**Введение**

Цель данной дипломной работы заключалась в разработке и внедрении серверного приложения для управления базой данных, что включало проектирование структуры БД, настройку конфигурационных параметров, а также проведение тестирования системы.

Основные задачи, которые решались в рамках работы:

1. Проектирование и создание структуры базы данных с использованием современных инструментов.
2. Настройка и оптимизация работы базы данных.
3. Отладка и обеспечение безопасности и доступности данных в системе.
4. Проведение тестирования системы для оценки её производительности и надёжности.

В результате проведенной работы удалось достичь:

1. Создания эффективной структуры базы данных, включая таблицы, индексы, последовательности, представления, функции и хранимые процедуры.
2. Настройки логирования и мониторинга базы данных.
3. Организации процесса резервного копирования и восстановления данных.
4. Разработки механизма переключения реплики базы данных при отказе основного сервера.

**1. Описание структуры базы данных**

**1.1 Таблицы и поля.**

Ниже представлена структура интернет-магазина:

1. City.

id **int4**

city **varchar**(50)

1. remaining\_products.

id **int4**

id\_product **int4**

id\_city **int4** **L**

quantity\_in\_stock **int4**

1. product.

id **int4**

product\_name **varchar**(255

product\_category **varchar**(100)

price **numeric**(10, 2)

quantity **int4**

1. basket

id **int4**

id\_buyer **int4**

id\_product **int4**

product\_quantity **int4**

1. customers

id **int4**

**"FIO"** **varchar**(100)

address **varchar**(255)

**"phone number"** **varchar**(20)

email **varchar**(50)

1. reviews

id **int4**

id\_product **int4**

id\_buyer **int4**

rating **int4**

**"comment"** **text**

1. orders

id **int4**

**"order date"** **date**

**"order status"** **varchar**(50)

id\_buyer **int4**

**CONSTRAINT** **"Заказы\_pkey"** **PRIMARY** **KEY** (id)

1. provider

id **int4**

сompany\_name **varchar**(255)

contact\_person **varchar**(100)

phone\_number **varchar**(20)

email **varchar**(50)

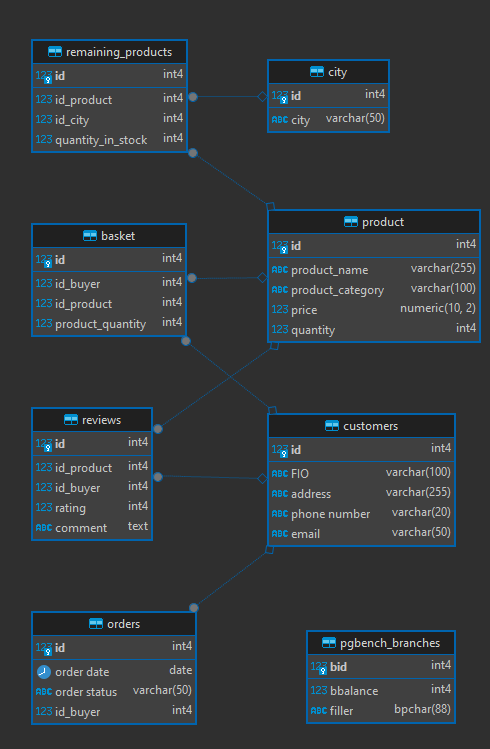
1. pgbench\_branches

bid **int4**

bbalance **int4**

filler **bpchar**(88)

**1.2 ER-диаграмма:**



**2.Типы объектов в базе данных**

**2.1 Индексы**. По предполагаемому количеству заказов в сутки от 1000 до 5000 в разных городах, были созданы индексы исполняющие функции улучшения скорости выполнения запросов.

**CREATE** **INDEX** idx\_basket\_id\_customer **ON** public.basket **USING** btree (id\_buyer);

**CREATE** **INDEX** idx\_product\_category **ON** public.product **USING** btree (product\_category);

**CREATE** **INDEX** idx\_customer\_id **ON** public.customers **USING** btree (id);

**CREATE** **INDEX** idx\_reviews\_id\_product **ON** public.reviews **USING** btree (id\_product);

**CREATE** **INDEX** idx\_order\_data **ON** public.orders **USING** btree (**"order date"**);

**CREATE** **INDEX** idx\_orders\_id\_customer **ON** public.orders **USING** btree (id\_buyer);

**2.2 Хранимые процедуры и функции.**

Функции и хранимые процедуры в SQL, как и в любом другом языке программирования, обеспечивают возможность повторного использования и гибкость. В данной работе были созданы:

1. Добавление нового товара;
2. Обновление информации о товаре;
3. Удаление товара;
4. Оформление заказа;
5. Создание записи о заказе;
6. Добавление товаров в заказ;
7. Подсчёт общей стоимости заказа.

**2.3 Роли.**

Для создания ролей был написан скрипт на python, который генерирует случайный пароли и создаёт пользователя. После создания user присваивает к определённой роли. На данный момент grant для ролей были заполнены вручную. Ниже представлены роли (скрип)

1. marketer
2. analyst
3. designer
4. engineer
5. seospecialist
6. worker
7. thirdpartyservice
8. admin

**2.4 Табличные представления.**

Для выгрузки списка товаров в XML-формате было создано табличное представление generate\_xml для вызова процедуры generate\_and\_download\_xml

**CREATE** **OR** **REPLACE** **VIEW** generate\_xml **AS**

**SELECT** r.id, p.product\_name, r.quantity\_in\_stock

**FROM** public.remaining\_products r

**JOIN** public.product p **ON** p.id = r.id\_product;

**3.Конфигурация.**

**3.1 Логирование.**

SQL-операций:

* log\_destination = ‘csvlog’ # Журнал будет записываться в формате CSV
* logging\_collector = on # Включение сборщика записей
* log\_directory = ‘pg\_log’ # Каталог, в котором будет сохранен журнал (его необходимо предварительно создать)
* log\_filename = ‘postgresql-%Y-%m-%d\_%H%M%S.log’ # Шаблон имени файла журнала
* log\_statement = ‘all’ # Журналирование всех SQL-операций
* log\_min\_duration\_statement = 500 # Журналирует операции, выполняющиеся дольше указанного времени в миллисекундах.

Postgres:

* log\_connections = on # Эта настройка определяет, будут ли журналироваться попытки установки соединения с базой данных.
* log\_duration = on # Включение записи продолжительности выполнения каждого SQL запроса в журнал. Полезно для мониторинга и оптимизации производительности.
* log\_line\_prefix = '%m [%p] %q%u@%d ' #Данная настройка используется для форматирования каждой строки журнала.
* log\_statement = 'ddl' #Эта настройка определяет какие операторы SQL будут журналироваться. Оператор "ddl" указывает журналировать только операции изменения схемы (Data Definition Language), такие как создание, изменение и удаление таблиц и индексов.
* log\_timezone = 'Etc/UTC' #Устанавливает временную зону для журнальных записей. В данном случае, журнал будет отображать временную зону "Etc/UTC" (Скоординированное мировое время).

**3.2 Тестирование.**

Для настройки репликации и автоматического переключения при падении мастера с использованием репликационного менеджера repmgr, я использовала:

* Основной сервер (master): 192.168.68.105
* Реплицированный сервер (slave): 192.168.68.106

**Шаг 1: Установка PostgreSQL и repmgr**

Установила repmgr на обоих серверах:

* sudo apt-get update
* sudo apt install repmgr

**Шаг 2: Основная настройка на мастере (192.168.68.105)**

**Настройка PostgreSQL**

Отредактировала файл конфигурации postgresql.conf:

* Настройки указаны ниже в п. Архивирование.

Добавила следующие строки для разрешения подключения к slave в файл pg\_hba.conf:

*# Allow replication connections from the slave server*

* host    replication     admin           192.168.68.106/24         md5
* host all all 0.0.0.0/0 md5

Перезапустила PostgreSQL для применения изменений.

**Настройка repmgr.conf**

/etc/repmgr/15/repmgr.conf:

* node\_id=1
* node\_name='master'
* conninfo='host=192.168.68.105 user=admin dbname=Diplom connect\_timeout=2'
* data\_directory='/var/lib/postgresql/15/main'

**Подключение к БД и пароли прописаны .pgss**

* postgres@student:~$ repmgr -f /etc/repmgr/15/repmgr.conf primary register
* INFO: connecting to primary database...
* NOTICE: attempting to install extension "repmgr"
* NOTICE: "repmgr" extension successfully installed
* NOTICE: primary node record (ID: 1) registered

**Шаг 3: Настройка слейва (192.168.68.106)**

1. **Настройка реплики.**

* vi postgresql.conf:
* listen\_addresses = '\*'
* hot\_standby = on

Добавила следующие строки для разрешения подключения к master в файл pg\_hba.conf:

*# Allow replication connections from master and repmgr connections*

* host replication replicator 192.168.68.105/32 md5
* host all all 0.0.0.0/0 md5

1. **Клонирование мастера.**

* repmgr -h 192.168.68.105 -U admin -d Diplom -f /etc/repmgr/15/repmgr.conf standby clone --force
* sudo systemctl start postgresql

1. **Регистрация реплики**

* sudo -i -u postgres
* repmgr -f /etc/repmgr/15/repmgr.conf standby register

**Шаг 4: Настройка автоматического переключения**

Для автоматического failover на обоих серверах настраивается repmgrd:

* failover=automatic
* promote\_command='repmgr standby promote -f /etc/repmgr/15/repmgr.conf --log-to-file'
* follow\_command='repmgr standby follow -f /etc/repmgr/15/repmgr.conf --log-to-file --upstream-node-id=%n'

Запуск repmgrd на каждом сервере:

* sudo -i -u postgres
* repmgrd -f /etc/repmgr/15/repmgr.conf –daemonize

**Шаг 5: Проверка настройки**

Проверка статуса репликации и установки кластеров:

* repmgr cluster show

Role | Name | Upstream | Connection String

---- | ---- | -------- | -----------------

\* master | node1 | |

standby | node2 | node1 | host=192.168.68.105 user=admin dbname=Diplom connect\_timeout=2

**3.3 Архивирование.**

Прогнозирование требуемого места на диске для хранения резервных копий и WAL-логов может зависеть от нескольких факторов, таких как объем данных, кратность резервирования и политика хранения. Вот примерный прогноз, основанный на предоставленной информации:

**1. Ежедневные данные заказов:**

Предположим, что среднее количество заказов составляет 3000 в день. Если каждый заказ имеет средний размер в 1 МБ (включая все прикрепленные файлы, изображения, детали заказа и т.д.), то ежедневно будет создаваться 3000 МБ (или 3 ГБ) новых данных.

**2. Резервные копии:**

Для интернет-магазина важно регулярно создавать резервные копии для обеспечения безопасности данных. Рекомендуется делать регулярные полные бэкапы базы данных, а также инкрементальные бэкапы WAL-логов.

Частота создания резервных копий может быть задана каждые 24 часа. Размер полной резервной копии будет примерно равен размеру данных, с учетом приращений внутри этого периода времени. Размер инкрементальных бэкапов будет зависеть от объема изменений в базе данных, которые произошли с момента предыдущего бэкапа.

Предположим, что размер полной резервной копии равен 3 ГБ (см. пункт 1) и размер инкрементального бэкапа WAL-логов составляет 100 МБ в сутки. При создании ежедневного полного бэкапа и ежедневного инкрементального бэкапа в течение года (365 дней), общий размер резервных копий будет примерно равен:

= (3 ГБ \* 365) + (100 МБ \* 365)  
= 1095 ГБ + 36.5 ГБ  
≈ 1131.5 ГБ или около 1.1 ТБ

**3. Хранение и удаление устаревших копий и WAL-логов:**

Важно установить политику хранения и удаления старых резервных копий и WAL-логов в соответствии с требованиями вашего бизнеса и сохранности данных.

Рекомендуется сохранять несколько последних полных копий, чтобы можно было восстановить данные в случае повреждения или потери. Удаление устаревших копий может осуществляться, например, через 1 месяц (настроены 40 дней) после их создания.

WAL-логи также можно хранить определенное время. Они используются для восстановления данных до определенного момента времени. После завершения символьного восстановления к точке во времени, соответствующей последней резервной копии, устаревшие WAL-логи могут быть удалены.

Важно сконфигурировать Politicy-Based Retention (PBR) для управления удалением старых копий и WAL-логов в соответствии с вашими требованиями по сохранности данных и пространству на диске.

1. **Настройка конфигурации WAL archive:**

wal\_level = archive

fsync = on

wal\_buffers = 32MB

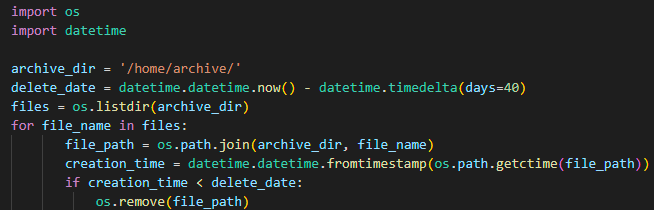
max\_wal\_size = 1GB

min\_wal\_size = 80MB

archive\_mode = on

archive\_command = archive\_command = 'rsync -av %p /home/archive/%f && find /home/archive -type f -name "\*.gz" -mtime +40 -delete'

1. **Автоматическое удаление python:**



Данный скрипт можно добавить в планировщик задач такой как cron, чтобы он выполнял регулярно. Так же требуется настроить права доступа к директории архива, чтобы пользователь имел доступ для удаления файлов.

**3.4 Резервное копирование.**

Для резервного копирования используется pg\_dump. Эта команда создаст резервную копию базы данных Diplom и сохранит её в файл backup.sql

* pg\_dump -h 192.168.68.105 -U admin -d Diplom -w > ./backup/backup\_$(date +"%Y-%m-%d").sql

Данное копирование происходит автоматически, с помощью планировщика задач cron.