

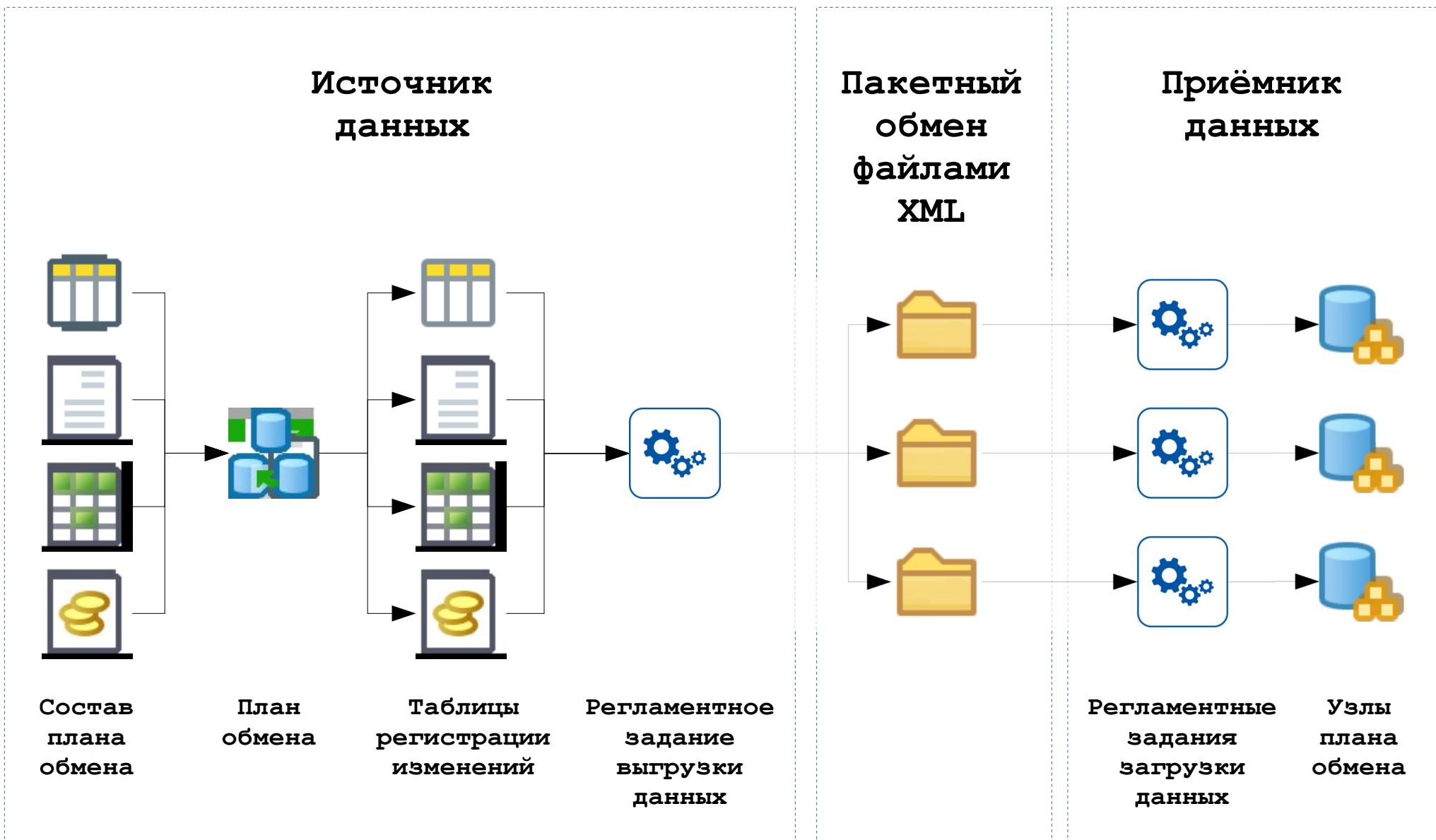
РИБ 2.0

**Альтернатива
планам обмена**

**Методика перехода
на регистры сведений**

<https://zhichkin.github.io/distributed-info-bases-2-0.pdf>

Схема типового обмена данными 1С (РИБ)



Механизм регистрации изменений 1С

Общий модуль "ОбменДаннымиПереопределяемый"

- Функция "ПолучитьПланыОбмена"
- Процедура "ПриКоллизииИзмененийДанных"

Подписки на события "Перед записью" и "Перед удалением" "ОбменДанными[имя плана обмена]ПередЗаписью"

Модуль менеджера плана обмена

Функция "ИспользоватьМеханизмРегистрацииОбъектов"

По умолчанию = ИСТИНА

Общий модуль "ОбменДаннымиСобытия"

- Процедура "МеханизмРегистрацииОбъектовПередЗаписью"
- Функция "ОбъектМодифицированДляПланаОбмена"
- Процедура "ВыполнитьПравилаРегистрацииОбъектовДляПланаОбмена"
- Процедура "ДополнитьПолучателей"
Объект.ОбменДанными.Получатели.Добавить (Элемент) ;
- Процедура "ПроверитьКоллизиюИзмененийДанных"

Альтернативная регистрация изменений должна размещаться в подписках на события "При записи" и "Перед удалением"

Альтернатива планам обмена (РИБ)

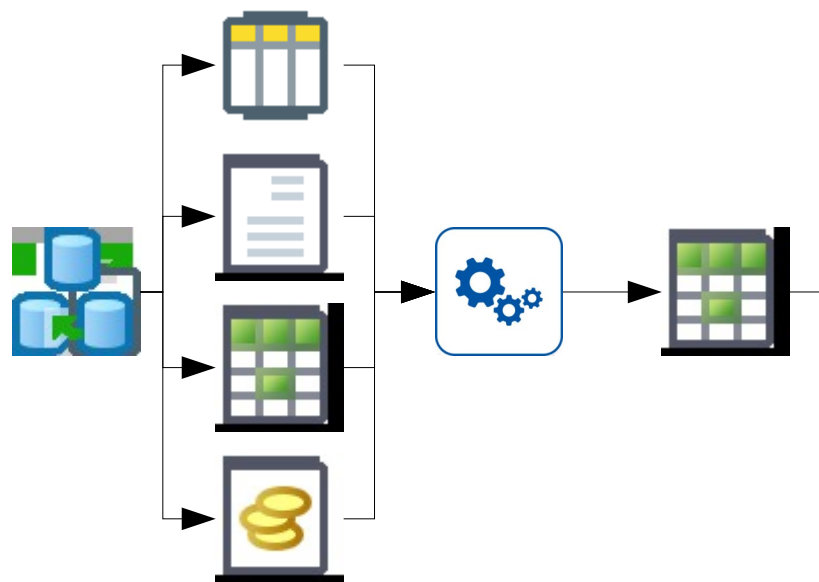
1. Альтернативная планам обмена регистрация изменений
2. Формирование списка получателей с учётом механизмов БСП
3. Структура регистров исходящих и входящих сообщений
4. Архитектура RabbitMQ: топология "звезда", двусторонний обмен
5. Запись в исходящую очередь
6. Гарантия строгого порядка регистрации и доставки данных
7. Представление данных в формате JSON (сериализатор)
8. Чтение из входящей очереди (отравленные сообщения)
9. Эволюция схем данных (прямая и обратная совместимость)
10. Конвертация данных между источником и приёмником
11. Настройка транспорта DaJet Stream (два с половиной скрипта =)
12. Обнаружение и разрешение коллизий данных

Наиболее частые и важные вопросы:

1. Правила регистрации объектов
2. Правила конвертации объектов
3. Гарантия последовательности!
4. Доставка изменений конфигурации и обновления

Схема обмена данными РИБ на RabbitMQ

Источник данных



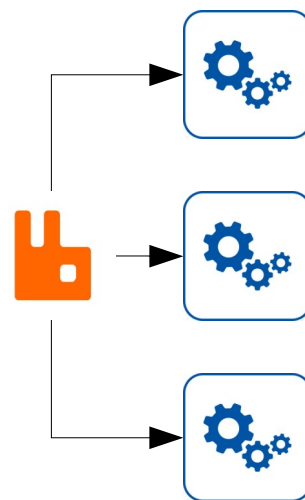
План обмена

Состав плана обмена

Подписки на события объектов

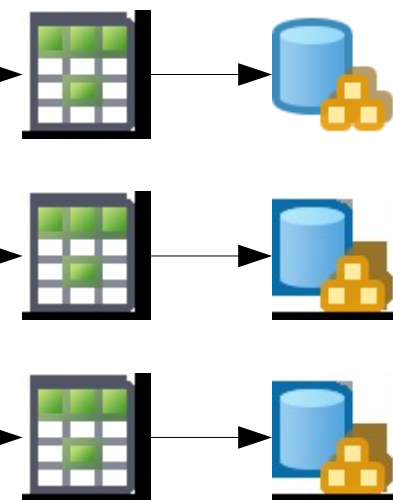
Исходящая таблица очереди

RabbitMQ Топики и очереди



Маршрутизация по типам сообщений и/или кодам получателей

Приёмник данных



Входящие таблицы очереди

Узлы плана обмена

Регистрация изменения данных

Формирование исходящих сообщений

Подписки на события объектов

"ПриЗаписи" и "ПередУдалением"

```
Набор = РегистрыСведений.ИсходящаяОчередь.СоздатьНаборЗаписей();
```

```
Сообщение = Набор.Добавить();
```

```
Сообщение.НомерСообщения = ТекущаяУниверсальнаяДатаВМиллисекундах();
```

```
Сообщение.Отправитель = ПолучитьКодЭтогоУзла();
```

```
Сообщение.Получатели = СформироватьСписокПолучателей(); // csv
```

```
Сообщение.ТипСообщения = Источник.Метаданные().ПолноеИмя();
```

```
Сообщение.ТелоСообщения = СформироватьТелоСообщения(Источник);
```

```
Набор.ОбменДанными.Загрузка = Истина;
```

```
Набор.Записать(Ложь);
```

Всегда только добавление новых записей !!!

Выбор способа регистрации изменений объектов:

1. Отслеживание изменений (ключ объекта)
2. Захват изменения данных (данные объекта)

Формирование исходящих сообщений

Формирование списка получателей

```
МассивУзловПолучателей = Новый Массив();
```

```
Для Каждого Получатель Из Источник.ОбменДанными.Получатели Цикл
```

```
    МассивУзловПолучателей.Добавить (Получатель);
```

```
КонецЦикла;
```

```
    ВЫБРАТЬ УзелОбменаСсылка, КодУзлаДляОбмена  
        ИЗ РегистрСведений.НастройкиОбменаRabbitMQ  
        ГДЕ Узел В (&МассивУзловПолучателей)  
        И ИспользоватьОбменДаннымиRabbitMQ
```

```
Для Каждого УзелОбменаСсылка Из МассивУзловПолучателей Цикл
```

```
    // Отменяем регистрацию изменения в плане обмена !!!  
    Источник.ОбменДанными.Получатели.Удалить (УзелОбменаСсылка);
```

```
КонецЦикла;
```

Чтение входящих сообщений

Загрузка данных объектов
"пакетами" по N сообщений

ВходящиеСообщенияRabbitMQ

⊖  Измерения

 НомерСообщения

⊖  Ресурсы

 Отправитель

 ТипСообщения

 ТелоСообщения

⊖  Реквизиты

 ОписаниеОшибки

 КоличествоОшибок

ВЫБРАТЬ ПЕРВЫЕ 1000

НомерСообщения,

Отправитель,

ТипСообщения,

ТелоСообщения,

КоличествоОшибок

ИЗ

РегистрСведений.ВходящиеСообщения

УПОРЯДОЧИТЬ ПО

НомерСообщения ВОЗР

Чтение входящих сообщений

"Бесконечный" цикл

```
МассивСообщений = ПолучитьМассивСообщений();
```

```
Пока МассивСообщений.Количество() > 0 Цикл
```

```
    Для Каждого СтруктураСообщения Из МассивСообщений Цикл
```

```
        Успех = ОбработатьВходящееСообщение (СтруктураСообщения);
```

```
        Если Не Успех Тогда
```

```
            Возврат;
```

```
        КонецЕсли;
```

```
    КонецЦикла;
```

```
МассивСообщений = ПолучитьМассивСообщений();
```

```
КонецЦикла;
```

Обработка входящего сообщения

```
Если Сообщение.КоличествоОшибок = МаксимумПопытокДляЗагрузки Тогда  
    Возврат; // Блокируем обработку очереди из-за "отравленного" сообщения  
КонецЕсли;
```

```
НачатьТранзакцию();
```

Попытка

```
    Объект = ПолучитьОбъектДанных(Сообщение); // Десериализация данных сообщения
```

```
    Объект.ОбменДанными.Загрузка = Истина;
```

```
    Объект.ОбменДанными.Получатели.Очистить();
```

```
    Объект.ОбменДанными.Получатели.Автозаполнение = Ложь;
```

```
    Объект.ОбменДанными.Отправитель = ОпределитьОтправителя(Сообщение.Отправитель);
```

```
    Объект.ДополнительныеСвойства.Вставить("ОтключитьМеханизмРегистрацииОбъектов");
```

```
    Объект.Записать(); // Предварительно можно определить режим записи
```

```
    ЗафиксироватьПолучениеСообщения(Сообщение); // Необязательные действия
```

```
    УдалитьВходящееСообщение(Сообщение);
```

```
    ЗафиксироватьТранзакцию();
```

Исключение

```
    ОтменитьТранзакцию();
```

```
    ЗафиксироватьОшибкуВходящейОчереди(Сообщение, ОписаниеОшибки());
```

КонецПопытки;

"Отравленное" сообщение

Фиксация ошибки обработки сообщения

Попытка

```
Набор = РегистрыСведений.ВходящаяОчередь.СоздатьНаборЗаписей();  
Набор.Отбор.НомерСообщения.Установить(Сообщение.НомерСообщения);  
Набор.Прочитать();
```

```
Если Набор.Количество() = 0 Тогда  
    Возврат; // Сообщение уже обработали и удалили
```

Иначе

```
    Запись = Набор[0];  
КонецЕсли;
```

```
Запись.ОписаниеОшибки = ТекстОписанияОшибки;  
Запись.КоличествоОшибок = Запись.КоличествоОшибок + 1;
```

```
Набор.Записать();
```

Исключение

```
    ЗаписьЖурналаРегистрации(ОписаниеОшибки());  
КонецПопытки;
```

Структура сообщения RabbitMQ




| | |
|---|--|
| Exchange | РИБ.ERP |
| Routing Key | Справочник.ТестовыйСправочник |
| Redelivered | o |
| Properties | <p>app_id: ЦБ</p> <p>type: Справочник.ТестовыйСправочник</p> <p>message_id: e40f28b4-fcef-f635-4132-c4aa491523f9</p> <p>priority: 0</p> <p>delivery_mode: 2</p> <p>headers: <u>CC: 01</u> 02 PG</p> <p>content_encoding: UTF-8</p> <p>content_type: application/json</p> |
| Payload 221 bytes Encoding: string | {"#type": "jcfg:CatalogObject.ТестовыйСправочник", "#value": {"Ref": "" |


Обмен сообщениями RabbitMQ


Центральная база – периферийный узел


← → ☆ План обмена RabbitMQ


Создать 


| Код | Наименование ↓ | Топик |
|---|----------------|---------|
|  ЦБ | ERP | РИБ.ERP |
|  N-001 | N-001 | |
|  N-002 | N-002 | |


 ИсходящиеСообщенияRabbitMQ


⊖  Измерения


 НомерСообщения

⊖  Ресурсы

 Отправитель

 Получатели

 ТипСообщения

 ТелоСообщения

Сообщение RabbitMQ

▶ Exchange

▶ message_id

▶ app_id

▶ CC

▶ type + Routing Key

▶ Payload

Схема обмена данными РИБ на RabbitMQ

Центральная база – периферийные узлы

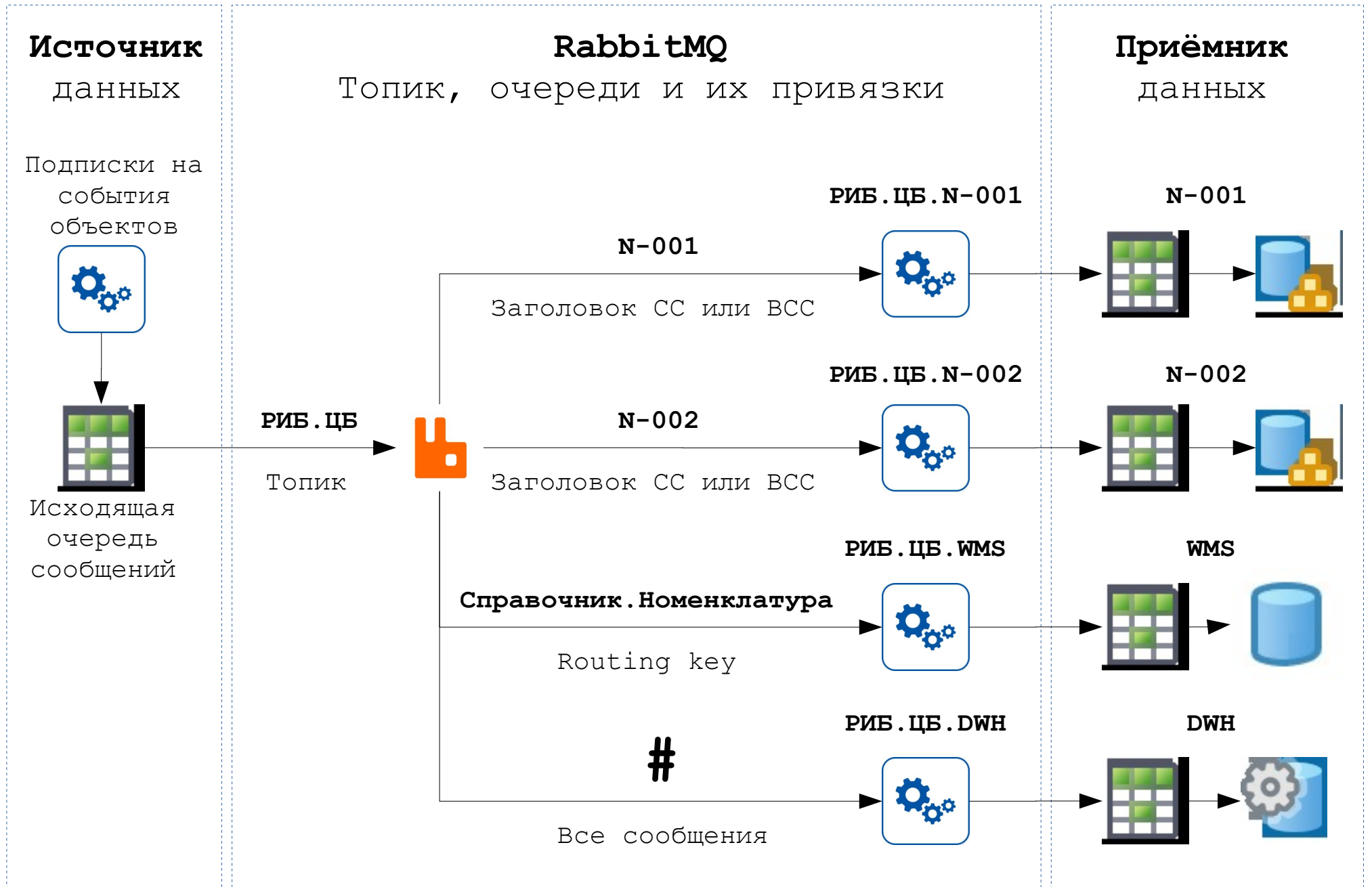
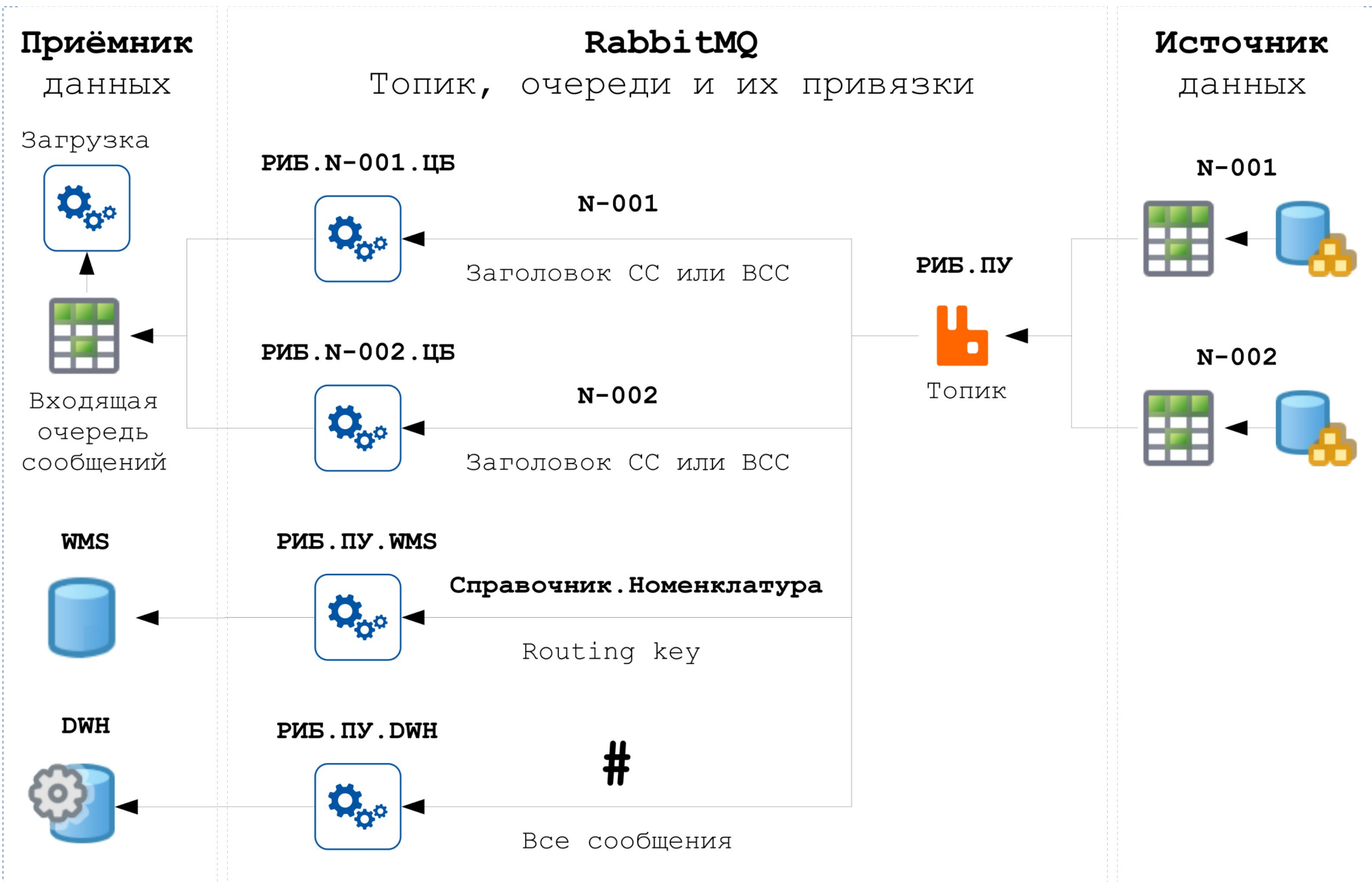


Схема обмена данными РИБ на RabbitMQ

Периферийный узел – центральная база



Отправка сообщений в RabbitMQ

Гарантия: at least once in order

Transactional outbox + polling publisher

```
DECLARE @message object
```

```
USE 'mssql://sql-server/rib-cb'
```

```
CONSUME TOP 1000
```

```
    НомерСообщения,
```

```
    Отправитель, Получатели,
```

```
    ТипСообщения, ТелоСообщения
```

```
INTO @message
```

```
FROM РегистрСведений.ИсходящиеСообщенияRabbitMQ
```

```
ORDER BY НомерСообщения ASC
```

```
PRODUCE 'amqp://guest:guest@localhost:5672/rib'
```

```
SELECT AppId      = @message.Отправитель
```

```
    , MessageId   = @message.НомерСообщения
```

```
    , Exchange    = 'РИБ.ERP'
```

```
    , RoutingKey  = @message.ТипСообщения
```

```
    , CarbonCopy  = @message.Получатели
```

```
    , Type        = @message.ТипСообщения
```

```
    , Body        = @message.ТелоСообщения
```


Приёмка сообщений из RabbitMQ

Однопоточно (для периферийного узла)

```
DECLARE @message object
```

```
USE 'pgsql://postgres:postgres@127.0.0.1:5432/rib-pg'
```

```
CONSUME 'amqp://guest:guest@localhost:5672/rib'
```

```
WITH QueueName = 'РИБ.ЦБ.PG'
```

```
INTO @message
```

```
INSERT РегистрСведений.ВходящиеСообщенияRabbitMQ
```

```
SELECT ДатаВремя = NOW()
```

```
    , НомерСообщения = VECTOR('so_inbox_sequence')
```

```
    , Отправитель = @message.AppId
```

```
    , ТипСообщения = @message.Type
```

```
    , ТелоСообщения = @message.Body
```

Приёмка сообщений из RabbitMQ

Многопоточно (для центрального узла)

```
DECLARE @message object

USE 'mssql://zhichkin/rib-cb'

DECLARE @node object
DECLARE @nodes array =
  SELECT ИмяОчереди = 'РИБ.' + Код + '.ЦБ'
  FROM ПланОбмена.ПланОбменаRabbitMQ
  WHERE Код <> 'ЦБ'

FOR EACH @node in @nodes MAXDOP UNBOUNDED

CONSUME 'amqp://guest:guest@localhost:5672/rib'
  WITH QueueName = @node.ИмяОчереди
  INTO @message

INSERT РегистрСведений.ВходящиеСообщенияRabbitMQ
SELECT ДатаВремя      = NOW()
      , НомерСообщения = VECTOR('so_inbox_sequence')
      , Отправитель    = @message.AppId
      , ТипСообщения   = @message.Type
      , ТелоСообщения  = @message.Body
```

Дополнительные материалы

1. Планы обмена 1С

<https://infostart.ru/1c/articles/899200/>

<https://infostart.ru/1c/articles/1346977/>

2. Обмен данными на регистрах сведений

<https://www.youtube.com/watch?v=ruF6Hp6zqUg>

<https://www.youtube.com/watch?v=pqEQYPzZpMw>

3. Гарантия строгой последовательности

регистрации и доставки изменений данных

<https://www.youtube.com/watch?v=s56E80anUVQ>

4. Формат обмена данными 1С JDTO

<https://infostart.ru/1c/articles/1481155/>

5. Эволюция схем данных (изменение метаданных)

<https://zhichkin.github.io/data-schema-evolution.pdf>

6. Обнаружение и разрешение коллизий данных

<https://infostart.ru/1c/articles/1377988/>

<https://infostart.ru/1c/articles/1469979/>

7. DaJet Stream: интеграционные скрипты (транспорт)

<https://infostart.ru/1c/tools/2077707/>

<https://www.youtube.com/watch?v=JuQYzqD7RwI>