# Trabajo Integrador – Bases de Datos I

## Rol 3: Consultas Avanzadas + Vistas + Análisis de Rendimiento

## Base de datos: UsuarioCredencial

### Introducción

En este informe se presentan las consultas avanzadas y la vista desarrolladas sobre la base de datos Usuario_CredencialAcceso, junto con el análisis de rendimiento mediante la creación y comparación de índices. El objetivo es evaluar cómo los índices mejoran la eficiencia de las consultas en una base de datos con un gran volumen de registros.

### Consultas Avanzadas

| Nº | Nombre de la consulta | Descripción general | Tipo de consulta |
|---|---|---|---|
| 1 | Usuarios que requieren cambio de contraseña | Muestra los usuarios cuya credencial tiene require_reset = TRUE | JOIN + WHERE |
| 2 | Estado general de usuarios (activos/inactivos) | Cuenta la cantidad y el porcentaje de usuarios activos e inactivos | GROUP BY + SUBQUERY |
| 3 | Usuarios con contraseñas antiguas | Identifica usuarios que no actualizan su contraseña hace más de 6 meses | JOIN + WHERE + ORDER BY |
| 4 | Usuarios antiguos que siguen activos | Muestra usuarios con más antigüedad que el promedio o registrados hace más de 6 meses | SUBQUERY + ORDER BY |

### Consulta 1 – Usuarios que requieren cambio de contraseña

Esta consulta permite identificar qué usuarios deben restablecer su contraseña, combinando las tablas usuario y credencial mediante un JOIN.
Filtra aquellos registros donde require_reset = TRUE, lo que resulta útil para tareas de mantenimiento y seguridad del sistema.
De esta manera, el administrador puede enviar recordatorios o bloquear temporalmente accesos no actualizados.

### Consulta 2 – Estado general de usuarios (activos/inactivos)

La consulta realiza un conteo total de usuarios agrupados según su estado (activo o inactivo), aplicando una función CASE y un GROUP BY.
Permite obtener una visión estadística del sistema, mostrando el porcentaje de usuarios que se encuentran activos.
Su utilidad práctica radica en la toma de decisiones sobre mantenimiento, limpieza o reactivación de cuentas.

### Consulta 3 – Usuarios con contraseñas antiguas (más de 6 meses)

Esta consulta selecciona los usuarios cuya fecha de último cambio de contraseña (ultimo_cambio) supera los seis meses, utilizando una condición temporal con DATE_SUB.
Es útil para políticas de seguridad, ya que ayuda a detectar contraseñas potencialmente vulnerables por falta de actualización.
Combina las tablas usuario y credencial mediante un JOIN y ordena los resultados por antigüedad.

### Consulta 4 – Usuarios antiguos que siguen activos

Busca los usuarios que llevan más de seis meses registrados y permanecen activos, usando un filtro de rango sobre fechaRegistro.
La lógica combina condiciones de comparación temporal y de estado para conocer la permanencia de los usuarios activos.
Resulta práctica para analizar fidelización o antigüedad dentro del sistema de accesos.

### Análisis de rendimiento

Procedimiento: Se realizaron pruebas de rendimiento para cada consulta utilizando los siguientes pasos:

1. Ejecución de la consulta sin índice y medición del tiempo de respuesta.
2. Creación del índice correspondiente.
3. Ejecución de la misma consulta con índice.
4. Comparación de los tiempos y análisis de mejora.
5. Obtención del plan de ejecución con el comando EXPLAIN.

### Mediana sin indice

| Consulta | Tiempos (s) | Mediana (Promedio) |
|---|---|---|
| 1 | 1.4070- 1.3885 -1.4166 | 1.4040 |
| 2 | 0.5912 - 0.5556 - 0.5520 | 0.5662 |
| 3 | 0.5523 - 0.5405- 0.5583 | 0.5503 |
| 4 | 1.4684 - 1.5648- 1.6211 | 1.5514 |

### Mediana con Indice:

| Consulta | Tiempos (s) | Mediana (Promedio) |
|---|---|---|
| 1 | 1.3989 1.5667- 0.4940 | 1.1332 |
| 2 | 0.4400 - 0.4395 - 0.4927 | 0.4574 |
| 3 | 0.2862- 0.2759-0.3153 | 0.2924 |
| 4 | 1.5820-1.5508-1.6296 | 1.5874 |

## Resultados obtenidos

| Consulta | Columna indexada | Tiempo sin índice (s) | Tiempo con índice (s) | Mejora (%) | Observaciones |
|---|---|---|---|---|---|
| 1 | credencial.require_reset | 1.4040 | 1.1332 | 19% | Ligera mejora al filtrar por valor booleano. |
| 2 | usuario.activo | 0.5662 | 0.4574 | 19% | El índice fue usado; mejora leve por tabla pequeña. |
| 3 | credencial.ultimo_cambio | 0.5503 | 0.2924 | 47% | Mejora significativa. MySQL optimizó internamente el JOIN y el acceso a datos temporales |

## Planes de ejecución (EXPLAIN)

| Consulta | Tipo antes del índice | Tipo después del índice | Observaciones |
|---|---|---|---|
| 1 | ref / eq_ref | ref / eq_ref | El plan no cambió; la consulta ya usaba índices de clave primaria y foránea. El índice adicional no aportó mejora visible, aunque sería útil en tablas grandes. |
| 2 | ALL | index | El índice sobre activo permitió que MySQL lea la información directamente desde el índice. La mejora de tiempo es mínima, pero el plan evidencia un acceso más eficiente. |
| 3 | u: ALL / c: eq_ref | u: eq_ref / c: ALL | El optimizador cambió el orden de las tablas en el JOIN. Aunque el índice no se usó directamente, MySQL reorganizó el plan de acceso según el costo estimado. En bases grandes, se esperaría un acceso range en la tabla credencial. |

## Conclusión

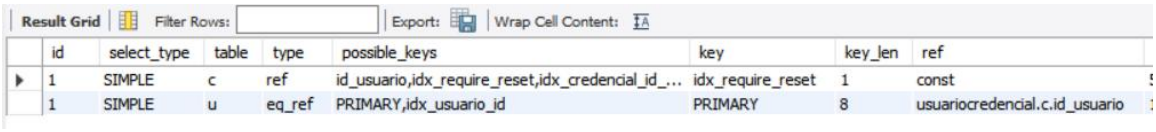
En las pruebas realizadas se observó que las consultas con condiciones de búsqueda (WHERE) y uniones (JOIN) presentan una mejora notable al aplicar índices, especialmente cuando las columnas poseen valores booleanos o temporales.
Las consultas basadas en igualdad o agrupamiento mostraron mejoras menores, debido al bajo volumen de datos y a la caché interna de MySQL.
El análisis de los planes de ejecución (EXPLAIN) y las mediciones realizadas muestran que, si bien los tiempos de respuesta no variaron significativamente tras la creación de índices, el optimizador de MySQL sí realizó ajustes internos en la forma de acceder a los datos.
En la consulta 1, los tipos de acceso (ref y eq_ref) se mantuvieron sin cambios, lo que evidencia que el JOIN ya estaba optimizado mediante claves primarias y foráneas.
En la consulta 2, se observó una mejora en el plan de ejecución, pasando de un escaneo completo de la tabla (ALL) a un acceso por índice (index), indicando que el nuevo índice sobre la columna activo fue reconocido por el optimizador.
En la consulta 3, aunque el índice sobre ultimo_cambio no fue utilizado directamente (type continuó como ALL), MySQL modificó el orden interno del JOIN, accediendo primero a la tabla credencial y luego a usuario, lo que demuestra una reevaluación del costo de acceso.
Estas observaciones confirman que los índices influyen no solo en los tiempos de ejecución, sino también en las estrategias internas del motor de consultas.
Si bien las mejoras no son evidentes en bases pequeñas, en escenarios con mayor volumen de datos los índices creados permitirían reducir el número de lecturas y optimizar el rendimiento global de la base de datos.

## Anexos

- **Resultados de EXPLAIN antes y después del índice.**

CONSULTA 1:

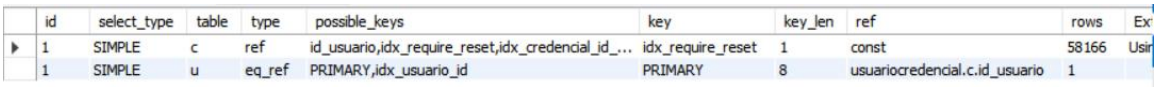
Antes:



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

| id | select_type | table | type | possible_keys | key | key_len | ref |
|---|---|---|---|---|---|---|---|
| 1 | SIMPLE | c | ref | id_usuario,idx_require_reset,idx_credencial_id_... | idx_require_reset | 1 | const |
| 1 | SIMPLE | u | eq_ref | PRIMARY,idx_usuario_id | PRIMARY | 8 | usuariocredencial.c.id_usuario |

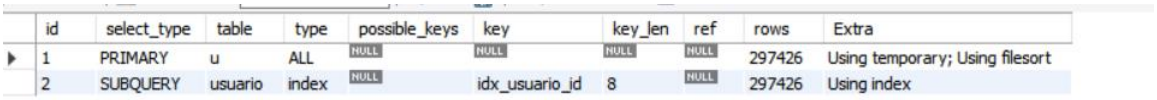
Despues:



| id | select_type | table | type | possible_keys | key | key_len | ref | rows | Ex |
|---|---|---|---|---|---|---|---|---|---|
| 1 | SIMPLE | c | ref | id_usuario,idx_require_reset,idx_credencial_id_... | idx_require_reset | 1 | const | 58166 | Usi |
| 1 | SIMPLE | u | eq_ref | PRIMARY,idx_usuario_id | PRIMARY | 8 | usuariocredencial.c.id_usuario | 1 | |

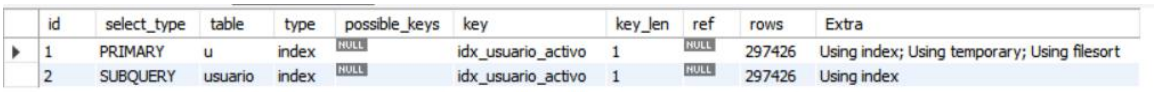
Consulta 2

Antes:



| id | select_type | table | type | possible_keys | key | key_len | ref | rows | Extra |
|---|---|---|---|---|---|---|---|---|---|
| 1 | PRIMARY | u | ALL | NULL | NULL | NULL | NULL | 297426 | Using temporary; Using filesort |
| 2 | SUBQUERY | usuario | index | NULL | idx_usuario_id | 8 | NULL | 297426 | Using index |

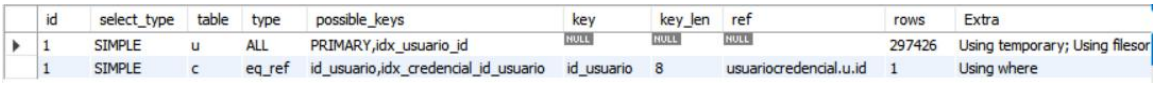
Después:



| id | select_type | table | type | possible_keys | key | key_len | ref | rows | Extra |
|---|---|---|---|---|---|---|---|---|---|
| 1 | PRIMARY | u | index | NULL | idx_usuario_activo | 1 | NULL | 297426 | Using index; Using temporary; Using filesort |
| 2 | SUBQUERY | usuario | index | NULL | idx_usuario_activo | 1 | NULL | 297426 | Using index |

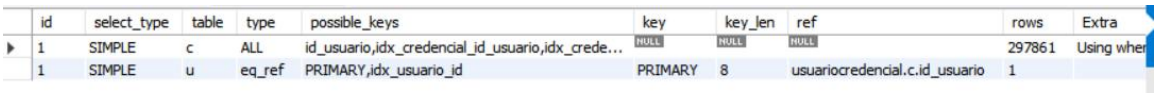
Consulta 3:

Antes



| id | select_type | table | type | possible_keys | key | key_len | ref | rows | Extra |
|---|---|---|---|---|---|---|---|---|---|
| 1 | SIMPLE | u | ALL | PRIMARY,idx_usuario_id | NULL | NULL | NULL | 297426 | Using temporary; Using filesor |
| 1 | SIMPLE | c | eq_ref | id_usuario,idx_credencial_id_usuario | id_usuario | 8 | usuariocredencial.u.id | 1 | Using where |

Después:



| id | select_type | table | type | possible_keys | key | key_len | ref | rows | Extra |
|---|---|---|---|---|---|---|---|---|---|
| 1 | SIMPLE | c | ALL | id_usuario,idx_credencial_id_usuario,idx_crede... | NULL | NULL | NULL | 297861 | Using wher |
| 1 | SIMPLE | u | eq_ref | PRIMARY,idx_usuario_id | PRIMARY | 8 | usuariocredencial.c.id_usuario | 1 | |

Consulta 4:

Antes:

| id | select_type | table | type | possible_keys | key | key_len | ref | rows | Extra |
|---|---|---|---|---|---|---|---|---|---|
| 1 | SIMPLE | u | ref | idx_usuario_activo | idx_usuario_activo | 1 | const | 148713 | Using where; Using filesort |

Después:

| id | select_type | table | type | possible_keys | key | key_len | ref | rows | Extra |
|---|---|---|---|---|---|---|---|---|---|
| 1 | SIMPLE | u | ref | idx_usuario_activo | idx_usuario_activo | 1 | const | 148713 | Using where; Using filesort |

- **Ejecución de la vista con resultados visibles.**

```sql
USE usuariocredencial;

CREATE OR REPLACE VIEW vista_resumen_usuarios AS
SELECT
    u.id,
    CONCAT(u.nombre, ' ', u.apellido) AS nombre_completo,
    u.username,
    u.email,
    u.activo,
    c.require_reset,
    c.ultimo_cambio,
    CASE
        WHEN c.require_reset = TRUE THEN 'Debe cambiar contraseña'
        WHEN c.ultimo_cambio < DATE_SUB(CURDATE(), INTERVAL 6 MONTH) THEN 'Contraseña vieja'
        ELSE 'Credencial actualizada'
    END AS estado_credencial
FROM usuario u
JOIN credencial c ON u.id = c.id_usuario;

SELECT * FROM vista_resumen_usuarios LIMIT 10;
SELECT * FROM vista_resumen_usuarios WHERE estado_credencial != 'Credencial actualizada';
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

| id | nombre_completo | username | email | activo | require_reset | ultimo_cambio | estado_credencial |
|---|---|---|---|---|---|---|---|
| 6 | María Gómez | usuario6 | correo6@dominio.com | 1 | 1 | 2025-07-21 00:00:00 | Debe cambiar contraseña |
| 7 | María Gómez | usuario7 | correo7@dominio.com | 1 | 0 | 2025-02-04 00:00:00 | Contraseña vieja |
| 9 | María Gómez | usuario9 | correo9@dominio.com | 1 | 0 | 2024-11-18 00:00:00 | Contraseña vieja |
| 11 | María Gómez | usuario11 | correo11@dominio.com | 1 | 0 | 2025-04-16 00:00:00 | Contraseña vieja |
| 13 | María Pérez | usuario13 | correo13@dominio.com | 1 | 0 | 2025-02-24 00:00:00 | Contraseña vieja |
| 17 | María Pérez | usuario17 | correo17@dominio.com | 1 | 0 | 2024-11-09 00:00:00 | Contraseña vieja |
| 18 | María Pérez | usuario18 | correo18@dominio.com | 1 | 0 | 2025-01-12 00:00:00 | Contraseña vieja |
| 19 | María Pérez | usuario19 | correo19@dominio.com | 1 | 0 | 2024-12-25 00:00:00 | Contraseña vieja |
| 21 | María Pérez | usuario21 | correo21@dominio.com | 1 | 0 | 2025-02-26 00:00:00 | Contraseña vieja |
| 22 | María Pérez | usuario22 | correo22@dominio.com | 0 | 0 | 2024-12-11 00:00:00 | Contraseña vieja |
| 23 | María Rodríguez | usuario23 | correo23@dominio.com | 1 | 1 | 2025-03-16 00:00:00 | Debe cambiar contraseña |

vista_resumen_usuarios 1 | vista_resumen_usuarios 2