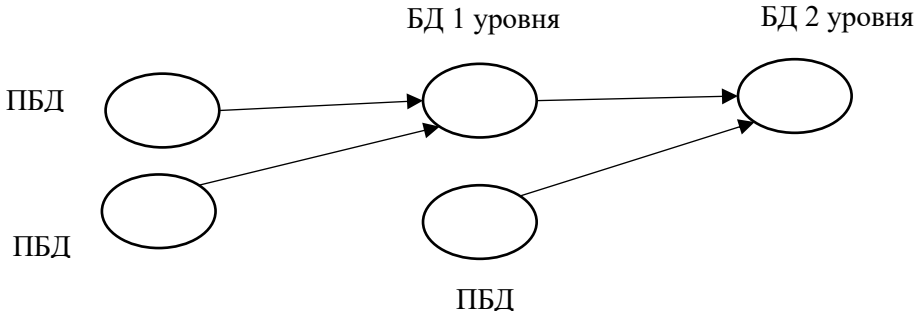
1. **Цель работы**

Изучить технологии тиражирования данных.

2. **Задание**

Схема репликации: Каскадное однонаправленное тиражирование. Изменение (вставка, модификация, удаление), выполненное в одной из ПБД, тиражируется в базу данных следующего уровня.



Запуск репликатора: Через определенный интервал времени в секундах, задаваемый при запуске программы РД.

Разрешение коллизий: В пользу более раннего обновления.

3. **Структура баз данных**

В данной работе перенос данных производится между базами данных, под которыми подразумеваются 5 таблиц, одинаковой структуры и единого содержимого, находящихся в одной схеме данных. Три таблицы являются периферийными базами данных (ПБД) нижнего уровня, одна – базой данных первого уровня (БД 1 уровня), и еще одна – базой данных второго уровня (БД 2 уровня).

Все изменения выполняются только в периферийных базах данных (ПБД) и затем транслируются на следующий уровень.

Были созданы следующие таблицы в схеме pmib6702:

- БД 1 уровня: db_level1;
- БД 2 уровня: db_level2;
- 3 ПБД: pdb1, pdb2, pdb3.

Структура каждой созданной таблицы выглядит следующим образом:

Уникальный идентификатор	Название детали	Город детали	Тип операции	Дата выполнения операции

В PostgreSQL:

n_det	name	town	type_oper	date_oper

4. **Структура журнала изменений**

Так как в схеме репликации каскадного однонаправленного тиражирования pdb1 и pdb2 вносят данные в БД 1-го уровня, а pdb3 в БД 2-го уровня, то для корректной работы, нам понадобится три журнала изменений. Первый журнал нужен для отслеживания коллизий при переносе данных в db_level1, второй – для отслеживания поступивших транзакций в pdb3, и третий – для отслеживания коллизий при переносе данных из db_level1 и pdb3 в db_level2.

Журнал изменений представлен в виде таблицы приложения (DataGridView). В нем отражается информация о каждой операции (вставка\удаление\обновление).

Структура журнала изменений выглядит следующим образом:

Дата и время изменения БД	Название операции и имя БД	Данные до изменения	Данные после изменения

В поле «Данные до изменения» записывается результат операций Вставка и Удаление, а поле «Данные после изменения» остается пустым. Для операции Изменение поле «Данные после изменения» заполняется соответствующей информацией.

В приложении:

Журнал pdb1 и pdb2

Дата и время изменения	Операция	Данные до изменения	Данные после изменения
2019-12-12 14:48:18	Удаление из pdb1	12 Крепление Новосибирск Вставка в pdb1 2019-12-12 14:48:18	
2019-12-12 14:48:24	Обновление в pdb2	1 Винт Новосибирск Обновление в pdb2 2019-12-12 14:47:27	1 Винт Новосибирск Обновление в pdb2 2019-12-12 14:48:24
2019-12-12 14:47:23	Удаление из pdb2	11 Крепление Новосибирск Вставка в pdb2 2019-12-12 14:47:23	
2019-12-12 13:45:45	Удаление из pdb2	10 Трубка Афины Начальная вставка 2019-12-12 13:45:45	

Журнал pdb3

Дата и время изменения	Операция	Данные до изменения	Данные после изменения
12.12.2019 14:49:20	Удаление из pdb3	10 Трубка Афины Начальная вставка 2019-12-12 13:45:45	
12.12.2019 14:49:25	Удаление из pdb3	9 Трубка Рим Начальная вставка 2019-12-12 13:45:45	
12.12.2019 14:49:26	Удаление из pdb3	8 Удлинитель Париж Начальная вставка 2019-12-12 13:45:45	
12.12.2019 14:49:31	Удаление из pdb3	7 Шпилька Лондон Начальная вставка 2019-12-12 13:45:45	

Журнал pdb1,pdb2,pdb3

Дата и время изменения	Операция	Данные до изменения	Данные после изменения
2019-12-12 14:48:18	Удаление из pdb1	12 Крепление Новосибирск Вставка в pdb1 2019-12-12 14:48:18	
2019-12-12 14:48:24	Обновление в pdb2	1 Винт Новосибирск Обновление в pdb2 2019-12-12 14:47:27	1 Винт Новосибирск Обновление в pdb2 2019-12-12 14:48:24
2019-12-12 14:47:23	Удаление из pdb2	11 Крепление Новосибирск Вставка в pdb2 2019-12-12 14:47:23	
2019-12-12 13:45:45	Удаление из pdb2	10 Трубка Афины Начальная вставка 2019-12-12 13:45:45	
		9 Трубка Рим Начальная	

5. Структура журнала содержимого таблиц

Журнал содержимого таблиц представлен в виде текстового файла content_log.txt. В нем находятся исходные данные таблиц (идентичные для каждой). После каждой операции (вставка\удаление\обновление) измененные данные таблиц заносятся в конец файла.

6. Порядок работы с базами данных

Примечание: все действия по вставке, обновлению или удалению строк выполняются в форме транзакций.

Проектируемая система будет выполнять следующие действия:

1. Инициализация данных (ИД);
2. Имитация работы системы (ИРС):
    1) Программа имитации работы системы с определенным интервалом в несколько секунд (интервал задается в начале запуска) моделирует процесс работы информационной системы, выполняя следующие действия:
        - Случайным образом выбирает одну из условных баз данных (pdb1, pdb2, pdb3).
        - Случайным образом выбирает одну из операций (вставка / обновление / удаление).
        - Выполняет выбранную операцию для выбранной базы данных.
3. Происходит блокировка доступа к таблицам для программы имитации работы системы (ИРС).
4. Параллельно с работой программы имитации работы данных (ИРС) запускаем программу репликации данных (РД).
5. Останавливаем работу программ имитации работы данных (ИРС) и репликации данных (РД).
6. Анализируем по журналу изменений и журналу содержимого правильность работы схемы репликации.

7. Инициализация данных (ИД)

В качестве инициализации данных используем SQL-скрипт, запущенный через PhpPgAdmin. Таблицы содержат 10 записей, а также едины по структуре и содержимому.

n_det	name	town	type_oper	date_oper
1	Винт	Париж	Начальная вставка	2019-12-10 23:00:56
2	Винт	Рим	Начальная вставка	2019-12-10 23:00:56
3	Кулачок	Афины	Начальная вставка	2019-12-10 23:00:56
4	Кулачок	Афины	Начальная вставка	2019-12-10 23:00:56
5	Кулачок	Лондон	Начальная вставка	2019-12-10 23:00:56
6	Шпилька	Осло	Начальная вставка	2019-12-10 23:00:56
7	Шпилька	Лондон	Начальная вставка	2019-12-10 23:00:56
8	Удлинитель	Париж	Начальная вставка	2019-12-10 23:00:56
9	Трубка	Рим	Начальная вставка	2019-12-10 23:00:56
10	Трубка	Афины	Начальная вставка	2019-12-10 23:00:56

8. Имитация работы системы (ИРС)

Вставка

В выбранную периферийную базу данных вставляется строка, в которой:

- в поле даты/времени заносится текущее время вставки;
- в поле идентификации операции заносится значение «вставка в БДi», где i = 1,2… .

В журнале изменений (файл данных) запоминается:

- время вставки;
- база данных, в которую выполняется вставка;
- вставленная строка.

Пример выполнения операции

Вставим данную строку в pdb2:

13	Крепление	Новосибирск	Вставка в pdb2	2019-12-11 16:31:39

В журнале изменений она будет выглядеть так:

2019-12-11 16:31:39	Вставка в pdb2	13 Крепление Новосибирск Вставка в pdb2	

Удаление

В выбранной периферийной базе данных удаляется строка с максимальным oid.

Перед удалением в журнале изменений запоминается:

- время удаления;
- база данных, из которой выполняется удаление;
- удаляемая строка.

Пример выполнения операции

Удалим строку из pdb1.

В журнале изменений она будет выглядеть так:

11.12.2019 16:32:46	Удаление из pdb1	12 Крепление Новосибирск Вставка в pdb1 2019-12-11 16:16:34	

Изменение

В выбранной базе данных обновляется строка с минимальным oid, в которой:

- в поле даты/времени заносится текущее время обновления;
- в поле идентификации операции заносится значение «обновление в БДi», где i = 1,2… .

В журнале изменений запоминается:

- время обновления;
- база данных, в которую выполняется обновление;
- старое и новое состояние обновляемой строки.

Пример выполнения операции

Изменим строку из pdb1:

3	Кулачок	Афины	Начальная вставка	2019-12-11 15:16:04

После обновления она примет вид:

3	Кулачок	Новосибирск	Обновление в pdb1	2019-12-11 16:52:16

В журнале изменений она будет выглядеть так:

2019-12-11 16:52:16	Обновление в pdb1	3 Кулачок Афины Начальная вставка 2019-12-11 15:16:04	3 Кулачок Новосибирск Обновление в pdb1 2019-12-11 16:52:16

9. Репликация данных (РД)

Алгоритм репликации строится на основании состояния журнала изменений. Через определенный интервал времени в секундах, задаваемый при запуске, доступ ко всем ПБД блокируется и начинается репликация данных.

9.1. Репликация без коллизий

Последовательность действий:

- Данные в журнале сортируются по времени.
- Данные из журнала pdb1 и pdb2 транслируются в db_level1.
- Данные из журнала pdb3 объединяются с журналом pdb1 и pdb2.
- Объединенные данные сортируются по времени.
- Объединенные данные транслируются в db_level2.
- Происходит остановка программ.
- Фиксируется в журнале содержимого:
  - Текущее время;
  - Содержимое всех таблиц условных БД;

Пример выполнения:

Зададим интервал времени, равный 5сек.

В журнал с pdb1 и pdb2 помещаются данные:

Журнал pdb1 и pdb2

Дата и время изменения	Операция	Данные до изменения	Данные после изменения
12.12.2019 17:17:18	Удаление из pdb2	10 Трубка Афины Начальная вставка 2019-12-12 15:50:35	
12.12.2019 17:17:22	Удаление из pdb2	9 Трубка Рим Начальная вставка 2019-12-12 15:50:35	
12.12.2019 17:17:24	Удаление из pdb2	8 Удлинитель Париж Начальная вставка 2019-12-12 15:50:35	

В журнал pdb3:

Журнал pdb3

Дата и время изменения	Операция	Данные до изменения	Данные после изменения
12.12.2019 17:17:16	Вставка в pdb3	11 Крепление Новосибирск Вставка в pdb3	
12.12.2019 17:17:21	Вставка в pdb3	12 Крепление Новосибирск Вставка в pdb3	

Объединенный журнал:

Журнал pdb1,pdb2,pdb3

Дата и время изменения	Операция	Данные до изменения	Данные после изменения
12.12.2019 17:17:16	Вставка в pdb3	11 Крепление Новосибирск Вставка в pdb3	
12.12.2019 17:17:18	Удаление из pdb2	10 Трубка Афины Начальная вставка 2019-12-12 15:50:35	
12.12.2019 17:17:21	Вставка в pdb3	12 Крепление Новосибирск Вставка в pdb3	
12.12.2019 17:17:22	Удаление из pdb2	9 Трубка Рим Начальная вставка 2019-12-12 15:50:35	
12.12.2019 17:17:24	Удаление из pdb2	8 Удлинитель Париж Начальная вставка 2019-12-12 15:50:35	

Таблица pdb1:

n_det	name	town	type_oper	date_oper
1	Винт	Париж	Начальная вставка	2019-12-12 15:50:35
2	Винт	Рим	Начальная вставка	2019-12-12 15:50:35
3	Кулачок	Афины	Начальная вставка	2019-12-12 15:50:35
4	Кулачок	Афины	Начальная вставка	2019-12-12 15:50:35
5	Кулачок	Лондон	Начальная вставка	2019-12-12 15:50:35
6	Шпилька	Осло	Начальная вставка	2019-12-12 15:50:35
7	Шпилька	Лондон	Начальная вставка	2019-12-12 15:50:35
8	Удлинитель	Париж	Начальная вставка	2019-12-12 15:50:35
9	Трубка	Рим	Начальная вставка	2019-12-12 15:50:35
10	Трубка	Афины	Начальная вставка	2019-12-12 15:50:35

Таблица pdb2:

n_det	name	town	type_oper	date_oper
1	Винт	Париж	Начальная вставка	2019-12-12 15:50:35
2	Винт	Рим	Начальная вставка	2019-12-12 15:50:35
3	Кулачок	Афины	Начальная вставка	2019-12-12 15:50:35
4	Кулачок	Афины	Начальная вставка	2019-12-12 15:50:35
5	Кулачок	Лондон	Начальная вставка	2019-12-12 15:50:35
6	Шпилька	Осло	Начальная вставка	2019-12-12 15:50:35
7	Шпилька	Лондон	Начальная вставка	2019-12-12 15:50:35

Таблица pdb3:

n_det	name	town	type_oper	date_oper
1	Винт	Париж	Начальная вставка	2019-12-12 15:50:35
2	Винт	Рим	Начальная вставка	2019-12-12 15:50:35
3	Кулачок	Афины	Начальная вставка	2019-12-12 15:50:35
4	Кулачок	Афины	Начальная вставка	2019-12-12 15:50:35
5	Кулачок	Лондон	Начальная вставка	2019-12-12 15:50:35
6	Шпилька	Осло	Начальная вставка	2019-12-12 15:50:35
7	Шпилька	Лондон	Начальная вставка	2019-12-12 15:50:35
8	Удлинитель	Париж	Начальная вставка	2019-12-12 15:50:35
9	Трубка	Рим	Начальная вставка	2019-12-12 15:50:35
10	Трубка	Афины	Начальная вставка	2019-12-12 15:50:35
11	Крепление	Новосибирск	Вставка в pdb3	2019-12-12 17:16:12
12	Крепление	Новосибирск	Вставка в pdb3	2019-12-12 17:16:16

Так как в журнале нет коллизий, то db_level1 примет вид:

n_det	name	town	type_oper	date_oper
1	Винт	Париж	Начальная вставка	2019-12-12 15:50:35
2	Винт	Рим	Начальная вставка	2019-12-12 15:50:35
3	Кулачок	Афины	Начальная вставка	2019-12-12 15:50:35
4	Кулачок	Афины	Начальная вставка	2019-12-12 15:50:35
5	Кулачок	Лондон	Начальная вставка	2019-12-12 15:50:35
6	Шпилька	Осло	Начальная вставка	2019-12-12 15:50:35
7	Шпилька	Лондон	Начальная вставка	2019-12-12 15:50:35

A db_level2:

n_det	name	town	type_oper	date_oper
1	Винт	Париж	Начальная вставка	2019-12-12 15:50:35
2	Винт	Рим	Начальная вставка	2019-12-12 15:50:35
3	Кулачок	Афины	Начальная вставка	2019-12-12 15:50:35
4	Кулачок	Афины	Начальная вставка	2019-12-12 15:50:35
5	Кулачок	Лондон	Начальная вставка	2019-12-12 15:50:35
6	Шпилька	Осло	Начальная вставка	2019-12-12 15:50:35
7	Шпилька	Лондон	Начальная вставка	2019-12-12 15:50:35
13	Крепление	Новосибирск	Вставка в pdb3	2019-12-12 17:17:16
14	Крепление	Новосибирск	Вставка в pdb3	2019-12-12 17:17:21

Примечание: Добавляемые в таблицы ключи – уникальные идентификаторы для всех таблиц.

Рассмотрим алгоритм, когда при транслировании выявлены коллизии.

9.2. Репликация с коллизиями

Так как в разных ПБД мы можем изменять данные с уникальными идентификаторами, возникает ситуация, когда не понятно, какую именно запись транслировать в БД уровня i. К примеру, была произведена вставка строки с id=2 в pdb1 и обновление данных с таким же id в pdb2. Что именно транслировать?

В таких случаях решение будет зависеть от времени произведенной операции. Та операция, которая была произведена раньше, и будет транслирована в БД уровня i.

Последовательность действий:

- Данные из журнала pdb1 и pdb2 сортируются по времени и проверяются на коллизии:
    - Если есть коллизии, то разрешаем их в пользу более раннего обновления;
- Данные из журнала pdb1 и pdb2 транслируются в db_level1.
- Данные из журнала pdb3 объединяются с журналом pdb1 и pdb2, сортируются по времени и проверяются на коллизии.
    - Если есть коллизии, то разрешаем их в пользу более раннего обновления;
- Транслируем данные в db_level2.
- Происходит остановка программ.
- Фиксируем в журнале содержимого:
    - Текущее время;
    - Содержимое всех таблиц условных БД;

Пример выполнения:

Зададим интервал времени, равный 10сек.

В журнал с pdb1 и pdb2 помещаются записи:

Журнал pdb1 и pdb2

Дата и время изменения	Операция	Данные до изменения	Данные после изменения
12.12.2019 18:49:40	Обновление в pdb1	1 Винт Париж Начальная вставка 2019-12-12 17:46:57	1 Винт Новосибирск Обновление в pdb1 2019-12-12 18:48:35
12.12.2019 18:49:41	Удаление из pdb2	10 Трубка Афины Начальная вставка 2019-12-12 17:46:57	
12.12.2019 18:49:43	Удаление из pdb2	9 Трубка Рим Начальная вставка 2019-12-12 17:46:57	

В данном журнале коллизий не обнаружено и данные транслируются в db_level1.

Посмотрим на журнал pdb3:

Журнал pdb3

Дата и время изменения	Операция	Данные до изменения	Данные после изменения
12.12.2019 18:49:37	Обновление в pdb3	1 Винт Париж Начальная вставка 2019-12-12 17:46:57	1 Винт Новосибирск Обновление в pdb3 2019-12-12 18:48:32

В журнале pdb3 коллизий нет. Но при транслировании в db_level2, появится коллизия между первой записью журнала pdb1, pdb2, и первой записью журнала pdb3. Разрешим ее в пользу более раннего обновления, т.е. удалим запись из журнала pdb2.

В db_level2 будут переданы записи:

Журнал pdb1,pdb2,pdb3

Дата и время изменения	Операция	Данные до изменения	Данные после изменения
12.12.2019 18:49:37	Обновление в pdb3	1 Винт Париж Начальная вставка 2019-12-12 17:46:57	1 Винт Новосибирск Обновление в pdb3 2019-12-12 18:48:32
12.12.2019 18:49:41	Удаление из pdb2	10 Трубка Афины Начальная вставка 2019-12-12 17:46:57	
12.12.2019 18:49:43	Удаление из pdb2	9 Трубка Рим Начальная вставка 2019-12-12 17:46:57	

Таблица pdb1 примет вид:

n_det	name	town	type_oper	date_oper
2	Винт	Рим	Начальная вставка	2019-12-12 17:46:57
3	Кулачок	Афины	Начальная вставка	2019-12-12 17:46:57
4	Кулачок	Афины	Начальная вставка	2019-12-12 17:46:57
5	Кулачок	Лондон	Начальная вставка	2019-12-12 17:46:57
6	Шпилька	Осло	Начальная вставка	2019-12-12 17:46:57
7	Шпилька	Лондон	Начальная вставка	2019-12-12 17:46:57
8	Удлинитель	Париж	Начальная вставка	2019-12-12 17:46:57
9	Трубка	Рим	Начальная вставка	2019-12-12 17:46:57
10	Трубка	Афины	Начальная вставка	2019-12-12 17:46:57
1	Винт	Новосибирск	Обновление в pdb1	2019-12-12 18:48:35

Таблица pdb2:

n_det	name	town	type_oper	date_oper
1	Винт	Париж	Начальная вставка	2019-12-12 17:46:57
2	Винт	Рим	Начальная вставка	2019-12-12 17:46:57
3	Кулачок	Афины	Начальная вставка	2019-12-12 17:46:57
4	Кулачок	Афины	Начальная вставка	2019-12-12 17:46:57
5	Кулачок	Лондон	Начальная вставка	2019-12-12 17:46:57
6	Шпилька	Осло	Начальная вставка	2019-12-12 17:46:57
7	Шпилька	Лондон	Начальная вставка	2019-12-12 17:46:57
8	Удлинитель	Париж	Начальная вставка	2019-12-12 17:46:57

Таблица pdb3:

n_det	name	town	type_oper	date_oper
2	Винт	Рим	Начальная вставка	2019-12-12 17:46:57
3	Кулачок	Афины	Начальная вставка	2019-12-12 17:46:57
4	Кулачок	Афины	Начальная вставка	2019-12-12 17:46:57
5	Кулачок	Лондон	Начальная вставка	2019-12-12 17:46:57
6	Шпилька	Осло	Начальная вставка	2019-12-12 17:46:57
7	Шпилька	Лондон	Начальная вставка	2019-12-12 17:46:57
8	Удлинитель	Париж	Начальная вставка	2019-12-12 17:46:57
9	Трубка	Рим	Начальная вставка	2019-12-12 17:46:57
10	Трубка	Афины	Начальная вставка	2019-12-12 17:46:57
1	Винт	Новосибирск	Обновление в pdb3	2019-12-12 18:48:32

Таблица db_level1:

n_det	name	town	type_oper	date_oper
2	Винт	Рим	Начальная вставка	2019-12-12 17:46:57
3	Кулачок	Афины	Начальная вставка	2019-12-12 17:46:57
4	Кулачок	Афины	Начальная вставка	2019-12-12 17:46:57
5	Кулачок	Лондон	Начальная вставка	2019-12-12 17:46:57
6	Шпилька	Осло	Начальная вставка	2019-12-12 17:46:57
7	Шпилька	Лондон	Начальная вставка	2019-12-12 17:46:57
8	Удлинитель	Париж	Начальная вставка	2019-12-12 17:46:57
1	Винт	Новосибирск	Обновление в pdb1	2019-12-12 18:49:40

Таблица db_level2:

n_det	name	town	type_oper	date_oper
2	Винт	Рим	Начальная вставка	2019-12-12 17:46:57
3	Кулачок	Афины	Начальная вставка	2019-12-12 17:46:57
4	Кулачок	Афины	Начальная вставка	2019-12-12 17:46:57
5	Кулачок	Лондон	Начальная вставка	2019-12-12 17:46:57
6	Шпилька	Осло	Начальная вставка	2019-12-12 17:46:57
7	Шпилька	Лондон	Начальная вставка	2019-12-12 17:46:57
8	Удлинитель	Париж	Начальная вставка	2019-12-12 17:46:57
1	Винт	Новосибирск	Обновление в pdb3	2019-12-12 18:49:37

10. Выводы

Если выбирать между репликацией изменений (анализ журнала изменений) и репликацией по состоянию (сравнение записей одной таблицы с записями другой), то репликация на основе изменений выигрывает в пользу меньшего объема информации, передаваемой базам высшего уровня, а также быстродействии, из-за отсутствия надобности затрагивать записи, которые остались в базе без модификации на данном цикле транзакций.

Каскадное однонаправленное тиражирование в сравнении с, к примеру, однонаправленным тиражированием будет лучшим выбором, т.к. данные, внесенные в ПБД, транслируются сразу в несколько последующих баз, а не только в одну, а значит, будут с большим шансом сохранены при какой-либо критической ситуации. Но у него есть и свои минусы. При недостаточной бдительности можно допустить удаление (или обновление) большего объема данных, чем ожидалось.