a mayoría de nosotros somos consumidores y generadores de datos en distintas actividades cotidianas: al comunicarnos por celular, al reservar un pasaje o al comprar en el supermercado. En nuestra vida diaria llevamos a cabo actividades que requieren acceso a bases de datos.

La inestabilidad de la base de datos puede hacer que el sistema se comporte de forma no deseada. Aparte de todo, si la base de datos se bloquea, el sistema es totalmente inútil.

Asi que la prueba de base de datos es útil para descubrir tales vulnerabilidades en la construcción de la base de datos. Por qué y cuánto es importante lo veremos en este artículo junto con algunas herramientas para automatizar este proceso.

No importa si se trata de una web, escritorio o móvil, cliente-servidor, peer-to-peer, empresa o negocio individual; la base de datos es necesaria en todas partes en el backend.

A medida que aumenta la complejidad de la aplicación, surge la necesidad de una base de datos más sólida y segura. De la misma forma, para las aplicaciones con alta frecuencia de transacciones (Por ejemplo, Aplicación bancaria o financiera), se combina la necesidad de una herramienta de base de datos con todas las funciones.

Hay varios Herramientas de base de datos que están disponibles en el mercado Por ejemplo, MS-Access, MS SQL Server, SQL Server, Oracle, Oracle Financial, MySQL, PostgreSQL, DB2, Toad, Admirer, etc. Estas herramientas varían en costo, solidez, características y seguridad. Cada uno de estos tiene sus propios beneficios e inconvenientes.



¿Porque debemos probar las Bases de datos?

Debemos hacerlo por 4 muy importantes razones:

1. **Mapeo de datos**

En los sistemas de software, los datos a menudo viajan hacia adelante y hacia atrás desde la UI (interfaz de usuario) a la base de datos de back-end y viceversa.

Siempre que se realiza una determinada acción en el front-end de una aplicación, se invoca una acción CRUD (Crear, Recuperar, Actualizar y Eliminar) correspondiente en el back-end. Un tester tendrá que verificar si se invoca la acción correcta y si la acción invocada en sí misma es exitosa o no

      2. Validación de propiedades ACID

Atomicidad, consistencia, aislamiento y durabilidad. Cada transacción que realiza una base de datos debe cumplir estas cuatro propiedades.



* **Atomicidad**significa que una transacción falla o pasa. Esto significa que incluso si una sola parte de la transacción falla, significa que toda la transacción ha fallado. Por lo general, esto se llama la regla de 'todo o nada'.
* **Consistencia :** Una transacción siempre dará como resultado un estado válido de la base de datos
* **Aislamiento :** Si hay varias transacciones y se ejecutan todas a la vez, el resultado / estado de la base de datos debería ser el mismo que si se ejecutaran una tras otra.
* **Durabilidad :** Una vez que se realiza y se confirma una transacción, ningún factor externo como la pérdida de energía o el bloqueo deberían poder cambiarla

**3.Integridad de los datos**

Para cualquiera de los Operaciones CRUD , los valores / estado actualizados y más recientes de los datos compartidos deben aparecer en todos los formularios y pantallas. El valor no debe actualizarse en una pantalla y mostrar un valor anterior en otra.

Cuando la aplicación está en ejecución, el usuario final utiliza principalmente las operaciones 'CRUD' facilitadas por la herramienta de base de datos .

**C: Crear** - Cuando el usuario 'Guarda' cualquier transacción nueva, se realiza la operación 'Crear'.

**R: recupera**r - Cuando el usuario 'Buscar' o 'Ver' cualquier transacción guardada, se realiza la operación 'Recuperar'.

**U: actualización** - Cuando el usuario 'Edita' o 'Modifica' un registro existente, se realiza la operación de 'Actualización' de la base de datos.

**D: Eliminar**- Cuando un usuario 'Elimina' cualquier registro del sistema, se realiza la operación 'Eliminar' de la base de datos.

La recomendación es que diseñe sus casos de prueba de base de datos de manera que incluyan la verificación de los datos en todos los lugares en los que parece para ver si son consistentemente los mismos.

**4. Conformidad con las reglas comerciales**

Una mayor complejidad en las bases de datos significa componentes más complicados como restricciones relacionales, disparadores, procedimientos almacenados, etc. Por lo tanto, los probadores tendrán que generar consultas SQL apropiadas para validar estos objetos complejos.

¿Que Debemos probar de las Bases de Datos?

1. Transacciones

**Estas son las declaraciones de uso común:**

COMENZAR TRANSACCIÓN TRANSACCIÓN #  
FIN DE LA TRANSACCIÓN TRANSACCIÓN #  
La declaración Rollback asegura que la base de datos permanece en un estado consistente.  
  
**TRANSACCIÓN ROLLBACK #**  
Después de que se ejecuten estas declaraciones, use un Seleccionar para asegurarse de que los cambios se hayan reflejado.  
  
**SELECCIONAR \* DE TABLENAME**

  2.Esquemas de Bases de Datos

Identificar los Requisitos en base a los cuales opera la Base de Datos.

**Requisitos de muestra:**

* Las claves primarias se crearán antes de que se creen otros campos.
* Las claves externas deben estar completamente indexadas para una fácil recuperación y búsqueda.
* Nombres de campo que comienzan o terminan con ciertos caracteres.
* Campos con la restricción de que ciertos valores se pueden insertar o no.
* Utilice uno de los siguientes métodos según la relevancia:
* Consulta SQL DESC para validar el esquema.

3.**Triggers**

Cuando un evento determinado tiene lugar en una mesa determinada, se puede auto-instruir un fragmento de código (un disparador) para que se ejecute.

Por ejemplo, un nuevo estudiante se unió a una escuela. El estudiante está tomando 2 clases: matemáticas y ciencias. El estudiante se agrega a la 'tabla de estudiantes'. Un activador podría agregar al estudiante a las tablas de materias correspondientes una vez que se agregue a la tabla de estudiantes.

El método común para probar es ejecutar la consulta SQL incrustada en el Trigger de forma independiente primero y registrar el resultado. Continúe con la ejecución del Trigger como un todo.

**4.Procedimientos Almacenados**

Los procedimientos almacenados son más o menos similares a las funciones definidas por el usuario. Estos pueden ser invocados por declaraciones de procedimiento de llamada / procedimiento de ejecución y la salida suele estar en forma de conjuntos de resultados.

**5.Restricciones de Campo**

El valor predeterminado, el valor único y la clave externa:  
Realizar una operación de front-end que ejercite la condición del objeto de la base de datos  
Valide los resultados con una consulta SQL.  
Verificar el valor predeterminado para un determinado campo es bastante simple. Es parte de la validación de reglas comerciales.

**Pasos para preparar nuestro ambiente de pruebas**

1. Ejecutar una prueba
2. Compruebe el resultado de la prueba
3. Validar según los resultados esperados
4. Informar los hallazgos a las respectivas partes interesadas

**Que se recomienda:**

* Escriba consultas usted mismo
* Observe los datos en cada tabla:
* Obtenga consultas de los desarrolladores:
* Utilice las herramientas de prueba

**Practica los comandos basicos para la escritura de Scripts:**

* SELECT
* ALTER
* ORDER BY
* FROM