	Lehrveranstaltung	Databases and Information Systems 2020		
	Aufgabenzettel	3		
	STiNE-Gruppe 14	Simon Weidmann, Aram Yesildeniz		
	Ausgabe	12. Mai 2020	Abgabe	19. Mai 2020

4.1 Isolation Levels and SQL

a)

How can you determine the currently set isolation level?

```
SHOW TRANSACTION ISOLATION LEVEL;
```

What is the default isolation level of PostgreSQL?

Read Committed is the default isolation level in PostgreSQL.

How can the isolation level be changed during a session in PostgreSQL?

For current transaction:

```
SET TRANSACTION ISOLATION LEVEL
{ SERIALIZABLE | REPEATABLE READ | READ COMMITTED | READ UNCOMMITTED }
```

For default transaction characteristics:

```
SET SESSION CHARACTERISTICS AS TRANSACTION ISOLATION LEVEL
{ SERIALIZABLE | REPEATABLE READ | READ COMMITTED | READ UNCOMMITTED }
```

b)

c)


d)

e)

SQL Scripts

Create Database

```
CREATE DATABASE dis
WITH
OWNER = postgres
ENCODING = 'UTF8'
CONNECTION LIMIT = -1;
```

	Lehrveranstaltung	Databases and Information Systems 2020		
	Aufgabenzettel	3		
	STiNE-Gruppe 14	Simon Weidmann, Aram Yesildeniz		
	Ausgabe	12. Mai 2020	Abgabe	19. Mai 2020

Create Tables


```
CREATE TABLE public.estate_agent
(
    agent_login text,
    agent_name text,
    agent_address text,
    agent_password text,
    PRIMARY KEY (agent_login)
);
```

```
ALTER TABLE public.estate_agent
    OWNER to postgres;
```

```
CREATE TABLE public.estate
(
    estate_id serial,
    city text,
    postal_code integer,
    street text,
    street_number text,
    square_area integer,
    manager text,
    PRIMARY KEY (estate_id),
    CONSTRAINT manager FOREIGN KEY (manager)
        REFERENCES public.estate_agent (agent_login) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
);
```

```
ALTER TABLE public.estate
    OWNER to postgres;
```

```
CREATE TABLE public.apartment
(
    floor integer,
    rent text,
    rooms text,
    balcony boolean,
    kitchen boolean,
```

	Lehrveranstaltung	Databases and Information Systems 2020		
	Aufgabenzettel	3		
	STiNE-Gruppe 14	Simon Weidmann, Aram Yesildeniz		
	Ausgabe	12. Mai 2020	Abgabe	19. Mai 2020

```

CONSTRAINT apartment_pkey PRIMARY KEY (estate_id),
CONSTRAINT manager FOREIGN KEY (manager)
    REFERENCES public.estate_agent (agent_login) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID
)
INHERITS (public.estate);

ALTER TABLE public.apartment
    OWNER to postgres;


CREATE TABLE public.house
(
    floors integer,
    price text,
    garden boolean,
    CONSTRAINT house_pkey PRIMARY KEY (estate_id),
    CONSTRAINT manager FOREIGN KEY (manager)
        REFERENCES public.estate_agent (agent_login) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)
INHERITS (public.estate)
TABLESPACE pg_default;

ALTER TABLE public.house
    OWNER to postgres;

CREATE TABLE public.person
(
    id serial,
    first_name text,
    last_name text,
    address text,
    PRIMARY KEY (id)
);

ALTER TABLE public.person
    OWNER to postgres;

```

	Lehrveranstaltung	Databases and Information Systems 2020		
	Aufgabenzettel	3		
	STiNE-Gruppe 14	Simon Weidmann, Aram Yesildeniz		
	Ausgabe	12. Mai 2020	Abgabe	19. Mai 2020


```
CREATE TABLE public.contract
(
    contract_number serial,
    contract_date date,
    place text,
    PRIMARY KEY (contract_number)
);
```

```
ALTER TABLE public.contract
    OWNER to postgres;
```

```
CREATE TABLE public.tenancy_contract
(
    start_date date,
    duration text,
    additional_costs text,
    person_id integer,
    apartment_id integer,
    CONSTRAINT tenancy_contract_pkey PRIMARY KEY (contract_number),
    CONSTRAINT person_id FOREIGN KEY (person_id)
        REFERENCES public.person (id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID,
    CONSTRAINT apartment_id FOREIGN KEY (apartment_id)
        REFERENCES public.apartment (estate_id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
)
INHERITS (public.contract);
```

```
ALTER TABLE public.tenancy_contract
    OWNER to postgres;
```

```
CREATE TABLE public.purchase_contract
(
    installment_amount text,
    interest_rate text,
    person_id integer,
    house_id integer,
    CONSTRAINT purchase_contract_pkey PRIMARY KEY (contract_number),
```

	Lehrveranstaltung	Databases and Information Systems 2020		
	Aufgabenzettel	3		
	STiNE-Gruppe 14	Simon Weidmann, Aram Yesildeniz		
	Ausgabe	12. Mai 2020	Abgabe	19. Mai 2020

```

CONSTRAINT person_id FOREIGN KEY (person_id)
    REFERENCES public.person (id) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID,
CONSTRAINT house_id FOREIGN KEY (house_id)
    REFERENCES public.house (estate_id) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID
)
INHERITS (public.contract);

ALTER TABLE public.purchase_contract
    OWNER to postgres;

```

Insert Estate Agent

```


INSERT INTO public.estate_agent(
    agent_login, agent_name, agent_address, agent_password)
VALUES ('testagent', 'testname', 'testaddress', 'testpassword');

```

4.2 Lock Conflicts

- a)
- b)
- c)
- d)
- e)

Create an apartment, an estate agent and a tenancy contract with your java application. Validate that they are in the database (e.g. by using a screenshot of application and database).

	Lehrveranstaltung	Databases and Information Systems 2020		
	Aufgabenzettel	3		
	STiNE-Gruppe 14	Simon Weidmann, Aram Yesildeniz		
	Ausgabe	12. Mai 2020	Abgabe	19. Mai 2020

Create Estate Agent

Estate Agent Management:

- [1] Create Agent
- [2] Update Agent
- [3] Delete Agent
- [4] Back to Main Menu

Input: 1

Login: peter

Name: pane


Address: somewhere

Password: 42

Agent [peter, pane, somewhere, 42] created.

- > Procedures
- > 1.3 Sequences
- ✓ > Tables (8)
 - > apartment
 - > contract
 - > estate
 - > estate_agent
 - > house
 - > person
 - > purchase_contract
 - > tenancy_contract
- > Trigger Functions

	Data Output	Explain	Messages	Notifications
	agent_login [PK] text	agent_name text	agent_address text	agent_password text
1	newagent	newagent	asdf	asdf
2	peter	pane	somewhere	42
3	testagent	testname	testaddress	testpassword

	Lehrveranstaltung	Databases and Information Systems 2020		
	Aufgabenzettel	3		
	STiNE-Gruppe 14	Simon Weidmann, Aram Yesildeniz		
	Ausgabe	12. Mai 2020	Abgabe	19. Mai 2020

Create Apartment

Estate Management. Logged in as [peter]:

- [1] Create Estate
- [2] Update Estate
- [3] Delete Estate
- [4] Back to Main Menu

Input: 1

Create new Estate. Only [peter] can perform actions:

- [1] Create Apartment
- [2] Create House
- [3] Back to Estate Menu

Input: 1

City: *hamburg*

Postalcode: *420202*

Street: *super street*

Street Number: *777*

Square Area: *120*

Floor: *3*

Rent: *1200*

Rooms: *5*

Balcony (1 = yes): *1*

Kitchen (1 = yes): *1*

Apartment [13, hamburg, 420202, super street, 777, 120, peter, 3, 1200, 5, true, true] created.

1.3 Sequences

Tables (8)

apartment

contract

estate

estate_agent

house

person

purchase_contract

tenancy_contract


Data Output

Explain

Messages

Notifications

	estate_id [PK] integer	city text	postal_code integer	street text	street_number text	square_area integer	manager text	floor integer	rent text	rooms text	balcony boolean	kitchen boolean
1	6	123		123 123	123		123 testagent		12 312	3123	false	false
2	12	1241		124 124	124		124 testagent		124 124	124	false	false
3	13	hamburg	420202	super street	777		120 peter	3	1200	5	true	true

	Lehrveranstaltung	Databases and Information Systems 2020		
	Aufgabenzettel	3		
	STiNE-Gruppe 14	Simon Weidmann, Aram Yesildeniz		
	Ausgabe	12. Mai 2020	Abgabe	19. Mai 2020

Create Tenancy Contract

```

Contract Menu:
[1] Insert Person
[2] Sign/Create Tenancy Contract
[3] Sign/Create Purchase Contract
[4] Show Contracts
[5] Back
Input: 2
Estate ID: 13
Tenant ID: 1
Start Date (Format DD.MM.YYYY): 10.10.2020
Duration: 12 months
Additional Costs: 1340
Contract Date (Format DD.MM.YYYY): 08.05.2020
Settlement Place: 123
Contract has been saved to database.

```

	contract_number	contract_date	place	start_date	duration	additional_costs	person_id	apartment_id
	[PK] integer	date	text	date	text	text	integer	integer
1		2020-05-08	123	2020-10-10	12 months	123 €		6
2		2020-05-08	123	2020-10-10	12 months	1340	1	13

Create a contract with a non-existing estate. Does it work? Why/Why not?

No it does not work. The foreign key 'apartment-id' in the tenancy contract table needs to be valid. If the entered id is not existing in the corresponding estate table, an exception will be thrown:

ERROR: insert or update on table tenancy-contract violates foreign key constraint person-id

Detail: Key (person-id)=(123) is not present in table person.

Which inheritance model did you choose and why?

Horizontal: Postgresql offers the INHERITS keyword, which enables horizontal partitioning. Therefore we used this keyword because it is intuitive to use easy to implement.

Create an apartment, and let your application crash between inserting the estate information and inserting the apartment information. What is the effect on your database state?

Since it is only possible to create an apartment or house in one go, this use case will not happen. If the application crashes while the user inserts information for the estate he wants to create, the record will not be saved.