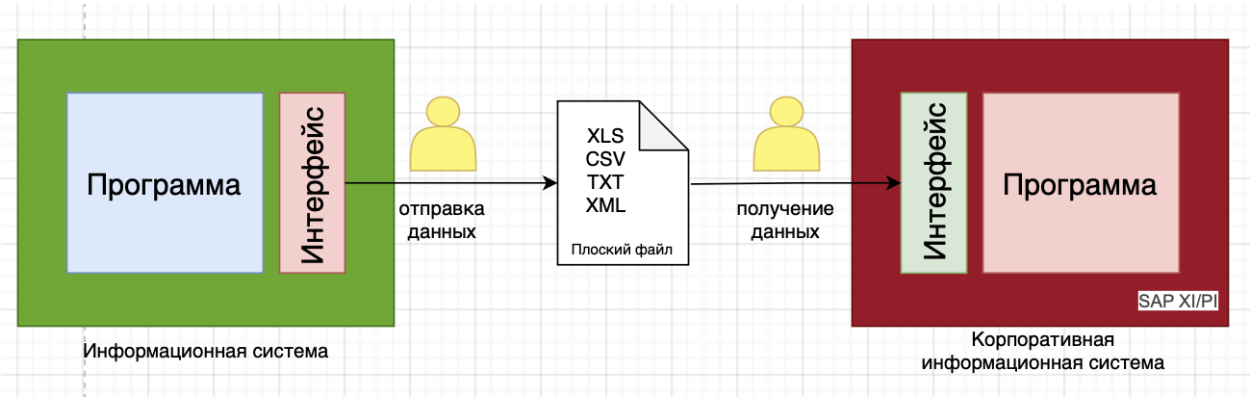
# WB Task 32

- **Интеграция на основе файлового обмена**




Преимущества:

1. SAP XI/PI обеспечивает интеграцию данных в режиме онлайн, а также высокий уровень безопасности, поддержку открытых стандартов взаимодействия и механизмы централизованного мониторинга.
2. Применение механизмов экспорта/импорта плоских файлов (Flat Files) является одним из самых быстрых и дешевых, с точки зрения программной реализации и стоимости, способов интеграции данных ИС. Данный способ интеграции применим в случаях, когда обмен данных ИС носит разовый или достаточно редкий характер.
3. Подходит для обмена данными между разнородными системами.

Недостатки:

1. Высокая задержка из-за периодичности обмена данными:

   *Решение*: настроить триггеры для моментальной обработки файлов или использовать расписание с более частыми интервалами обновлений.

2. Нет гарантии доставки:

   *Решение*: внедрить механизмы подтверждения доставки, такие как создание сопроводительных файлов или метаданных для отслеживания обработки файла.

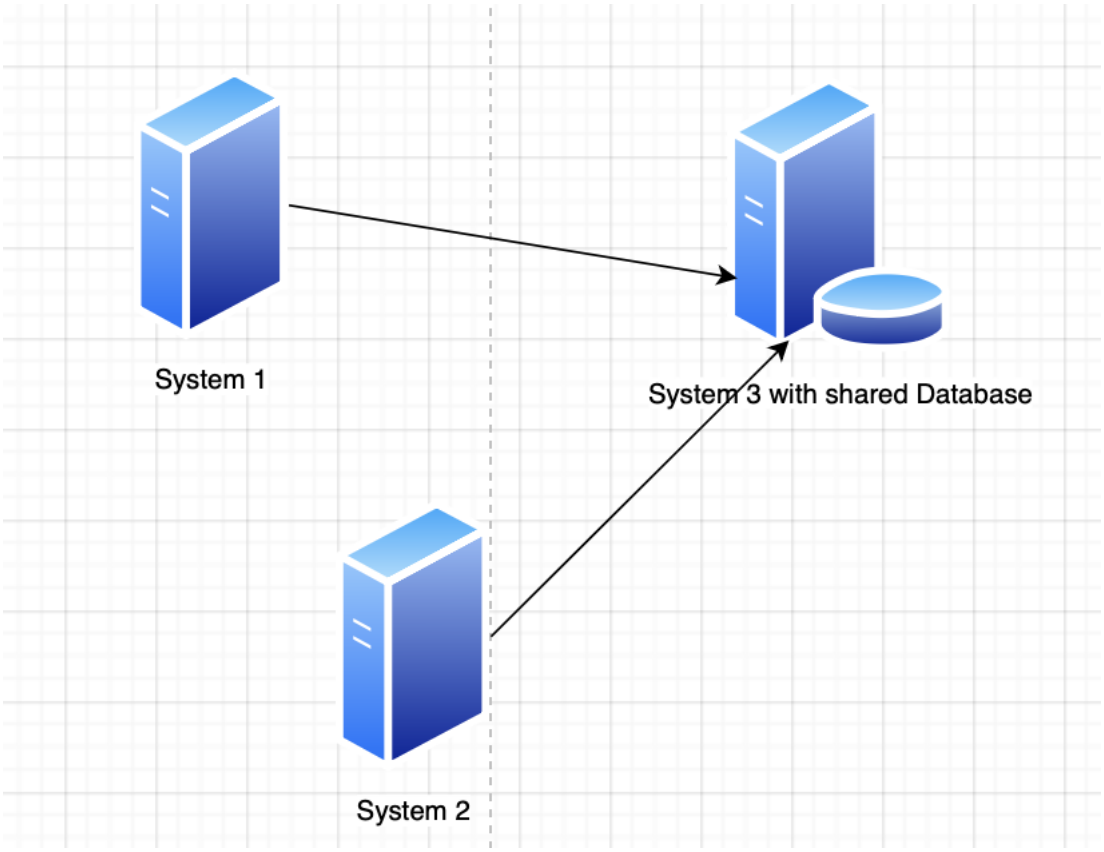
3. Риски конфиденциальности и безопасности данных:

   *Решение*: шифровать файлы перед их сохранением и настраивать строгие права доступа к файловой системе.

4. Ограниченная масштабируемость и возможное увеличение объема файлов

   *Решение*: организовать архивирование старых файлов и очистку неактуальных данных, оптимизировать формат файлов, использовать сжатие для больших данных.

- **Интеграция на основе использования прямого доступа в БД**




Преимущества:

- Целостность и непротиворечивость данных: все системы имеют доступ к одному хранилищу и одинаковым данным.
- Отсутствие распределённых транзакций.
- Данные не дублируются.
- Простота реализации.

Недостатки:

- БД — единая точка отказа для всех интегрируемых ИС.

Решение:

- Использовать кластеризацию базы данных и резервное копирование для повышения отказоустойчивости.
- Настроить репликацию данных и балансировку нагрузки между несколькими экземплярами БД.

- Снижение производительности из-за возможных блокировок и разной нагрузки от разных ИС.
  Решение:
  - Внедрить механизмы разделения нагрузки, используя реплики для операций чтения, чтобы избежать блокировок.
  - Оптимизировать запросы и индексацию, а также настроить транзакции для минимизации блокировок.

- Избыточность модели данных из-за необходимости закрыть потребности разных внешних систем и бизнес-процессов.
  Решение:
  - Использовать представления (views) или схемы данных, адаптированные под каждую систему, сохраняя основную модель данных неизменной.
  - Применять уровень абстракции, который позволяет скрывать детали реализации и избегать дублирования данных.

- Нужно много внимания уделить масштабированию БД.
  Решение:
  - Разделить данные на разделы (шардинг) и использовать горизонтальное масштабирование.

  - Использовать архитектуру микросервисов с распределенными базами данных, если возможно.

- Нужно много внимания уделить вопросам безопасности.
  Решение:
  - Внедрить ролевую модель доступа с минимально необходимыми привилегиями для каждой системы.

  - Шифровать данные и использовать мониторинг доступа для предотвращения несанкционированного доступа.

- Большой объём ручных изменений из-за высокого уровня связанности: все внешние системы должны учитывать все возможные изменения в структуре хранения.
  Решение:

- Внедрить API или уровень абстракции, который отделяет интеграционные системы от внутренней структуры БД, что снизит зависимость.
- Применять версионирование схемы данных и документировать изменения для облегчения обновлений.

- Сложность обновлений и миграции.
  Решение:
  - Использовать инструменты миграции данных (например, Liquibase или Flyway) для автоматизации и управления изменениями в структуре БД.

  - Планировать миграции на периоды с низкой нагрузкой и тестировать обновления на тестовой среде перед развертыванием в рабочей среде.

- Низкий уровень абстракции при разработке.
  Решение:
  - Внедрить слои абстракции, такие как ORM (Object-Relational Mapping) для управления доступом к данным и улучшения читаемости и поддержки кода.

  - Использовать библиотеку доступа к данным или микросервисный подход для минимизации прямого взаимодействия с БД.

- **Сервисная интеграция**


Преимущества:

- Гибкость и независимость от платформ.
- Хорошо масштабируется и предоставляет четкий интерфейс.

Недостатки:

- Зависимость от доступности сервиса:

  *Решение*: внедрить механизмы кеширования данных и обработку ошибок, используя резервные сервисы и механизм очередей.

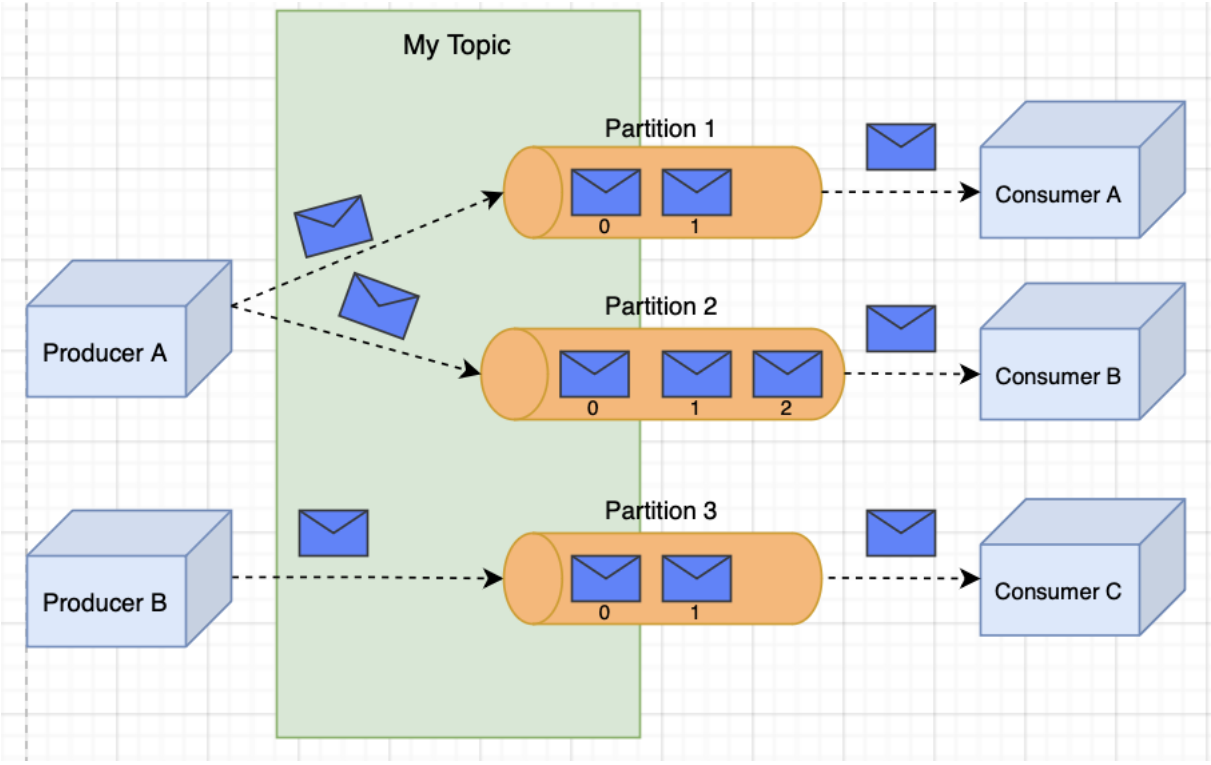
- Сложность управления версиями API:

  *Решение*: использовать версионирование API и чётко документировать изменения, чтобы старые клиенты не пострадали.

- Повышенные требования к безопасности:

  *Решение*: использовать OAuth2, JWT или другие методы аутентификации и авторизации для защиты данных и доступа.

- **Интеграция на основе использования брокера сообщений**




Преимущества:

- Кафка гарантирует, что **то, что попало в Кафку, из нее не пропадет.**
- Записать сообщение в топик
- Прочитать записи из топика
- Закоммитить место до которого дочитали
- Высокая производительность и масштабируемость
- Отказоустойчивость
- Надежность хранения данных
- Поддержка асинхронного взаимодействия

Недостатки:

- Ограниченные возможности удаления и изменения сообщений

  *Решение*: использовать политику ретенции для удаления старых данных по истечении времени или при достижении объема, освобождая место на диске.

- Высокая сложность настройки отказоустойчивости и репликации

*Решение*: использовать инструменты мониторинга и настройки (например, Zookeeper для координации) и регулярно проверять репликацию и распределение нагрузки.

- Высокая задержка при использовании режима высокой гарантии (acks)

  *Решение*: настраивать режимы подтверждения (acks) в зависимости от требований к задержкам и надежности для оптимального баланса.

- Ограничение параллелизма по числу партиций

  *Решение*: при проектировании системы заранее определить необходимое количество партиций, чтобы обеспечить масштабируемость.

- Сложность управления offset'ами

  *Решение*: использовать автоматический commit offset'ов или ручное управление в зависимости от потребностей надежности и нагрузки.