DB Testing

DB Testing - для чего тестировщик обращается к базе данных?

Тестирование базы данных – Обзор

Тестирование базы данных включает в себя выполнение проверки достоверности данных, проверку целостности данных, проверку производительности, связанную с базой данных, и тестирование процедур, триггеров и функций в базе данных.

Пример

Рассмотрим приложение, которое фиксирует сведения о ежедневных транзакциях пользователей и сохраняет их в базе данных. С точки зрения тестирования базы данных необходимо выполнить следующие проверки −

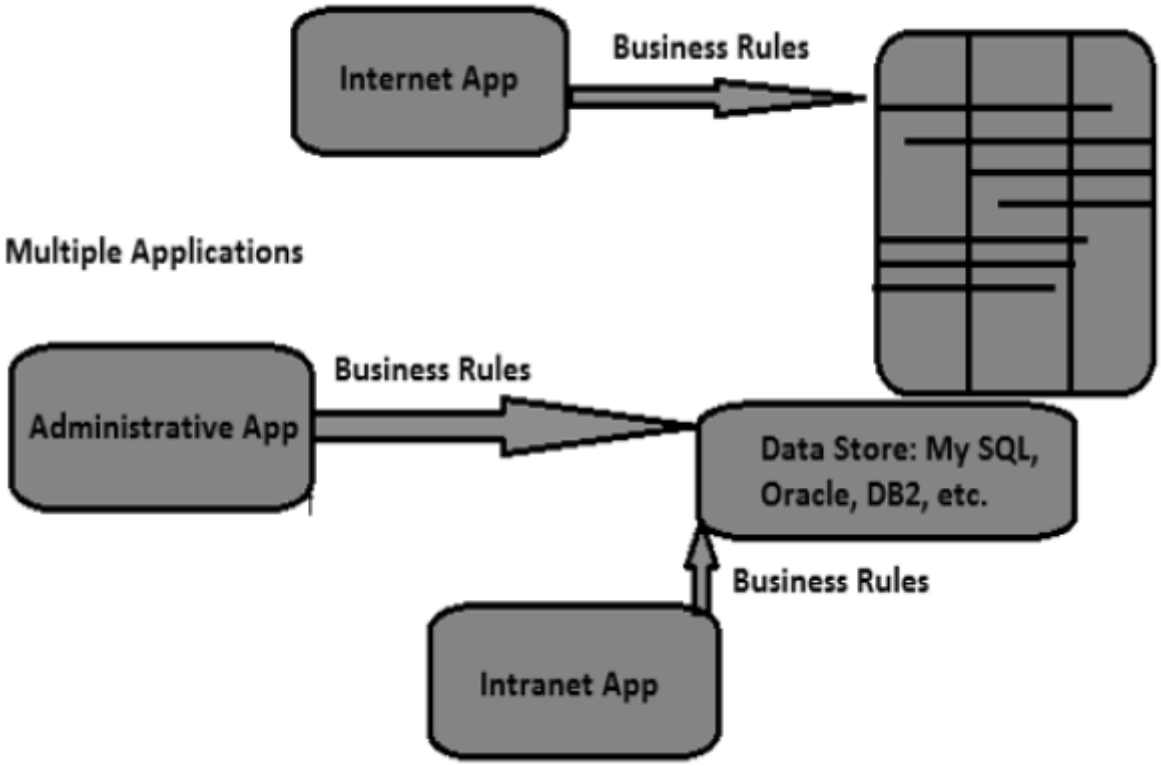
- Информация о транзакциях из приложения должна храниться в базе данных, и она должна предоставлять пользователю правильную информацию.
- Информация не должна быть потеряна при загрузке в базу данных.
- Должны храниться только завершенные транзакции, а все незавершенные операции должны быть прерваны приложением.
- Необходимо поддерживать авторизацию доступа к базе данных. Не должен предоставляться несанкционированный или несанкционированный доступ к пользовательской информации.

Зачем вам нужно выполнять тестирование базы данных?

Существует несколько причин, по которым выполняется тестирование базы данных. Необходимо выполнить проверку целостности, проверки и согласованности данных в базе данных, поскольку серверная система отвечает за хранение данных, и доступ к ней осуществляется для нескольких целей.

Ниже приведены некоторые общие причины для тестирования базы данных −

- Чтобы упростить вызовы серверной части базы данных, разработчики расширяют использование **представлений** и **хранимых** процедур.
- Эти **хранимые** процедуры и **представления** содержат важные задачи, такие как вставка сведений о клиентах (имя, контактная информация и т.д.) и данных о продажах. Эти задачи необходимо протестировать на нескольких уровнях.
- **Тестирование черного ящика**, выполняемое во внешнем интерфейсе, важно, но затрудняет локализацию проблемы. Тестирование в серверной системе повышает надежность данных. Именно поэтому тестирование базы данных выполняется на серверной системе.
- В базе данных данные поступают из нескольких приложений, и существует вероятность того, что в базе данных хранятся вредоносные или неправильные данные. Поэтому необходимо регулярно проверять компоненты базы данных. Кроме того, следует регулярно проверять целостность и согласованность данных.



Тестирование базы данных против интерфейсного тестирования

Тестирование базы данных отличается от тестирования пользовательского интерфейса.

Тестирование базы данных	Тестирование пользовательского интерфейса
Тестирование базы данных известно как проверка достоверности и целостности данных или внутреннее тестирование.	Тестирование пользовательского интерфейса или интерфейсное тестирование также называется тестированием приложений или тестированием графического интерфейса.
Тестирование базы данных включает в себя тестирование внутренних компонентов, которые не видны пользователям. Сюда входят компоненты базы данных и системы СУБД, такие как My SQL, Oracle.	Тестирование пользовательского интерфейса включает в себя проверку функциональных возможностей приложения и его компонентов, таких как формы, графики, меню, отчеты и т.д. Эти компоненты создаются с использованием интерфейсных инструментов разработки, таких как VB.net , C #, Delphi и т.д.
Тестирование базы данных включает в себя проверку хранимых процедур, представлений, схем в базе данных, таблиц, индексов, ключей, триггеров, проверки данных и проверку согласованности данных.	Тестирование пользовательского интерфейса включает в себя проверку функциональности приложения, кнопок, форм и полей, календаря и изображений, навигации с одной страницы на другую и общей функциональности приложения.
Для выполнения тестирования БД тестировщику необходимо досконально знать концепцию базы данных, такие как процедуры и функции, представления, индексы, ключи и хорошее практическое владение SQL.	Для выполнения тестирования пользовательского интерфейса тестировщику необходимо хорошее понимание бизнес-требований, функциональных знаний приложений, программирования и т. Д.
Данные поступают из нескольких разнородных источников данных через веб-приложения, приложения интрасети и различные другие приложения.	Данные вводятся в приложения вручную. Оно включает в себя функциональное тестирование интерфейсных приложений.

Тестирование базы данных – Типы

В зависимости от функции и структуры базы данных тестирование БД можно разделить на три категории−

- **Структурное тестирование базы** данных − оно касается тестирования таблиц и столбцов, тестирования схемы, тестирования хранимых процедур и представлений, проверки триггеров и т. Д.
- **Функциональное тестирование** − включает в себя проверку функциональности базы данных с точки зрения пользователя. Наиболее распространенным типом функционального тестирования являются тестирование "белого ящика" и "черного ящика".

- **Нефункциональное тестирование** − включает в себя нагрузочное тестирование, тестирование рисков в базе данных, стресс-тестирование, минимальные системные требования и касается производительности базы данных.

Структурное тестирование базы данных

Структурное тестирование базы данных включает проверку тех компонентов базы данных, которые не доступны конечным пользователям. Оно включает в себя все компоненты репозитория, которые используются для хранения данных и не изменяются конечными пользователями. Администраторы баз данных, хорошо владеющие хранимыми процедурами SQL и другими концепциями, обычно выполняют это тестирование.

Обсуждаются общие компоненты, протестированные в отношении структурного тестирования −

Тестирование схемы / сопоставления

Оно включает в себя проверку объектов интерфейсного приложения с помощью сопоставления объектов базы данных.

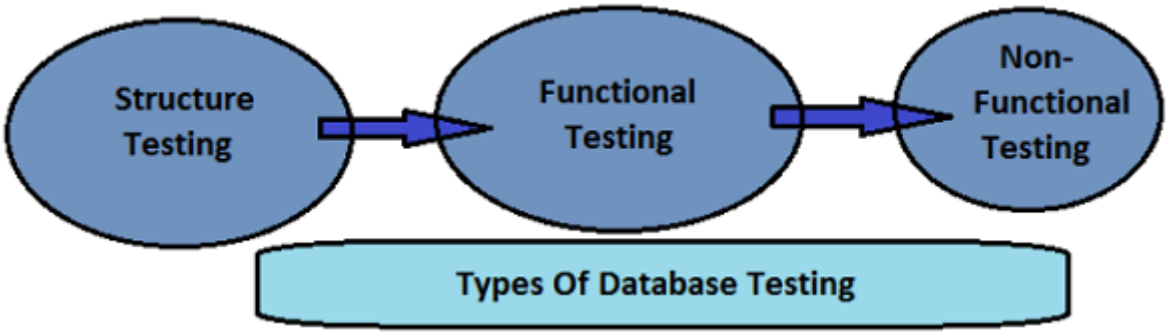
В тестировании схемы −

- Иногда случается, что объекты приложения конечного пользователя неправильно сопоставлены или несовместимы с объектами базы данных. Поэтому требуется проверка правильности различных форматов схем, связанных с базами данных.
- Требуется найти не сопоставленные объекты в базе данных, такие как таблицы, представления, столбцы и т. Д. Требуется.

На рынке существуют различные инструменты, которые можно использовать для выполнения сопоставления объектов в схемах.

Пример − В Microsoft SQL Server тестировщик может писать простые запросы для проверки и проверки схем в базе данных.

Если тестировщик хочет внести изменения в структуру таблицы, он / она должен убедиться, что все **хранимые** процедуры, имеющие эту таблицу, совместимы с этим изменением.



Тестирование хранимых процедур и представлений

В этом тестировании тестировщик гарантирует, что ручное выполнение хранимых процедур и представлений приведет к требуемому результату.

Тестировщик обеспечивает −

- Если это позволяет выполнять требуемые триггеры, как ожидалось.
- Если команда разработчиков выполнила все циклы и условия, передав входные данные приложениям в процедурах.
- Если в базе данных есть какие-либо неиспользуемые хранимые процедуры.
- Операции обрезки применяются правильно, когда данные извлекаются из требуемых таблиц в базе данных.
- Проверка общей интеграции модулей хранимых процедур в соответствии с требованиями тестируемого приложения.
- Используются механизмы обработки исключений и ошибок.

Наиболее распространенными инструментами, используемыми для тестирования хранимых процедур, являются **LINQ**, **SP Test tool** и др.

Тестирование триггеров

При тестировании триггеров тестировщик должен обеспечить следующее −

- Соблюдаются ли соглашения о кодировании на этапе кодирования триггеров.
- Посмотрите, выполняются ли триггеры, соответствующие требуемым условиям.
- Правильно ли триггер обновляет данные после их выполнения.
- Проверка функциональности триггеров обновления / вставки / удаления в тестируемом приложении.

Тестирование таблиц и столбцов

Ключевыми областями, рассматриваемыми в этом тестировании, являются−

- Проверка типов данных в базе данных на соответствие значениям полей во интерфейсном приложении.
- Проверка соответствия длины поля данных в базе данных длине типов данных в приложении.
- Проверка наличия в базе данных каких-либо не сопоставленных таблиц или столбцов из объектов поля приложения.
- Соглашения об именовании таблиц и столбцов базы данных проверяются, соответствуют ли они бизнес-требованиям или нет.
- Проверка ключей и индексов в базе данных, т.Е. Первичные и внешние ключи в таблицах определяются в соответствии с требованиями.
- Проверьте, совпадают ли первичные ключи и соответствующие им внешние ключи в двух таблицах.
- Убедитесь, что сохраняются уникальные и ненулевые характеристики ключей.
- Длина и тип данных ключей и индексов поддерживаются в соответствии с требованиями.

Проверка сервера базы данных

Проверка сервера базы данных включает проверку −

- Если сервер базы данных может обрабатывать ожидаемое количество транзакций в соответствии с бизнес-требованиями.
- Если сведения о конфигурации серверов баз данных соответствуют бизнес-требованиям.
- Если авторизация пользователя поддерживается в соответствии с требованиями.

Функциональное тестирование

Функциональное тестирование выполняется с учетом точки зрения конечного пользователя; соответствуют ли требуемые транзакции и операции, выполняемые конечными пользователями, бизнес-спецификациям.

Тестирование черного ящика

Тестирование черного ящика включает проверку интеграции базы данных для проверки функциональности. Тестовые примеры просты и используются для проверки входящих и исходящих данных из функции.

Для тестирования функциональности базы данных используются различные методы, такие как метод построения графиков причинно-следственных связей, разделение эквивалентности и анализ граничных значений.

Его **преимущества** заключаются в следующем −

- Оно довольно простое и выполняется на ранних стадиях разработки.
- Стоимость разработки тест-кейсов меньше по сравнению с тестированием "белого ящика".

Его недостатки заключаются в следующем −

- Не удается обнаружить несколько ошибок
- Неизвестно, сколько программ нужно протестировать.

Тестирование белого ящика

Тестирование белого ящика касается внутренней структуры базы данных, а детали спецификации скрыты от пользователей. Оно включает в себя тестирование триггеров базы данных и логических представлений, которые будут поддерживать рефакторинг базы данных.

Он выполняет модульное тестирование функций базы данных, триггеров, представлений, SQL-запросов и т. Д. Этот тип тестирования проверяет таблицы базы данных, модели данных, схему базы данных и т. Д. Он проверяет правила ссылочной целостности. Он выбирает значения таблицы по умолчанию для проверки согласованности базы данных.

Наиболее распространенными методами, используемыми для выполнения тестирования "белого ящика", являются покрытие условий, покрытие решений, покрытие утверждений и т. Д.

Ошибки кодирования могут быть обнаружены при тестировании "белого ящика", поэтому внутренние ошибки в базе данных могут быть устранены. Ограничение тестирования белого ящика заключается в том, что инструкции SQL не покрываются.

Нефункциональное тестирование

Нефункциональное тестирование включает в себя выполнение нагрузочного тестирования, стресс-тестирования, проверку минимальных системных требований на соответствие бизнес-спецификациям, выявление рисков и оптимизацию производительности базы данных.

Нагрузочное тестирование

Основная цель нагрузочного тестирования - проверить, влияет ли большинство выполняемых транзакций на производительность базы данных.

При нагрузочном тестировании тестировщик проверяет −

- Время отклика для выполнения транзакций для нескольких удаленных пользователей.
- Время, затрачиваемое базой данных на выборку определенных записей.

Примеры нагрузочного тестирования в различных типах тестирования −

- Повторный запуск наиболее часто используемой транзакции для оценки производительности системы баз данных.
- Загрузка серии больших файлов из Интернета.
- Одновременный запуск нескольких приложений на компьютере или сервере.

Стресс-тестирование

Стресс-тестирование выполняется для определения точки останова системы. В этом тестировании приложение загружается таким образом, что в какой-то момент система выходит из строя. Эта точка называется точкой **останова** системы баз данных.

Определение состояния транзакций базы данных требует значительных усилий. Требуется правильное планирование, чтобы избежать любых временных и финансовых проблем.

Наиболее часто используемыми инструментами стресс-тестирования являются **LoadRunner** и **WinRunner**.

Давайте рассмотрим **пример** стресс-тестирования. Приложение CRM может принимать максимальную пользовательскую нагрузку в 50000 одновременных пользователей. Предположим, вы увеличиваете нагрузку до 51000 и выполняете некоторые транзакции, такие как обновление записей или добавление записи. Как только вы выполняете транзакцию, приложение может

синхронизироваться с системой баз данных. Итак, следующий тест должен выполняться с пользовательской нагрузкой 52000. Иногда стресс-**тестирование также называют испытанием на усталость**.

Тестирование базы данных – Процессы

Процесс выполнения тестирования базы данных аналогичен тестированию других приложений. Тестирование базы данных можно описать с помощью ключевых процессов, приведенных ниже.

- Настройка среды
- Запустите тест
- Проверьте результат теста
- Проверка в соответствии с ожидаемыми результатами
- Сообщите о результатах соответствующим заинтересованным сторонам

Для разработки тестовых примеров используются различные инструкции SQL. Наиболее распространенным оператором SQL, который используется для выполнения тестирования БД, является оператор **Select** . Помимо этого, также могут использоваться различные инструкции DDL, DML, DCL.

Пример − Создание, вставка, выбор, обновление и т.д.

Этапы тестирования базы данных

Тестирование БД не является утомительным процессом и включает в себя различные этапы жизненного цикла тестирования базы данных в соответствии с процессами тестирования.

Ключевыми этапами тестирования базы данных являются−

- Проверка начального состояния
- Тестовый запуск
- Проверка результатов в соответствии с ожидаемым результатом
- Генерация результатов

Первым этапом тестирования БД является проверка начального состояния базы данных перед началом процесса тестирования. Затем поведение базы данных проверяется для определенных тестовых случаев. В соответствии с полученными результатами настраиваются тестовые примеры.

Для успешного тестирования базы данных при каждом отдельном тестировании выполняется приведенный ниже рабочий процесс.

- **Очистка базы** данных − Если в базе данных есть проверяемые данные, их следует очистить.
- **Настройка устройства** − это включает в себя ввод данных в базу данных и проверку текущего состояния базы данных.
- **Выполните тестирование, проверьте результаты и сгенерируйте результаты** − тест выполняется, и выходные данные проверяются. Если результат соответствует ожидаемым результатам, следующий шаг - сгенерировать результаты в соответствии с требованиями. В противном случае тестирование повторяется для поиска ошибок в базе данных.

Тестирование базы данных – Методы

В этой главе описываются наиболее распространенные методы, которые используются для выполнения тестирования базы данных.

Тестирование схемы базы данных

Как упоминалось ранее, оно включает в себя тестирование каждого объекта в схеме.

Проверка баз данных и устройств

- Проверка имени базы данных
- Проверка устройства данных, устройства регистрации и устройства дампа
- Проверка, достаточно ли места выделено для каждой базы данных
- Проверка настройки параметров базы данных

Проверка правил таблиц, столбцов, типов столбцов

Проверьте приведенные ниже пункты, чтобы выяснить различия между фактическими и применяемыми настройками.

- Имя всех таблиц в базе данных
- Имена столбцов для каждой таблицы
- Типы столбцов для каждой таблицы

- **Нулевое** значение проверено или нет
- Привязано ли значение по умолчанию к правильным столбцам таблицы
- Определения правил для исправления имен таблиц и прав доступа

Ключ и индексы

Проверьте ключ и индексы в каждой таблице −

- Первичный ключ для каждой таблицы
- Внешние ключи для каждой таблицы
- Типы данных между столбцом внешнего ключа и столбцом в других табличных индексах, кластеризованные или некластеризованные, уникальные или не уникальные

Тесты хранимых процедур

Оно включает в себя проверку того, определена ли хранимая процедура, и сравнение выходных результатов. При тестировании хранимой процедуры проверяются следующие пункты −

- Имя хранимой процедуры
- Имена параметров, типы параметров и т.д.
- **Вывод** − содержит ли вывод много записей. Выполняется удаление нулевых строк или извлекается только несколько записей.
- Какова функция хранимой процедуры и что хранимая процедура не должна делать?
- Передача примеров входных запросов для проверки, извлекает ли хранимая процедура правильные данные.
- **Параметры хранимой процедуры** − вызов хранимой процедуры с граничными данными и с действительными данными. Сделайте каждый параметр недействительным один раз и запустите процедуру.
- **Возвращаемые значения** − проверьте значения, возвращаемые хранимой процедурой. В случае сбоя должно быть возвращено значение, отличное от нуля.
- **Проверка сообщений об ошибках** − внесите изменения таким образом, чтобы хранимая процедура завершилась с ошибкой, и генерируйте каждое

сообщение об ошибке по крайней мере один раз. Проверьте все сценарии исключений, если нет предопределенного сообщения об ошибке.

Триггерные тесты

При тестировании триггера тестировщик должен выполнить следующие задачи –

- Убедитесь, что имя триггера указано правильно.
- Проверьте триггер, если он сгенерирован для определенного столбца таблицы.
- Проверка обновления триггера.
- Обновите запись действительными данными.
- Обновите запись с неверными данными и устраните все ошибки запуска.
- Обновите запись, если на нее все еще ссылается строка в другой таблице.
- Обеспечьте откат транзакций при возникновении сбоя.
- Выясните все случаи, в которых триггер не должен откатывать транзакции.

Сценарии настройки сервера

Необходимо выполнить два типа тестов −

- Настройка базы данных с нуля и
- Для настройки существующей базы данных.

Интеграционные тесты SQL Server

Интеграционные тесты следует выполнять после завершения тестирования компонентов.

- Хранимые процедуры должны вызываться интенсивно для выбора, вставки, обновления и удаления записей в разных таблицах, чтобы обнаружить любые конфликты и несовместимость.
- Любые конфликты между схемой и триггерами.
- Любые конфликты между хранимыми процедурами и схемой.
- Любые конфликты между хранимыми процедурами и триггерами.

Метод функционального тестирования

Функциональное тестирование может быть выполнено путем разделения базы данных на модули в соответствии с функциональностью. Функциональные возможности бывают следующих двух типов −

- **Тип 1** − В тестировании типа 1 узнайте особенности проекта. Для каждой основной функции найдите схему, триггеры и хранимые процедуры, отвечающие за реализацию этой функции, и поместите их в функциональную группу. Затем протестируйте каждую группу вместе.
- **Тип 2** − При тестировании типа 2 границы функциональных групп в серверной части не очевидны. Вы можете проверить поток данных и посмотреть, где вы можете проверить данные. Начните с интерфейса.

Выполняется следующий процесс −

- Когда служба получает запрос или сохраняет данные, вызываются некоторые хранимые процедуры.
- Процедуры обновят некоторые таблицы.
- Эти хранимые процедуры будут местом для начала тестирования, а эти таблицы будут местом для проверки результатов тестирования.

Стресс-тестирование

Стресс-тестирование включает в себя получение списка основных функций базы данных и соответствующих хранимых процедур. Выполните приведенные ниже шаги для стресс−тестирования -

- Напишите тестовые сценарии для тестирования этих функций, и каждая функция должна быть проверена хотя бы один раз за полный цикл.
- Выполняйте тестовые сценарии снова и снова в течение определенного периода времени.
- Проверка файлов журнала на наличие взаимоблокировок, сбоев в памяти, повреждения данных и т.д.

Контрольное тестирование

Если в вашей базе данных нет проблем с данными или ошибок, можно проверить производительность системы. Низкая производительность системы

может быть обнаружена при тестировании тестов, проверив параметры, приведенные ниже −

- Производительность на уровне системы
- Определите наиболее часто используемые функции / функции
- Сроки – максимальное время, минимальное время и среднее время выполнения функций
- Объем доступа

Тестирование базы данных с помощью интерфейса

Внутренние ошибки также иногда можно обнаружить, выполняя интерфейсное тестирование. Вы можете выполнить простые шаги, приведенные ниже, чтобы обнаружить ошибки с помощью интерфейсного тестирования.

- Напишите запросы из интерфейса и выполните поиск.
- Выберите существующую запись, измените значения в некоторых полях и сохраните запись. (Оно включает в себя инструкцию UPDATE или хранимые процедуры update и триггеры update.)
- Вставьте новый пункт меню в интерфейсное окно. Заполните информацию и сохраните запись. (Оно включает в себя инструкции INSERT или хранимые процедуры вставки и триггеры удаления.)
- Выберите существующую запись, нажмите на кнопку УДАЛИТЬ или УДАЛИТЬ и подтвердите удаление. (Оно включает в себя инструкцию DELETE или хранимые процедуры удаления и триггеры удаления.)
- Повторите эти тестовые примеры с неверными данными и посмотрите, как реагирует база данных.

Тестирование базы данных – Сценарии

В этой главе мы рассмотрим некоторые распространенные сценарии тестирования базы данных в отношении различных методов тестирования.

Тестирование структурированной базы данных

Ниже приведены общие сценарии тестирования структурированных баз данных –

- Проверка имени базы данных, проверка устройства передачи данных, устройства ведения журнала и устройства дампа, проверка, достаточно ли места выделено для каждой базы данных, и проверка настройки параметров базы данных.
- Имена всех таблиц в базе данных, имена столбцов для каждой таблицы, типы столбцов для каждой таблицы, проверка нулевого значения или нет. Проверьте ключ и индексы в каждой таблице: первичный ключ для каждой таблицы, внешние ключи для каждой таблицы.
- Типы данных между столбцом внешнего ключа и столбцом в других табличных индексах, кластеризованные или некластеризованные, уникальные или не уникальные.

Функциональное тестирование базы данных

Распространенными сценариями тестирования базы данных в отношении **функционального тестирования базы** данных являются:

- Найдите схему, триггеры и хранимые процедуры, отвечающие за реализацию этой функции, и объедините их в функциональную группу, а затем каждую группу можно протестировать вместе.
- Проверьте поток данных и посмотрите, где вы можете проверить данные. Начните с интерфейса.

Тестирование нефункциональной базы данных

Распространенными сценариями тестирования базы данных в отношении **тестирования нефункциональной базы** данных являются–

- Напишите тестовые сценарии для тестирования основных функций, и каждая функция должна быть проверена хотя бы один раз за полный цикл.
- Выполняйте тестовые сценарии снова и снова в течение определенного периода времени.
- Проверка файлов журнала на наличие взаимоблокировок, сбоев из-за нехватки памяти, повреждения данных и т. Д.
- Пишите запросы из внешнего интерфейса и выполняйте поисковые запросы. Выберите существующую запись, измените значения в некоторых

полях и сохраните запись. (Это включает в себя инструкцию ОБНОВЛЕНИЯ или хранимые процедуры обновления, триггеры обновления.)

- Вставьте новый пункт меню в интерфейсное окно. Заполните информацию и сохраните запись. (Это включает инструкции INSERT или хранимые процедуры вставки, триггеры удаления.)
- Выберите существующую запись, нажмите на кнопку УДАЛИТЬ или УДАЛИТЬ и подтвердите удаление. (Это включает в себя оператор удаления или хранимые процедуры удаления, триггеры удаления.)
- Повторите эти тестовые примеры с неверными данными и посмотрите, как реагирует база данных.

Тестирование базы данных – Объекты

Схемы, **таблицы**, **хранимые процедуры** и **триггеры** являются ключевыми объектами базы данных. Мы уже поделились типами тестирования БД и сценариями тестирования для этих объектов базы данных.

Схемы

Схема базы данных определяет структуру системы баз данных в формате, поддерживаемом системой управления базами данных. Схема относится к тому, как структурирована база данных (состоящая из таблиц базы данных в случае реляционных баз данных).

Схема базы данных представляет собой набор формул, называемых ограничениями целостности, налагаемых на базу данных. Эти ограничения целостности обеспечивают совместимость между частями схемы.

В реляционной базе данных схема состоит из таблиц, полей, представлений, индексов, пакетов, процедур, функций, триггеров, типов, материализованных представлений, синонимов, ссылок на базу данных и других элементов.

Схемы обычно хранятся в словаре данных. Хотя схема определяется на текстовом языке базы данных, этот термин часто используется для обозначения графического изображения структуры базы данных. Другими словами, схема - это структура базы данных, которая определяет объекты в базе данных.

Типичными схемами, используемыми в хранилище данных, являются−

- Звездообразная схема

- Схема снежинок
- Схема галактики

Таблицы в базе данных

В реляционной базе данных таблица используется для упорядочивания информации по строкам и столбцам.

Пример − Таблица клиентов содержит такую информацию, как идентификатор клиента, адреса, номера телефонов и т. Д. В виде ряда столбцов.

Каждая отдельная часть данных представляет собой поле в таблице. Столбец состоит из всех записей в одном поле, например телефонных номеров всех клиентов. Поля организованы в виде записей, которые представляют собой полные наборы информации (например, набор информации о конкретном клиенте), каждый из которых содержит строку.

Хранимые процедуры

Хранимая процедура - это серия инструкций SQL, хранящихся в базе данных в скомпилированном виде, и несколько программ могут совместно использовать ее. Использование хранимых процедур может быть полезным для поддержания целостности данных, контроля доступа к данным и повышения производительности.

Триггеры

Триггер базы данных - это код, который выполняется в ответ на определенные события в конкретной таблице или представлении в базе данных. Триггер в основном используется для поддержания целостности информации в базе данных.

Тестирование базы данных – Целостность данных

Целостность данных важна в базе данных. Оно включает проверку данных перед вставкой, обновлением и удалением. Для проверки записей справочной таблицы должны быть установлены триггеры.

Для проверки целостности данных необходимо выполнить следующие операции −

- Вам необходимо проверить основные столбцы в каждой таблице и проверить, существуют ли неверные данные. (Символы в поле имени, отрицательный процент и т.д.)
- Найдите противоречивые данные и вставьте их в соответствующие таблицы и посмотрите, не возникает ли какой-либо сбой.
- Вставьте дочерние данные перед вставкой родительских данных. Попробуйте удалить запись, на которую все еще ссылаются данные в другой таблице.
- Если данные в таблице обновляются, проверьте, обновляются ли другие соответствующие данные. Необходимо убедиться, что реплицируемые серверы или базы данных синхронизированы и содержат согласованную информацию.

Тестирование базы данных – Сопоставление данных

Сопоставление данных в базе данных - одна из ключевых концепций, которую должен проверять каждый тестировщик. Обычно тестировщики должны проверить сопоставление полей интерфейса пользователя с соответствующим полем базы данных.

Эта информация приведена в спецификации требований к программному обеспечению или спецификации бизнес-требований SRS / BRS document. Если сопоставление не предусмотрено, вам необходимо проверить часть кодирования.

Когда вы выполняете какое-либо действие во внешнем интерфейсе приложения, вызывается соответствующее действие CRUD, и тестировщик должен проверять, является ли каждое вызванное действие успешным или нет.

Ключевые аспекты сопоставления данных

Ниже приведены ключевые аспекты сопоставления данных −

- Чтобы проверить поля в формах пользовательского интерфейса / интерфейса и сопоставить их с соответствующей таблицей базы данных. Эта информация о сопоставлении определена в документах требований, как указано выше.

- Для любого действия, выполняемого во внешнем интерфейсе приложения, на серверной части инициируется соответствующее CRUD-действие "Создать, извлечь, обновить и удалить".
- Тестировщик должен будет проверить, вызвано ли правильное действие, и само по себе вызванное действие является успешным или нет.

Этапы тестирования сопоставления данных

Ниже приведены шаги, выполняемые для тестирования сопоставления данных –

- **Шаг 1** − Сначала проверьте наличие синтаксических ошибок в каждом скрипте.
- **Шаг 2**. Далее необходимо проверить сопоставление таблиц, сопоставление столбцов и сопоставление типов данных.
- **Шаг 3** − Проверьте сопоставление данных поиска.
- **Шаг 4** − Запускайте каждый сценарий, если записи не существуют в целевых таблицах.
- **Шаг 5** − Запускайте каждый сценарий, когда записи уже существуют в целевых таблицах.

Тестирование базы данных – Производительность

Приложение с большим временем отклика и низкой производительностью может привести к огромным проблемам. Нагрузочное тестирование базы данных используется для выявления любых проблем с производительностью перед развертыванием приложений баз данных для конечных пользователей.

Нагрузочное тестирование базы данных помогает вам разрабатывать приложения для баз данных с точки зрения производительности, надежности и масштабируемости. Нагрузочное тестирование приложений баз данных включает в себя тестирование производительности и масштабируемости вашего приложения базы данных при различной нагрузке пользователя.

Тестирование нагрузки на базу данных включает в себя моделирование реальной нагрузки пользователя для целевого приложения базы данных. Это

поможет вам определить, как ведет себя ваше приложение базы данных, когда несколько пользователей обращаются к нему одновременно.

Нагрузочное тестирование

Основная цель нагрузочного тестирования - проверить, влияет ли большинство выполняемых транзакций на производительность базы данных. При нагрузочном тестировании необходимо проверить следующие аспекты −

- Следует проверить время отклика для выполнения транзакций для нескольких удаленных пользователей.
- Для обычных транзакций необходимо включить одну редактируемую транзакцию, чтобы проверить производительность базы данных для транзакций этого типа.
- При обычных транзакциях вы должны включить одну транзакцию без редактирования, чтобы проверить производительность базы данных для транзакций такого типа.
- Следует проверять время, затрачиваемое базой данных на выборку определенных записей.

Стресс-тестирование

Стресс-тестирование выполняется для определения точки **останова** системы. Здесь приложение загружается таким образом, что в какой-то момент система выходит из строя. Эта точка называется точкой останова системы баз данных. Стресс-тестирование также известно как **тестирование на усталость**.

Определение состояния транзакций базы данных требует значительных усилий. Требуется правильное планирование, чтобы избежать каких-либо временных и финансовых проблем.

Наиболее распространенными инструментами стресс-тестирования являются **LoadRunner** и **WinRunner**.

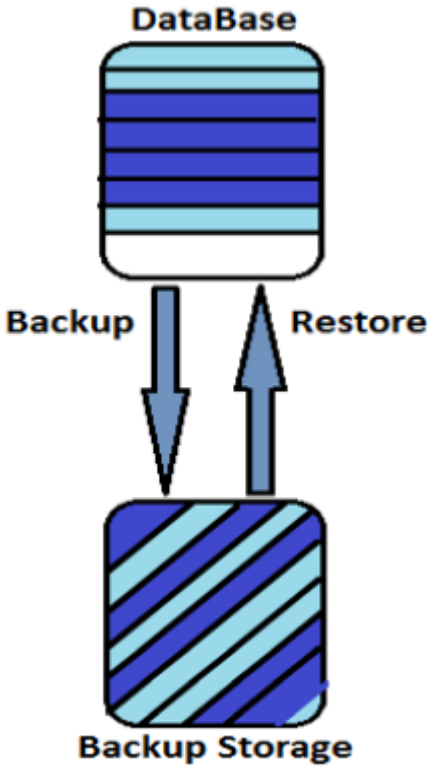
Тестирование базы данных – Инструменты

Существуют различные инструменты, предоставляемые поставщиками, которые можно использовать для генерации тестовых данных, управления тестовыми данными и выполнения тестирования базы данных, такого как нагрузочное тестирование и регрессионное тестирование.

Ниже приведены несколько распространенных используемых инструментов.**Тестирование базы данных – резервное копирование**

Sr.No	Категория и описание	Примеры
1	**Инструменты нагрузочного тестирования** Эти инструменты используются для создания высокой нагрузки на вашу базу данных, что позволяет определить, будет ли ландшафт вашей системы соответствовать потребностям вашего бизнеса.	Производительность веб-сайта Просмотр Rad Mercury
2	**Средства защиты данных** Эти инструменты используются для обеспечения соответствия и стандартов в соответствии с правилами информационной безопасности.	IBM Optim конфиденциальность данных
3	**Инструменты генератора тестовых данных** Тестировщик использует эти инструменты для генерации тестовых данных для системы баз данных. Это в основном требуется, когда у вас есть огромное количество данных, и вам нужен образец для выполнения тестирования БД. Обычно используется для нагрузочного и стресс-тестирования.	Фабрика данных Генератор данных DTM Турбо-данные
4	**Инструмент управления тестовыми данными** Эти инструменты используются для поддержания контроля версий для тестовых данных. Вы должны определить ожидаемые результаты, а затем сравнить их с фактическими результатами тестов.	IBM Optim Test Data Management
5	**Инструменты для выполнения модульного тестирования** Эти инструменты используются для выполнения регрессионного тестирования вашей базы данных.	SQLUnit TSQLUnit DbFit DBUnit

Наиболее важной частью роста организации являются ее данные. В случае сбоя системы необходимо восстановить данные. Резервное копирование - это точная копия базы данных, которая поможет вам восстановить ваши данные в случае любой потери данных.



Рассмотрим финансовую компанию, у которой есть данные, относящиеся к ее клиентам, такие как номер A / C, имена клиентов, кредитные и дебетовые платежи, продолжительность и т. Д. Как такая организация справится с давлением потери такой важной информации в случае сбоя данных?

По этой причине вы создаете резервные копии данных, чтобы в случае любого сбоя диска, контроллера диска и т. Д. вы можете полагаться на резервную копию, чтобы восстановить ее в базе данных.

Типы резервных копий данных

Существует два типа резервного копирования, которые можно использовать −

- **Физическое резервное** копирование. Физическое резервное копирование включает в себя резервное копирование с помощью сторонних средств резервного копирования, таких как Veritas Net Back, IBM Tivoli Manager или user manager, с использованием утилит операционной системы.
- **Логическое резервное** копирование − Логическое резервное копирование базы данных включает в себя создание резервных копий логических объектов, таких как таблицы, индексы, процедуры и т.д.

Пример − Одним из распространенных инструментов для резервного копирования данных является **Oracle Recovery Manager (RMAN)**, который

является утилитой Oracle для резервного копирования базы данных.

RMAN состоит из двух компонентов −

- **Целевая база** данных, для которой требуется резервное копирование.
- **RMAN**клиент используется для выполнения команд для резервного копирования данных.

ПРОВЕРКА РЕЗЕРВНОЙ КОПИИ используется для проверки того, можете ли вы создать действительную резервную копию файлов базы данных. Это гарантирует −

- Если имеется резервная копия для физических или логических объектов базы данных.
- Если для бесценных данных настроены регулярные резервные копии.
- Если средство резервного копирования соответствует требованиям организации к резервному копированию.

Тестирование базы данных – восстановление

Тестирование восстановления базы данных используется для обеспечения восстановления базы данных. Тестирование восстановления позволяет выяснить, работает ли приложение должным образом, и проверить получение бесценных данных, которые были бы потеряны, если бы ваш метод восстановления не был настроен должным образом.

Вы также проверяете, работают ли несколько важных процессов без сбоев, чтобы гарантировать, что восстановление данных пройдет гладко на этапе тестирования.

Для восстановления базы данных можно выполнить следующие проверки −

- Любые ошибки или ошибки в программном обеспечении резервного копирования, и вам необходимо устранить эти проблемы на более ранней стадии.
- Вам необходимо провести тестирование восстановления, чтобы вы знали, что делать в случае чрезвычайной ситуации.
- Вам необходимо проверить потребности в тестировании восстановления, чтобы вы могли спланировать эффективную стратегию восстановления.

- Вы также должны знать, как можно восстановить документы.

Вам необходимо запустить тесты восстановления на ранней стадии проекта. Это позволяет удалять и удалять из системы ошибки любого типа. Вот список некоторых важных моментов, которые следует учитывать во время тестирования −

- Промежуток времени, когда в системе базы данных происходят изменения или модификации.
- Период, в течение которого вы хотите выполнить свой план восстановления.
- Чувствительность данных в системе баз данных. Чем важнее данные, тем чаще вам нужно будет тестировать программное обеспечение.

Общие этапы тестирования резервного копирования и восстановления базы данных

При тестировании восстановления базы данных необходимо выполнить тест в реальной среде, чтобы проверить, действительно ли система или данные могут быть восстановлены в случае каких-либо сбоев или любых других непредвиденных событий в бизнес-среде.

Ниже приведены общие действия, выполняемые при тестировании восстановления базы данных −

- Тестирование системы баз данных
- Тестирование файлов SQL
- Тестирование частичных файлов
- Тестирование резервного копирования данных
- Тестирование средства резервного копирования
- Тестирование резервных копий журналов

Тестирование базы данных – Безопасность

Тестирование безопасности базы данных проводится для поиска лазеек в механизмах безопасности, а также для выявления уязвимостей или слабых мест системы базы данных.

Основной целью тестирования безопасности базы данных является выявление уязвимостей в системе и определение того, защищены ли ее данные и ресурсы от потенциальных злоумышленников. Тестирование безопасности определяет способ эффективного выявления потенциальных уязвимостей при регулярном выполнении.

Ниже приведены основные цели выполнения тестирования безопасности базы данных −

- Аутентификация
- Авторизация
- Конфиденциальность
- Доступность
- Целостность
- Устойчивость

Типы угроз в системе баз данных

Внедрение SQL

Это наиболее распространенный тип атаки в системе баз данных, когда вредоносные инструкции SQL вставляются в систему баз данных и выполняются для получения важной информации из системы баз данных. Эта атака использует лазейки в реализации пользовательских приложений. Чтобы предотвратить это, следует тщательно обрабатывать поля пользовательского ввода.

Повышение привилегий в базе данных

В этой атаке пользователь уже имеет некоторый доступ к системе баз данных, и он только пытается повысить этот уровень доступа, чтобы он / она мог выполнять некоторые несанкционированные действия в системе баз данных.

Отказ в обслуживании

При этом типе атаки злоумышленник делает систему базы данных или ресурс приложения недоступными для законных пользователей. Приложения также могут быть атакованы способами, которые делают приложение, а иногда и всю машину непригодной для использования.

Несанкционированный доступ к данным

Другим типом атаки является получение несанкционированного доступа к данным в приложении или системе баз данных. Несанкционированный доступ включает в себя −

- Несанкционированный доступ к данным через пользовательские приложения
- Несанкционированный доступ к путем мониторинга доступа других
- Несанкционированный доступ к повторно используемой информации аутентификации клиента

Подделка идентификационных данных

При подделке идентификационных данных хакер использует учетные данные пользователя или устройства для запуска атак на сетевые узлы, кражи данных или обхода контроля доступа к системе базы данных. Для предотвращения этой атаки требуются меры по смягчению последствий на уровне ИТ-инфраструктуры и сети.

Манипулирование данными

При атаке с использованием данных хакер изменяет данные, чтобы получить некоторое преимущество или нанести ущерб имиджу владельцев баз данных.

Методы тестирования безопасности базы данных

Тестирование на проникновение

Тест на проникновение - это атака на компьютерную систему с целью поиска лазеек в системе безопасности, потенциального получения доступа к ней, ее функциям и данным.

Поиск рисков

Определение рисков - это процесс оценки и принятия решения о риске, связанном с типом потери и возможностью возникновения уязвимости. Это определяется внутри организации различными собеседованиями, обсуждениями и анализом.

Тест на внедрение SQL

Оно включает в себя проверку вводимых пользователем данных в полях приложения. Например, ввод специального символа, такого как ‘,’ или ‘;’, в любом текстовом поле в пользовательском приложении должен быть запрещен. Когда возникает ошибка базы данных, это означает, что пользовательский ввод вставляется в некоторый запрос, который затем выполняется приложением. В таком случае приложение уязвимо для внедрения SQL.

Эти атаки представляют большую угрозу для данных, поскольку злоумышленники могут получить доступ к важной информации из базы данных сервера. Чтобы проверить точки входа SQL-инъекций в ваше веб-приложение, найдите код из своей базы кода, в котором выполняются прямые запросы MySQL к базе данных, принимая некоторые пользовательские вводимые данные.

Тестирование SQL-инъекций может выполняться для скобок, запятых и кавычек.

Взлом пароля

Это самая важная проверка при выполнении тестирования системы баз данных. Чтобы получить доступ к важной информации, хакеры могут использовать инструмент для взлома паролей или могут угадать общее имя пользователя / пароль. Эти распространенные пароли легко доступны в Интернете, а также инструменты для взлома паролей существуют свободно.

Поэтому во время тестирования необходимо проверить, поддерживается ли в системе политика паролей. В случае любых банковских и финансовых приложений необходимо установить строгую политику паролей для всех критически важных информационных систем баз данных.

Аудит безопасности системы баз данных

Аудит безопасности - это процесс оценки политик безопасности компании через регулярные промежутки времени, чтобы определить, соблюдаются ли необходимые стандарты или нет. В соответствии с бизнес-требованиями для определения политики безопасности могут использоваться различные стандарты безопасности, а затем может быть выполнена оценка установленных политик в соответствии с этими стандартами.

Примером наиболее распространенных стандартов безопасности являются ISO 27001, BS15999 и др.

Инструменты тестирования безопасности базы данных

На рынке доступны различные инструменты тестирования системы, которые можно использовать для тестирования ОС и проверки приложений. Некоторые из наиболее распространенных инструментов обсуждаются ниже.

Прокси-сервер Zed Attack

Это инструмент для тестирования на проникновение для поиска уязвимостей в веб-приложениях. Он предназначен для использования людьми с широким опытом работы в области безопасности и поэтому идеально подходит для разработчиков и функциональных тестировщиков, которые новички в тестировании на проникновение. Обычно используется для Windows, Linux, Mac OS.

Paros

Все данные HTTP и HTTPS между сервером и клиентом, включая файлы cookie и поля формы, могут быть перехвачены и изменены с помощью этих сканеров. Он используется для кроссплатформенных Java JRE / JDK 1.4.2 или выше.

Инструментарий социального инженера

Это инструмент с открытым исходным кодом, и атаке подвергаются человеческие элементы, а не системный элемент. Это позволяет отправлять электронные письма, Java-апплеты и т. Д., Содержащие код атаки. Это предпочтительно для Linux, Apple Mac OS X и Microsoft Windows.

Skipfish

Этот инструмент используется для проверки их сайтов на наличие уязвимостей. Отчеты, создаваемые инструментом, предназначены для использования в качестве основы для профессиональных оценок безопасности веб-приложений. Это предпочтительно для Linux, FreeBSD, macOS X и Windows.

Vega

Это мультиплатформенный инструмент веб-безопасности с открытым исходным кодом, который используется для поиска случаев внедрения SQL,

межсайтовых сценариев (XSS) и других уязвимостей в веб-приложениях. Это предпочтительно для Java, Linux и Windows.

Wapiti

Wapiti - это веб-инструмент с открытым исходным кодом, который сканирует веб-страницы веб-приложения и проверяет наличие сценариев и форм, в которые он может вводить данные. Он построен на Python и может обнаруживать ошибки обработки файлов, инъекции базы данных, XSS, LDAP и CRLF, обнаружение выполнения команд.

Веб-скарабей

Он написан на Java и используется для анализа приложений, которые взаимодействуют по протоколам HTTP / HTTPS. Этот инструмент в первую очередь предназначен для разработчиков, которые могут сами писать код. Этот инструмент не зависит от операционной системы.

Тестирование базы данных – проблемы

Для успешного тестирования базы данных тестировщик должен собрать требования из всех источников, например, технические и функциональные требования. Существует вероятность того, что некоторые требования находятся на высоком уровне, поэтому необходимо разбить эти требования на мелкие части. Тестирование базы данных - сложная задача, и тестировщики сталкиваются со многими проблемами при выполнении этого тестирования. Наиболее распространенными проблемами тестирования базы данных являются−

Область тестирования слишком велика

Тестировщик должен определить тестовые элементы при тестировании базы данных, иначе у него может не быть четкого понимания того, что он будет тестировать, а что нет. Поэтому, если вы четко понимаете требования, вы можете потратить много времени на тестирование некритичных объектов в базе данных.

Когда у вас есть список объектов для тестирования, следующим шагом является оценка усилий, необходимых для разработки тестов и выполнения тестов для каждого элемента тестирования. В зависимости от их структуры и

размера данных выполнение некоторых тестов базы данных может занять много времени.

Поскольку размер базы данных слишком велик, становится большой проблемой определить объекты, которые необходимо протестировать, и те, которые следует исключить.

Уменьшенная тестовая база данных

Обычно тестировщикам предоставляется копия базы данных разработки для тестирования. В этой базе данных мало данных, которых достаточно для запуска приложения. Таким образом, необходимо протестировать систему разработки, промежуточной и рабочей базы данных.

Изменения в структуре базы данных

Это одна из распространенных проблем при тестировании БД. Иногда случается, что вы разрабатываете или выполняете тест, а структура базы данных была изменена в это время. Это необходимо для того, чтобы вы были в курсе изменений, внесенных в базу данных во время тестирования.

После изменения структуры базы данных следует проанализировать влияние изменений и модифицировать тесты. Кроме того, если тестовую базу данных используют несколько пользователей, вы не будете уверены в результатах тестирования, поэтому убедитесь, что тестовая база данных используется только для тестирования.

Еще одна проблема при тестировании БД заключается в том, что вы запускаете несколько тестов одновременно. Вы должны запускать по одному тесту за раз, по крайней мере, для тестов производительности. Вы не хотите, чтобы ваша база данных выполняла несколько задач и занижала производительность.

Сложные планы тестирования

Структура базы данных обычно сложна и содержит огромные данные, поэтому существует вероятность того, что вы повторно выполняете неполные или одни и те же тесты. Поэтому необходимо создать план тестирования и действовать соответствующим образом, регулярно проверяя прогресс.