Цель работы

Получить навыки обеспечения конфиденциальности баз данных методом шифрования.

Формулировка задания

1. Для своего варианта выполнения лабораторных работ определите приватные (требующие шифрования):

a. По крайней мере одно отношение;

b. По крайней мере 2 столбца разного типа данных в отношении, в котором также будут присутствовать незашифрованные данные.

2. Реализовать шифрование в СУБД (для всех допустимых в бесплатной версии режимах), включая:

a. Шифрование столбцов таблицы (см.п.1)

b. Шифрование всей таблицы (см.п.1)

c. Шифрование файлов данных на диске (и только)

d. Шифрование данных при передаче.

e. Сквозное шифрование всех объектов, включая данные в памяти (если возможно).

3. При использовании штатных режимов шифрования, укажите, какие задачи из перечисленных они позволяют решить.

4. Укажите для каждой из задач 2.a – 2.e возможно ли ее решение с разным уровнем конфиденциальности (для пользователей с разным доступом). Реализуйте примеры где это возможно.

5. Для каждого практического примера оцените изменение производительности и объема данных, проведя для измерения производительности 6-10 тестов на объеме данных, обеспечивающим должный уровень нагрузки (задержку не менее нескольких секунд).

Описание варианта №1:

Данные	Пользователи
Больница	

Врач (персональные данные включая ФИО и номер(а) документа(ов), специальность, квалификация, стаж, предыдущие места работы, рейтинг, зарплата) **Пациент** (персональные данные включая ФИО и номер(а) документа(ов), тип страховки, рейтинг) **История болезни** (приемы у врачей, диагнозы, анализы, назначения) **Лабораторный журнал** (пациент, врач, анализы, дата сдачи, дата получения, значения измеренных показателей, стоимость) **Расписание приемов** (врач, пациент, кабинет, стоимость, начало приема, конец приема) **Кабинеты** (тип кабинета, номер, специализации принимающих врачей)	Врач Пациент Администратор

Ход работы

1. Определение приватных полей

Для шифрования была выбрана таблица *doctor*. Это обусловлено тем, что данные о докторах являются персональными и требуют защиты. В качестве защищаемых столбцов были выбраны столбцы *salary* и *qualification*.

2. Реализация шифрования в СУБД

a. Шифрование столбцов таблицы

Модуль *pgcrypto* позволяет хранить в зашифрованном виде избранные поля. Установка *pgcrypto:*

```
CREATE EXTENSION IF NOT EXISTS pgcrypto;
```

Для шифрования данных использовались функции *pgp_sym_encrypt()*, *pgp_sym_decrypt()* из этого модуля. Это функции симметричного шифрования, по умолчанию используется алгоритм *AES128*.

Выбранные ранее столбцы каждого кортежа отношения *doctor* были зашифрованы и расшифрованы. Для типов данных отличных от символьных пришлось создавать вспомогательный столбец для хранения в зашифрованном виде. Результаты приведены на *рисунках 1, 2*. На *рисунке 3* представлены функции зашифрования и расшифрования.

Data Output | Explain | Messages | Notifications

	passport [PK] character var	fio text	rating real	qualification qualification_type	experience integer	salary integer	encrypt_salary text	encrypt_qualification text
1	1	doctor1	4.862504	highest	6	-1	\xc30d0407030227d59fd327...	\xc30d040703028a88269c3ab...
2	2	doctor2	1.147986	highest	7	-1	\xc30d0407030215685efe61...	\xc30d04070302364cbc07db7...
3	3	doctor3	8.36608	highest	0	-1	\xc30d040703025617011932...	\xc30d04070302db63d375467...
4	4	doctor4	3.9618855	highest	8	-1	\xc30d0407030215008f771a...	\xc30d04070302374ead33fca4...
5	5	doctor5	8.88909	highest	6	-1	\xc30d04070302e778423cff7...	\xc30d04070302f66303a6a7cc...
6	6	doctor6	0.98011255	highest	8	-1	\xc30d0407030295b7693a38...	\xc30d040703026cd56b45fdee...
7	7	doctor7	7.159309	highest	7	-1	\xc30d040703029a159c6b5b...	\xc30d040703024db40b605f57...
8	8	doctor8	7.9670796	highest	2	-1	\xc30d04070302a3f73506bb...	\xc30d04070302d818e1e50c7...
9	9	doctor9	6.7669935	highest	8	-1	\xc30d040703022289451b14...	\xc30d04070302254b47ca6c4...
10	10	doctor10	3.6266928	highest	1	-1	\xc30d0407030282f2c6030f7...	\xc30d0407030254cb4c49fae5...
11	11	doctor11	5.400999	highest	4	-1	\xc30d0407030247e5b4ff28f...	\xc30d040703026e46b491432...

Рисунок 1 – Шифрование отдельных столбцов

Data Output | Explain | Messages | Notifications

	passport [PK] character var	fio text	rating real	qualification qualification_type	experience integer	salary integer	encrypt_salary text	encrypt_qualification text
1	1	doctor1	4.862504	highest	6	69601	[null]	[null]
2	2	doctor2	1.147986	second	7	98935	[null]	[null]
3	3	doctor3	8.36608	first	0	97148	[null]	[null]
4	4	doctor4	3.9618855	second	8	28370	[null]	[null]
5	5	doctor5	8.88909	highest	6	52730	[null]	[null]
6	6	doctor6	0.98011255	second	8	67607	[null]	[null]
7	7	doctor7	7.159309	highest	7	87919	[null]	[null]
8	8	doctor8	7.9670796	second	2	15927	[null]	[null]
9	9	doctor9	6.7669935	first	8	37695	[null]	[null]
10	10	doctor10	3.6266928	second	1	91088	[null]	[null]
11	11	doctor11	5.400999	highest	4	86889	[null]	[null]

Рисунок 2 – Расшифрование отдельно зашифрованных столбцов

```
-- Добавление столбцов для несимвольных данных
ALTER TABLE doctor ADD COLUMN encrypt_salary text;
ALTER TABLE doctor ADD COLUMN encrypt_qualification text;

-- Шифрование отдельных столбцов
UPDATE doctor
SET encrypt_salary = pgp_sym_encrypt(salary::TEXT, 'secretPass'),
    encrypt_qualification = pgp_sym_encrypt(qualification::TEXT, 'secretPass'),
    salary = -1,
    qualification = 'highest';

-- Расшифрование отдельных столбцов
UPDATE doctor
SET salary = pgp_sym_decrypt(encrypt_salary::bytea, 'secretPass')::int,
    qualification = pgp_sym_decrypt(encrypt_qualification::bytea, 'secretPass')::qualification_type,
    encrypt_salary = NULL,
    encrypt_qualification = NULL;
```

Рисунок 3 – Функции зашифрования и расшифрования отдельных столбцов

b. Шифрование всей таблицы

В *PostgreSQL* нет режима шифрования всей таблицы, поэтому он был реализован путем шифрования каждого столбца аналогично предыдущему пункту. Однако есть некоторые ограничения. Если первичный ключ, не содержит

чувствительных данных, можно его оставить в открытом виде. Либо необходимо поменять все ссылки на поля таблицы. Результаты приведены на *рисунках 4, 5*. На *рисунках 6, 7* представлены функции зашифрования и расшифрования.

Data Output | Explain | Messages | Notifications

	passport [PK] character var	fio text	rating real	qualification qualification_type	experience integer	salary integer	encrypt_rating text	encrypt_experience text	encrypt_salary text	encrypt_qualification text
1	1	\xc30d0...	-1	highest	-1	-1	\xc30d040703...	\xc30d04070302bb...	\xc30d040703...	\xc30d040703022e8a...
2	2	\xc30d0...	-1	highest	-1	-1	\xc30d040703...	\xc30d0407030247...	\xc30d040703...	\xc30d04070302e090...
3	3	\xc30d0...	-1	highest	-1	-1	\xc30d040703...	\xc30d040703022e...	\xc30d040703...	\xc30d0407030276c4f...
4	4	\xc30d0...	-1	highest	-1	-1	\xc30d040703...	\xc30d0407030223...	\xc30d040703...	\xc30d040703021161...
5	5	\xc30d0...	-1	highest	-1	-1	\xc30d040703...	\xc30d04070302a5...	\xc30d040703...	\xc30d040703029630...
6	6	\xc30d0...	-1	highest	-1	-1	\xc30d040703...	\xc30d0407030259...	\xc30d040703...	\xc30d040703021fd78...
7	7	\xc30d0...	-1	highest	-1	-1	\xc30d040703...	\xc30d040703024e...	\xc30d040703...	\xc30d04070302eafe9...
8	8	\xc30d0...	-1	highest	-1	-1	\xc30d040703...	\xc30d0407030224...	\xc30d040703...	\xc30d040703021ae8...
9	9	\xc30d0...	-1	highest	-1	-1	\xc30d040703...	\xc30d0407030238...	\xc30d040703...	\xc30d0407030231f6b...
10	10	\xc30d0...	-1	highest	-1	-1	\xc30d040703...	\xc30d040703029b...	\xc30d040703...	\xc30d0407030229dd...
11	11	\xc30d0...	-1	highest	-1	-1	\xc30d040703...	\xc30d040703023d...	\xc30d040703...	\xc30d04070302bc76...

Рисунок 4 – Шифрование всего отношения

	passport [PK] character var	fio text	rating real	qualification qualification_type	experience integer	salary integer	encrypt_rating text	encrypt_experience text	encrypt_salary text	encrypt_qualification text
1	1	doctor1	8.668035	highest	10	15709	[null]	[null]	[null]	[null]
2	2	doctor2	1.3663625	second	4	69961	[null]	[null]	[null]	[null]
3	3	doctor3	9.488684	first	0	47098	[null]	[null]	[null]	[null]
4	4	doctor4	1.3543628	second	5	84670	[null]	[null]	[null]	[null]
5	5	doctor5	5.672027	highest	9	83241	[null]	[null]	[null]	[null]
6	6	doctor6	4.2378945	second	2	9918	[null]	[null]	[null]	[null]
7	7	doctor7	1.5167006	highest	4	6164	[null]	[null]	[null]	[null]
8	8	doctor8	1.1530706	second	6	82240	[null]	[null]	[null]	[null]
9	9	doctor9	0.16020408	first	4	36343	[null]	[null]	[null]	[null]
10	10	doctor10	4.2051806	second	8	14366	[null]	[null]	[null]	[null]
11	11	doctor11	9.632319	highest	7	36661	[null]	[null]	[null]	[null]

Рисунок 5 – Расшифрование всего отношения

```
-- Добавление дополнительного столбца
ALTER TABLE doctor ADD COLUMN encrypt_rating text;
ALTER TABLE doctor ADD COLUMN encrypt_experience text;
ALTER TABLE doctor ADD COLUMN encrypt_salary text;
ALTER TABLE doctor ADD COLUMN encrypt_qualification text;

-- Шифрование всей таблицы
UPDATE doctor
SET fio = pgp_sym_encrypt(fio::TEXT, 'secretPass'),
    encrypt_rating = pgp_sym_encrypt(rating::TEXT, 'secretPass'),
    encrypt_experience = pgp_sym_encrypt(experience::TEXT, 'secretPass'),
    encrypt_salary = pgp_sym_encrypt(salary::TEXT, 'secretPass'),
    encrypt_qualification = pgp_sym_encrypt(qualification::TEXT, 'secretPass'),
    rating = -1,
    salary = -1,
    experience = -1,
    qualification = 'highest';
```

Рисунок 6 – Функция зашифрования всего отношения

```
-- Расшифрование всей таблицы
UPDATE doctor
SET fio = pgp_sym_decrypt(fio::bytea, 'secretPass')::text,
    rating = pgp_sym_decrypt(encrypt_rating::bytea, 'secretPass')::real,
    experience = pgp_sym_decrypt(encrypt_experience::bytea, 'secretPass')::int,
    salary = pgp_sym_decrypt(encrypt_salary::bytea, 'secretPass')::int,
    qualification = pgp_sym_decrypt(encrypt_qualification::bytea, 'secretPass')::qualification_type,
    encrypt_rating = NULL,
    encrypt_experience = NULL,
    encrypt_salary = NULL,
    encrypt_qualification = NULL;
```

Рисунок 7 – Функция расшифрования всего отношения

c. Шифрование файлов данных на диске (TDE)

PostgreSQL не поддерживает шифрование файлов на физическом диске. Если существует необходимость хранить все данные зашифрованными, то можно воспользоваться технологией *Transparent Data Encryption (TDE)*, или любой другой для шифрования файловой системы.

Шифрование хранилища может выполняться на уровне файловой системы или на уровне блоков. Для шифрования была использована встроенная в *Windows* утилита *EFS*. Чтобы включить *EFS*-шифрование для папки, нужно просто включить соответствующий атрибут в ее свойствах *(см. рис. 8-9).*

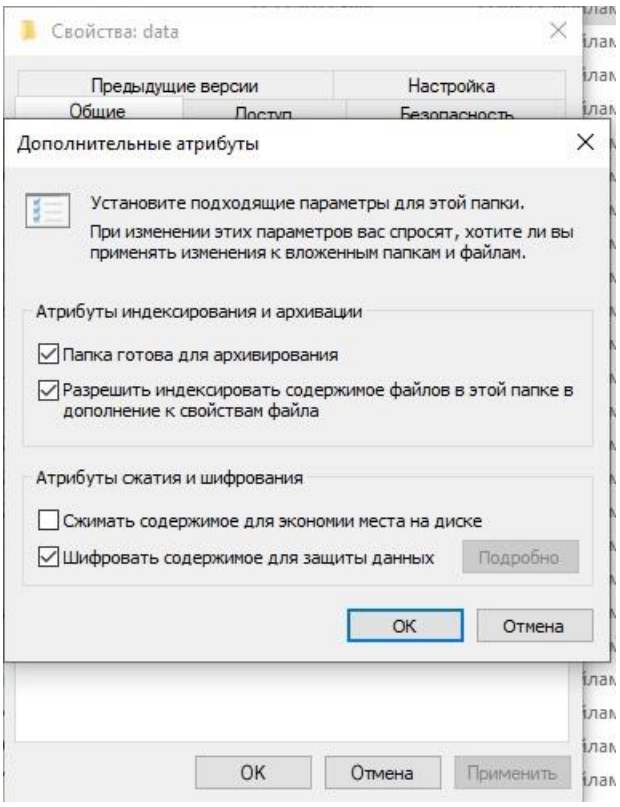

Рисунок 8 – Включение шифрования папки data

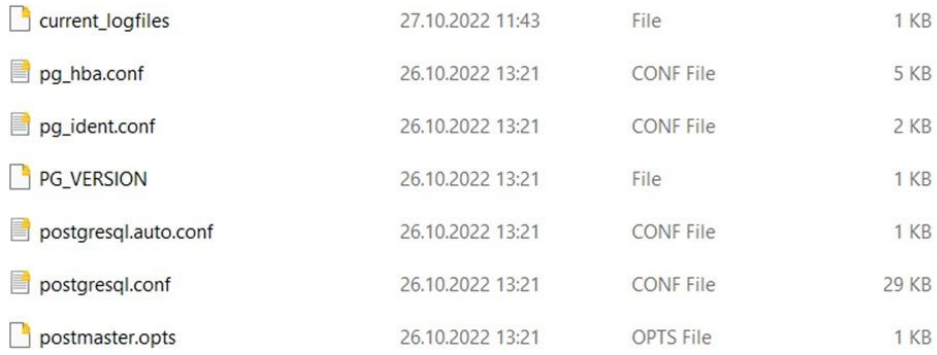

Рисунок 9 – Зашифрованные данные

d. *Шифрование данных при передаче (TLS)*

SSL-соединения шифруют все данные, передаваемые по сети: пароль, запросы и возвращаемые данные. Файл *pg_hba.conf* позволяет администраторам указать, для каких узлов будут разрешены незашифрованные соединения *(host)*, а для каких будет требоваться *SSL (hostssl)*. Необходимо добавить соединение с шифрованием *(см. рис. 10)*.

```
# Allow replication connections from localhost, by a user with the
# replication privilege.
local   replication     all                                     scram-sha-256
host    replication     all             127.0.0.1/32            scram-sha-256
host    replication     all             ::1/128                 scram-sha-256
hostssl   all           all             10.0.3.0/24             scram-sha-256
```

Рисунок 10 – Добавление соединения с шифрованием

Также необходимо сгенерировать сертификат и ключ к нему. Генерация реализована с помощью утилиты *openssl (см. рис. 11)*.

```
marug@ubuntu:~$ openssl req -new -x509 -days 356 -nodes -out server.crt -keyout
 server.key -subj "/CN=localhost"
Generating a RSA private key
.+++++
.....................+++++
writing new private key to 'server.key'
-----
```

Рисунок 11 – Генерация ключа и сертификата

Также был изменён конфигурационный файл *postgresql.conf*. Необходимо разрешить соединение с *SSL* и прописать пути до сертификата и ключа.

```
ssl = on
ssl_ca_file = 'C:\\Program Files\\PostgreSQL\\14\\data\\root.crt'
ssl_cert_file = 'C:\\Program Files\\PostgreSQL\\14\\data\\server.crt'
ssl_key_file = 'C:\\Program Files\\PostgreSQL\\14\\data\\server.key'
```

```
ssl_prefer_server_ciphers = on
ssl_min_protocol_version = 'TLSv1.3'
```

На *рисунках 12, 13* представлен результат соединения.

```
Server [localhost]:
Database [postgres]: encrypteddb
Port [5432]:
Username [postgres]:
Password for user postgres:
psql (14.5)
WARNING: Console code page (437) differs from Windows code page (1252)
         8-bit characters might not work correctly. See psql reference
         page "Notes for Windows users" for details.
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, bits: 256, compression: off)
Type "help" for help.
```

Рисунок 12 – Сообщение при входе

```
2022-10-29 11:29:49.201 MSK [19632] LOG:  connection authenticated: identity="postgres" method=scram-sha-256
(C:/Program Files/PostgreSQL/14/data/pg_hba.conf:89)
2022-10-29 11:29:49.201 MSK [19632] LOG:  connection authorized: user=postgres database=encrypteddb
application_name=psql SSL enabled (protocol=TLSv1.3, cipher=TLS_AES_256_GCM_SHA384, bits=256)
```

Рисунок 13 – Сообщение в журнале

e. Сквозное шифрование

Сквозное шифрование также называется шифрованием на стороне клиента. Если системный администратор сервера, где работает база данных, не является доверенным, клиент должен сам шифровать данные; тогда незашифрованные данные никогда не появятся на этом сервере. В этом случае клиент шифрует данные, прежде чем передавать их серверу, а получив из базы данных результаты, он расшифровывает их для использования.

Шифровать данные нужно перед отправкой сообщения. Для этого можно также воспользоваться утилитой командной строки *openssl*. На *рисунке 14* представлен пример шифрования данных.

```
marug@ubuntu:~$ echo "doctor1" | openssl enc -base64
ZG9jdG9yMQo=
marug@ubuntu:~$ echo "2.2719963" | openssl enc -base64
Mi4yNzE5OTYzCg==
marug@ubuntu:~$ echo "highest" | openssl enc -base64
aGlnaGVzdAo=
marug@ubuntu:~$ echo "7" | openssl enc -base64
Nwo=
marug@ubuntu:~$ echo "80237" | openssl enc -base64
ODAyMzcK
```

Рисунок 14 – Шифрование данных с помощью утилиты openssl

На *рисунке 15* представлен пример отправки зашифрованных сообщений и результат запроса этих данных.

```
127 INSERT INTO doctor (passport, fio, encrypt_salary, encrypt_qualification, encrypt_rating, encrypt_experience)
128 VALUES ('1', 'ZG9jdG9yMQo=', 'ODAyMzcK', 'aGlnaGVzdAo=', 'Mi4yNzE5OTYzCg==', 'Nwo=');
129
130 SELECT * FROM doctor
131     WHERE encrypt_rating IS NOT NULL;
132
```

Data Output | Explain | Messages | Notifications

	passport [PK] character var	fio text	rating real	qualification qualification_type	experience integer	salary integer	encrypt_salary text	encrypt_qualification text	encrypt_rating text	encrypt_experience text
1	1	ZG9jdG9...	[null]	[null]	[null]	[null]	ODAyMzcK	aGlnaGVzdAo=	Mi4yNzE5OTY...	Nwo=

Рисунок 15 – Отправка зашифрованных сообщений и результат запроса

На *рисунке 16* представлено расшифрование полученных данных.

```
marug@ubuntu:~$ echo "ZG9jdG9yMQo=" | openssl enc -d -base64
doctor1
marug@ubuntu:~$ echo "Mi4yNzE5OTYzCg==" | openssl enc -d -base64
2.2719963
marug@ubuntu:~$ echo "aGlnaGVzdAo=" | openssl enc -d -base64
highest
marug@ubuntu:~$ echo "Nwo=" | openssl enc -d -base64
7
marug@ubuntu:~$ echo "ODAyMzcK" | openssl enc -d -base64
80237
```

Рисунок 16 – Расшифрование данных с помощью утилиты openssl

3. Результаты тестирования производительности и объёма данных

Для выполнения тестов в таблицу *doctor* было добавлено 100000 записей. Результаты тестирования режимов шифрования на производительность с помощью выполнения запроса представлены в *таблице 1*.

	Без шифрования	**Шифрование 2 столбцов**	**Шифрование всех столбцов**	**TDE**	**TLS**	**На стороне клиента**
Среднее время, мс	216	12496	17326	2925	2698	15693
Общий размер файла БД на диске, Мб	25,33	64,2	107,3	25,33	25,33	72,9

Таблица 1 – Результаты тестов

Без шифрования доступ к данным и занимаемое на диске место наиболее эффективны.

Шифрование нескольких (2) столбцов значительно увеличило время запроса. Это объясняется накладными расходами на расшифрование во время выполнения

запроса. Также увеличился размер данных, так как увеличился минимальный размер блока после шифрования.

Шифрование всей таблицы ещё значительно увеличило занимаемое пространство по той же причине. Время доступа также увеличилось.

Наконец, шифрование при передаче, очевидно, никак не отразилось на занимаемом на диске пространстве, ведь данные хранятся в открытом виде. Время же доступа к данным немного выросло, но меньше, чем при шифровании столбцов.

Вывод

В ходе выполнения лабораторной работы были изучены доступные режимы шифрования в СУБД *PostgreSQL*. Были исследованы их влияние на ресурсы системы – как на память, так и на время. Были получены навыки по настройке шифрования.