

# Conciliación de extractos de hoteles

## Trabajo Práctico Especial de Bases de Datos 2

### Integrantes:

- Pablo Alejandro Costesich, 50109
- Horacio Miguel Gómez, 50825
- Juan Pablo Orsay, 49373
- Sebastián Maio, 50386
- Agustin Golmar, 53396
- Conciliación de extractos de hoteles
  - Introducción
  - Situación inicial
  - Análisis de la Solución
    - Problemas Encontrados
      - Los tablespaces no son auto-extensibles
      - Supplier.status nunca se usa
    - Optimizaciones Realizadas
      - Mejora 0 - Eliminar STATEMENT LOCATOR de conciliate booking
      - Mejora 1 - Columna RECORD LOCATOR en PAYMENT ORDER y HOTEL STATEMENT
      - Mejora 2 - Eliminación completa del paso intermedio conciliate statement
      - Mejora 3 - Conciliation STATUS
      - Mejora 4 - Una sola query (resolviendo el problema N+1)
      - Mejora 5 - Final Stage
  - Resultados
    - Plan de Ejecución
    - Recomendaciones
  - Anexo
    - Mejora 0
    - Mejora 1
    - Mejora 2
    - Mejora 3
    - Mejora 4
    - Mejora 5

## Introducción

Una empresa de turismo desea optimizar un proceso de conciliación. Este proceso contrasta los extractos que les envían los hoteles contra los registros de ventas de la empresa, y sirve para verificar el pago que se le debe hacer a los hoteles proveedores por las reservas generadas. Por

desgracia, no hay documentación asociada al proceso. La empresa que realizó el desarrollo original no entregó la documentación esperada ya que terminaron su contrato de mala manera: hubo una historia de incumplimientos en fechas de entrega y la performance del sistema no es la esperada.

## Situación inicial

Tanto en el schema de la base como en los procedures intervinientes en la conciliación, se notan muchas inconsistencias y redundancias de operaciones. Esto es comprobado al ejecutar las consultas, las cuales presentan tiempos de ejecución elevados. Estos están disponibles en el anexo.

## Análisis de la Solución

### Problemas Encontrados

#### Los tablespaces no son auto-extensibles

Los tablespaces `TEAM1_DATA` y `TEAM1_INDEXES` no son *autoextensibles*, por lo cual la generación de índices, las inserciones o el cálculo de nuevas conciliaciones acabarían bloqueando los procesos llevados a cabo por la empresa de turismo. Para solventar la falta de espacio físico, se alteraron las propiedades de ambos tablespaces mediante:

```
ALTER DATABASE DATAFILE '$ORACLE_HOME/dbs/team1_data.ora'
    AUTOEXTEND ON;

ALTER DATABASE DATAFILE '$ORACLE_HOME/dbs/team1_indexes.ora'
    AUTOEXTEND ON;
```

#### Supplier.status nunca se usa

En la tabla `Supplier`, existe el campo `status`. Este campo nunca es utilizado en el procedure por lo que en el caso de que exista algún registro que no sea `status='ACTIVE'`, no va a ser tenido en cuenta en el Procedure.

Se decidió no cambiar esta lógica porque el objetivo de esta trabajo es sólo performance.

## Optimizaciones Realizadas

### Mejora 0 - Eliminar `STATEMENT_LOCATOR` de `conciliate_booking`

El procedure `conciliate_booking` recibe como primer parámetro `STATEMENT_LOCATOR` y en la tabla `CONCILIATION` se está guardando el mismo. Este parámetro carece de total sentido debido a que un `HOTEL_STATEMENT` posee ya una *Primary Key* (`HOTEL_STATEMENT.ID`). Además esta columna no posee índice (como sí posee la PK) y el tipo de la columna `CONCILIATION.STATEMENT_LOCATOR` es inconsistente con el de la columna `HOTEL_STATEMENT.STATEMENT_LOCATOR`.

Por todas estas razones se removió la columna `STATEMENT_LOCATOR` de `CONCILIATION` y se quitó el primer parámetro de `conciliate_booking` acordemente. (El segundo parámetro es la PK

mencionada `pHsId`).

No se removió en el contexto de este cambio la columna `STATEMENT_LOCATOR` de `HOTEL_STATEMENT` porque luego en un cambio más grande esta columna sera removida por completo.

### Mejora 1 - Columna `RECORD_LOCATOR` en `PAYMENT_ORDER` y `HOTEL_STATEMENT`

Una de las primeras mejoras analizadas fue la posibilidad de modificar el uso de `RECORD_LOCATOR` por directamente una referencia a la *Primary Key* como se muestra a continuación:

```
ALTER TABLE PAYMENT_ORDER
  ADD HOTEL_STATEMENT_ID NUMBER(10,0) REFERENCES HOTEL_STATEMENT(ID);

MERGE INTO PAYMENT_ORDER po
  USING HOTEL_STATEMENT hs
  ON (lower(po.RECORD_LOCATOR) = lower(hs.RECORD_LOCATOR))
  WHEN MATCHED THEN UPDATE SET po.HOTEL_STATEMENT_ID = hs.ID;
```

Dado que una `PAYMENT_ORDER` se genera a partir de un `HOTEL_STATEMENT` en principio parece una buena solución.

Al analizar el resultado de esta query notamos que No existe una correlación 1 a 1 entre estas dos tablas. Existen códigos en `RECORD_LOCATOR` que existen en `PAYMENT_ORDER` y no en `HOTEL_STATEMENT` y viceversa. Esto sugiere que el concepto de `RECORD_LOCATOR` es de orden superior al de estas dos tablas, por ejemplo puede ser que una `PAYMENT_ORDER` se pueda relacionar con otros tipos de pagos que no sean `HOTEL_STATEMENT` en otros ámbitos y es por esto que decidimos mantener dicha columna.

Mantener esta columna de "incierto origen", requiere que se tenga mas cuidado con el cambio realizado, es por esto que se decidió agregar el siguiente código a nuestro update:

```
ALTER TABLE HOTEL_STATEMENT
  ADD CONSTRAINT HS_RECORD_LOCATOR_UPPER
    CHECK (upper(RECORD_LOCATOR) = RECORD_LOCATOR);
CREATE UNIQUE INDEX HOTEL_STATEMENT_RECORD_LOCATOR
  ON HOTEL_STATEMENT(RECORD_LOCATOR);

ALTER TABLE PAYMENT_ORDER
  ADD CONSTRAINT PO_RECORD_LOCATOR_UPPER
    CHECK (upper(RECORD_LOCATOR) = RECORD_LOCATOR);
CREATE UNIQUE INDEX PAYMENT_ORDER_RECORD_LOCATOR
  ON PAYMENT_ORDER(RECORD_LOCATOR);
```

De esta manera se mueve el chequeo de las mayúsculas a la hora del insert (en lugar de nuestro script de conciliación), podemos confiar en los datos y dejar de hacer consultas que no entran por índice. A su vez se agregó el índice `UNIQUE` para garantizar que sea unívoco el acceso al mismo.

Éste último cambio nos lleva a pensar una limitación que tiene el sistema actual, el `RECORD_LOCATOR` es de tipo `CHAR(6 BYTE)`, la cardinalidad de este tipo es mucho menor a la de nuestras PKs de `NUMBER(10, 0)`, por lo que en algún momento el sistema se va a quedar sin `RECORD_LOCATOR` que generar. Se pensó en cambiar la columna de `RECORD_LOCATOR` por otra cosa, pero debido a que el origen de este dato es incierto se decidió dejarlo como está y hacer esta mención en el informe.

## Mejora 2 - Eliminación completa del paso intermedio `conciliate_statement`

El script `conciliate_all_statements` estaba iterando por las rows de `hotel_statement` en `STATUS='PENDING'` y luego en el `LOOP` invocando a `conciliate_statement` con un único parámetro `STATEMENT_LOCATOR`. Ya hemos mencionado la innecesidad de utilizar la columna `STATEMENT_LOCATOR`, la ausencia de índice en ella. La utilización de `STATEMENT_LOCATOR` hace que tengamos que realizar una query *extra* para hallar los `HOTEL_STATEMENT` *candidatos* porque previamente se tuvo que desambiguar con `DISTINCT` debido a que el `STATEMENT_LOCATOR` no representa unívocamente una row.

Por último esta procedure hace una búsqueda del `SUPPLIER` para obtener los valores `vTolPercentage` y `vTolMax` para finalmente invocar a `conciliate_booking`.

Debido a que todo este paso intermedio es innecesario podemos remover la clausula `conciliate_statement` por completo como así también la columna `STATEMENT_LOCATOR` de `HOTEL_STATEMENT` y realizar directamente en `conciliate_all_statements` un `LOOP` conteniendo toda la información necesaria y llamar directamente a `conciliate_booking` desde allí:

```
-- Conciliacion de todos los extractos pendientes
PROCEDURE conciliate_all_statements AS
BEGIN
    -- Recorro los extractos pendientes
    FOR R IN (
        SELECT
            hs.ID, hs.SUPPLIER_ID, hs.RECORD_LOCATOR, hs.AMOUNT, hs.CURRENCY,
            s.CONCILIATION_TOLERANCE_PERC, s.CONCILIATION_TOLERANCE_MAX
        FROM hotel_statement hs
        JOIN supplier s ON s.ID = hs.SUPPLIER_ID
        WHERE LTRIM(RTRIM(hs.STATUS)) = 'PENDING'
    ) LOOP
        -- Concilio una reserva
        dbms_output.put_line(' Conciliating booking '||R.RECORD_LOCATOR);
        conciliate_booking(
            R.ID,R.SUPPLIER_ID,R.RECORD_LOCATOR,R.AMOUNT,R.CURRENCY,
            R.CONCILIATION_TOLERANCE_PERC, R.CONCILIATION_TOLERANCE_MAX
        );
        -- El extracto debe procesarse completo
        COMMIT;
    END LOOP;
END conciliate_all_statements;
```

## Mejora 3 - Conciliation STATUS

En la conciliación se utiliza el concepto de `STATUS`. Las tablas `CONCILIATION`, `HOTEL_STATEMENT` y

`PAYMENT_ORDER` todas poseen una columna `STATUS`, pero las mismas no tienen índice y tienen tipos diferentes. Además en el paquete se implementan condiciones que por más de que haya un índice, no sería utilizado: `rtrim(ltrim(po.status)) = 'PENDING'`. Por último, nos llamó la atención que si bien las tablas `HOTEL_STATEMENT` y `PAYMENT_ORDER` tienen la columna `STATUS`, no parece tener sentido que esto sea así, dado que es un dato propio de la conciliación, evidencia de esto es que se replica el mismo valor de `STATUS` en todas las tablas correspondientes (dependiendo del resultado).

Debido a que la cantidad de `STATUS` valores esperados es limitada, se decidió utilizar `NUMBER(1, 0)` para el tipo de dato de la `PK` y se agregó un `BITMAP INDEX` en la tabla `CONCILIATION` para su utilización. Esto mejora tanto en `size` como en `performance` el chequeo de `STATUS`.

Por todo lo mencionado se decidió extraer el concepto de `STATUS` a una nueva tabla `CONCILIATION_STATUS` y reemplazar el `CHAR` type por una `PK` a la misma; de esta forma nos aseguramos que existe el índice, los tipos coinciden y no hace falta realizar transformaciones como el `rtrim` y `ltrim`. Se creó una *vista* `CONCILIATION_WS` para que el usuario final pueda tener una experiencia similar a la anterior mostrando el `STATUS.NAME` en lugar de `STATUS_ID`.

También se quitaron las columnas `STATUS` de `HOTEL_STATEMENT` y `PAYMENT_ORDER`, pero en estas dos no fue reemplazado por el `STATUS_ID` porque nos pareció irrelevante la información para estas tablas. No obstante se crearon las *vistas* `HOTEL_STATEMENT_WS` y `PAYMENT_ORDER_WS` que realizan el correspondiente `JOIN` a `CONCILIATION` para mostrar el estado correspondiente (si en la tabla `CONCILIATION` no hay un registro relacionado se muestra 'PENDING').

Por último se destaca el cambio de condiciones de `WHERE ltrim(rtrim(...))` por `ANTIJOINS` en los siguientes casos: `conciliate_all_statements`:

```
...
-         WHERE LTRIM(RTRIM(hs.STATUS)) = 'PENDING'
+         WHERE hs.id NOT IN (
+             SELECT hotel_statement_id
+             FROM conciliation
+         )
...
```

`conciliate_booking`:

```
...
-         and rtrim(ltrim(po.status)) = 'PENDING';
+         and po.id NOT IN (
+             SELECT payment_order_id
+             FROM conciliation
+             WHERE payment_order_id IS NOT NULL
+         );
...
```

## Mejora 4 - Una sola query (resolviendo el problema N+1)

En esta etapa nos queda un script que hace dos operaciones: 1. Busca `HOTEL_STATEMENTS` en estado `PENDING` para analizar. 2. Por cada resultado en `HOTEL_STATEMENT` llama a `conciliate_booking` 1. Busca las `PAYMENT_ORDERS` relacionadas a través de `RECORD_LOCATOR`. 2. Realiza una serie de chequeos y graba en `CONCILIATION` el record con el resultado en base a diferentes escenarios posibles.

Este es un claro ejemplo del antipatrón N+1, primero hacemos una query de búsqueda (1.) y luego por cada resultado realizamos otra query para buscar mas información relacionada (2.2). Esto afecta a la performance porque tener un quiebre desde la lógica del script no permite al motor de base de datos realizar las optimizaciones pertinentes a la hora de buscar los datos todos juntos.

Para poder ir a buscar `PAYMENT_ORDERS` en la misma query original es importante destacar que tiene que hacerse a través de un `LEFT JOIN` porque nos interesa atrapar el caso de `NOT_FOUND` que se venia comprobando como una `EXCEPTION WHEN NO_DATA_FOUND`. En su lugar revisaremos cuando llegue un `NULL` en `vPoId` significa que no encontró la orden que queríamos.

Con este cambio nos quedan dos procedures en nuestra implementación final: 1.

`CONCILIATE_PKG.conciliate_all_statements`. Es público (definido en el package) y realiza la query que levanta toda la información necesaria para la conciliación. 2.

`CONCILIATE_PKG.conciliate_booking`. Es privada, recibe todos los parámetros necesarios para realizar una conciliación y contiene la lógica de negocio para definir si la conciliación es correcta guardando el resultado en la tabla `CONCILIATION`.

## Mejora 5 - Final Stage

Finalmente todo lo que queda es preparar el sistema para el despliegue final, lo que \* implica reconstruir los índices con el estado actual, y computar las \* estadísticas completas sobre todas las tablas para mejorar el CBO.

```
@4_n+1AntipatternFix.sql
```

```
/* Index Rebuild */
ALTER INDEX HOTEL_STATEMENT_PK REBUILD;
ALTER INDEX HOTEL_STATEMENT_RECORD_LOCATOR REBUILD;
ALTER INDEX PAYMENT_ORDER_PK REBUILD;
ALTER INDEX PAYMENT_ORDER_RECORD_LOCATOR REBUILD;
ALTER INDEX PO2_IDX REBUILD;
ALTER INDEX STATUS_ID_IDX REBUILD;
ALTER INDEX SUPPLIER_PK REBUILD;

/* Table Statistics */
EXEC DBMS_STATS.gather_table_stats('BDII_TEAM1', 'CONCILIATION_STATUS', cascade
=> TRUE);
EXEC DBMS_STATS.gather_table_stats('BDII_TEAM1', 'HOTEL_STATEMENT', cascade =>
TRUE);
EXEC DBMS_STATS.gather_table_stats('BDII_TEAM1', 'PAYMENT_ORDER', cascade =>
TRUE);
```

```
EXEC DBMS_STATS.gather_table_stats('BDII_TEAM1', 'SUPPLIER', cascade => TRUE);

/* Index Statistics */
EXEC DBMS_STATS.gather_index_stats('BDII_TEAM1', 'HOTEL_STATEMENT_PK');
EXEC DBMS_STATS.gather_index_stats('BDII_TEAM1',
'HOTEL_STATEMENT_RECORD_LOCATOR');
EXEC DBMS_STATS.gather_index_stats('BDII_TEAM1', 'PAYMENT_ORDER_PK');
EXEC DBMS_STATS.gather_index_stats('BDII_TEAM1',
'PAYMENT_ORDER_RECORD_LOCATOR');
EXEC DBMS_STATS.gather_index_stats('BDII_TEAM1', 'PO2_IDX');
EXEC DBMS_STATS.gather_index_stats('BDII_TEAM1', 'STATUS_ID_IDX');
EXEC DBMS_STATS.gather_index_stats('BDII_TEAM1', 'SUPPLIER_PK');
```

## Resultados

	Run 1	Run 2	Run 3	Run 4	Average	Variación	%
BASE TIME:	2.272	2.546	2.246	2.062	2.282	-	
UPDATE 0:	2.644	1.422	1.547	1.297	1.728	0.554	24.28
UPDATE 1:	1.579	1.368	1.351	1.313	1.403	0.325	18.81
UPDATE 2:	1.796	1.843	1.61	1.547	1.699	-0.296	-21.10
UPDATE 3:	1.594	1.062	1.407	1.249	1.328	0.371	21.84
UPDATE 4:	0.734	1.438	1.14	1.375	1.172	0.156	11.75
UPDATE 5:	0.844	0.796	0.863	0.703	0.802	0.37	31.57

## Plan de Ejecución

## Recomendaciones

## Anexo

## Mejora 0

```
/**
 * Elimina la columna STATEMENT_LOCATOR de la tabla CONCILIATION, y evita
 * actualizar dicha propiedad durante la conciliación.
 */

ALTER TABLE CONCILIATION DROP COLUMN STATEMENT_LOCATOR;

create or replace PACKAGE BODY CONCILIATE_PKG AS

    -- Conciliacion de una reserva
    PROCEDURE conciliate_booking ( pHsId NUMBER, pSupplier NUMBER, pRecordLocator
    VARCHAR,
                                pAmount NUMBER, pCurrency VARCHAR, vTolPercentage NUMBER,
    vTolMax NUMBER ) AS
        vPoId NUMBER(10);
        vAmount NUMBER(10,2);
        vCurrency CHAR(3);
        vStatus CHAR(20);
```

```

vCheckinDate DATE;
vCheckoutDate DATE;
BEGIN

    -- Buscar la PO asociada
    select /*+MERGE*/ po.ID, po.TOTAL_COST, po.TOTAL_COST_CURRENCY, po.STATUS,
po.CHECKIN, po.CHECKOUT
    into vPoId, vAmount, vCurrency, vStatus, vCheckinDate, vCheckoutDate
    from PAYMENT_ORDER po, SUPPLIER s
    where po.supplier_id = pSupplier
    and po.supplier_id = s.id
    and lower(po.record_locator) = lower(pRecordLocator)
    and rtrim(ltrim(po.status)) = 'PENDING';

    -- Si no paso la fecha de checkout no se puede pagar aun
    IF vCheckOutDate>SYSDATE THEN
        -- Registrar que la reserva aun no puede conciliarse por estar
pendiente su fecha de checkout
        dbms_output.put_line('      Checkout Pending');
        INSERT INTO CONCILIATION (
            ID, HOTEL_STATEMENT_ID, PAYMENT_ORDER_ID,
            CONCILIATED_AMOUNT, CONCILIATED_AMOUNT_CURRENCY,
            ADJUSTMENT_AMOUNT, ADJUSTMENT_AMOUNT_CURRENCY,
            STATUS, CREATED, MODIFIED)
        VALUES (CONCILIATION_SEQ.nextval, pHsId, null,
            null, null, null, null,
            'CHECKOUT_PENDING',sysdate,sysdate);
        UPDATE HOTEL_STATEMENT SET STATUS = 'CHECKOUT_PENDING', MODIFIED =
SYSDATE
        WHERE ID = pHsId;
    -- Si la moneda de conciliacion y la del hotelero no coinciden
    ELSIF vCurrency NOT LIKE pCurrency THEN
        -- Registrar que la moneda indicada en el extracto no es la correcta
        dbms_output.put_line('      Wrong Currency');
        INSERT INTO CONCILIATION (
            ID, HOTEL_STATEMENT_ID, PAYMENT_ORDER_ID,
            CONCILIATED_AMOUNT, CONCILIATED_AMOUNT_CURRENCY,
            ADJUSTMENT_AMOUNT, ADJUSTMENT_AMOUNT_CURRENCY,
            STATUS, CREATED, MODIFIED)
        VALUES (CONCILIATION_SEQ.nextval, pHsId, null,
            null, null, null, null,
            'WRONG_CURRENCY',sysdate,sysdate);
        UPDATE HOTEL_STATEMENT SET STATUS = 'WRONG_CURRENCY', MODIFIED =
SYSDATE
        WHERE ID = pHsId;
    -- Si el monto solicitado por el hotelero esta dentro de los limites de
tolerancia
    ELSIF ( ((vAmount-pAmount)<((vTolPercentage/100)*pAmount)) AND
((vAmount-pAmount)<vTolMax) ) THEN
        -- Registrar que se aprueba la conciliacion de la reserva
        dbms_output.put_line('      Conciliated');
        INSERT INTO CONCILIATION (
            ID, HOTEL_STATEMENT_ID, PAYMENT_ORDER_ID,
            CONCILIATED_AMOUNT, CONCILIATED_AMOUNT_CURRENCY,
            ADJUSTMENT_AMOUNT, ADJUSTMENT_AMOUNT_CURRENCY,
            STATUS, CREATED, MODIFIED)
        VALUES (CONCILIATION_SEQ.nextval, pHsId, vPoId,

```



```

        pAmount, pCurrency, round(vAmount-pAmount,2), pCurrency,
        'CONCILIATED',sysdate,sysdate);
    UPDATE HOTEL_STATEMENT SET STATUS = 'CONCILIATED', MODIFIED = SYSDATE
    WHERE ID = pHsId;
    UPDATE PAYMENT_ORDER SET STATUS = 'CONCILIATED', MODIFIED = SYSDATE
    WHERE ID = vPoId;
    -- Si el monto solicitado por el hotelero no esta dentro de los limites de
tolerancia
    ELSE
        -- Registrar que la reserva no puede conciliarse por diferencia de
monto
        dbms_output.put_line('    Error Tolerance');
        INSERT INTO CONCILIATION (
            ID, HOTEL_STATEMENT_ID, PAYMENT_ORDER_ID,
            CONCILIATED_AMOUNT, CONCILIATED_AMOUNT_CURRENCY,
            ADJUSTMENT_AMOUNT, ADJUSTMENT_AMOUNT_CURRENCY,
            STATUS, CREATED, MODIFIED)
        VALUES (CONCILIATION_SEQ.nextval, pHsId, vPoId,
            pAmount, pCurrency, null, null,
            'ERROR_TOLERANCE',sysdate,sysdate);
        UPDATE HOTEL_STATEMENT SET STATUS = 'ERROR_TOLERANCE', MODIFIED =
SYSDATE
        WHERE ID = pHsId;
    END IF;

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        -- Registrar que no se encontro una reserva de las caracteriticas que
el hotelero indico
        dbms_output.put_line('    Not Found');
        INSERT INTO CONCILIATION (
            ID, HOTEL_STATEMENT_ID, PAYMENT_ORDER_ID,
            CONCILIATED_AMOUNT, CONCILIATED_AMOUNT_CURRENCY,
            ADJUSTMENT_AMOUNT, ADJUSTMENT_AMOUNT_CURRENCY,
            STATUS, CREATED, MODIFIED)
        VALUES (CONCILIATION_SEQ.nextval, pHsId, null,
            null, null, null, null,
            'NOT_FOUND',sysdate,sysdate);
        UPDATE HOTEL_STATEMENT SET STATUS = 'NOT_FOUND'
        WHERE ID = pHsId;
    END conciliate_booking;

-- Conciliacion de un extracto
PROCEDURE conciliate_statement ( pStatementLocator VARCHAR ) AS
    vTolPercentage NUMBER(10,2);
    vTolMax NUMBER(10,2);
BEGIN

    dbms_output.put_line('Conciliating statement '||pStatementLocator);

    -- Recorro las reservas reclamadas por el hotelero en su extracto
    FOR R IN (
        SELECT hs.ID, hs.SUPPLIER_ID, hs.RECORD_LOCATOR, hs.AMOUNT, hs.CURRENCY
        FROM hotel_statement hs
        WHERE hs.statement_locator = pStatementLocator
        AND LTRIM(RTRIM(hs.STATUS)) = 'PENDING'
    )

```

```

) LOOP
    -- Recupero los parametros de tolerancia del proveedor
    dbms_output.put_line(' Retrieving supplier '||R.SUPPLIER_ID);
    SELECT s.CONCILIATION_TOLERANCE_PERC, s.CONCILIATION_TOLERANCE_MAX
    INTO vTolPercentage, vTolMax
    FROM supplier s
    WHERE s.ID = R.SUPPLIER_ID;

    -- Concilio una reserva
    dbms_output.put_line(' Conciliating booking '||R.RECORD_LOCATOR);

    conciliate_booking(R.ID,R.SUPPLIER_ID,R.RECORD_LOCATOR,R.AMOUNT,R.CURRENCY,vTolPercentage,vTolMax);
    END LOOP;

    -- El extracto debe procesarse completo
    COMMIT;

    END conciliate_statement;

    -- Conciliacion de todos los extractos pendientes
    PROCEDURE conciliate_all_statements AS
    BEGIN

    -- Recorro los extractos pendientes
    FOR R IN (
        SELECT distinct hs.statement_locator
        FROM hotel_statement hs
        WHERE LTRIM(RTRIM(hs.STATUS)) = 'PENDING'
    ) LOOP
        -- Concilio el extracto actual
        conciliate_statement(R.STATEMENT_LOCATOR);
    END LOOP;

    END conciliate_all_statements;

END CONCILIATE_PKG;

```

## Mejora 1

```

/**
 * Aplica el UPDATE-0, y además optimiza el uso del RECORD_LOCATOR, forzando
 * el uso de mayúsculas en las inserciones, y agregando índices UNIQUE sobre
 * dichas columnas para HOTEL_STATEMENT y PAYMENT_ORDER.
 *
 * No es necesario refactorizar las inserciones porque todos los
 * RECORD_LOCATOR ya se encuentran en mayúsculas.
 */

@0_removeStatementLocator.sql

ALTER TABLE HOTEL_STATEMENT
    ADD CONSTRAINT HS_RECORD_LOCATOR_UPPER
        CHECK (upper(RECORD_LOCATOR) = RECORD_LOCATOR);

```

```

CREATE UNIQUE INDEX HOTEL_STATEMENT_RECORD_LOCATOR
  ON HOTEL_STATEMENT(RECORD_LOCATOR)
  TABLESPACE TEAM1_INDEXES;

ALTER TABLE PAYMENT_ORDER
  ADD CONSTRAINT PO_RECORD_LOCATOR_UPPER
    CHECK (upper(RECORD_LOCATOR) = RECORD_LOCATOR);

CREATE UNIQUE INDEX PAYMENT_ORDER_RECORD_LOCATOR
  ON PAYMENT_ORDER(RECORD_LOCATOR)
  TABLESPACE TEAM1_INDEXES;

create or replace PACKAGE BODY CONCILIATE_PKG AS

  -- Conciliacion de una reserva
  PROCEDURE conciliate_booking ( pHsId NUMBER, pSupplier NUMBER, pRecordLocator
  VARCHAR,
                                pAmount NUMBER, pCurrency VARCHAR, vTolPercentage NUMBER,
  vTolMax NUMBER ) AS
    vPoId NUMBER(10);
    vAmount NUMBER(10,2);
    vCurrency CHAR(3);
    vStatus CHAR(20);
    vCheckinDate DATE;
    vCheckoutDate DATE;
  BEGIN

    -- Buscar la PO asociada
    select /*+MERGE*/ po.ID, po.TOTAL_COST, po.TOTAL_COST_CURRENCY, po.STATUS,
  po.CHECKIN, po.CHECKOUT
    into vPoId, vAmount, vCurrency, vStatus, vCheckinDate, vCheckoutDate
    from PAYMENT_ORDER po, SUPPLIER s
    where po.supplier_id = pSupplier
    and po.supplier_id = s.id
    and po.record_locator = pRecordLocator
    and rtrim(ltrim(po.status)) = 'PENDING';

    -- Si no paso la fecha de checkout no se puede pagar aun
    IF vCheckOutDate>SYSDATE THEN
      -- Registrar que la reserva aun no puede conciliarse por estar
  pendiente su fecha de checkout
      dbms_output.put_line('      Checkout Pending');
      INSERT INTO CONCILIATION (
        ID, HOTEL_STATEMENT_ID, PAYMENT_ORDER_ID,
        CONCILIATED_AMOUNT, CONCILIATED_AMOUNT_CURRENCY,
        ADJUSTMENT_AMOUNT, ADJUSTMENT_AMOUNT_CURRENCY,
        STATUS, CREATED, MODIFIED)
      VALUES (CONCILIATION_SEQ.nextval, pHsId, null,
        null, null, null, null,
        'CHECKOUT_PENDING',sysdate,sysdate);
      UPDATE HOTEL_STATEMENT SET STATUS = 'CHECKOUT_PENDING', MODIFIED =
  SYSDATE

      WHERE ID = pHsId;

    -- Si la moneda de conciliacion y la del hotelero no coinciden
    ELSIF vCurrency NOT LIKE pCurrency THEN
      -- Registrar que la moneda indicada en el extracto no es la correcta
      dbms_output.put_line('      Wrong Currency');

```

```

        INSERT INTO CONCILIATION (
            ID, HOTEL_STATEMENT_ID, PAYMENT_ORDER_ID,
            CONCILIATED_AMOUNT, CONCILIATED_AMOUNT_CURRENCY,
            ADJUSTMENT_AMOUNT, ADJUSTMENT_AMOUNT_CURRENCY,
            STATUS, CREATED, MODIFIED)
        VALUES (CONCILIATION_SEQ.nextval, pHsId, null,
            null, null, null, null,
            'WRONG_CURRENCY',sysdate,sysdate);
        UPDATE HOTEL_STATEMENT SET STATUS = 'WRONG_CURRENCY', MODIFIED =
SYSDATE
        WHERE ID = pHsId;
        -- Si el monto solicitado por el hotelero esta dentro de los limites de
tolerancia
        ELSIF ( ((vAmount-pAmount)<((vTolPercentage/100)*pAmount)) AND
((vAmount-pAmount)<vTolMax) ) THEN
            -- Registrar que se aprueba la conciliacion de la reserva
            dbms_output.put_line('    Conciliated');
            INSERT INTO CONCILIATION (
                ID, HOTEL_STATEMENT_ID, PAYMENT_ORDER_ID,
                CONCILIATED_AMOUNT, CONCILIATED_AMOUNT_CURRENCY,
                ADJUSTMENT_AMOUNT, ADJUSTMENT_AMOUNT_CURRENCY,
                STATUS, CREATED, MODIFIED)
            VALUES (CONCILIATION_SEQ.nextval, pHsId, vPoId,
                pAmount, pCurrency, round(vAmount-pAmount,2), pCurrency,
                'CONCILIATED',sysdate,sysdate);
            UPDATE HOTEL_STATEMENT SET STATUS = 'CONCILIATED', MODIFIED = SYSDATE
            WHERE ID = pHsId;
            UPDATE PAYMENT_ORDER SET STATUS = 'CONCILIATED', MODIFIED = SYSDATE
            WHERE ID = vPoId;
            -- Si el monto solicitado por el hotelero no esta dentro de los limites
de tolerancia
            ELSE
                -- Registrar que la reserva no puede conciliarse por diferencia de
monto
                dbms_output.put_line('    Error Tolerance');
                INSERT INTO CONCILIATION (
                    ID, HOTEL_STATEMENT_ID, PAYMENT_ORDER_ID,
                    CONCILIATED_AMOUNT, CONCILIATED_AMOUNT_CURRENCY,
                    ADJUSTMENT_AMOUNT, ADJUSTMENT_AMOUNT_CURRENCY,
                    STATUS, CREATED, MODIFIED)
                VALUES (CONCILIATION_SEQ.nextval, pHsId, vPoId,
                    pAmount, pCurrency, null, null,
                    'ERROR_TOLERANCE',sysdate,sysdate);
                UPDATE HOTEL_STATEMENT SET STATUS = 'ERROR_TOLERANCE', MODIFIED =
SYSDATE
                WHERE ID = pHsId;
            END IF;

        EXCEPTION
            WHEN NO_DATA_FOUND THEN
                -- Registrar que no se encontro una reserva de las caracteriticas que
el hotelero indico
                dbms_output.put_line('    Not Found');
                INSERT INTO CONCILIATION (
                    ID, HOTEL_STATEMENT_ID, PAYMENT_ORDER_ID,
                    CONCILIATED_AMOUNT, CONCILIATED_AMOUNT_CURRENCY,
                    ADJUSTMENT_AMOUNT, ADJUSTMENT_AMOUNT_CURRENCY,

```

```

        STATUS, CREATED, MODIFIED)
VALUES (CONCILIATION_SEQ.nextval, pHsId, null,
        null, null, null, null,
        'NOT_FOUND',sysdate,sysdate);
UPDATE HOTEL_STATEMENT SET STATUS = 'NOT_FOUND'
WHERE ID = pHsId;
END conciliate_booking;

-- Conciliacion de un extracto
PROCEDURE conciliate_statement ( pStatementLocator VARCHAR ) AS
    vTolPercentage NUMBER(10,2);
    vTolMax NUMBER(10,2);
BEGIN

    dbms_output.put_line('Conciliating statement '||pStatementLocator);

    -- Recorro las reservas reclamadas por el hotelero en su extracto
    FOR R IN (
        SELECT hs.ID, hs.SUPPLIER_ID, hs.RECORD_LOCATOR, hs.AMOUNT, hs.CURRENCY
        FROM hotel_statement hs
        WHERE hs.statement_locator = pStatementLocator
        AND LTRIM(RTRIM(hs.STATUS)) = 'PENDING'
    ) LOOP
        -- Recupero los parametros de tolerancia del proveedor
        dbms_output.put_line(' Retrieving supplier '||R.SUPPLIER_ID);
        SELECT s.CONCILIATION_TOLERANCE_PERC, s.CONCILIATION_TOLERANCE_MAX
        INTO vTolPercentage, vTolMax
        FROM supplier s
        WHERE s.ID = R.SUPPLIER_ID;

        -- Concilio una reserva
        dbms_output.put_line(' Conciliating booking '||R.RECORD_LOCATOR);

        conciliate_booking(R.ID,R.SUPPLIER_ID,R.RECORD_LOCATOR,R.AMOUNT,R.CURRENCY,vTolPercentage,vTolMax);
    END LOOP;

    -- El extracto debe procesarse completo
    COMMIT;

END conciliate_statement;

-- Conciliacion de todos los extractos pendientes
PROCEDURE conciliate_all_statements AS
BEGIN

    -- Recorro los extractos pendientes
    FOR R IN (
        SELECT distinct hs.statement_locator
        FROM hotel_statement hs
        WHERE LTRIM(RTRIM(hs.STATUS)) = 'PENDING'
    ) LOOP
        -- Concilio el extracto actual
        conciliate_statement(R.STATEMENT_LOCATOR);
    END LOOP;

```

```

        END conciliate_all_statements;

END CONCILIATE_PKG;

```

## Mejora 2

```

/**
 * Aplica el UPDATE-1 y, además, remueve por completo el PSM intermedio
 * CONCILIATE_STATEMENT ya que es innecesario. Como efecto colateral, ya no
 * es requerido mantener el STATEMENT_LOCATOR en la tabla HOTEL_STATEMENT.
 */

@1_improveRecordLocator.sql

ALTER TABLE HOTEL_STATEMENT DROP COLUMN STATEMENT_LOCATOR;

CREATE OR REPLACE PACKAGE CONCILIATE_PKG AS
    PROCEDURE conciliate_all_statements;
END CONCILIATE_PKG;

/

CREATE OR REPLACE PACKAGE BODY CONCILIATE_PKG AS

    -- Conciliacion de una reserva
    PROCEDURE conciliate_booking ( pHsId NUMBER, pSupplier NUMBER, pRecordLocator
    VARCHAR,
                                pAmount NUMBER, pCurrency VARCHAR, vTolPercentage NUMBER,
    vTolMax NUMBER ) AS
        vPoId NUMBER(10);
        vAmount NUMBER(10,2);
        vCurrency CHAR(3);
        vStatus CHAR(20);
        vCheckinDate DATE;
        vCheckoutDate DATE;
    BEGIN

        -- Buscar la PO asociada
        select /*+MERGE*/ po.ID, po.TOTAL_COST, po.TOTAL_COST_CURRENCY, po.STATUS,
    po.CHECKIN, po.CHECKOUT
        into vPoId, vAmount, vCurrency, vStatus, vCheckinDate, vCheckoutDate
        from PAYMENT_ORDER po, SUPPLIER s
        where po.supplier_id = pSupplier
        and po.supplier_id = s.id
        and po.record_locator = pRecordLocator
        and rtrim(ltrim(po.status)) = 'PENDING';

        -- Si no paso la fecha de checkout no se puede pagar aun
        IF vCheckOutDate>SYSDATE THEN
            -- Registrar que la reserva aun no puede conciliarse por estar
    pendiente su fecha de checkout
            dbms_output.put_line('      Checkout Pending');
            INSERT INTO CONCILIATION (
                ID, HOTEL_STATEMENT_ID, PAYMENT_ORDER_ID,

```

```

        CONCILIATED_AMOUNT, CONCILIATED_AMOUNT_CURRENCY,
        ADJUSTMENT_AMOUNT, ADJUSTMENT_AMOUNT_CURRENCY,
        STATUS, CREATED, MODIFIED)
VALUES (CONCILIATION_SEQ.nextval, pHsId, null,
        null, null, null, null,
        'CHECKOUT_PENDING',sysdate,sysdate);
UPDATE HOTEL_STATEMENT SET STATUS = 'CHECKOUT_PENDING', MODIFIED =
SYSDATE

WHERE ID = pHsId;
-- Si la moneda de conciliacion y la del hotelero no coinciden
ELSIF vCurrency NOT LIKE pCurrency THEN
    -- Registrar que la moneda indicada en el extracto no es la correcta
    dbms_output.put_line('    Wrong Currency');
    INSERT INTO CONCILIATION (
        ID, HOTEL_STATEMENT_ID, PAYMENT_ORDER_ID,
        CONCILIATED_AMOUNT, CONCILIATED_AMOUNT_CURRENCY,
        ADJUSTMENT_AMOUNT, ADJUSTMENT_AMOUNT_CURRENCY,
        STATUS, CREATED, MODIFIED)
    VALUES (CONCILIATION_SEQ.nextval, pHsId, null,
        null, null, null, null,
        'WRONG_CURRENCY',sysdate,sysdate);
    UPDATE HOTEL_STATEMENT SET STATUS = 'WRONG_CURRENCY', MODIFIED =
SYSDATE

WHERE ID = pHsId;
-- Si el monto solicitado por el hotelero esta dentro de los limites de
tolerancia
    ELSIF ( ((vAmount-pAmount)<((vTolPercentage/100)*pAmount)) AND
((vAmount-pAmount)<vTolMax) ) THEN
        -- Registrar que se aprueba la conciliacion de la reserva
        dbms_output.put_line('    Conciliated');
        INSERT INTO CONCILIATION (
            ID, HOTEL_STATEMENT_ID, PAYMENT_ORDER_ID,
            CONCILIATED_AMOUNT, CONCILIATED_AMOUNT_CURRENCY,
            ADJUSTMENT_AMOUNT, ADJUSTMENT_AMOUNT_CURRENCY,
            STATUS, CREATED, MODIFIED)
        VALUES (CONCILIATION_SEQ.nextval, pHsId, vPoId,
            pAmount, pCurrency, round(vAmount-pAmount,2), pCurrency,
            'CONCILIATED',sysdate,sysdate);
        UPDATE HOTEL_STATEMENT SET STATUS = 'CONCILIATED', MODIFIED = SYSDATE
        WHERE ID = pHsId;
        UPDATE PAYMENT_ORDER SET STATUS = 'CONCILIATED', MODIFIED = SYSDATE
        WHERE ID = vPoId;
        -- Si el monto solicitado por el hotelero no esta dentro de los limites
de tolerancia
    ELSE
        -- Registrar que la reserva no puede conciliarse por diferencia de
monto
        dbms_output.put_line('    Error Tolerance');
        INSERT INTO CONCILIATION (
            ID, HOTEL_STATEMENT_ID, PAYMENT_ORDER_ID,
            CONCILIATED_AMOUNT, CONCILIATED_AMOUNT_CURRENCY,
            ADJUSTMENT_AMOUNT, ADJUSTMENT_AMOUNT_CURRENCY,
            STATUS, CREATED, MODIFIED)
        VALUES (CONCILIATION_SEQ.nextval, pHsId, vPoId,
            pAmount, pCurrency, null, null,
            'ERROR_TOLERANCE',sysdate,sysdate);
        UPDATE HOTEL_STATEMENT SET STATUS = 'ERROR_TOLERANCE', MODIFIED =

```

```

SYSDATE
        WHERE ID = pHsId;
    END IF;

    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            -- Registrar que no se encontro una reserva de las caracteriticas que
            el hotelero indico
            dbms_output.put_line('    Not Found');
            INSERT INTO CONCILIATION (
                ID, HOTEL_STATEMENT_ID, PAYMENT_ORDER_ID,
                CONCILIATED_AMOUNT, CONCILIATED_AMOUNT_CURRENCY,
                ADJUSTMENT_AMOUNT, ADJUSTMENT_AMOUNT_CURRENCY,
                STATUS, CREATED, MODIFIED)
            VALUES (CONCILIATION_SEQ.nextval, pHsId, null,
                null, null, null, null,
                'NOT_FOUND',sysdate,sysdate);
            UPDATE HOTEL_STATEMENT SET STATUS = 'NOT_FOUND'
            WHERE ID = pHsId;
    END conciliate_booking;

    -- Conciliacion de todos los extractos pendientes
    PROCEDURE conciliate_all_statements AS
    BEGIN
        -- Recorro los extractos pendientes
        FOR R IN (
            SELECT
                hs.ID, hs.SUPPLIER_ID, hs.RECORD_LOCATOR, hs.AMOUNT,
            hs.CURRENCY,
                s.CONCILIATION_TOLERANCE_PERC, s.CONCILIATION_TOLERANCE_MAX
            FROM hotel_statement hs
            JOIN supplier s ON s.ID = hs.SUPPLIER_ID
            WHERE LTRIM(RTRIM(hs.STATUS)) = 'PENDING'
        ) LOOP
            -- Concilio una reserva
            dbms_output.put_line('    Conciliating booking '||R.RECORD_LOCATOR);
            conciliate_booking(
                R.ID,R.SUPPLIER_ID,R.RECORD_LOCATOR,R.AMOUNT,R.CURRENCY,
                R.CONCILIATION_TOLERANCE_PERC, R.CONCILIATION_TOLERANCE_MAX
            );
            -- El extracto debe procesarse completo
            COMMIT;
        END LOOP;
    END conciliate_all_statements;

END CONCILIATE_PKG;

```

## Mejora 3

```

/**
 * Aplica el UPDATE-2 y, además, modifica la representación del parámetro
 * STATUS en todas las tablas, utilizando una representación más acorde y
 * eficiente. Adicionalmente, se generan algunas vistas para mantener

```



```

* compatibilidad con la representación antigua.
*
* @issue 1
*   Actualmente, el usuario BDII_TEAM1 no tiene privilegios suficientes
*   para crear vistas.
*/

```

```
@2_dropConciliateStatementPSM.sql
```

```

CREATE TABLE CONCILIATION_STATUS (
    "ID" NUMBER(1,0),
    "NAME" CHAR(20 BYTE),
    PRIMARY KEY ("ID") ENABLE
) TABLESPACE TEAM1_DATA;

```

```

INSERT INTO CONCILIATION_STATUS VALUES (1, 'CHECKOUT_PENDING');
INSERT INTO CONCILIATION_STATUS VALUES (2, 'WRONG_CURRENCY');
INSERT INTO CONCILIATION_STATUS VALUES (3, 'ERROR_TOLERANCE');
INSERT INTO CONCILIATION_STATUS VALUES (4, 'NOT_FOUND');
INSERT INTO CONCILIATION_STATUS VALUES (5, 'CONCILIATED');

```

```
ALTER TABLE CONCILIATION ADD STATUS_ID NUMBER(1,0);
```

```

/* Necesario para poder crear un BITMAP INDEX en una IOT */
ALTER TABLE CONCILIATION MOVE MAPPING TABLE;

```

```

CREATE BITMAP INDEX STATUS_ID_IDX ON CONCILIATION(STATUS_ID)
    TABLESPACE TEAM1_INDEXES;

```

```

ALTER TABLE CONCILIATION ADD CONSTRAINT STATUS_ID_FK
    FOREIGN KEY (STATUS_ID) REFERENCES CONCILIATION_STATUS(ID);

```

```

ALTER TABLE CONCILIATION DROP COLUMN STATUS;
ALTER TABLE HOTEL_STATEMENT DROP COLUMN STATUS;
ALTER TABLE PAYMENT_ORDER DROP COLUMN STATUS;

```

```

/*CREATE OR REPLACE VIEW CONCILIATION_WS AS
SELECT
    c.ID, c.HOTEL_STATEMENT_ID, c.PAYMENT_ORDER_ID,
    c.CONCILIATED_AMOUNT, c.CONCILIATED_AMOUNT_CURRENCY,
    c.ADJUSTMENT_AMOUNT, c.ADJUSTMENT_AMOUNT_CURRENCY,
    cs.NAME STATUS, c.CREATED, c.MODIFIED
FROM CONCILIATION c
JOIN CONCILIATION_STATUS cs ON cs.ID = c.STATUS_ID;

```

```

CREATE OR REPLACE VIEW HOTEL_STATEMENT_WS AS
SELECT
    hs.ID, hs.RECORD_LOCATOR, hs.SUPPLIER_ID, hs.AMOUNT,
    hs.CURRENCY, COALESCE(c.STATUS, 'PENDING') STATUS,
    hs.CREATED, COALESCE(GREATEST(hs.MODIFIED, c.MODIFIED), hs.MODIFIED)
MODIFIED
FROM HOTEL_STATEMENT hs
LEFT JOIN CONCILIATION_WS c ON c.HOTEL_STATEMENT_ID = hs.ID;

```

```

CREATE OR REPLACE VIEW PAYMENT_ORDER_WS AS
SELECT
    po.ID, po.RECORD_LOCATOR, po.SUPPLIER_ID,

```

```

        po.TOTAL_AMOUNT, po.TOTAL_AMOUNT_CURRENCY,
        po.TOTAL_COST, po.TOTAL_COST_CURRENCY,
        COALESCE(c.STATUS, 'PENDING') STATUS, po.CHECKIN, po.CHECKOUT,
        po.CREATED, COALESCE(GREATEST(po.MODIFIED, c.MODIFIED), po.MODIFIED)
MODIFIED
FROM PAYMENT_ORDER po
LEFT JOIN CONCILIATION_WS c ON c.PAYMENT_ORDER_ID = po.ID;*/

CREATE OR REPLACE PACKAGE CONCILIATE_PKG AS
    C_STATUS_CHECKOUT_PENDING CONSTANT SMALLINT := 1;
    C_STATUS_WRONG_CURRENCY CONSTANT SMALLINT := 2;
    C_STATUS_ERROR_TOLERANCE CONSTANT SMALLINT := 3;
    C_STATUS_NOT_FOUND CONSTANT SMALLINT := 4;
    C_STATUS_CONCILIATED CONSTANT SMALLINT := 5;

    PROCEDURE conciliate_all_statements;
END CONCILIATE_PKG;

/

CREATE OR REPLACE PACKAGE BODY CONCILIATE_PKG AS

    -- Conciliacion de una reserva
    PROCEDURE conciliate_booking ( pHSId NUMBER, pSupplier NUMBER, pRecordLocator
VARCHAR,
                                pAmount NUMBER, pCurrency VARCHAR, vTolPercentage NUMBER,
vTolMax NUMBER ) AS
        vPoId NUMBER(10);
        vAmount NUMBER(10,2);
        vCurrency CHAR(3);
        vCheckinDate DATE;
        vCheckoutDate DATE;
    BEGIN
        dbms_output.put_line('Conciliate_booking - pHSId: ' || pHSId || ' pSupplier:
' || pSupplier || ' pRecordLocator: ' || pRecordLocator);

        -- Buscar la PO asociada
        select /*+MERGE*/ po.ID, po.TOTAL_COST, po.TOTAL_COST_CURRENCY,
po.CHECKIN, po.CHECKOUT
        into vPoId, vAmount, vCurrency, vCheckinDate, vCheckoutDate
        from PAYMENT_ORDER po, SUPPLIER s
        where po.supplier_id = pSupplier
        and po.supplier_id = s.id
        and po.record_locator = pRecordLocator
        and po.id NOT IN (
            SELECT payment_order_id
            FROM conciliation
            WHERE payment_order_id IS NOT NULL
        );

        dbms_output.put_line('vPoId: ' || vPoId);

        -- Si no paso la fecha de checkout no se puede pagar aun
        IF vCheckOutDate>SYSDATE THEN
            -- Registrar que la reserva aun no puede conciliarse por estar
pendiente su fecha de checkout
            dbms_output.put_line('      Checkout Pending');
            INSERT INTO CONCILIATION (
                ID, HOTEL_STATEMENT_ID, PAYMENT_ORDER_ID,

```

```

        CONCILIATED_AMOUNT, CONCILIATED_AMOUNT_CURRENCY,
        ADJUSTMENT_AMOUNT, ADJUSTMENT_AMOUNT_CURRENCY,
        STATUS_ID, CREATED, MODIFIED)
VALUES (CONCILIATION_SEQ.nextval, pHsId, null,
        null, null, null, null,
        C_STATUS_CHECKOUT_PENDING,sysdate,sysdate);
-- Si la moneda de conciliacion y la del hotelero no coinciden
ELSIF vCurrency NOT LIKE pCurrency THEN
    -- Registrar que la moneda indicada en el extracto no es la correcta
    dbms_output.put_line('    Wrong Currency');
    INSERT INTO CONCILIATION (
        ID, HOTEL_STATEMENT_ID, PAYMENT_ORDER_ID,
        CONCILIATED_AMOUNT, CONCILIATED_AMOUNT_CURRENCY,
        ADJUSTMENT_AMOUNT, ADJUSTMENT_AMOUNT_CURRENCY,
        STATUS_ID, CREATED, MODIFIED)
VALUES (CONCILIATION_SEQ.nextval, pHsId, null,
        null, null, null, null,
        C_STATUS_WRONG_CURRENCY,sysdate,sysdate);
-- Si el monto solicitado por el hotelero esta dentro de los limites de
tolerancia
    ELSIF ( ((vAmount-pAmount)<((vTolPercentage/100)*pAmount)) AND
((vAmount-pAmount)<vTolMax) ) THEN
        -- Registrar que se aprueba la conciliacion de la reserva
        dbms_output.put_line('    Conciliated');
        INSERT INTO CONCILIATION (
            ID, HOTEL_STATEMENT_ID, PAYMENT_ORDER_ID,
            CONCILIATED_AMOUNT, CONCILIATED_AMOUNT_CURRENCY,
            ADJUSTMENT_AMOUNT, ADJUSTMENT_AMOUNT_CURRENCY,
            STATUS_ID, CREATED, MODIFIED)
VALUES (CONCILIATION_SEQ.nextval, pHsId, vPoId,
        pAmount, pCurrency, round(vAmount-pAmount,2), pCurrency,
        C_STATUS_CONCILIATED,sysdate,sysdate);
-- Si el monto solicitado por el hotelero no esta dentro de los limites
de tolerancia
    ELSE
        -- Registrar que la reserva no puede conciliarse por diferencia de
monto
        dbms_output.put_line('    Error Tolerance');
        INSERT INTO CONCILIATION (
            ID, HOTEL_STATEMENT_ID, PAYMENT_ORDER_ID,
            CONCILIATED_AMOUNT, CONCILIATED_AMOUNT_CURRENCY,
            ADJUSTMENT_AMOUNT, ADJUSTMENT_AMOUNT_CURRENCY,
            STATUS_ID, CREATED, MODIFIED)
VALUES (CONCILIATION_SEQ.nextval, pHsId, vPoId,
        pAmount, pCurrency, null, null,
        C_STATUS_ERROR_TOLERANCE,sysdate,sysdate);
    END IF;

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        -- Registrar que no se encontro una reserva de las caracteriticas que
el hotelero indico
        dbms_output.put_line('    Not Found');
        INSERT INTO CONCILIATION (
            ID, HOTEL_STATEMENT_ID, PAYMENT_ORDER_ID,
            CONCILIATED_AMOUNT, CONCILIATED_AMOUNT_CURRENCY,
            ADJUSTMENT_AMOUNT, ADJUSTMENT_AMOUNT_CURRENCY,

```

```

        STATUS_ID, CREATED, MODIFIED)
VALUES (CONCILIATION_SEQ.nextval, pHsId, null,
        null, null, null, null,
        C_STATUS_NOT_FOUND,sysdate,sysdate);
END conciliate_booking;

-- Conciliacion de todos los extractos pendientes
PROCEDURE conciliate_all_statements AS
BEGIN
    -- Recorro los extractos pendientes
    FOR R IN (
        SELECT
            hs.ID, hs.SUPPLIER_ID, hs.RECORD_LOCATOR, hs.AMOUNT,
hs.CURRENCY,
            s.CONCILIATION_TOLERANCE_PERC, s.CONCILIATION_TOLERANCE_MAX
        FROM hotel_statement hs
        JOIN supplier s ON s.ID = hs.SUPPLIER_ID
        WHERE hs.id NOT IN (
            SELECT hotel_statement_id
            FROM conciliation
        )
    ) LOOP
        -- Concilio una reserva
        conciliate_booking(
            R.ID,R.SUPPLIER_ID,R.RECORD_LOCATOR,R.AMOUNT,R.CURRENCY,
            R.CONCILIATION_TOLERANCE_PERC, R.CONCILIATION_TOLERANCE_MAX
        );
        -- El extracto debe procesarse completo
        COMMIT;
    END LOOP;
END conciliate_all_statements;

END CONCILIATE_PKG;

```

## Mejora 4

```

/**
 * Aplica el UPDATE-3 y, además, resuelve el anti-patrón (N + 1), lo que
 * reduce considerablemente la cantidad de queries emitidas durante una
 * conciliación.
 */

@3_conciliationStatusRefactor.sql

CREATE OR REPLACE PACKAGE BODY CONCILIATE_PKG AS

    -- Conciliacion de una reserva
    PROCEDURE conciliate_booking ( pHsId NUMBER, pSupplier NUMBER, pRecordLocator
VARCHAR,
                                pAmount NUMBER, pCurrency VARCHAR, vTolPercentage NUMBER,
vTolMax NUMBER,
                                vPoId NUMBER, vAmount NUMBER, vCurrency CHAR, vCheckinDate
DATE, vCheckoutDate DATE

```

```

        ) AS

BEGIN
    dbms_output.put_line(
        'Conciliate_booking - pHsId: ' || pHsId
        || ' pSupplier: ' || pSupplier
        || ' pRecordLocator: ' || pRecordLocator
        || ' vPoId: ' || vPoId
    );
    -- Si no paso la fecha de checkout no se puede pagar aun
    IF vPoId IS NULL THEN
        dbms_output.put_line('      Not Found');
        INSERT INTO CONCILIATION (
            ID, HOTEL_STATEMENT_ID, PAYMENT_ORDER_ID,
            CONCILIATED_AMOUNT, CONCILIATED_AMOUNT_CURRENCY,
            ADJUSTMENT_AMOUNT, ADJUSTMENT_AMOUNT_CURRENCY,
            STATUS_ID, CREATED, MODIFIED)
        VALUES (CONCILIATION_SEQ.nextval, pHsId, null,
            null, null, null, null,
            C_STATUS_NOT_FOUND,sysdate,sysdate);
    ELSIF vCheckOutDate>SYSDATE THEN
        -- Registrar que la reserva aun no puede conciliarse por estar
        pendiente su fecha de checkout
        dbms_output.put_line('      Checkout Pending');
        INSERT INTO CONCILIATION (
            ID, HOTEL_STATEMENT_ID, PAYMENT_ORDER_ID,
            CONCILIATED_AMOUNT, CONCILIATED_AMOUNT_CURRENCY,
            ADJUSTMENT_AMOUNT, ADJUSTMENT_AMOUNT_CURRENCY,
            STATUS_ID, CREATED, MODIFIED)
        VALUES (CONCILIATION_SEQ.nextval, pHsId, null,
            null, null, null, null,
            C_STATUS_CHECKOUT_PENDING,sysdate,sysdate);
    -- Si la moneda de conciliacion y la del hotelero no coinciden
    ELSIF vCurrency NOT LIKE pCurrency THEN
        -- Registrar que la moneda indicada en el extracto no es la correcta
        dbms_output.put_line('      Wrong Currency');
        INSERT INTO CONCILIATION (
            ID, HOTEL_STATEMENT_ID, PAYMENT_ORDER_ID,
            CONCILIATED_AMOUNT, CONCILIATED_AMOUNT_CURRENCY,
            ADJUSTMENT_AMOUNT, ADJUSTMENT_AMOUNT_CURRENCY,
            STATUS_ID, CREATED, MODIFIED)
        VALUES (CONCILIATION_SEQ.nextval, pHsId, null,
            null, null, null, null,
            C_STATUS_WRONG_CURRENCY,sysdate,sysdate);
    -- Si el monto solicitado por el hotelero esta dentro de los limites de
    tolerancia
    ELSIF ( ((vAmount-pAmount)<((vTolPercentage/100)*pAmount)) AND
        ((vAmount-pAmount)<vTolMax) ) THEN
        -- Registrar que se aprueba la conciliacion de la reserva
        dbms_output.put_line('      Conciliated');
        INSERT INTO CONCILIATION (
            ID, HOTEL_STATEMENT_ID, PAYMENT_ORDER_ID,
            CONCILIATED_AMOUNT, CONCILIATED_AMOUNT_CURRENCY,
            ADJUSTMENT_AMOUNT, ADJUSTMENT_AMOUNT_CURRENCY,
            STATUS_ID, CREATED, MODIFIED)
        VALUES (CONCILIATION_SEQ.nextval, pHsId, vPoId,
            pAmount, pCurrency, round(vAmount-pAmount,2), pCurrency,
            C_STATUS_CONCILIATED,sysdate,sysdate);

```

```

-- Si el monto solicitado por el hotelero no esta dentro de los limites
de tolerancia
ELSE
    -- Registrar que la reserva no puede conciliarse por diferencia de
monto
    dbms_output.put_line('    Error Tolerance');
    INSERT INTO CONCILIATION (
        ID, HOTEL_STATEMENT_ID, PAYMENT_ORDER_ID,
        CONCILIATED_AMOUNT, CONCILIATED_AMOUNT_CURRENCY,
        ADJUSTMENT_AMOUNT, ADJUSTMENT_AMOUNT_CURRENCY,
        STATUS_ID, CREATED, MODIFIED)
    VALUES (CONCILIATION_SEQ.nextval, pHsId, vPoId,
        pAmount, pCurrency, null, null,
        C_STATUS_ERROR_TOLERANCE,sysdate,sysdate);
    END IF;
END conciliate_booking;

-- Conciliacion de todos los extractos pendientes
PROCEDURE conciliate_all_statements AS
BEGIN
    -- Recorro los extractos pendientes
    FOR R IN (
        SELECT
            hs.ID, hs.SUPPLIER_ID, hs.RECORD_LOCATOR, hs.AMOUNT,
hs.CURRENCY,
            s.CONCILIATION_TOLERANCE_PERC, s.CONCILIATION_TOLERANCE_MAX,
            po.ID vPoId, po.TOTAL_COST, po.TOTAL_COST_CURRENCY, po.CHECKIN,
po.CHECKOUT
        FROM hotel_statement hs
        JOIN supplier s ON s.ID = hs.SUPPLIER_ID
        LEFT JOIN payment_order po ON (
            po.RECORD_LOCATOR = hs.RECORD_LOCATOR
            AND po.supplier_id = hs.supplier_id
            AND po.id NOT IN (
                SELECT payment_order_id
                FROM conciliation
                WHERE payment_order_id IS NOT NULL
            )
        )
        WHERE hs.id NOT IN (
            SELECT hotel_statement_id
            FROM conciliation
        )
    ) LOOP
        -- Concilio una reserva
        conciliate_booking(
            R.ID,R.SUPPLIER_ID,R.RECORD_LOCATOR,R.AMOUNT,R.CURRENCY,
            R.CONCILIATION_TOLERANCE_PERC, R.CONCILIATION_TOLERANCE_MAX,
            R.vPoId, R.TOTAL_COST, R.TOTAL_COST_CURRENCY, R.CHECKIN,
R.CHECKOUT
        );
        -- El extracto debe procesarse completo
        COMMIT;
    END LOOP;
END conciliate_all_statements;

END CONCILIATE_PKG;

```

## Mejora 5

```
/**
 * Aplica el UPDATE-4 y prepara el sistema para el despliegue final, lo que
 * implica reconstruir los índices con el estado actual, y computar las
 * estadísticas completas sobre todas las tablas para mejorar el CBO.
 */

@4_n+1AntipatternFix.sql

/* Index Rebuild */
ALTER INDEX HOTEL_STATEMENT_PK REBUILD;
ALTER INDEX HOTEL_STATEMENT_RECORD_LOCATOR REBUILD;
ALTER INDEX PAYMENT_ORDER_PK REBUILD;
ALTER INDEX PAYMENT_ORDER_RECORD_LOCATOR REBUILD;
ALTER INDEX PO2_IDX REBUILD;
ALTER INDEX STATUS_ID_IDX REBUILD;
ALTER INDEX SUPPLIER_PK REBUILD;

/* Table Statistics */
EXEC DBMS_STATS.gather_table_stats('BDII_TEAM1', 'CONCILIATION_STATUS', cascade
=> TRUE);
EXEC DBMS_STATS.gather_table_stats('BDII_TEAM1', 'HOTEL_STATEMENT', cascade =>
TRUE);
EXEC DBMS_STATS.gather_table_stats('BDII_TEAM1', 'PAYMENT_ORDER', cascade =>
TRUE);
EXEC DBMS_STATS.gather_table_stats('BDII_TEAM1', 'SUPPLIER', cascade => TRUE);

/* Index Statistics */
EXEC DBMS_STATS.gather_index_stats('BDII_TEAM1', 'HOTEL_STATEMENT_PK');
EXEC DBMS_STATS.gather_index_stats('BDII_TEAM1',
'HOTEL_STATEMENT_RECORD_LOCATOR');
EXEC DBMS_STATS.gather_index_stats('BDII_TEAM1', 'PAYMENT_ORDER_PK');
EXEC DBMS_STATS.gather_index_stats('BDII_TEAM1',
'PAYMENT_ORDER_RECORD_LOCATOR');
EXEC DBMS_STATS.gather_index_stats('BDII_TEAM1', 'PO2_IDX');
EXEC DBMS_STATS.gather_index_stats('BDII_TEAM1', 'STATUS_ID_IDX');
EXEC DBMS_STATS.gather_index_stats('BDII_TEAM1', 'SUPPLIER_PK');
```