SOLID-ПРИНЦИПЫ ОБЪЕКТНО-ОРИЕНТИРОВАННОГО ПРОЕКТИРОВАНИЯ И ИХ РЕАЛИЗАЦИЯ НА ЯЗЫКЕ PHP

УО «Белорусский государственный университет информатики и радиоэлектроники»
г. Минск, Республика Беларусь

Нестер А.А.

Куликов С.С. – к.т.н., доцент

В работе рассмотрены принципы SOLID, используемые при построении объектно-ориентированных систем. Так как данный метод построения систем сегодня используются при построении многих приложений и систем, то данная тема является весьма актуальной.

SOLID-принципы- это аббревиатура пяти основных принципов дизайна классов в объектно-ориентированном проектировании — Singleresponsibility, Open-closed, Liskovsubstitution, Interfacesegregation и Dependencyinversion.

Принцип единственной ответственности гласит — «На каждый объект должна быть возложена одна единственная обязанность». Т.е. другими словами — конкретный класс должен решать конкретную задачу — ни больше, ни меньше. Чтобы проверить, соблюдается ли данный принцип, можно воспользоваться следующим правилом - проверяем, сколько у нас есть причин для изменения класса — если больше одной, то следует разбить данный класс.

Данный принцип гласит — "программные сущности должны быть открыты для расширения, но закрыты для модификации". На более простых словах это можно описать так — все классы, функции и т.д. должны проектироваться так, чтобы для изменения их поведения, нам не нужно было изменять их исходный код. Правило для проверки данного принципа - представляем наш класс как «чёрный ящик» и смотрим, возможно ли в таком случае изменить его поведение.

Принцип подстановки Барбары Лисков (Liskovsubstitution) гласит — «Объекты в программе могут быть заменены их наследниками без изменения свойств программы». Данный принцип широко используется в контрактном программировании. Для проверки соблюдения условия, необходимо проверить, не усилили ли мы предусловия выполнения участка кода и не ослабили ли постусловия. Если это произошло — то принцип не соблюдается

Принцип разделения интерфейса (Interfacesegregation) гласит, что «Много специализированных интерфейсов лучше, чем один универсальный». Соблюдение этого принципа необходимо для того, чтобы классы-клиенты использующий/реализующий интерфейс знали только о тех методах, которые они используют, что ведёт к уменьшению количества неиспользуемого кода. Проверка соблюдения принципа происхожит следующим образом - Проверяем, насколько много интерфейс содержит методов и насколько разные функции накладываются на эти методы, и если необходимо — разбиваем интерфейсы.

Принцип инверсии зависимостей (DependencyInvertion) гласит — «Зависимости внутри системы строятся на основе абстракций. Модули верхнего уровня не зависят от модулей нижнего уровня. Абстракции не должны зависеть от деталей. Детали должны зависеть от абстракций». Данное определение можно сократить — «зависимости должны строится относительно абстракций, а не деталей». Для проверки выполнения принципа используем следующее правило - Проверяем, зависят ли классы от каких-то других классов(непосредственно инстанцируют объекты других классов и т.д) и если эта зависимость имеет место, заменяем на зависимость от абстракции.

Список использованных источников:

1. Роберт С. Мартин, Джеймс В. Ньюкирк, Роберт С. Косс Быстрая разработка программ. Принципы, примеры, практика, стр. 254-316
2. Мартин Р., Мартин М. Принципы, паттерны и методики гибкой разработки на языке C# , стр.357-404

ПРИМЕНЕНИЕ ДОКУМЕНТООРИЕНТИРОВАННЫХ NOSQL СУБД В СОВРЕМЕННЫХ ВЕБ-ПРИЛОЖЕНИЯХ

УО «Белорусский государственный университет информатики и радиоэлектроники»
г. Минск, Республика Беларусь

Н.А. Хмурович, О.А. Мацкевич

Куликов С.С. – к.т.н., доцент

В последнее время происходит стремительный рост популярности веб-технологий. Он обусловлен прежде всего тем, что веб остается единственной платформой, приложения которой одинаково хорошо работают на любом аппаратном обеспечении. С увеличением количества устройств, предоставляющих выход в интернет, возрастает популярность веб-приложений. Важной задачей становится хранение и

быстрая агрегация большого объема данных. Серьёзными требованиями к современным веб-приложениям являются быстрый отклик и максимальная производительность при работе с большим объемом информации.

Современные веб-приложения обрабатывают большое количество запросов и действий пользователей. Такого рода информация имеет ценность с точки зрения поведенческого анализа субъекта.

Реляционные СУБД стали стандартом при проектировании слоя хранения данных в современных приложениях. Современные реляционные СУБД соответствуют требованиям ACID: Atomicity — Атомарность, Consistency — Согласованность, Isolation — Изолированность, Durability — Надежность. В основу реляционной модели заложены жесткие схемы, строго регламентирующие структуру и связи сущностей. Реляционные базы данных обеспечивают надёжное хранение данных, атомарность крупных операций и постоянную согласованность. Однако для достижения подобного поведения используются различные механизмы, которые накладывают пенальти на производительность и ограничивают возможности масштабирования.

С приходом огромных массивов информации и распределенных систем стало ясно, что обеспечить для них одновременно транзакционность набора операций и получить высокую доступность и быстрый отклик — невозможно.

Эта идея была положена в основу CAP теоремы, которая гласит о том, что реализация распределенных вычислений не может достичь одновременно трёх свойств: Consistency (Согласованность), Availability (Доступность), PartitionTolerance (Терпимость к разделению)

Напрактикесуществуютсистемытипа CA (Availability, Consistency), CP (Consistency, Partition tolerance), AP (Availability, Partition tolerance).

С лавинообразным ростом количества пользователей и информации увеличились требования к производительности хранилищ данных. Необходимость обрабатывать большое количество информации за разумное время столкнулась с проблемой вертикальной масштабируемости баз данных.

Выходом из ситуации является горизонтальное масштабирование, когда несколько независимых серверов соединяются между собой, и каждый владеет своей репликой или частью данных, и обрабатывает только часть запросов. В такой архитектуре для повышения мощности хранилища достаточной мерой является добавление нового сервера в кластер. Процедурами шардинга, репликации, обеспечением отказоустойчивости, перераспределения данных в случае добавления дополнительного сервера в таких системах занимается СУБД.

Общие характеристики NoSQLдокументоориентированных СУБД:

1. Представление данных в виде агрегатов. Документоориентированные хранилища пропагандируют агрегирование и встраивание данных в документ, чтобы оперировать с этими сущностями как с целостными объектами.
2. В большинстве своём это opensource решения, они бесплатны.
3. Возможность автоматически распределять данные между серверами.
4. Использование памяти, прозрачное кэширование - содержимое коллекций активно кэшируется для выборки.

Вопрос применения документоориентированных СУБД в современных веб-приложениях зависит от конечной аудитории пользователей и планируемой нагрузки. Очевидно, что упрощение процесса разработки и закладка в возможность горизонтального масштабирования также является весомым аргументом при выборе СУБД. Современные NoSQL СУБД могут стать очень эффективным инструментом для хранения данных современных веб-приложений.

Список использованных источников:

1. BASE: An ACID alternative - Dan Pritchett, http://queue.acm.org/detail.cfm?id=1394128
2. Why NoSQL? - Couchbase, http://www.couchbase.com/why-nosql/nosql-database
3. http://ru.wikipedia.org/wiki/ACID

ОЦЕНКА ЭФФЕКТИВНОСТИ ПРИМЕНЕНИЯ ЧАСТОТНО-КОНТЕКСТНОГО АНАЛИЗА ТЕКСТОВОЙ ИНФОРМАЦИИ

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Потараев В. В.

Серебряная Л. В. – канд. техн. наук, доцент

В современных информационных системах содержится огромное количество текстовой информации. Существуют различные методы анализа текстовой информации. Рассмотрим эффективность применения частотно-контекстной классификации при решении задачи выбора текста, наиболее полно отражающего некоторую тему.

Предположим, необходимо выбрать один наиболее содержательный текст из множества текстов по данному вопросу. Очевидно, что в этом случае можно использовать классификацию текстовых данных.