problemas de sincronización**. Mantener la coherencia y la integridad de los datos en un ambiente concurrente puede ser complicado y requiere técnicas y herramientas específicas para evitar problemas de sincronización.
3. **Pérdida de rendimiento**. Si no se implementa correctamente, la concurrencia puede tener efectos negativos en el rendimiento.
4. **Dificultad en la detección de errores**. Al ser concurrente, detectar un error puede ser difícil, ya que varios usuarios o procesos pueden estar modificando los mismos datos al mismo tiempo.
5. **Mayor complejidad**. Mantener un sistema de concurrencia en una base de datos es más complejo que en un sistema sin concurrencia.



1. INTRODUCCIÓN A LA CONSISTENCIA DE DATOS Y CONCURRENCIA

- Nace la **serializabilidad**:
  - Los procesos de un conjunto de transacciones simultáneas deben dar el mismo resultado que si se ejecutaran en serie las transacciones individuales y que si cada una de las transacciones tuviera un uso exclusivo del sistema.
- TIPOS DE PLANES:
  - A. **Plan en serie**. No intercala las operaciones de diferentes transacciones.
  - B. **Plan equivalente**. Si para cualquier estado de la BD, el efecto de ejecución de un plan es idéntico al de la ejecución de otro.
  - C. **Plan serializable**. Es aquel plan que es equivalente a un plan en serie.

1. INTRODUCCIÓN A LA CONSISTENCIA DE DATOS Y CONCURRENCIA

- ¿DE QUÉ MANERA HACE POSIBLE LA CONCURRENCIA ORACLE 11g?
  - Presenta vistas (distintas versiones) de los datos a los múltiples usuarios simultáneamente, con cada vista consistente en un punto en el tiempo.
- PROBLEMA QUE SE PLANTEA:
  - Debido a que pueden existir diferentes versiones de los bloques de datos de las distintas vistas, las transacciones pueden leer la versión de los datos comprometidos en el momento requerido por una consulta y devolver resultados que son coherentes con un solo momento.

1. INTRODUCCIÓN A LA CONSISTENCIA DE DATOS Y CONCURRENCIA

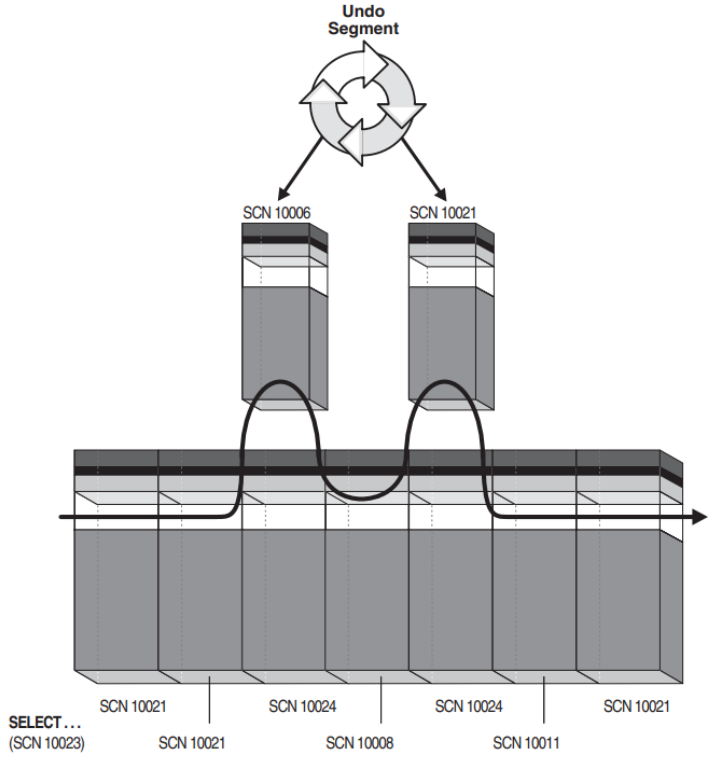
- Oracle tiene la habilidad de tener varias versiones de datos simultáneamente.
- Las consultas a las bases de datos pueden ser de dos tipos:
  - **Consultas de lectura coherente**. Los datos devueltos por una consulta están comprometidos y son consistentes con respecto a un punto en el tiempo.
  - **Consultas sin bloqueo**. Los lectores y escritores de datos no se bloquean entre sí.
- Oracle garantiza que los datos devueltos por una sola consulta están comprometidos y son consistentes con respecto a un único punto en el tiempo.

2. VISIÓN GENERAL DE LOS NIVELES DE AISLAMIENTO

- Al existir varias versiones de datos simultáneamente, las bases de datos Oracle crean un conjunto de datos de lectura consistente cuando una tabla se consulta y actualiza simultáneamente. Este objetivo lo consigue haciendo uso de → **undo data**.
- ¿CÓMO LO HACE?
  - Cada vez que un usuario modifica datos, Oracle crea segmentos que contienen los valores antiguos de los datos que han sido modificados, pero no confirmados recientemente.
  - La base de datos podrá usar éstas "instantáneas" de datos en diferentes puntos en el tiempo para poder proporcionar vistas coherentes de lectura de los datos y permitir consultas sin bloqueos.

1. INTRODUCCIÓN A LA CONSISTENCIA DE DATOS Y CONCURRENCIA

- ¿CÓMO SE GARANTIZA EL ORDEN DE LAS TRANSACCIONES?
  - Haciendo uso de un mecanismo llamado "SCN".
  - Cuando la sentencia "SELECT" entra en fase de ejecución, la base de datos determina el valor del SCN registrado en el momento en que la consulta comenzó a ejecutarse (10023).
  - En la figura vemos que los bloques después del 10023 indican cambios de datos (2 bloques 10024).
  - La sentencia "SELECT" requiere una versión del bloque que sea coherente con los cambios confirmados.
  - La base de datos copia los nuevos bloques de datos en un buffer y aplica el "undo" para recuperar versiones anteriores.



2. VISIÓN GENERAL DE LOS NIVELES DE AISLAMIENTO

- Para que una declaración SQL sea consistente en un único punto en el tiempo depende del nivel de aislamiento de la transacción y la naturaleza de la consulta.
- Hay distintos niveles de aislamiento:
  - **Serializable** (por defecto). Afecta sólo a quien lo elige y no al resto de usuarios de la BD.
  - **Repeatable Read**. Evita el problema de "lectura no repetible".
  - **Read Commited**. Evita la lectura sucia, ya que la transacción sólo podrá leer datos que han sido reafirmados por el commit de otra transacción.
  - **Read Uncommited**. Es el nivel más permisivo. Una transacción que tenga este nivel de aislamiento puede ver valores que otra transacción ha escrito, o deja ver valores que otra haya borrado, a pesar de que ésta no haya hecho el commit. Este nivel de aislamiento permite lecturas sucias, lecturas no repetibles y lecturas fantasmas.

CONFLICTO / NIVEL AISLAMIENTO	ACTUALIZACION PERDIDA	LECTURA NO CONFIRMADA	LECTURA NO REPETIBLE Y ANALISIS INCONSISTENTE (EXCEPTO FANTASMAS)	FANTASMAS
READ UNCOMMITED	SI	NO	NO	NO
READ COMMITED	SI	SI	NO	NO
REPEATABLE READ	SI	SI	SI	NO
SERIALIZABLE	SI	SI	SI	SI

SI= Si Protege / **NO=** No Protege

2. VISIÓN GENERAL DE LOS NIVELES DE AISLAMIENTO

- EJEMPLOS DE LOS NIVELES DE AISLAMIENTO
  - Supongamos que tenemos una base de datos con la siguiente relación:

VENDE (bar, cerveza, precio)

CP: (bar, cerveza)

  - Vamos a suponer que el bar "Fran" vende dos marcas de cerveza:

Cruzcampo (2€) y Mahou (2,5€)

  - Imaginemos ahora que existen dos usuarios en la BD: "Fran" y "Javi"

2. VISIÓN GENERAL DE LOS NIVELES DE AISLAMIENTO

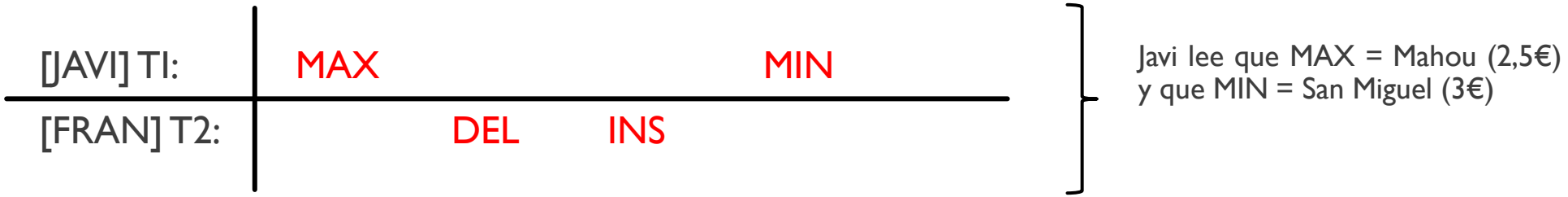
- Estos usuarios desean realizar las siguientes transacciones:
  - T1: Javi quiere saber cuál es la cerveza más cara y más barata del bar "Fran".
  - T2: Fran (el dueño del bar), procede a eliminar las cervezas Cruzcampo y Mahou para ahora vender sólo cervezas San Miguel a 3€.
- Las consultas serían las siguientes:
  - JAVI [T1]: select max (precio) from VENDE where bar = "Fran" **MAX**

    select min (precio) from VENDE where bar = "Fran" **MIN**
  - FRAN [T2]: delete from VENDE where bar = "Fran" **DEL**

    insert into VENDE values ('Fran', 'San Miguel', 3) **INS**
- Si ambas transacciones se ejecutan a la vez, solamente podríamos afirmar que MAX va antes que MIN en T1 y que DEL va antes que INS en T2.

2. VISIÓN GENERAL DE LOS NIVELES DE AISLAMIENTO

- Un posible plan de ejecución de ambas transacciones podría ser el siguiente:



- ¿CÓMO SERÍA EL NIVEL SERIALIZABLE?
  - Si Javi ejecuta sus instrucciones haciendo uso de este nivel, verá la BD antes o después de que Fran ejecute sus instrucciones, pero nunca en medio.
  - La elección de este nivel sólo afecta al que lo elige.
  - Si Fran ejecuta T2 con nivel serializable y Javi no, entonces Javi podría ver los datos como si T1 se ejecutara a la mitad de la de Fran.

2. VISIÓN GENERAL DE LOS NIVELES DE AISLAMIENTO

- ¿CÓMO SERÍA EL NIVEL READ COMMITED?
  - Este nivel evita la "Lectura sucia" o no confirmada. La transacción sólo podrá leer los datos que hayan sido confirmados (commit) por otra transacción.
  - Ejemplo de lectura sucia:

Javi lee que MAX = San Miguel (3€) y que MIN = San Miguel (3€), aunque este dato nunca llegue a existir

  - En este nivel se asegura que Javi no lea 3€ si Pepe hace rollback.

2. VISIÓN GENERAL DE LOS NIVELES DE AISLAMIENTO

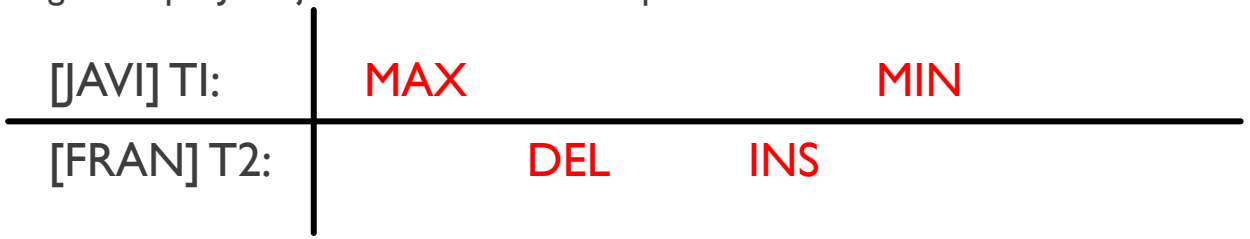
- Este nivel es más permisivo que el serializable, lo que genera problemas como el siguiente

Sería factible en este nivel siempre que Fran haga commit. En ese caso, Javi verá MAX = Mahou (2,5€) y MIN = San Miguel (3€)

- En este nivel se asegura que Javi no lea 3€ si Pepe hace rollback.

2. VISIÓN GENERAL DE LOS NIVELES DE AISLAMIENTO

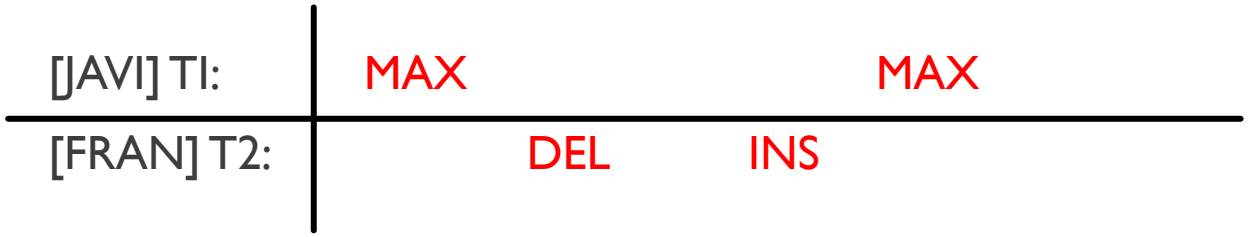
- ¿CÓMO SERÍA EL NIVEL REPETEATBLE READ?
  - Este nivel evita la lectura no repetible. Es similar al anterior añadiendo la restricción de que "en toda transacción, todo lo que se mostró en una lectura inicial, debe ser mostrado si se ejecuta de nuevo posteriormente la lectura".
  - Supongamos que Javi ejecuta T1 con nivel "Repeatble Read":



El sistema debe asegurar que, al ejecutar MIN, se vean además del valor 3€, los valores 2€ y 2,5€ vistos en MAX.

En este caso, Javi verá: MAX = 2,5€ y MIN = 2€, aunque estos datos no reflejan el estado actual de la BD.

  - Este nivel vuelve a ser más permisivo que el serializable, por lo que generará problemas como:



Si Javi ejecuta T1 con este nivel, lo que lee en el 1º MAX, lo lee también en el segundo MAX.

Sin embargo, realmente, obtiene en el primer MAX = 2,5€ y en el segundo MAX = 3€.

2. VISIÓN GENERAL DE LOS NIVELES DE AISLAMIENTO

- ¿CÓMO SERÍA EL NIVEL READ UNCOMMITED?
  - Este es el nivel más permisivo y el que causa más problemas.
  - Una transacción que se ejecute con este nivel de aislamiento puede ver valores que haya escrito otra transacción o dejar de ver valores que ésta borre, a pesar de que la transacción no haya hecho commit.
  - Javi podría ver el valor 3€ como precio MAX o MIN, a pesar de que pepe posteriormente haga INS y haga RoolBack.
  - Problemas que causa este nivel de aislamiento:
    - Lectura sucia.
    - Lectura no repetible.
    - Lectura fantasma.

3. VISIÓN GENERAL DE LOS MECANISMOS DE BLOQUEOS DE TRANSACCIÓN DE ORACLE

- ¿QUÉ SON LOS MECANISMOS DE BLOQUEOS?
  - Son recursos compartidos que usan varios usuarios para garantizar la integridad de los datos en condiciones de concurrencia.
  - Hay dos tipos de bloqueos en la base de datos:
    - **Bloqueos exclusivos** (Exclusive Locks o X Lock). Una transacción obtiene un bloqueo exclusivo cuando modifica datos.
    - **Bloqueos compartidos** (Share Lock o S Lock). Permite compartir el recurso bloqueado, dependiendo de las operaciones involucradas. Varios usuarios que leen datos pueden compartir los datos, manteniendo bloqueos compartidos para evitar el acceso simultáneo de un escritor que necesita un bloqueo exclusivo. Varias transacciones pueden adquirir bloqueos compartidos en el mismo recurso.
  - Se puede hacer uso de un bloqueo exclusivo en un recurso, como una fila o tabla, pero, sin embargo, se pueden tener muchos bloqueos compartidos en un solo recurso.

3. VISIÓN GENERAL DE LOS MECANISMOS DE BLOQUEOS DE TRANSACCIÓN DE ORACLE

- Los bloqueos afectan tanto a operaciones de lectura como de escritura. Una de ellas se encarga de consultar el recurso mientras que la escritura de modificarlo.
- Las siguientes reglas resumen el comportamiento de las bases de datos Oracle para las operaciones de lectura y escritura:
  - Una fila se bloquea sólo cuando la modifica un escritor.

    Cuando se actualiza una fila, la transacción adquiere un bloqueo sólo para dicha fila.
  - Un escritor de una fila bloquea a otro escritor simultáneo de dicha fila.

    Si una transacción modifica una fila, el bloqueo evita que otra transacción modifique la misma fila simultáneamente.
  - Un lector nunca bloquea a un escritor.

    Un escritor podría modificar la fila que un lector está consultando. Sin embargo, para las sentencias SELECT, FOR UPDATE, que son de tipo especial ya que dichas sentencias bloquean la fila que está leyendo.
  - Un escritor nunca bloquea a un lector.

    Cuando un escritor cambia una fila, la base de datos hace uso de la operación "**undo**", para proporcionar una vista a los lectores coherente de la fila.

3. VISIÓN GENERAL DE LOS MECANISMOS DE BLOQUEOS DE TRANSACCIÓN DE ORACLE

- ¿CUÁNDO SE LIBERA UN BLOQUEO?
  - Las bases de datos de Oracle liberan todos los bloqueos adquiridos cuando se confirma o se retrocede, es decir, cuando se hace un commit o un rollback.
  - Sin embargo, sólo las transacciones que no están esperando ningún recurso que esté bloqueado, puede adquirir bloqueos en los recursos disponibles.
  - Las transacciones que están en espera siguen esperando hasta que la transacción original se confirma o se revierte por completo.
- Se pueden producir interbloqueos cuando dos o más transacciones intentan bloquear los mismos recursos y ninguno de ellos puede continuar hasta que uno de ellos libere el bloqueo.
- ¿QUÉ HACE LA BASE DE DATOS CUANDO SE PRODUCE UN INTERBLOQUEO?
  - Los detecta automáticamente y los resuelve deshaciendo una declaración que esté involucrada en dicho interbloqueo, liberando un conjunto de bloqueos de fila en conflicto.
  - La base de datos devuelve un mensaje correspondiente a la transacción que sufre dicha reversión.

3.VISIÓN GENERAL DE LOS MECANISMOS DE BLOQUEOS DE TRANSACCIÓN DE ORACLE

- EJEMPLO DE INTERBLOQUEO:

Time	Session 1	Session 2
t1	SQL> UPDATE employees SET salary = salary*1.1 WHERE employee_id = 200; -- prompt does not return	SQL> UPDATE employees salary = salary*1.1 WHERE employee_id = 100; -- prompt does not return
t2	UPDATE employees * ERROR at line 1: ORA-00060: deadlock detected while waiting for resource SQL>	
t3	SQL> COMMIT; Commit complete.	
t4		1 row updated. SQL>
t5		SQL> COMMIT; Commit complete.

4. VISIÓN GENERAL DE LOS BLOQUEOS AUTOMÁTICOS

Un bloqueo automático en una base de datos es una medida de seguridad que se utiliza para evitar que varios procesos o transacciones accedan a la misma información al mismo tiempo y causen conflictos o inconsistencias.

En Oracle los bloqueos automáticos se dividen en las siguientes categorías:

- Bloqueos DML (Lenguaje de manipulación de datos)
- Bloqueos DDL (Lenguaje de definición de datos)
- Bloqueos del sistema

4. VISIÓN GENERAL DE LOS BLOQUEOS AUTOMÁTICOS BLOQUEOS DML

Dentro de los propios bloqueos DML existen varios tipos que pueden ser de fila o tabla:

- Bloqueos compartidos (Shared Locks): Este tipo de bloqueo se dan operaciones de lectura. Cuando se aplica este bloqueo otras transacciones pueden seguir leyendo del recurso (tabla completa o fila) pero, en ningún caso, pueden modificarlo.
- Bloqueos exclusivos (Exclusive Locks): En este tipo de bloqueo otras transacciones no pueden leer ni actualizar el recurso.

Los bloqueos DML (tanto de fila como de tabla) pueden causar problemas de rendimiento y bloquear el acceso a datos de forma innecesaria. Por lo tanto, la mejor práctica es liberarlos tan pronto como sea posible.

4. VISIÓN GENERAL DE LOS BLOQUEOS AUTOMÁTICOS BLOQUEOS DML EJEMPLOS

Bloqueos de fila TX:

Transacción 1	Transacción 2
SELECT id, name FROM user WHERE id IN (1, 2);	SELECT id, name FROM user WHERE id IN (1, 2);
UPDATE user SET name="newNameT1" WHERE id=1	
	UPDATE user SET name="newNameT1" WHERE id=2
SELECT id, name FROM user WHERE id IN (1, 2);	SELECT id, name FROM user WHERE id IN (1, 2);

TX

TX

1- newNameT1
2-originalName

1- originalName
2-newNameT2

4. VISIÓN GENERAL DE LOS BLOQUEOS AUTOMÁTICOS BLOQUEOS DML CLAVES AJENAS

Se realiza un bloqueo de tabla completo cuando:

- No hay índices en la foreign key de la tabla referenciada
- Se modifica una primary key en la tabla principal

No se realiza un bloqueo de tabla cuando:

- La foreign key está indexada
- Se inserta en la tabla principal

4. VISIÓN GENERAL DE LOS BLOQUEOS AUTOMÁTICOS BLOQUEOS DDL

Las bases de datos realizan automáticamente bloqueos DDL si cualquier transacción DDL(CREATE, ALTER, DROP) lo necesitan, los usuarios no pueden requerir bloqueos DDL de forma específica.

Ejemplo

```
ALTER TABLE user RENAME COLUMN name TO user_name;
```

Esta consulta provocaría en un bloqueo DDL en la tabla user, de forma que otras transacciones no tendrán acceso a esta hasta que ya transacción haya terminado

5. VISIÓN GENERAL DE LOS BLOQUEOS DE DATOS MANUALES

Las bases de datos Oracle realizan bloqueos de forma automática para garantizar la concurrencia y la integridad de los datos. De todas formas, también nos proporciona mecanismos para redefinir estos bloqueos automáticos.
Aplicar bloqueos manuales puede ser útil en las siguientes situaciones:

- Se requiere consistencia a nivel de transacción para las lecturas (lectura repetible)
- Se requiere que una transacción tenga acceso exclusivo a un recurso

También pueden modificarse los bloqueos a nivel de sesión utilizando la sentencia ALTER SESSION

5. VISIÓN GENERAL DE LOS BLOQUEOS DE DATOS MANUALES EJEMPLOS

Sentencias de bloqueo manual
SET TRANSACTION ISOLATION LEVEL
LOCK TABLE
SELECT * FROM TABLE FOR UPDATE

6. VISIÓN GENERAL DE LOS BLOQUEOS DEFINIDOS POR EL USUARIO

Oracle también nos permite definir nuestros propios bloqueos. Los casos en los que esta función puede ser útil son muy variados y dependen mucho del tipo de aplicación.

Los bloqueos de usuario de identifican con el prefijo UL.

El servicio de gestión se encuentra en el paquete DBMS_LOCK en Oracle.

Función	Uso
ALLOCATE_UNIQUE	Asigna un identificador único al bloqueo
REQUEST	Solicita el bloqueo en el modo especificado
RELEASE	Libera el bloqueo