Міністерство освіти і науки України

Національний технічний університет України

“Київський політехнічний інститут”

Кафедра АСОІУ

**ЗВІТ**

про виконання комп’ютерного практикуму № 2

з дисципліни

“ Програмні додатки з використанням баз даних”

по темі

«Наповнення бази даних. Застосування індексів»

Варіант 6

|  |  |  |
| --- | --- | --- |
| Прийняв: |  | Виконав: |
| Cт.вик. Клименко О. М. |  | студент 3-го курсу  гр. ІП-51 ФІОТ  Зарічковий Олександр Анатолійович |

Київ – 2017

**ЗМІСТ:**

[1 Завдання 3](#_Toc495230724)

[2 Виконання Завдання 4](#_Toc495230725)

# Завдання

**Завдання 1**:

Кожен студент згідно предметної області за варіантом повинен вибрати

таблицю, що містить такі поля як прізвище, імя та по-батькові:

1) Написати програмний додаток (бажано веб-додаток), що має:

а) кнопку «Очистити таблицю»;

б) поле «Ввести кількість рядків» (кількість рядків може бути 500 тис. і

більше);

в) кнопку «Наповнити таблицю». Вивести час, за який відбулося

наповнення вибранної таблиці заданною у пункті б) кількістю рядків. При

наповненні таблиці, звертаю увагу на те, що треба наповнити ще одну

таблицю, що пов’язана з цією для виконання запиту з JOIN.

2) Виконати без індекса запити з WHERE, JOIN та вивести час виконання

кожного запиту.

3) Створити індекси, які на вашу думку дадуть прискорення виконання

попередніх запитів з WHERE, JOIN та вивести час виконання кожного

запиту.

**Завдання 2** (повторення теми «Індекси» - матеріалу попереднього семестру):

1) Описати які типи індексів використовує вибрана вами СУБД і як кожен

тип реалізований алгоритмічно (наприклад, B-tree індекс та принцип його

роботи).

2) Для яких запитів який тип індексу використовує вибрана вами СУБД,

навести приклади, до кожного типу індексу хоча один запит (наприклад,

B-tree індекс використовується, якщо до індексованого поля

застосовуються оператори порівняння). Приклади виключно з вашого

варіанту.

**Преметна область**: Навчання з охорони праці

# Виконання Завдання

1. Завдання по написанню запитів

import mysql.connector

import numpy as np

from tqdm import tqdm

from time import time

NUMBER\_OF\_GENERATED\_SAMPLES = 1000000

def create\_listeners\_table(db\_cursor):

sql = """

CREATE TABLE `Listeners` (

`ListenerId` int(11) NOT NULL AUTO\_INCREMENT,

`FirstName` varchar(255) NOT NULL,

`LastName` varchar(255) NOT NULL,

PRIMARY KEY (`ListenerId`))

ENGINE=InnoDB

"""

db\_cursor.execute(sql)

def create\_tests\_table(db\_cursor):

sql = """

CREATE TABLE `Tests` (

`TestId` int(11) NOT NULL AUTO\_INCREMENT,

`ListenerId` int(11) NOT NULL,

`Others` int(11) NULL,

PRIMARY KEY (`TestId`))

ENGINE=InnoDB

"""

db\_cursor.execute(sql)

def create\_tests\_table\_index(db\_cursor):

sql = """

CREATE TABLE `Tests` (

`TestId` int(11) NOT NULL AUTO\_INCREMENT,

`ListenerId` int(11) NOT NULL,

`Others` int(11) NULL,

PRIMARY KEY (`TestId`),

FOREIGN KEY (ListenerId)

REFERENCES Listeners(ListenerId)

ON UPDATE CASCADE

ON DELETE CASCADE

)

ENGINE=InnoDB

"""

db\_cursor.execute(sql)

def create\_listeners\_table\_index(db\_cursor):

sql = """

CREATE TABLE `Listeners` (

`ListenerId` int(11) NOT NULL AUTO\_INCREMENT,

`FirstName` varchar(255) NOT NULL,

`LastName` varchar(255) NOT NULL,

PRIMARY KEY (`ListenerId`),

INDEX names USING BTREE (`FirstName`))

ENGINE=InnoDB

"""

db\_cursor.execute(sql)

def drop\_test(db\_cursor):

sql = """

DROP TABLE tests

"""

db\_cursor.execute(sql)

def drop\_listeners(db\_cursor):

sql = """

DROP TABLE listeners

"""

db\_cursor.execute(sql)

def insert\_tests\_table(db\_cursor):

np.random.seed(0xCAFFE)

random\_listeners\_id = np.random.randint(low=1, high=NUMBER\_OF\_GENERATED\_SAMPLES,

size=(NUMBER\_OF\_GENERATED\_SAMPLES, ))

random\_other = np.random.randint(low=1, high=NUMBER\_OF\_GENERATED\_SAMPLES,

size=(NUMBER\_OF\_GENERATED\_SAMPLES, ))

data = np.hstack((random\_listeners\_id.reshape(-1, 1),

random\_other.reshape(-1, 1)))

sql = """

INSERT INTO tests (ListenerId, Others)

VALUES (%s, %s)

"""

for row\_index in tqdm(range(NUMBER\_OF\_GENERATED\_SAMPLES)):

first, second = data[row\_index]

db\_cursor.execute(sql, (str(first), str(second)))

def insert\_listeners\_table(db\_cursor):

np.random.seed(0xCAFFE)

first\_names = np.array(['aaa', 'bbb', 'ccc'])

second\_names = np.array(['ddd', 'eee', 'ggg', 'jjj'])

random\_indexes\_first = np.random.randint(low=0, high=first\_names.shape[0],

size=(NUMBER\_OF\_GENERATED\_SAMPLES, ))

random\_indexes\_second = np.random.randint(low=0, high=second\_names.shape[0],

size=(NUMBER\_OF\_GENERATED\_SAMPLES, ))

data = np.hstack((first\_names[random\_indexes\_first].reshape(-1, 1),

second\_names[random\_indexes\_second].reshape(-1, 1)))

sql = """

INSERT INTO listeners (FirstName, LastName)

VALUES (%s, %s)

"""

for row\_index in tqdm(range(NUMBER\_OF\_GENERATED\_SAMPLES)):

first\_name, second\_name = data[row\_index]

db\_cursor.execute(sql, (str(first\_name), str(second\_name)))

def select\_with\_join\_and\_where(db\_cursor):

sql = """

SELECT \*

FROM listeners as L

LEFT JOIN tests as T

ON T.ListenerId = L.ListenerId

WHERE FirstName like 'aaa'

"""

current\_time = time()

db\_cursor.execute(sql)

for \_ in db\_cursor:

pass

print('Time elapsed: %s sec' % (time() - current\_time))

def main():

operation\_list = [

('Create listeners table', create\_listeners\_table),

('Create tests table', create\_tests\_table),

('Inserting %d elements to listeners table' % NUMBER\_OF\_GENERATED\_SAMPLES, insert\_listeners\_table),

('Inserting %d elements to tests table' % NUMBER\_OF\_GENERATED\_SAMPLES, insert\_tests\_table),

('Selecting', select\_with\_join\_and\_where),

('Droping test', drop\_test),

('Droping listeners', drop\_listeners),

('Create listeners table with index', create\_listeners\_table\_index),

('Create tests table with index', create\_tests\_table\_index),

('Inserting %d elements to listeners table' % NUMBER\_OF\_GENERATED\_SAMPLES, insert\_listeners\_table),

('Inserting %d elements to tests table' % NUMBER\_OF\_GENERATED\_SAMPLES, insert\_tests\_table),

('Selecting', select\_with\_join\_and\_where),

('Droping test', drop\_test),

('Droping listeners', drop\_listeners),

]

connection = mysql.connector.connect(user='ALEXKIRNAS',

password='1234',

host='127.0.0.1',

database='database')

db\_cursor = connection.cursor()

for name, function in operation\_list:

print(name, end=':\n')

function(db\_cursor)

connection.commit()

print("Finished")

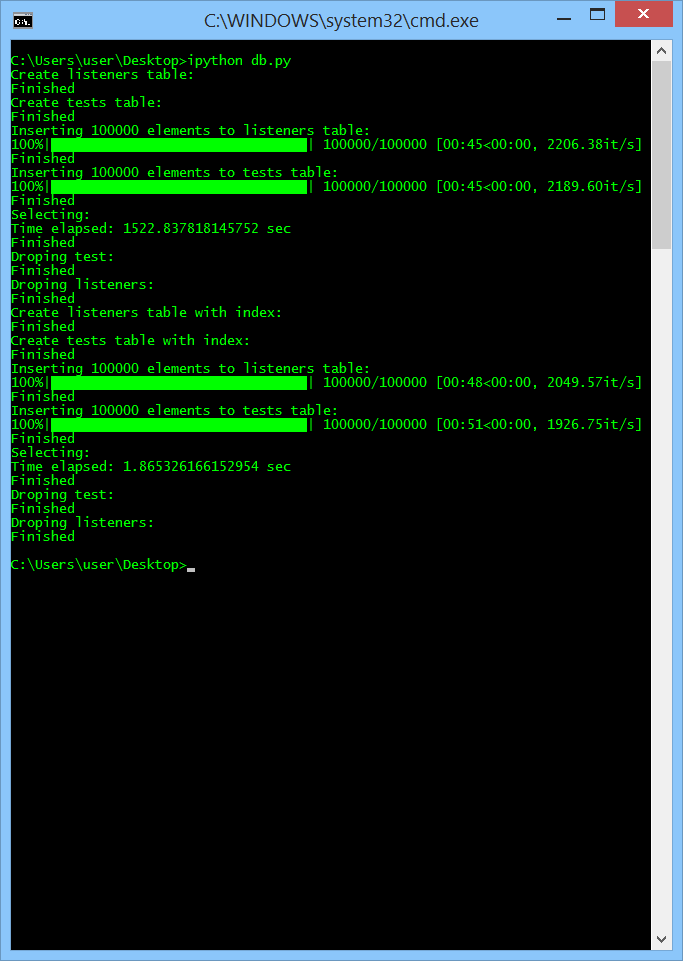
db\_cursor.close()

connection.close()

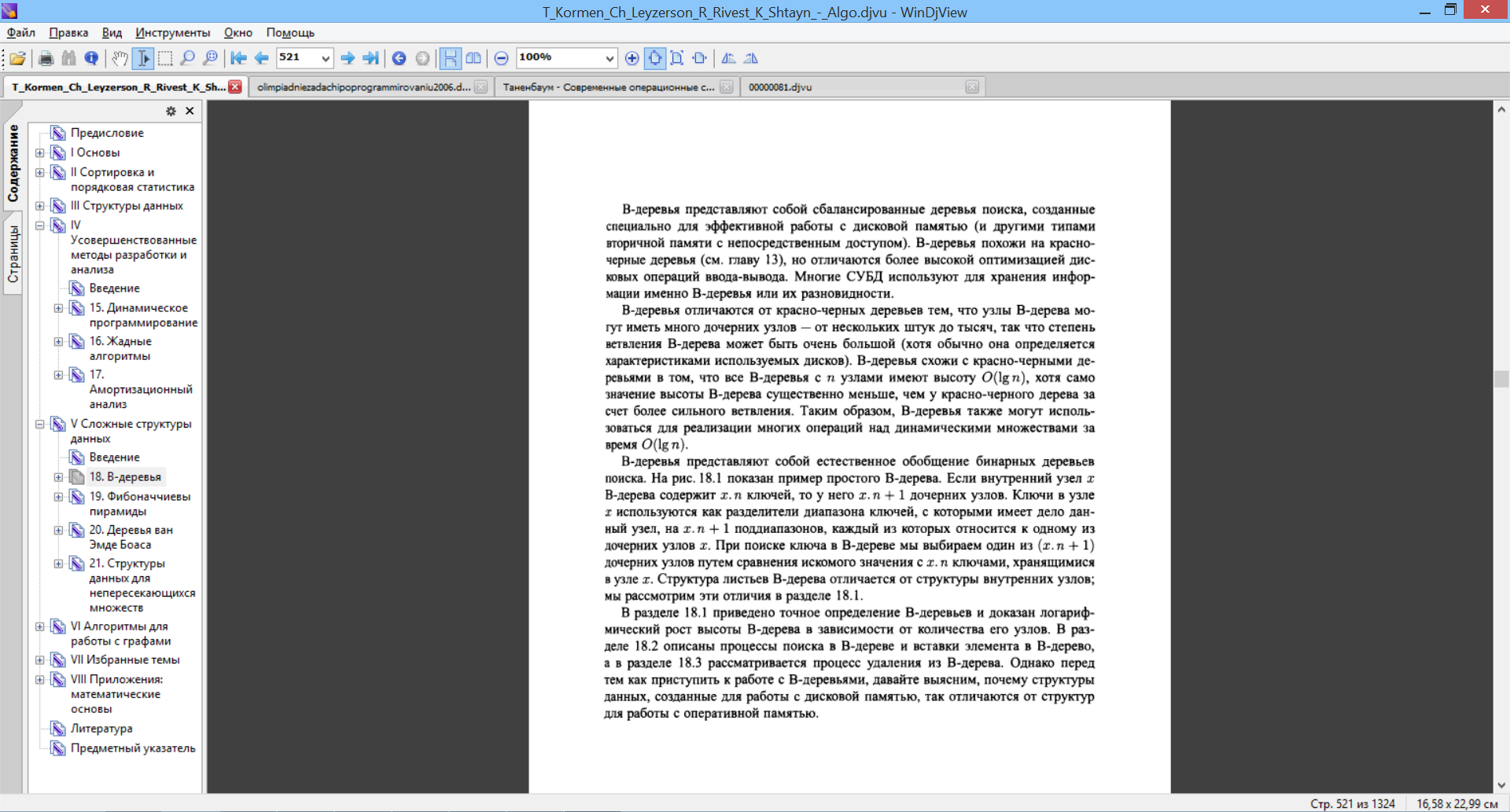
if \_\_name\_\_ == '\_\_main\_\_':

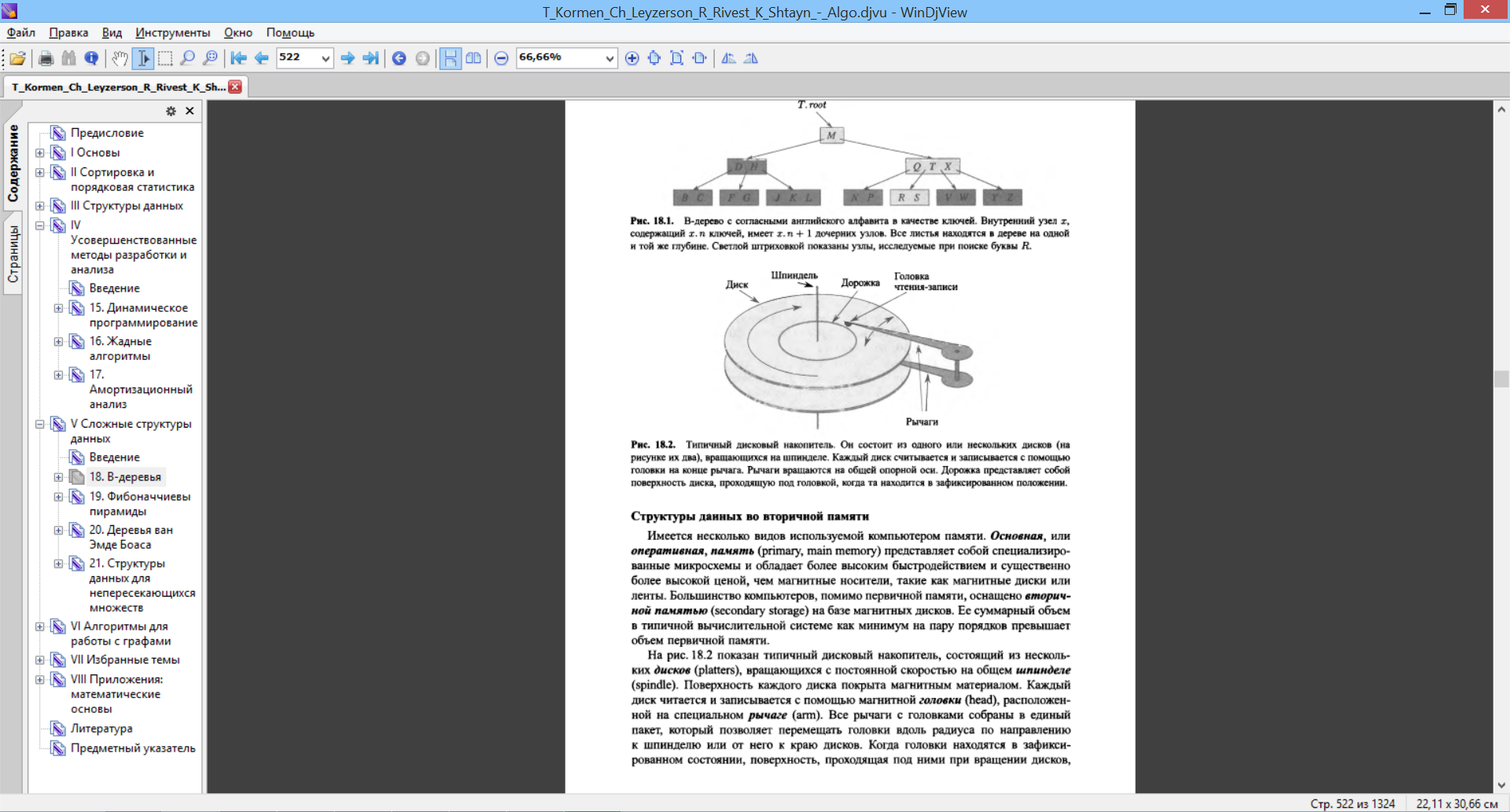
main()

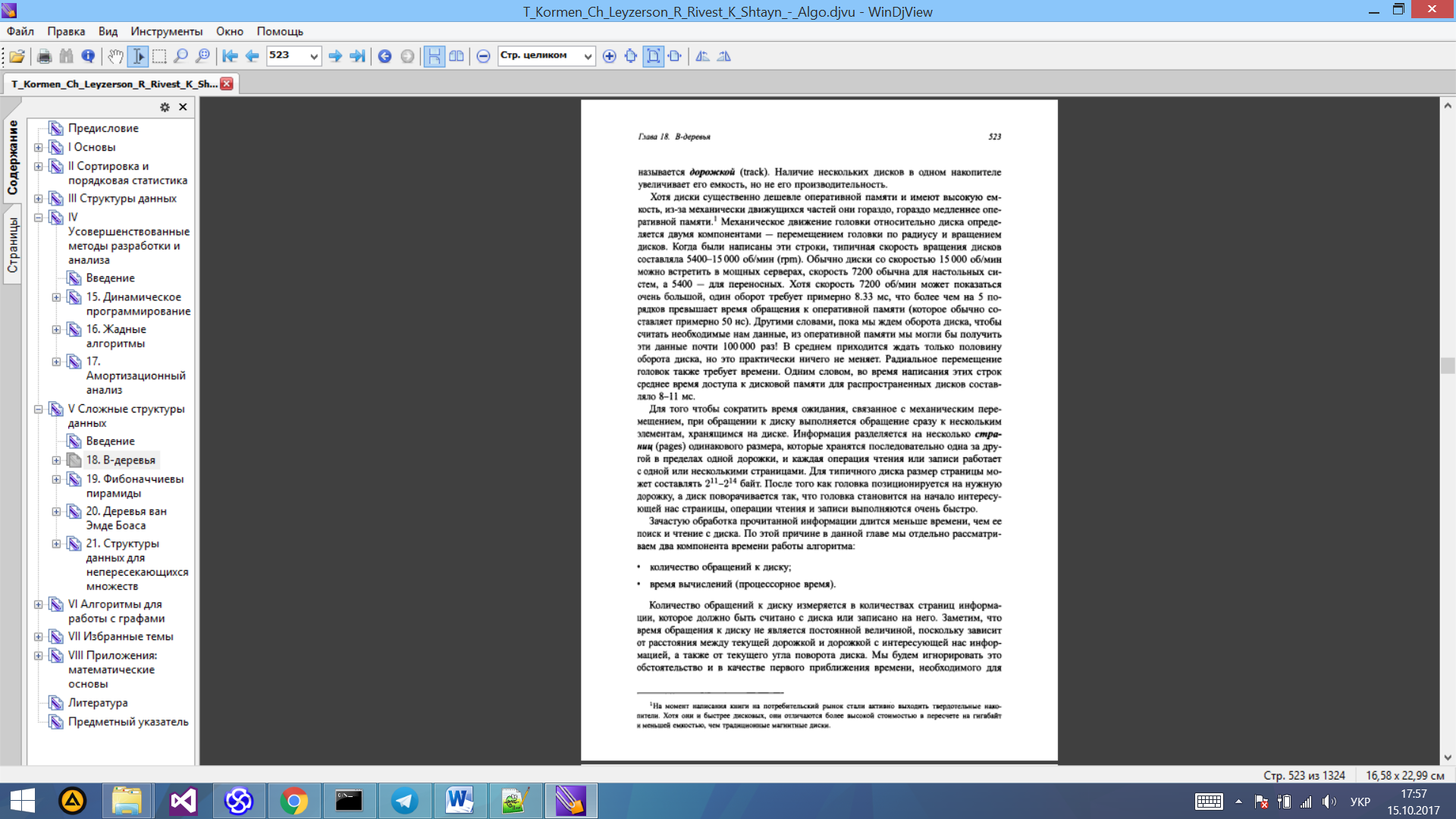
*Результат виконання програми*:

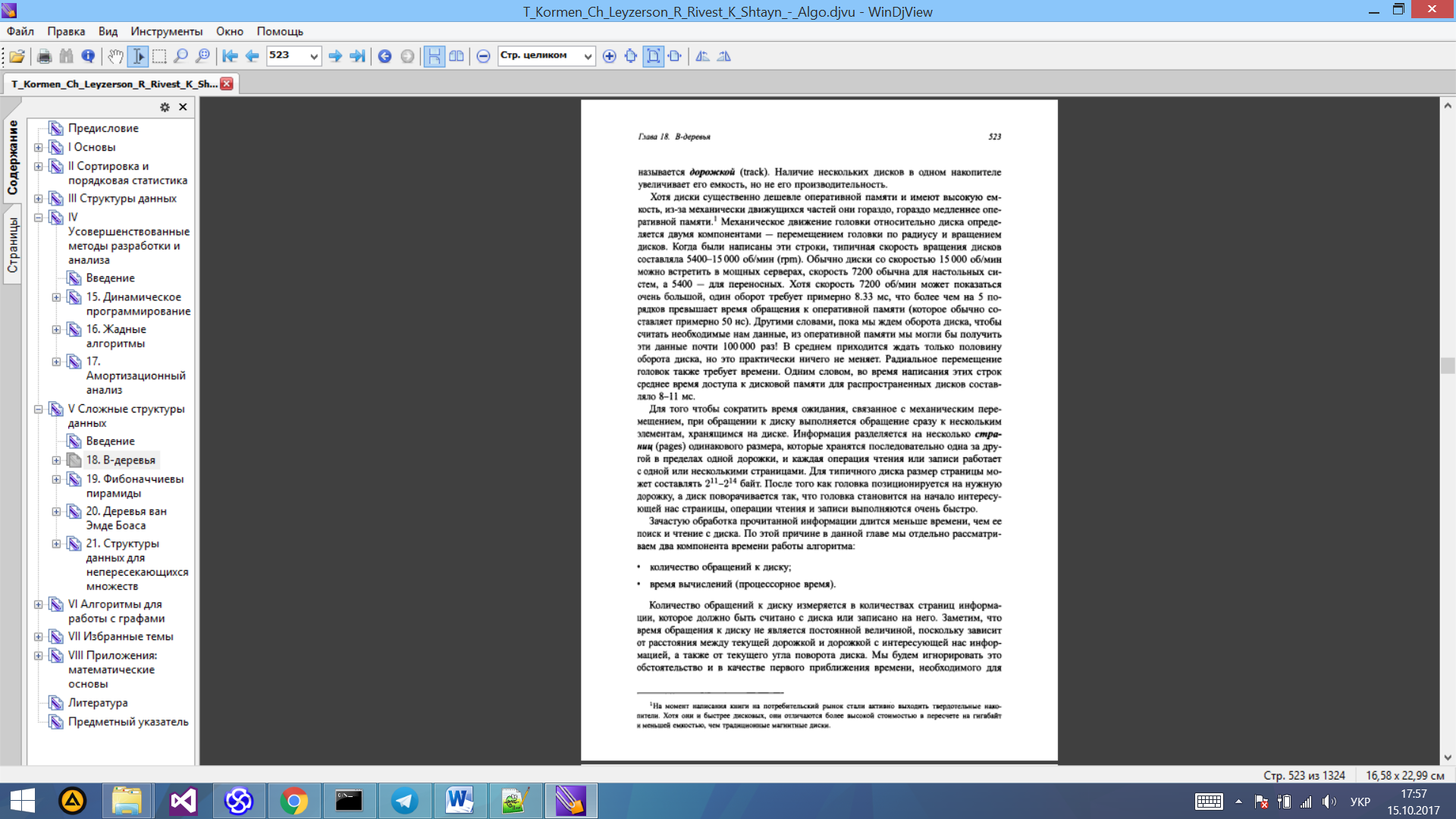


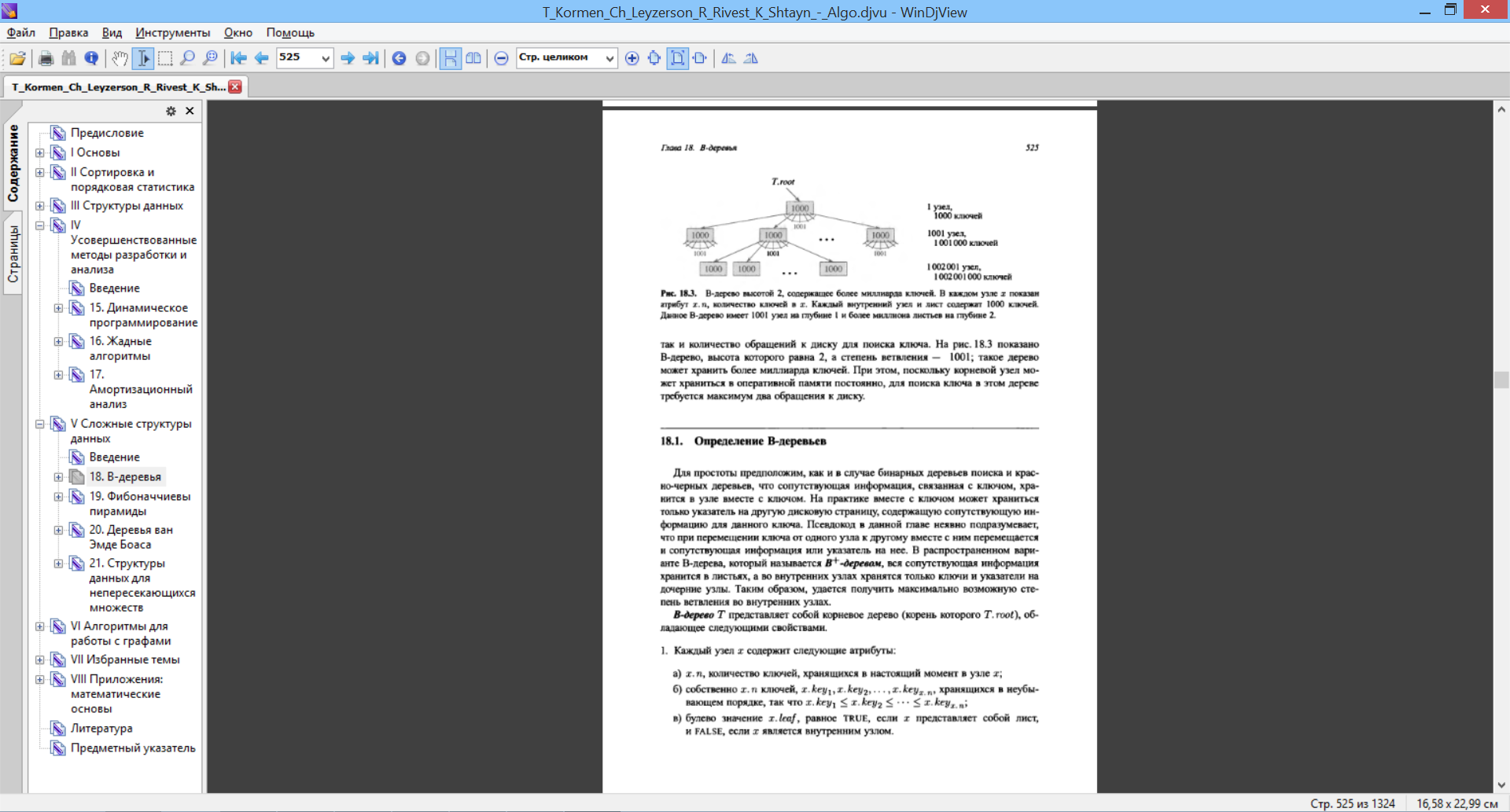
1. Індекси:
   1. Б-дерева

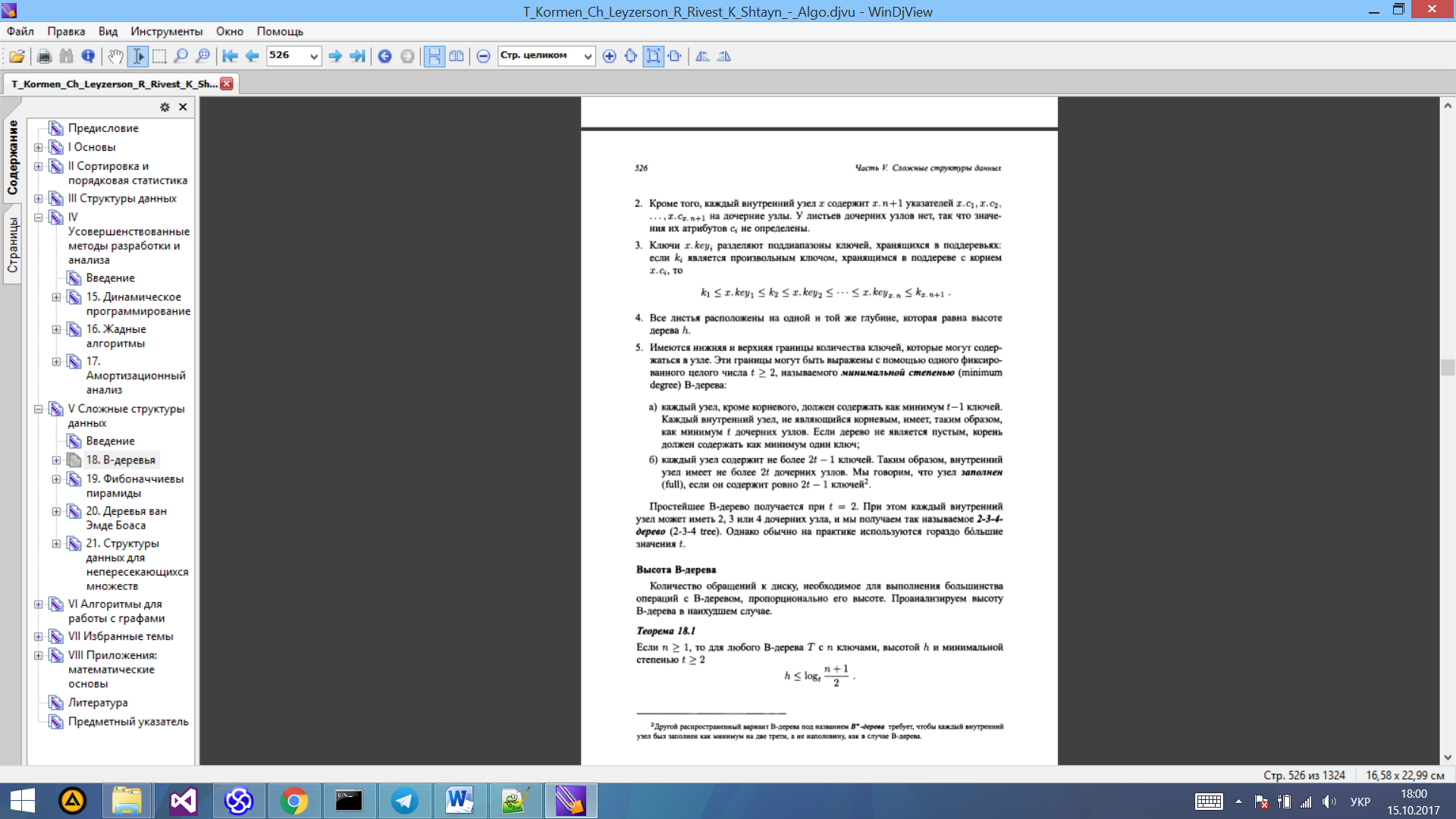


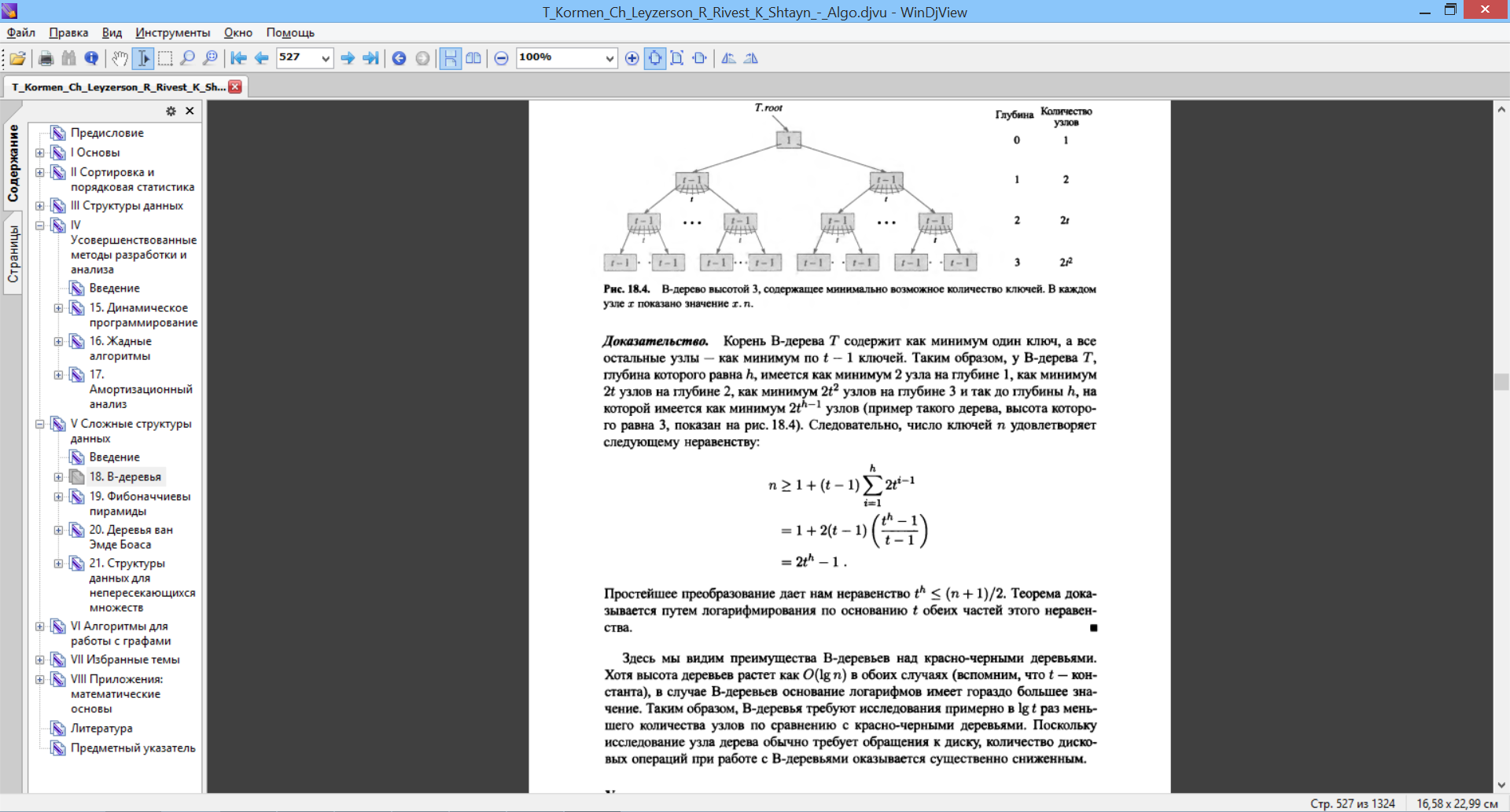












* 1. Для яких запитів який тип індексу використовує вибрана вами СУБД

Обрана мною СУБД для всіх операцій пошуку, порівняння, при виконанні операції WHERE, BETWEEN використовую некластерний індекс який базується на B-деревах (за умови що по даному полю побудований індекс).