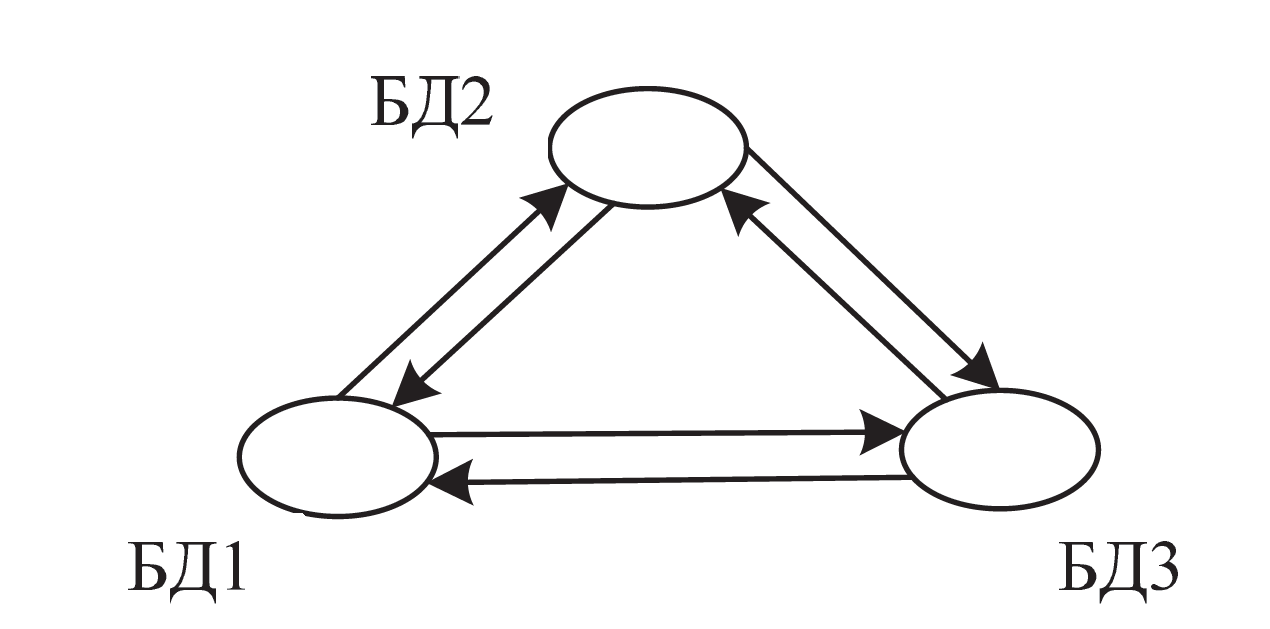
|  |  |  |
| --- | --- | --- |
| Министерство науки и высшего образования  Российской Федерации | | |
| Федеральное государственное бюджетное  образовательное учреждение высшего образования | | |
| «Новосибирский государственный технический университет» | | |
| NSTU | | |
| Кафедра теоретической и прикладной информатики | | |
|  | | |
| Курсовая работа | | |
| по дисциплине «Технологии баз данных» | | |
|  | | |
| **Технологии тиражирования данных** | | |
|  | | |
|  | Факультет: | ПМИ |
| Группа: | ПМИ-81 |
| Вариант: | 11 |
| Студент: | Парышков Дмитрий |
|  |  |
|  |  |
| Преподаватели: | Стасышин Владимир Михайлович |
|  | Стасышина Татьяна Леонидовна |
|  | | |
| Новосибирск | | |
| 2021 | | |

1. **Цель работы**

Изучить технологию тиражирования данных.

1. **Текст задания**
2. Ознакомиться с технологиями тиражирования данных.
3. Изучить схемы репликации данных.
4. Для выбранного варианта спроектировать и программно реализовать модель репликации данных.
5. Оформить отчет и продемонстрировать работу разработанной модели репликации данных.
6. **Вариант задания**

*Схема репликации*- многонаправленное тиражирование. Все базы данных функционально равноправны. Изменение (вставка, модификация, удаление), выполненное в одной из баз данных, тиражируется во все другие базы данных распределенной системы.



*Запуск репликатора* **-** через определенный интервал времени в секундах, задаваемый при запуске программы РД.

*Разрешение коллизий* **-** в зависимости от приоритета базы данных.

1. **Ход выполнения работы**

При использовании схемы много направленного тиражирования все базы данных функционально равноправны. Изменение (вставка, модификация, удаление), выполненное в одной из баз данных, тиражируется во все другие базы данных распределенной системы. Допустим, были внесены изменения в БД1, то в данном случае необходимо произвести тиражирование в БД2 и БД3. После репликации данных все базы данных должны иметь одинаковое содержание. Чтобы контролировать этот процесс задан репликатор, которой запускается через определенный интервал времени в секундах, задаваемый при запуске программы РД. При репликации данных могут возникать коллизии, в нашем случае они решаются в пользу приоритета базы данных.

Приоритет баз данных выглядит следующем образом:

БД1 – маленький приоритет.

БД2 – средний приоритет.

БД3 – большой приоритет.

Все работы с таблицами базы данных будем выполнять в форме транзакций.

Разрабатываемое программное обеспечение состоит из трёх программ:

1. Программа инициализации данных (ИД);

2. Программа имитации работы системы (ИРС);

3. Программа репликации данных (РД).

Далее опишем алгоритмы выполнения каждой из этих программ.

**Тема базы данных – смартфоны.**

Структура базы данных максимально упрощена, т.к. огромное количество характеристик устройств в нашем случае не требуется. Будет достаточно 5-8 атрибутов.

Каждая база данных (БД1, БД2, БД3) будет представлена в виде таблицы со следующей структурой:

* Title – название смартфона.
* Company – название компании.
* Memory – объем памяти устройства.
* Cpu – процессор.
* Year – год выпуска.
* battary\_vol – объем батареи.
* Operation – последняя операции с данным элементом таблицы.
* op\_date – время последней операции.

Скрипт для создания таблицы:

CREATE TABLE table\_bd1(

title character(20) NOT NULL,

company character(20) NOT NULL,

memory integer NOT NULL,

cpu character(20) NOT NULL,

year integer NOT NULL,

battary\_vol integer NOT NULL,

operation character(30),

op\_date timestamp without time zone NOT NULL

) with oids;

Скрипт для заполнения таблицы:

INSERT INTO table\_bd1 VALUES

('Iphone 11','Apple','128','A13','2019','1850','initial insert', date\_trunc('second', current\_timestamp) AT TIME ZONE '-7 UTC'),

('Iphone X','Apple','64','A10','2017','1800','initial insert', date\_trunc('second', current\_timestamp) AT TIME ZONE '-7 UTC'),

('Iphone 12','Apple','256','A14','2020','2010','initial insert', date\_trunc('second', current\_timestamp) AT TIME ZONE '-7 UTC'),

('Pixel 3','Google','128','Snapdragon 845','2018','1900','initial insert', date\_trunc('second', current\_timestamp) AT TIME ZONE '-7 UTC'),

('Pixel 4','Google','128','Snapdragon 855','2019','2100','initial insert', date\_trunc('second', current\_timestamp) AT TIME ZONE '-7 UTC'),

('Iphone 13','Apple','512','A15 Bionic','2021','2590','initial insert', date\_trunc('second', current\_timestamp) AT TIME ZONE '-7 UTC'),

('Pixel 5','Google','128','Snapdragon 765G','2020','2150','initial insert', date\_trunc('second', current\_timestamp) AT TIME ZONE '-7 UTC'),

('Pixel 6','Google','256','Google Tensor','2021','2700','initial insert', date\_trunc('second', current\_timestamp) AT TIME ZONE '-7 UTC'),

('Iphone 8','Apple','32','A10','2017','1980','initial insert', date\_trunc('second', current\_timestamp) AT TIME ZONE '-7 UTC'),

('P30','Huawei','128','HiSilicon Kirin 980','2018','2560','initial insert', date\_trunc('second', current\_timestamp) AT TIME ZONE '-7 UTC'),

('P40','Huawei','128','HiSilicon Kirin 990','2019','3400','initial insert', date\_trunc('second', current\_timestamp) AT TIME ZONE '-7 UTC'),

('P50','Huawei','256','HiSilicon Kirin 1000','2020','4000','initial insert', date\_trunc('second', current\_timestamp) AT TIME ZONE '-7 UTC');

Данные действия также проводятся для создания и заполнения таблиц table\_bd2 и table\_bd3

*Вид исходной таблицы в PhpMyAdmin:*

Изображение выглядит как стол

Автоматически созданное описание

Для каждой из трёх баз данных необходимо фиксировать изменения, которые были в ней произведены, для дальнейшей обработки репликатором.

Структура таблицы изменений Log:

* num\_chan – номер изменения.
* title – комбинация названия устройства и компании даёт ему уникальный идентификатор.
* Companyю.
* Operation – производимая операция.
* Date – время и дата производимой операции.

Данная таблица создаётся в программе.

Следующий шаг – реализация программы-симулятора. Программа написана на языке ESQL/C в виде консольного приложения.

*Алгоритм работы программы:* случайным образом выбирается одна трёх баз данных, далее также случайным образом выбирается одно из действий: вставка, обновление, удаление. Если выполняется операция обновления, то в выбранной базе данных обновляется строка с минимальным oid. Если выполняется операция удаления, то в выбранной базе данных удаляется строка с максимальным oid. Данная операция выполняется и заносится в журнал изменений Log.

После определённого количества операций и времени (их число и количество задаёт пользователь), запускается программа репликации и базы данных блокируется для их синхронизации. Алгоритм репликации, реализованный в курсовой работе, основывается на основании состояния журнала изменений (репликации изменений). Программа репликации обрабатывает журнал изменений Log и выполняет процесс синхронизации. При выполнении репликации могут возникать коллизии:

***Вставка*:** Проблема в случае со вставкой заключается в том, что строка, которую необходимо вставить уже имеется в остальных таблицах, но дата её изменения позже, чем дата для вставки. Тогда придётся просто обновить эту же строку в таблицах, где она уже имеется.

**Удаление:** В одной из таблиц может не оказаться строки, которую по ходу репликации необходимо будет удалить. Тогда придётся предварительно проверять, имеется ли эта строка в таблице, прежде чем предпринимать попытку удаления отсутствующей строки. Может также возникнуть ситуация, когда запись удалена более поздним запросом.

**Обновление:** Если две разных таблицы изменяют одни и те же данные, приоритет отдаётся базе данных с более высоким приоритетом.

1. **Результат работы программы**

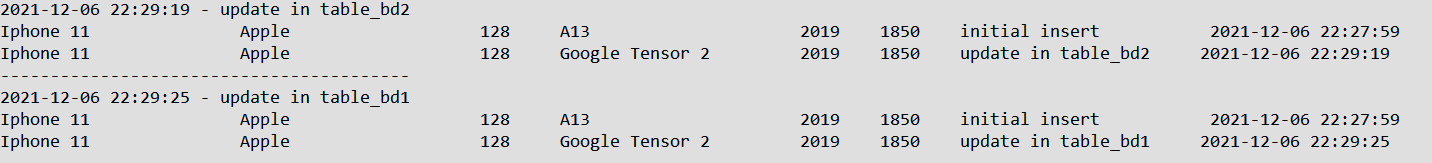
Во всех случаях все таблицы изначально в исходном состоянии.

Для начала произведем все операции по отдельности.

**Изменение – программа произведет две случайные модификации и выполнит репликацию**

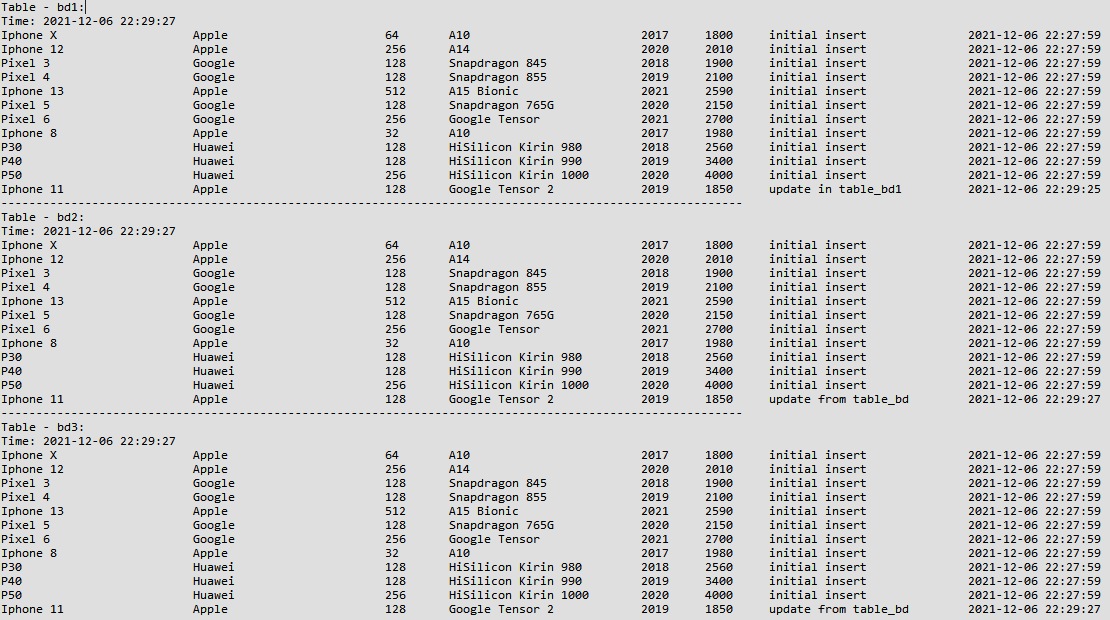
*Журнал изменений:*

*(первая строка – старая строка, вторая строка - модифицированная)*



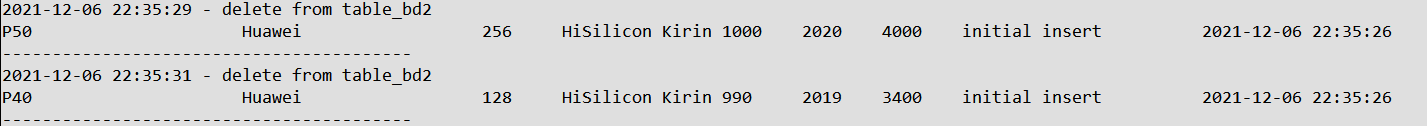
Как можно видеть, возникла коллизия, две таблицы обновили одни и те же данные, в данном случае приоритет был отдан базе данных – table\_bd2

*Состояние таблиц после репликации:*

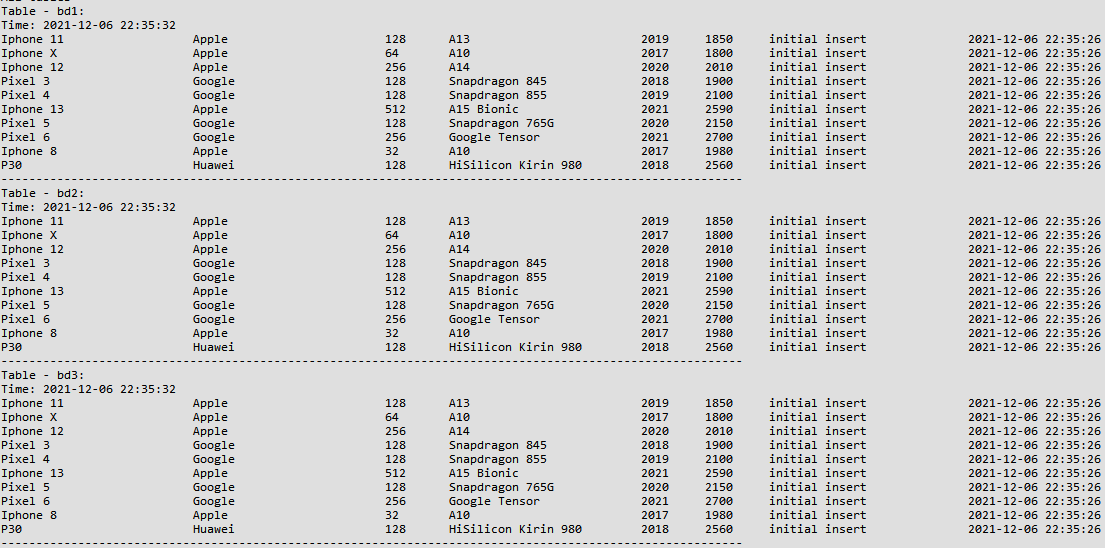


**Удаление – программа удалит две строки с максимальным IOD из случайных таблиц**

*Журнал изменений:*

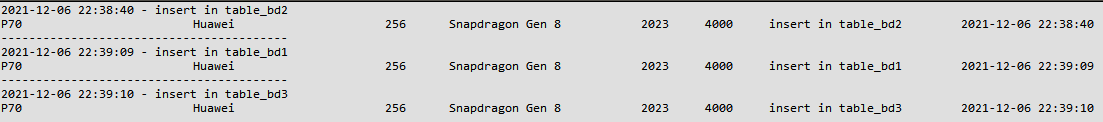


*Состояние таблиц после репликации:*



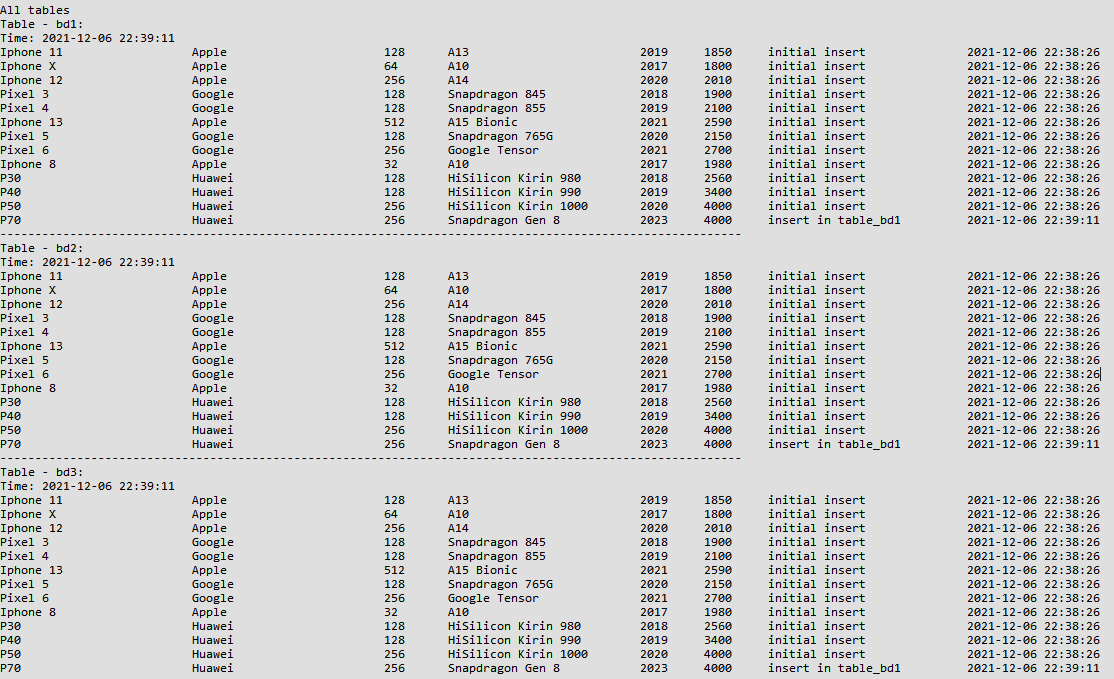
**Вставка – программа произведёт три случайные вставки**

*Журнал изменений:*



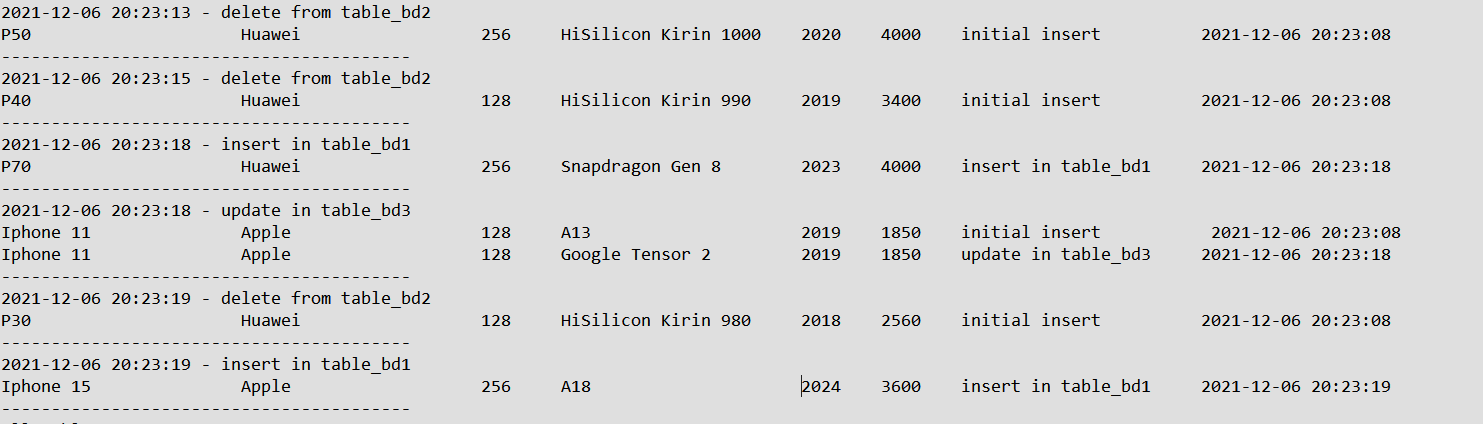
Возникла коллизия, во все три таблицы уже была выполнена вставка, в данном случае выбирается вставка из приоритетной базы данных и в других базах данных обновляется только время и место вставки (место – база данных номер три). В данном случае можно было бы оставить все три записи со своим временем в каждой базе данных, но мы решили сделать немного по-другому.

*Состояние таблиц после репликации:*

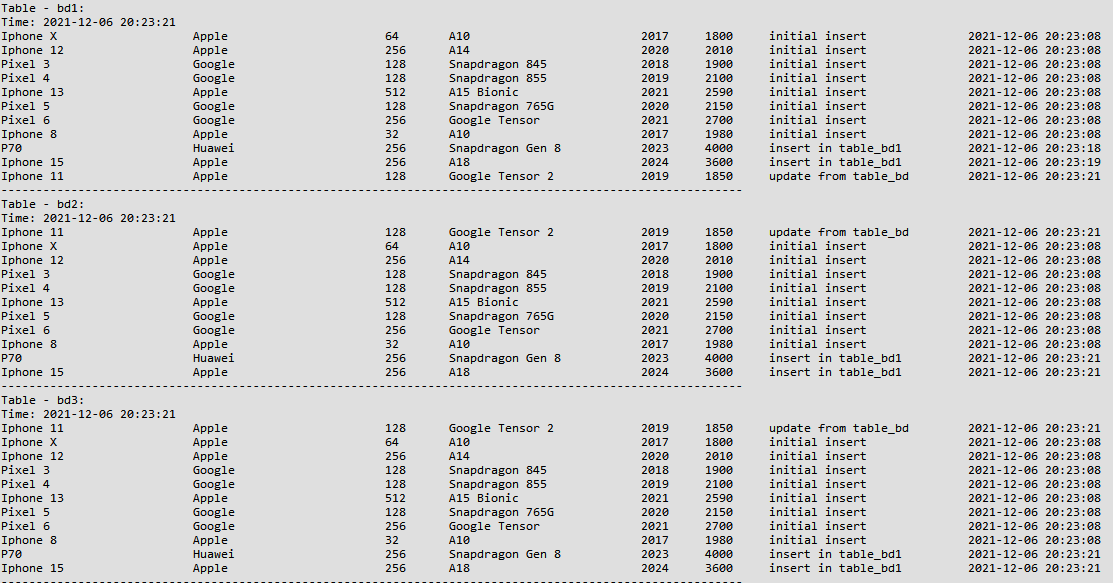


***Полная работа программы с множеством различных операций:***

*Журнал изменений:*



*Состояние таблиц после репликации:*



1. **Вывод**

В результате выполнения работы была разработана программа, позволяющая инициализировать схему баз данных для много направленного тиражирования, имитировать работу с реальными базами данными в виде таблиц, а также выполнять периодическую репликацию данных с разрешением коллизий в зависимости от приоритета базы данных. При репликации данных может происходить потеря данных, но после репликации, все таблицы содержат одинаковую информацию.