Таким образом, использование реляционных баз данных в анализе социальных сетей играет ключевую роль в понимании динамики взаимодействий между пользователями и открывает новые перспективы для развития социальных наук и цифрового маркетинга.

Литература

1. Гектор Гарсиа-Молина, Джеффри Д. Ульман и Дженнифер Уид. Системы баз данных: Полная книга / Гектор Гарсиа-Молина, Джеффри Д. Ульман и Дженнифер Уид
2. Мэтью А. Рассел. Анализ социальных сетей: анализ данных Facebook, Twitter, LinkedIn, Google+, GitHub и т. д. / Мэтью А. Рассел.
3. Кузнецов, С.Д. Базы данных: учебник для вузов / С.Д. Кузнецов. – 4-е изд. – М.: Академия, 2012.

УДК 004.658

МИГРАЦИИ БАЗ ДАННЫХ. ИНТЕГРАЦИЯ МИГРАЦИЙ В C#

Зеленухо А. Д., Мацкевич Н.Н.
Научный руководитель – Воронич Л.В., ассистент

Миграция базы данных — это процесс переноса данных и рабочих нагрузок с одного или нескольких платформ на более подходящее решение для хранения. Миграции включают в себя сложный, многоэтапный процесс, который обычно содержит оценку, преобразование схемы базы данных, преобразование сценариев, миграцию данных, функциональное тестирование, настройку производительности и многие другие этапы[1]. Компании осуществляют миграцию баз данных по разным причинам:

- Сократить расходы на ИТ за счет перехода на базу данных с более эффективным использованием ресурсов;
- Удовлетворить потребности бизнеса для более мощных систем хранения данных;
- Обновление до последней версии уже используемой базы данных;
- Выбор другого типа базы данных для выигрыша производительности и уменьшения задержки;

Существует два основных вида миграции базы данных: “большой взрыв” и “струйная миграция”.

При Большом взрыве команда закрывает текущую базу данных и переходит в новую среду. При этом стоит учитывать, чем больше данных, тем дольше длится процесс. Данный процесс сопровождается постоянным тестированием. Команда постоянно проверяет результаты миграции, чтобы на ранних этап исключить ошибки и убедиться, что все работает корректно.

Данный тип миграции всегда связан с проблемами доступности, и при ошибке весь процесс команда должна повторять заново. Из положительных моментов следует выделить простоту подхода “большого взрыва”. Этот процесс происходит в рамках ограниченного по времени мероприятия.
Миграция по принципу "большого взрыва" — это оптимальный вариант, когда команда может с самого начала определить точный объем работ или когда другие проекты диктуют сроки.
“Струйная миграция” предполагает поэтапное перемещение данных из источника в систему назначения. Данные переносятся постепенно в течение длительного периода, что требует синхронизации между исходной и целевой системами для поддержания согласованности данных. Такой подход сводит к минимуму время простоя и снижает компонент риска, связанный с миграцией “большого взрыва”. При снижении рисков выплывает проблема сложности данного подхода. От команды требуется постоянная синхронизация данных и дополнительные требования к ресурсам[2].
Выбор правильной стратегии миграции зависит от таких факторов, как размер базы данных, совместимость исходной и целевой системы, правила бизнеса. Бизнес сложная структура, которая может разработать свой план миграция, обусловленный потребностями, положительными и отрицательными сторонами других подходов.
После изучения понятия миграции базы данных, необходимо понимать, как выполнить миграцию базы данных. К сожалению, не существует стандарта, который бы определял, как производиться процесс миграции для всех баз данных. В общем процесс зависит от конкретного инструмента, однако можно выделить два основных способа произвести миграцию.
Способ 1: Использование библиотеки или фреймворка, зависящего от выбранного языка. Библиотеки и фреймворки имеют подробную документацию, которая помогает производить миграции. При этом способе команда создает файлы миграции с помощью командной строки. Иногда вам может понадобиться вручную написать пользовательский код для внесения изменений. В остальных случаях библиотека или выбранных фреймворк сделает все за вас.
Способ 2: Использование независимого программного обеспечения, ориентированного на миграцию баз данных. Что бы избежать привязки к конкретной библиотеке или фреймворку, может быть необходимость использование данного подхода. Существую такие инструменты как Flyway, Liquibase, однако способ привязывает к конкретной базе данных, так как данного программное обеспечении работает только с ограниченным кругом баз данных. Из положительных моментов стоит подметить, что программное обеспечение относительно простое в освоении и

предоставляет возможность генерации миграции с помощью командной строки и создавать пользовательские коды для схем миграций[3].
Миграции в C# в большинстве случаев реализованы с помощью Entity Framework Core (EF Core) от Microsoft. EF Core является кроссплатформенная и расширяемая версия популярной технологии доступа к данным Entity Framework с открытым исходным кодом. EF Core обновляет схему базы данных с помощью миграций, синхронизируя ее с моделью данных вашего приложения и сохраняя существующие данные в базе. Управление изменениями в базе данных очень важно, особенно в командной среде, где несколько разработчиков работают над одним приложением. Эта функция помогает эффективно организовать эти изменения[4]. По своей сути миграция представляет собой версию схемы вашей базы данных. После того как вы инициируете миграцию, EF Core активно генерирует код, который описывает процесс обновления схемы базы данных от существующего состояния до нового состояния, определенного вашей моделью данных .NET.
Давайте рассмотрим этапы создания миграции в C#:
1) Установить EF Core в проект через NuGet пакеты. Для этого в менеджере NuGet пакетов или в консоли добавьте пакет Microsoft.EntityFrameworkCore. Необходимо установить пакет используемой базы данных, например Microsoft.EntityFrameworkCore.SqlServer для использования SQL Server;
2) Создать классы C#, описывающие таблицы в базе данных;
3) Создать контекст базы данных (DbContext), который действует как мост между кодом C# и базой данных. При включении класса в контекст, открывается возможность сохранять, управлять, запрашивать данные из базы данных;
После подготовительных процессов в консоле используя команду “Add-Migration [name]” создается миграция, отражающая исходное состояние моделей в схеме базы данных. В папке Migration появиться файл с миграциями, который содержит методы UP и DOWN. UP методы отображают какие внести изменения в базу данных, а DOWN какие изменения откатить.
Когда миграция готова можно применять ее к базе данных командой “Update-Database”. Миграции в C# являются важным инструментом для управления жизненным циклом базы данных и поддержания ее согласованном состоянии с моделью приложения[5].
В заключении: миграция — это фундаментальные аспекты, обеспечивающие надежную основу для развития схемы базы данных в синхронизации с вашим приложением. Понимая и используя миграцию, вы можете гарантировать, что управление схемой вашей базы данных является систематическим, контролируется версиями и интегрировано в рабочий

процесс разработки. Независимо от того, добавляете ли вы новые функции, исправляете ошибки или отправляете данные для тестирования, миграции предлагают мощный набор инструментов для эффективного и результативного управления этими изменениями.

Литература

1) Amazon [Электронный ресурс]. – Режим доступа: https://aws.amazon.com/blogs/database/database-migration-what-do-you-need-to-know-before-you-start/. – Дата доступа: 25.04.2024.
2) PhoenixNap [Электронный ресурс]. – Режим доступа: https://phoenixnap.com/blog/database-migration. – Дата доступа: 25.04.2024.
3) CloudBees [Электронный ресурс]. – Режим доступа: https://www.cloudbees.com/blog/database-migration#choosing-the-best-option-for-you– Дата доступа: 24.04.2024.
4) .Net Code Chronicles [Электронный ресурс]. – Режим доступа: https://amarozka.dev/entity-framework-migrations/. – Дата доступа: 26.04.2024.
5) Forproger [Электронный ресурс]. – Режим доступа: https://forproger.ru/article/migracii-i-isxodnye-dannye-s-entity-framework-core. – Дата доступа: 26.04.2024.

УДК 004.223

СИМВОЛЬНЫЕ КОДИРОВКИ В СОВРЕМЕННЫХ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЯХ

Кабыш Я.А.
Научный руководитель – Воронич Л.В., ассистент

Стремительное развитие информационных технологий в современном мире привело к необходимости постоянной работы с текстовой информацией, которую нужно было как-то хранить в памяти компьютера. Для решения этой проблемы были придуманы различные методы кодировки информации, каждая из которых имеет свои преимущества и недостатки.
Первым методом кодировки стал формат ASCII (American standard code for information interchange) — это стандартный код символов, который используется для представления текстовой информации в компьютерных системах и обмена данными между различными устройствами и программами. ASCII использует 7-битный код (0-127) для представления основных символов английского алфавита, цифр, знаков препинания и