

Proyecto 2 SisTrans

Marco Alejandro Ramírez - Jhostin Aleck Sanchez

November 8, 2023

1 Cambios respecto a la entrega 1

Para esta entrega se hicieron muchos cambios en la base de datos, aunque antes la base de datos estaba normalizada, era muy complejo realizar consultas grandes, debido a que cada servicio tenía su propia tabla.

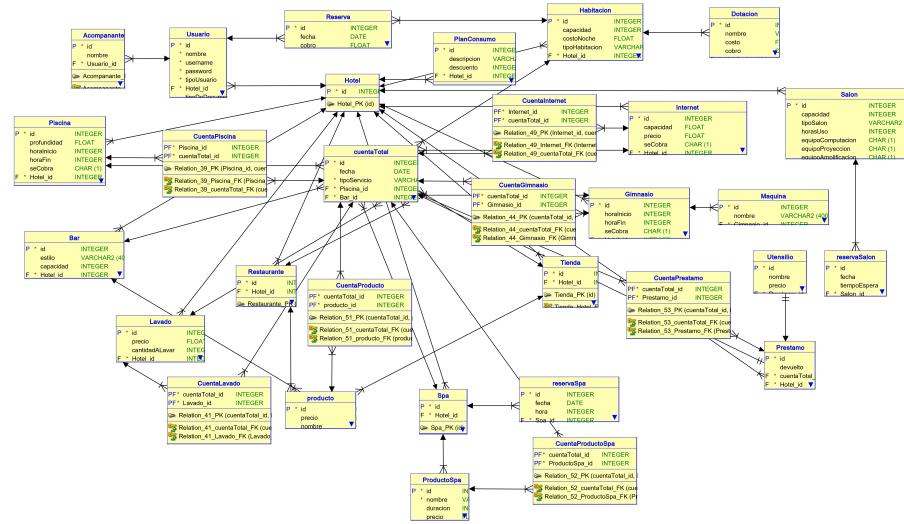


Figure 1: Diagrama relacional anterior

Para mejorar este aspecto, decidimos centralizar todos los servicios en una sola tabla. Ayudando a mejorar la complejidad de las consultas (menos JOIN's) y legibilidad de los diagramas.

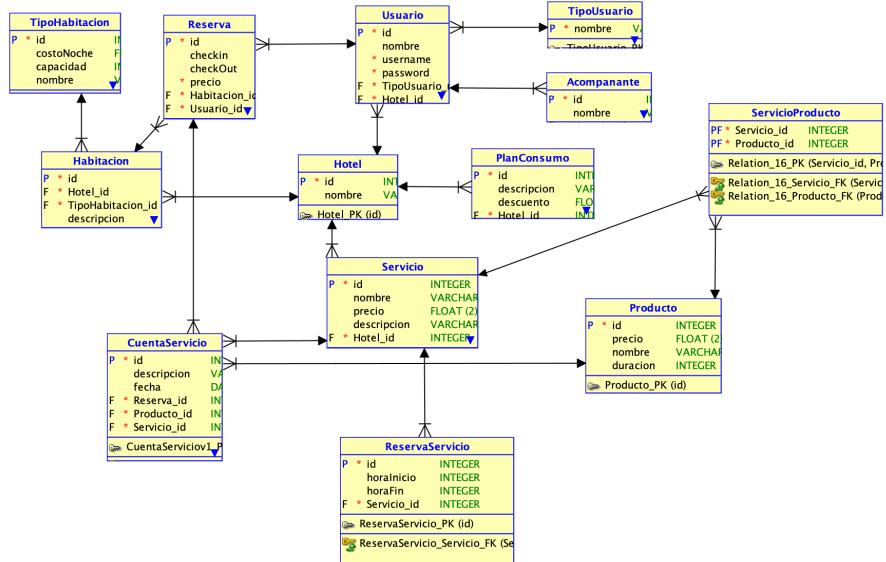


Figure 2: Diagrama relacional nuevo

Por otro lado, habían cosas a cambiar necesariamente, por ejemplo:

1. Crear la tabla tipoHabitacion, antes era solamente un atributo de en la tabla habitacion
 2. Crear la tabla tipoUsuario, antes era solamente un atributo de en la tabla usuario
 3. Unificar la reserva de servicio, porque antes era una reserva spa, reserva salón, etc.

2 Requerimiento Funcional 1

2.1 Query

Para el primer Requerimiento Funcional se debe haber muchos Left Joins para poder acceder a toda la información, debido a la **normalización** de las tablas. La query para este requerimiento es la siguiente:

```
SELECT
    r.habitacion_id AS habitacion_id,
    s.nombre AS servicio_nombre,
    SUM(
        CASE
            WHEN cs.producto_id IS NOT NULL THEN p.precio
            ELSE s.precio
        END
```

```

) AS dinero_recolectado
FROM
    reserva r
    JOIN cuentaservicio cs ON r.id = cs.reserva_id
    JOIN servicio s ON cs.servicio_id = s.id
    LEFT JOIN producto p ON cs.producto_id = p.id
WHERE
    r.checkin >= TRUNC(SYSDATE) - INTERVAL '1' YEAR
GROUP BY
    r.habitacion_id, s.nombre
ORDER BY
    r.habitacion_id, servicio_nombre;

```

2.2 Indices

La elección de los índices depende en gran medida de las necesidades y patrones de acceso específicos de tu aplicación y de la cantidad de datos en tu base de datos. Sin embargo, en general, las siguientes opciones pueden ser consideradas como las más útiles:

Índice en **cuentaservicio** en la columna fecha: Este índice es crucial para acelerar la búsqueda de registros en la tabla cuentaservicio basada en la fecha, y es esencial para la restricción de tiempo en tu consulta. Si tu consulta se ejecuta con frecuencia y es importante para tu aplicación, este índice es crítico.

Índice en **habitacion** en la columna id: Este índice ayudará a acelerar la búsqueda de registros en la tabla habitacion cuando se realiza la unión con otras tablas. Si las consultas que involucran habitacion son comunes en tu aplicación, este índice puede ser beneficioso.

Índice en **cuentaservicio** en las columnas reserva id, servicio id y producto id: Estos índices son útiles para acelerar las uniones entre las tablas cuentaservicio, servicio y producto. Si las consultas que involucran estas tablas son frecuentes en tu aplicación, estos índices pueden mejorar el rendimiento.

```

CREATE INDEX idx_cuentaservicio_fecha ON cuentaservicio (fecha);

CREATE INDEX idx_habitacion_id ON habitacion (id);

CREATE INDEX idx_reserva_habitacion_id ON reserva (habitacion_id);

CREATE INDEX idx_cuentaservicio_reserva_id ON
cuentaservicio (reserva_id);

CREATE INDEX idx_cuentaservicio_servicio_id ON
cuentaservicio (servicio_id);

CREATE INDEX idx_cuentaservicio_producto_id ON
cuentaservicio (producto_id);

```

3 Requerimiento Funcional 2

3.1 Query

Esta consulta solo necesita un Join, y es mucho más sencilla de ver, se tomaron como ejemplo las fechas 2022-01-01 y 2023-12-31.

```
SELECT
    s.id AS servicio_id,
    s.nombre AS servicio_nombre,
    COUNT(cs.id) AS cantidad_consumos
FROM
    servicio s
    JOIN cuentaservicio cs ON s.id = cs.servicio_id
WHERE
    cs.fecha >= TO_DATE('2022-01-01', 'YYYY-MM-DD') -- fecha 1
    AND cs.fecha <= TO_DATE('2023-12-31', 'YYYY-MM-DD') -- fecha 2
GROUP BY
    s.id, s.nombre
ORDER BY
    cantidad_consumos DESC
FETCH FIRST 20 ROWS ONLY;
```

3.2 Indices

La elección de los índices depende de la estructura de la base de datos y de cómo se utiliza en tu aplicación. Los indices que pensamos en tener en cuenta fueron:

Índice en **cuentaservicio** en la columna fecha: Este índice es fundamental para filtrar las transacciones en un período de tiempo dado y puede ser esencial si las consultas con restricciones de tiempo son comunes en tu aplicación.

Índice en **servicio** en la columna id: Este índice puede mejorar la eficiencia de la unión con la tabla servicio y es beneficioso si las consultas que involucran información detallada sobre los servicios son frecuentes.

Índice compuesto en **cuentaservicio** en las columnas **servicio id y fecha**: Este índice es útil para acelerar la búsqueda y agregación de consumos de servicios dentro del período de tiempo deseado. Es especialmente relevante si deseas optimizar el rendimiento de consultas que involucran estas dos columnas.

```
CREATE INDEX idx_cuentaservicio_fecha ON cuentaservicio (fecha);

CREATE INDEX idx_servicio_id ON servicio (id);

CREATE INDEX idx_cuentaservicio_servicio_id_fecha ON
cuentaservicio (servicio_id, fecha);
```

4 Requerimiento Funcional 3

4.1 Query

Esta consulta es larga, debido a que, tiene que calcular muchos valores en una misma consulta.

```
WITH ReservasPorHabitacion AS (
    SELECT
        h.id AS habitacion_id,
        COUNT(r.id) AS total_reservas
    FROM
        habitacion h
        LEFT JOIN reserva r ON h.id = r.habitacion_id
    WHERE
        r.checkin >= TRUNC(SYSDATE) - INTERVAL '1' YEAR
    GROUP BY
        h.id
),
DiasOcupadosPorHabitacion AS (
    SELECT
        r.habitacion_id,
        SUM(
            CASE
                WHEN r.checkin >= TRUNC(SYSDATE) - INTERVAL '1'
                    YEAR THEN
                    LEAST(r.checkout, TRUNC(SYSDATE)) -
                    GREATEST(r.checkin, TRUNC(SYSDATE)) -
                    INTERVAL '1' YEAR) + 1
                ELSE
                    LEAST(r.checkout, TRUNC(SYSDATE)) -
                    TRUNC(SYSDATE) + 1
            END
        ) AS total_dias_ocupados
    FROM
        reserva r
    WHERE
        r.checkout >= TRUNC(SYSDATE) - INTERVAL '1' YEAR
    GROUP BY
        r.habitacion_id
)
SELECT
    h.id AS habitacion_id,
    COALESCE(DO.total_dias_ocupados, 0) AS total_dias_ocupados,
    COALESCE(RP.total_reservas, 0) AS total_reservas,
    ROUND(
        CASE
            WHEN COALESCE(DO.total_dias_ocupados, 0) = 0 THEN 0
            ELSE (DO.total_dias_ocupados / 365) * 100
        END, 2
    ) AS indice_ocupacion
FROM
    habitacion h
LEFT JOIN ReservasPorHabitacion RP ON h.id = RP.habitacion_id
LEFT JOIN DiasOcupadosPorHabitacion DO ON h.id = DO.habitacion_id
ORDER BY
```

```
indice_ocupacion DESC;
```

4.2 Indices

En cuanto a las opciones de índices tenemos, hay 2 que son útiles para mejorar el rendimiento de la consulta que calcula el índice de ocupación de las habitaciones.

Índice en habitacion(id): Este índice es útil si se realizan búsquedas o consultas que involucran la tabla habitacion y necesitas una rápida recuperación de datos por id. Si las consultas que implican esta tabla son frecuentes, este índice es beneficioso.

Índices en reserva(habitacion id, checkin, checkout): Estos índices son útiles si las consultas frecuentes están relacionadas con la tabla reserva y se basan en los campos habitacion id, checkin y checkout. Estos índices facilitarán la búsqueda y la unión de datos al filtrar por fechas y habitaciones.

```
CREATE INDEX idx_habitacion_id ON habitacion(id);

CREATE INDEX idx_reserva_habitacion_id ON reserva(habitacion_id);
CREATE INDEX idx_reserva_checkin ON reserva(checkin);
CREATE INDEX idx_reserva_checkout ON reserva(checkout);
```

5 Requerimiento Funcional 4

5.1 Query

Esta consulta es corta, debido a que, solamente hay que tener cuenta si los valores son null's o no.

```
SELECT s.*
FROM servicio s
JOIN cuentaservicio cs ON s.id = cs.servicio_id
WHERE (:p_precio_min IS NULL OR s.precio >= :p_precio_min)
AND (:p_precio_max IS NULL OR s.precio <= :p_precio_max)
AND (:p_fecha_inicio IS NULL OR cs.fecha >= :p_fecha_inicio)
AND (:p_fecha_fin IS NULL OR cs.fecha <= :p_fecha_fin);
```

5.2 Indices

Al crear índices en las columnas que se utilizan en cláusulas WHERE y JOIN, se puede reducir significativamente el tiempo necesario para ejecutar las consultas, especialmente cuando las tablas contienen grandes conjuntos de datos. Aquí hay una explicación más detallada:

Índice en servicio(precio): Este índice permite una búsqueda eficiente de los registros en la tabla servicio basados en el precio. Dado que tu consulta filtra por el precio del servicio, este índice acelera la búsqueda y mejora el rendimiento.

Índice en servicio(id): Si no existe un índice en la columna id, crear uno puede ser beneficioso, ya que se utiliza en la unión entre las tablas servicio y cuentaservicio. Esto mejora la velocidad de recuperación de los datos de ambas tablas.

Índice en cuentaservicio(fecha): La consulta también filtra por fecha en la tabla cuentaservicio. Un índice en la columna fecha mejora la eficiencia de la búsqueda, especialmente cuando se utilizan condiciones de rango en las fechas.

Índice en cuentaservicio(servicio_id): Al igual que en el caso de la tabla servicio, si no existe un índice en la columna servicio_id, crearlo mejora el rendimiento de la unión entre las tablas servicio y cuentaservicio.

```
CREATE INDEX idx_servicio_precio ON servicio(precio);
CREATE INDEX idx_servicio_id ON servicio(id);

CREATE INDEX idx_cuentaservicio_fecha ON cuentaservicio(fecha);
CREATE INDEX idx_cuentaservicio_servicio_id ON
cuentaservicio(servicio_id);
```

6 Requerimiento Funcional 5

6.1 Query

Esta consulta es extensa, debido a que, se tienen que mostrar datos de varias tablas a la vez, entonces necesitamos hacer 4 joins.

```
SELECT
    u.nombre AS Usuario,
    s.nombre AS Servicio,
    cs.descripcion AS DescripcionServicio,
    cs.fecha AS FechaServicio,
    s.precio AS PrecioServicio,
    p.nombre AS Producto,
    p.precio AS PrecioProducto
FROM
    usuario u
    JOIN reserva r ON u.id = r.usuario_id
    JOIN habitacion hab ON r.habitacion_id = hab.id
    LEFT JOIN cuentaservicio cs ON r.id = cs.reserva_id
    LEFT JOIN servicio s ON cs.servicio_id = s.id
    LEFT JOIN servicioproducto sp ON s.id = sp.servicio_id
    LEFT JOIN producto p ON sp.producto_id = p.id
WHERE
    u.id = :given_user_id
    AND (
        (r.checkin BETWEEN TO_DATE(:start_date, 'YYYY-MM-DD')
        AND TO_DATE(:end_date, 'YYYY-MM-DD'))
        OR (r.checkout BETWEEN TO_DATE(:start_date, 'YYYY-MM-DD')
        AND TO_DATE(:end_date, 'YYYY-MM-DD')))
    )
ORDER BY
    cs.fecha;
```

6.2 Indices

Vamos a hacer un indice para cada una de las columnas involucradas en un JOIN o un WHERE, para agilizar lo máximo posible la consulta.

```
CREATE INDEX idx_usuario_id ON usuario(id);

CREATE INDEX idx_reserva_usuario_id ON reserva(usuario_id);
CREATE INDEX idx_reserva_checkin ON reserva(checkin);
CREATE INDEX idx_reserva_checkout ON reserva(checkout);

CREATE INDEX idx_cuentaservicio_reserva_id ON
cuentaservicio(reserva_id);

CREATE INDEX idx_servicio_id ON servicio(id);

CREATE INDEX idx_servicioproducto_servicio_id ON
servicioproducto(servicio_id);

CREATE INDEX idx_producto_id ON producto(id);
```

7 Requerimiento Funcional 6

7.1 Query

Esta consulta son realmente 3 a la vez, pero las pusimos en una sola para fines de facilidad a la hora de mostrarlo en el front

```
WITH ResumenDiario AS (
    SELECT
        TRUNC(r.checkin) AS fecha,
        COUNT(DISTINCT r.habitacion_id) AS habitaciones_ocupadas,
        SUM(
            CASE
                WHEN cs.producto_id IS NOT NULL THEN p.precio
                ELSE s.precio
            END
        ) AS ingresos
    FROM
        reserva r
        LEFT JOIN cuentaservicio cs ON r.id = cs.reserva_id
        LEFT JOIN servicio s ON cs.servicio_id = s.id
        LEFT JOIN producto p ON cs.producto_id = p.id
    GROUP BY
        TRUNC(r.checkin)
),
FechaMayorOcupacion AS (
    SELECT
        fecha,
        habitaciones_ocupadas
    FROM
        ResumenDiario
    WHERE
        habitaciones_ocupadas = (SELECT
```

```

        MAX(habitaciones_ocupadas) FROM ResumenDiario
),
FechaMayorIngresos AS (
    SELECT
        fecha,
        ingresos
    FROM
        ResumenDiario
    WHERE
        ingresos = (SELECT MAX(ingresos) FROM ResumenDiario)
),
FechaMenorDemandas AS (
    SELECT
        fecha,
        habitaciones_ocupadas
    FROM
        ResumenDiario
    WHERE
        habitaciones_ocupadas = (SELECT
            MIN(habitaciones_ocupadas) FROM ResumenDiario)
)
SELECT
    'Fecha de Mayor Ocupacion' AS tipo,
    TO_CHAR(fmo.fecha, 'YYYY-MM-DD') AS fecha,
    fmo.habitaciones_ocupadas AS valor
FROM
    FechaMayorOcupacion fmo
UNION ALL
SELECT
    'Fecha de Mayores Ingresos' AS tipo,
    TO_CHAR(fmi.fecha, 'YYYY-MM-DD') AS fecha,
    fmi.ingresos AS valor
FROM
    FechaMayorIngresos fmi
UNION ALL
SELECT
    'Fecha de Menor Demanda' AS tipo,
    TO_CHAR(fmd.fecha, 'YYYY-MM-DD') AS fecha,
    fmd.habitaciones_ocupadas AS valor
FROM
    FechaMenorDemandas fmd
WHERE
    fmd.fecha = (
        SELECT MIN(fecha)
        FROM FechaMenorDemandas
    );

```

7.2 Indices

Vamos a hacer un indice para cada una de las columnas involucradas en un JOIN o un WHERE, para agilizar lo máximo posible la consulta.

```

CREATE INDEX idx_usuario_id ON usuario(id);

CREATE INDEX idx_reserva_usuario_id ON reserva(usuario_id);

```

```

CREATE INDEX idx_reserva_checkin ON reserva(checkin);
CREATE INDEX idx_reserva_checkout ON reserva(checkout);

CREATE INDEX idx_cuentaservicio_reserva_id ON
cuentaservicio(reserva_id);

CREATE INDEX idx_servicio_id ON servicio(id);

CREATE INDEX idx_servicioproducto_servicio_id ON
servicioproducto(servicio_id);

CREATE INDEX idx_producto_id ON producto(id);

```

8 Requerimiento Funcional 7

8.1 Query

Esta consulta es muy larga por la cantidad de CASE's que toca hacer. Esto debido a que hay que mostrar la razón por la que es un buen cliente.

```

WITH BuenClienteCriterio AS (
    SELECT
        r.usuario_id AS usuario_id,
        SUM(
            CASE
                WHEN r.checkout >= TRUNC(SYSDATE) - INTERVAL '1'
                    YEAR THEN
                    r.checkout - GREATEST(r.checkin,
                    TRUNC(SYSDATE) - INTERVAL '1' YEAR) + 1
                ELSE
                    r.checkout - TRUNC(SYSDATE) + 1
            END
        ) AS total_dias_hospedaje,
        SUM(
            CASE
                WHEN cs.fecha >= TRUNC(SYSDATE) - INTERVAL '1'
                    YEAR THEN
                    CASE
                        WHEN cs.producto_id IS NOT NULL THEN
                            p.precio
                        ELSE s.precio
                    END
                ELSE
                    0
            END
        ) AS total_consumo_ultimo_anio
    FROM
        reserva r
        LEFT JOIN cuentaservicio cs ON r.id = cs.reserva_id
        LEFT JOIN servicio s ON cs.servicio_id = s.id
        LEFT JOIN producto p ON cs.producto_id = p.id
    GROUP BY
        r.usuario_id
)
SELECT

```

```

u.nombre AS Usuario,
u.username AS Username,
CASE
    WHEN BC.total_dias_hospedaje >= 14 OR
        BC.total_consumo_ultimo_anio >= 15000000 THEN
        'Buen_Cliente'
    ELSE
        'Cliente_Regular'
END AS EstadoCliente,
CASE
    WHEN BC.total_dias_hospedaje >= 14 AND
        BC.total_consumo_ultimo_anio >= 15000000 THEN
        'Hospedaje_de_1_a_2_semanas_y_consumo_superior_a_15000.00'
    WHEN BC.total_dias_hospedaje >= 14 THEN
        'Hospedaje_de_1_a_2_semanas'
    WHEN BC.total_consumo_ultimo_anio >= 15000000 THEN
        'Consumo_superior_a_15000.00'
    ELSE
        'No_cumple_con_los_criterios_para_ser_considerado_un_buen_cliente'
END AS Razon
FROM
    usuario u
    LEFT JOIN BuenClienteCriterio BC ON u.id = BC.usuario_id
WHERE
    BC.total_dias_hospedaje >= 14 OR
    BC.total_consumo_ultimo_anio >= 15000000;

```

8.2 Indices

Vamos a hacer un indice para cada una de las columnas involucradas en un JOIN o un WHERE, para agilizar lo máximo posible la consulta.

```

CREATE INDEX idx_reserva_usuario_id ON reserva(usuario_id);
CREATE INDEX idx_reserva_checkin ON reserva(checkin);
CREATE INDEX idx_reserva_checkout ON reserva(checkout);

CREATE INDEX idx_cuentaservicio_reserva_id ON
cuentaservicio(reserva_id);
CREATE INDEX idx_cuentaservicio_fecha ON cuentaservicio(fecha);

CREATE INDEX idx_servicio_id ON servicio(id);

CREATE INDEX idx_producto_id ON producto(id);

```

9 Requerimiento Funcional 8

9.1 Query

En esta consulta el DISTINCT no sirvió, entonces tuvimos que tener en cuenta otra forma de tener NO duplicar los servicios a mostrar. Se muestra la semana en la que falló no teniendo 3 o más servicios.

```

WITH ServiciosSolicitados AS (
    SELECT
        cs.servicio_id,
        TRUNC(cs.fecha, 'WW') AS semana,
        COUNT(*) AS solicitudes_semanales
    FROM
        cuentaservicio cs
    WHERE
        cs.fecha >= TRUNC(SYSDATE) - INTERVAL '1' YEAR
    GROUP BY
        cs.servicio_id, TRUNC(cs.fecha, 'WW')
),
ServiciosConRecuento AS (
    SELECT
        s.nombre AS Servicio,
        TO_CHAR(ss.semana, 'YYYY-MM-DD') AS Semana,
        ss.solicitudes_semanales AS SolicitudesSemanales,
        ROW_NUMBER() OVER (PARTITION BY s.id ORDER BY
            ss.solicitudes_semanales DESC) AS rn
    FROM
        servicio s
    LEFT JOIN ServiciosSolicitados ss ON s.id = ss.servicio_id
    WHERE
        (ss.solicitudes_semanales < 3 OR
        ss.solicitudes_semanales IS NULL)
)
SELECT
    Servicio,
    Semana,
    SolicitudesSemanales
FROM
    ServiciosConRecuento
WHERE
    rn = 1;

```

9.2 Indices

Vamos a hacer un indice para cada una de las columnas involucradas en un JOIN o un WHERE, para agilizar lo máximo posible la consulta.

```

CREATE INDEX idx_cuentaservicio_servicio_id ON
cuentaservicio(servicio_id);

CREATE INDEX idx_servicio_id ON servicio(id);

```

10 Requerimiento Funcional 9

10.1 Query

Esta consulta depende de lo que diga el usuario, entonces hay varias formas de hacerlo, está es una de las combinaciones. Agrupamiento es el item que decida el usuario, y ordenamiento también.

```

SELECT u.id, u.nombre, COUNT(cs.id) AS numero_de_veces,
MAX(cs.fecha) AS ultima_fecha_uso
FROM usuario u
JOIN reserva r ON u.id = r.usuario_id
JOIN cuentaservicio cs ON cs.reserva_id = r.id
JOIN servicio s ON cs.servicio_id = s.id
WHERE s.id = :servicioId
AND cs.fecha BETWEEN :fechaInicio AND :fechaFin
GROUP BY "u+agrupamiento" "u+
ORDERBY" + agrupamiento + "ordenamiento;

```

10.2 Indices

Vamos a hacer un indice para cada una de las columnas involucradas en un JOIN o un WHERE, para agilizar lo máximo posible la consulta.

```

CREATE INDEX idx_usuario_id ON usuario (id);

CREATE INDEX idx_reserva_usuario_id ON reserva (usuario_id);

CREATE INDEX idx_cuentaservicio_servicio_id ON cuentaservicio
(servicio_id);

CREATE INDEX idx_cuentaservicio_fecha ON cuentaservicio (fecha);

CREATE INDEX idx_cuentaservicio_reserva_servicio ON
cuentaservicio (reserva_id, servicio_id);

```

11 Requerimiento Funcional 10

11.1 Query

```

SELECT DISTINCT u.id, u.nombre
FROM usuario u
WHERE u.id NOT IN (
    SELECT DISTINCT u.id
    FROM usuario u
    JOIN reserva r ON u.id = r.usuario_id
    JOIN cuentaservicio cs ON r.id = cs.reserva_id
    WHERE cs.servicio_id = :servicioId
    AND cs.fecha BETWEEN :fechaInicio AND :fechaFin
)

```

11.2 Indices

Vamos a hacer un indice para cada una de las columnas involucradas en un JOIN o un WHERE, para agilizar lo máximo posible la consulta.

```
CREATE INDEX idx_usuario_id ON usuario (id);
```

```

CREATE INDEX idx_reserva_usuario_id ON reserva (usuario_id);

CREATE INDEX idx_cuentaservicio_reserva_id ON cuentaservicio
(reserva_id);

CREATE INDEX idx_cuentaservicio_servicio_id ON cuentaservicio
(servicio_id);

CREATE INDEX idx_cuentaservicio_fecha ON cuentaservicio (fecha);

```

12 Requerimiento Funcional 11

12.1 Query

En esta consulta el DISTINCT no sirvió, entonces tuvimos que tener en cuenta otra forma de tener NO duplicar los servicios a mostrar. Se muestra la semana en la que falló no teniendo 3 o más servicios.

```

WITH Semanas AS (
    SELECT DISTINCT
        TO_CHAR(DATE '2023-01-01' + (level - 1) * 7, 'YYYY-IW')
        AS semana,
        TO_CHAR(DATE '2023-01-01' + (level - 1) * 7, 'YYYY-MM-
        DD') AS inicio_semana,
        TO_CHAR(DATE '2023-01-01' + (level - 1) * 7 + 6, 'YYYY-
        MM-DD') AS fin_semana
    FROM dual
    CONNECT BY level <= 53
)
, ServiciosSemana AS (
    SELECT
        TO_CHAR(cs.fecha, 'YYYY-IW') AS semana,
        s.nombre AS servicio_nombre,
        COUNT(*) AS cantidad_consumos
    FROM cuentaservicio cs
    JOIN servicio s ON cs.servicio_id = s.id
    WHERE cs.fecha BETWEEN TO_DATE(:fecha_inicio, 'YYYY-MM-DD')
    AND TO_DATE(:fecha_fin, 'YYYY-MM-DD')
    GROUP BY TO_CHAR(cs.fecha, 'YYYY-IW'), s.nombre
)
, HabitacionesSemana AS (
    SELECT
        TO_CHAR(r.checkin, 'YYYY-IW') AS semana,
        h.id AS habitacion_id,
        COUNT(*) AS cantidad_solicitudes
    FROM reserva r
    JOIN habitacion h ON r.habitacion_id = h.id
    WHERE r.checkin BETWEEN TO_DATE(:fecha_inicio, 'YYYY-MM-DD')
    AND TO_DATE(:fecha_fin, 'YYYY-MM-DD')
    GROUP BY TO_CHAR(r.checkin, 'YYYY-IW'), h.id
)
, SemanaServicioMax AS (
    SELECT semana, servicio_nombre, cantidad_consumos,
        ROW_NUMBER() OVER (PARTITION BY semana ORDER BY
        cantidad_consumos DESC) AS rn_max

```

```

        FROM ServiciosSemana
    )
    , SemanaServicioMin AS (
        SELECT semana, servicio_nombre, cantidad_consumos,
               ROW_NUMBER() OVER (PARTITION BY semana ORDER BY
               cantidad_consumos ASC) AS rn_min
        FROM ServiciosSemana
    )
    , SemanaHabitacionMax AS (
        SELECT semana, habitacion_id, cantidad_solicitudes,
               ROW_NUMBER() OVER (PARTITION BY semana ORDER BY
               cantidad_solicitudes DESC) AS rn_max
        FROM HabitacionesSemana
    )
    , SemanaHabitacionMin AS (
        SELECT semana, habitacion_id, cantidad_solicitudes,
               ROW_NUMBER() OVER (PARTITION BY semana ORDER BY
               cantidad_solicitudes ASC) AS rn_min
        FROM HabitacionesSemana
    )
SELECT
    s.semana AS semana,
    (SELECT servicio_nombre FROM SemanaServicioMax WHERE semana
     = s.semana AND rn_max = 1) AS servicio_mas_consumido,
    (SELECT servicio_nombre FROM SemanaServicioMin WHERE semana
     = s.semana AND rn_min = 1) AS servicio_menos_consumido,
    (SELECT TO_CHAR(habitacion_id) FROM SemanaHabitacionMax
     WHERE semana = s.semana AND rn_max = 1) AS
     habitacion_mas_solicitada,
    (SELECT TO_CHAR(habitacion_id) FROM SemanaHabitacionMin
     WHERE semana = s.semana AND rn_min = 1) AS
     habitacion_menos_solicitada
FROM Semanas s
ORDER BY s.semana;

```

12.2 Indices

Vamos a hacer un indice para cada una de las columnas involucradas en un JOIN o un WHERE, para agilizar lo máximo posible la consulta.

```

CREATE INDEX idx_cuentaservicio_fecha ON cuentaservicio(fecha);

CREATE INDEX idx_cuentaservicio_servicio_fecha ON
cuentaservicio(servicio_id, fecha);

CREATE INDEX idx_reserva_checkin ON reserva(checkin);

CREATE INDEX idx_reserva_habitacion_checkin ON
reserva(habitacion_id, checkin);

```

13 Requerimiento Funcional 12

13.1 Query

En esta consulta el DISTINCT no sirvió, entonces tuvimos que tener en cuenta otra forma de tener NO duplicar los servicios a mostrar. Se muestra la semana en la que falló no teniendo 3 o más servicios.

```
WITH EstanciasPorTrimestre AS (
    SELECT u.id, u.nombre, COUNT(DISTINCT TRUNC(r.checkin, 'Q'))
    as NumTrimestres
    FROM usuario u
    INNER JOIN reserva r ON u.id = r.usuario_id
    GROUP BY u.id, u.nombre
    HAVING COUNT(DISTINCT TRUNC(r.checkin, 'Q')) >= 4
),
ServiciosCostosos AS (
    SELECT DISTINCT u.id, u.nombre
    FROM usuario u
    INNER JOIN reserva r ON u.id = r.usuario_id
    INNER JOIN cuentaservicio cs ON r.id = cs.reserva_id
    INNER JOIN servicio s ON cs.servicio_id = s.id
    WHERE s.precio > 300000
),
ServiciosDuraderos AS (
    SELECT DISTINCT u.id, u.nombre
    FROM usuario u
    INNER JOIN reserva r ON u.id = r.usuario_id
    INNER JOIN reservaservicio rs ON u.id = rs.usuario_id
    INNER JOIN servicio s ON rs.servicio_id = s.id
    WHERE (s.descripcion LIKE '%SPA%' OR s.descripcion LIKE
    '%reuniones%')
    AND (rs.horafin - rs.horainicio) > 4
)
SELECT u.id, u.nombre, u.username, u.hotel_id, 'Cumple\u00e1estancias
por\u00e1trimestre' as Justificacion
FROM usuario u
INNER JOIN EstanciasPorTrimestre ept ON u.id = ept.id
UNION
SELECT u.id, u.nombre, u.username, u.hotel_id, 'Consum
servicios\u00e1costosos' as Justificacion
FROM usuario u
INNER JOIN ServiciosCostosos sc ON u.id = sc.id
UNION
SELECT u.id, u.nombre, u.username, u.hotel_id, 'Consum
servicios\u00e1de\u00d1SPA\u00e1o\u00d1reuniones\u00e1duraderos' as Justificacion
FROM usuario u
INNER JOIN ServiciosDuraderos sd ON u.id = sd.id;
```

13.2 Indices

Vamos a hacer un indice para cada una de las columnas involucradas en un JOIN o un WHERE, para agilizar lo máximo posible la consulta.

```
CREATE INDEX idx_usuario_id ON usuario (id);
```

```

CREATE INDEX idx_usuario_nombre ON usuario (nombre);

CREATE INDEX idx_reserva_usuario_id ON reserva (usuario_id);
CREATE INDEX idx_reserva_checkin ON reserva (checkin);

CREATE INDEX idx_cuentaservicio_reserva_id ON cuentaservicio
(reserva_id);
CREATE INDEX idx_cuentaservicio_servicio_id ON cuentaservicio
(servicio_id);

CREATE INDEX idx_servicio_id ON servicio (id);
CREATE INDEX idx_servicio_precio ON servicio (precio);
CREATE INDEX idx_servicio_descripcion ON servicio (descripcion);

CREATE INDEX idx_reservaservicio_usuario_id ON reservaservicio
(usuario_id);
CREATE INDEX idx_reservaservicio_servicio_id ON reservaservicio
(servicio_id);

CREATE INDEX idx_estanciasportrimestre_id ON
EstanciasPorTrimestre (id);

CREATE INDEX idx_servicioscostosos_id ON ServiciosCostosos (id);

CREATE INDEX idx_serviciosduraderos_id ON ServiciosDuraderos (id);

```

13.3 Plan Acción

SELECT STATEMENT				
↳ SORT				
↳ UNION-ALL				
↳ HASH JOIN				
↳ Access Predicates				
↳ ID=EP7.ID				
↳ NESTED LOOPS				
↳ NESTED LOOPS				
↳ PARTITION BY COLLECT				
↳ VIEW				
↳ FILTER				
↳ Filter Predicates				
↳ COUNT(INTERNAL_FUNCTION(\$vm_col_1))>=4				
↳ HASH				
↳ VIEW	SYS.VM_NWVW_1	GROUP BY		
↳ HASH				
↳ HASH				
↳ NE				
↳ MERGE				
↳ RESERVA		FULL		
↳ USUARIO_PK		UNIQUE SCAN		
↳ Access Predicates				
↳ U.ID=R.USUARIO_ID				
↳ INDEX				
↳ Access Predicates				
↳ U.ID=EP7.ID				
↳ TABLE ACCESS	USUARIO	BY INDEX ROWID		
↳ TABLE ACCESS	USUARIO	FULL		
↳ HASH JOIN				
↳ Access Predicates				
↳ U.ID=R.USUARIO_ID				
↳ NESTED LOOPS				
↳ NESTED LOOPS				
↳ PARTITION BY COLLECT				
↳ HASH JOIN				
↳ Access Predicates				
↳ R.ID=CS.RESERVA_ID				
↳ NESTED LOOPS				
↳ STATISTICS				
↳ NESTED I		SEMI		
↳ TABLERESERVASERVICIO		FULL		
↳ TABLERESERVACO		BY INDEX ROWID		
↳ Filter Predicates				
↳ S.PRECIO>300000				
↳ INSERVICIO_PK				
↳ Access Predicates				
↳ CS.SERVICIO_ID=S.ID				
↳ TABLE ACCESS	RESERVA	BY INDEX ROWID		
↳ INDEX	RESERVA_PK	UNIQUE SCAN		
↳ Access Predicates				
↳ R.ID=CS.RESERVA_ID				
↳ TABLE ACCESS	RESERVA	FULL		
↳ INDEX	RESERVA_PK	UNIQUE SCAN		
↳ Access Predicates				
↳ U.ID=R.USUARIO_ID				
↳ TABLE ACCESS	USUARIO	BY INDEX ROWID		
↳ TABLE ACCESS	USUARIO	FULL		
↳ HASH JOIN				
↳ Access Predicates				
↳ U.ID=SD.ID				
↳ NESTED LOOPS				
↳ NESTED LOOPS				
↳ PARTITION BY COLLECT				
↳ VIEW				
↳ HASH		UNIQUE		
↳ NESTED LOOPS				
↳ NESTED I				
↳ MERGE		CARTESIAN		
↳ NE		SEMI		
↳ RESERVASERVICIO		FULL		
↳ Filter Predicates				
↳ AND				
↳ RS.USUARIO_ID IS NOT NULL				
↳ RS.HORAINI-RS.HORAFIN>4				
↳ SERVICIO		BY INDEX ROWID		
↳ Filter Predicates				
↳ OR				
↳ AND				
↳ S.DESCRIPCION LIKE '%SPAN%'				
↳ S.DESCRIPCION IS NOT NULL				
↳ S.DESCRIPCION IS NOT NULL				
↳ AND				
↳ S.DESCRIPCION LIKE '%espanol%'				
↳ S.DESCRIPCION IS NOT NULL				
↳ S.DESCRIPCION IS NOT NULL				
↳ SERVICIO_PK		UNIQUE SCAN		
↳ Access Predicates				
↳ RS.SERVICIO_ID=S.ID				
↳ BULK COLLECT	RESERVA	SORT		
↳ INDEX	RESERVA_PK	FULL		
↳ Access Predicates				
↳ U.ID=R.USUARIO_ID				
↳ Filter Predicates				
↳ U.ID=R.USUARIO_ID				
↳ TABLE ACCESS	USUARIO	BY INDEX ROWID		
↳ INDEX	USUARIO_PK	UNIQUE SCAN		
↳ Access Predicates				
↳ U.ID=SD.ID				
↳ TABLE ACCESS	USUARIO	BY INDEX ROWID		
↳ TABLE ACCESS	USUARIO	FULL		

Figure 3: Plan Acción Req 12