PREPARING TO DATABASE FINAL

Lecture 1

**SQL** (Structured Query Language) – is a standard language for accessing and manipulating databases.

**Database** – a set of related information

**Database System** – is a collection of data

**Data Models** – is an abstract model that organizes elements of data and standardizes how they relate to one another and to the properties of real-world entities (relational data model, E-R data model, Object based DM)

**DDL** (Data Definition Language) – **SQL** commands that can be used to define the database schema.

(Create, drop, alter)

CREATE TABLE corporation (corp\_id SMALLINT, name VARCHAR(30), CONSTRAINT pk\_corporation PRIMARY KEY (corp\_id));

**DML** (Data Manipulation Language) – Language for accessing and updating the data organized by the appropriate data model •DML also known as query language (insert, delete, update)

INSERT INTO corporation (corp\_id, name) VALUES (27, 'Acme Paper Corporation');

**DBMS** (Database Management System) – \*MySQL, \*Oracle DB, \*PostgreSQL, \*MS SQL Server

Why **PostgreSQL**? – is an enterprise-class relational database management system, on par with the very best proprietary database systems. (open source, fast, large number of data types, stored procedures and functions in numerous programming language(C, SQL, Python, JS)

**Transaction** – is a collection of operations that performs a single logical function in a database application

Lecture 2

**Database schema** – is the logical structure of the database

**Relational algebra** – theoretical foundation for database and SQL

•select: σ operation selects tuples that satisfy a given predicate

•project: ∏ operation that returns its argument relation, with certain attributes left out

•union: ∪ operation allows us to combine two relations

•set difference: – operation allows us to find tuples that are in one relation but are not in another.

•Cartesian product: x allows us to combine information from any two relations.

•rename: ρ

Lecture 3

**String operation:**

•percent (%). The % character matches any substring.

•underscore (\_). The \_ character matches any character.

**A concatenation operator [||]** is used for appending two string

▪Find the names of all instructors whose name includes the substring “dar”.

select name

from instructor

where name like '%dar%'

**null value** – value is a field with no value

**not null** – constraint simply specifies that a column must not assume the null value.

**Constraints** – (Primary key, foreign key, Not NULL, Unique, Check)

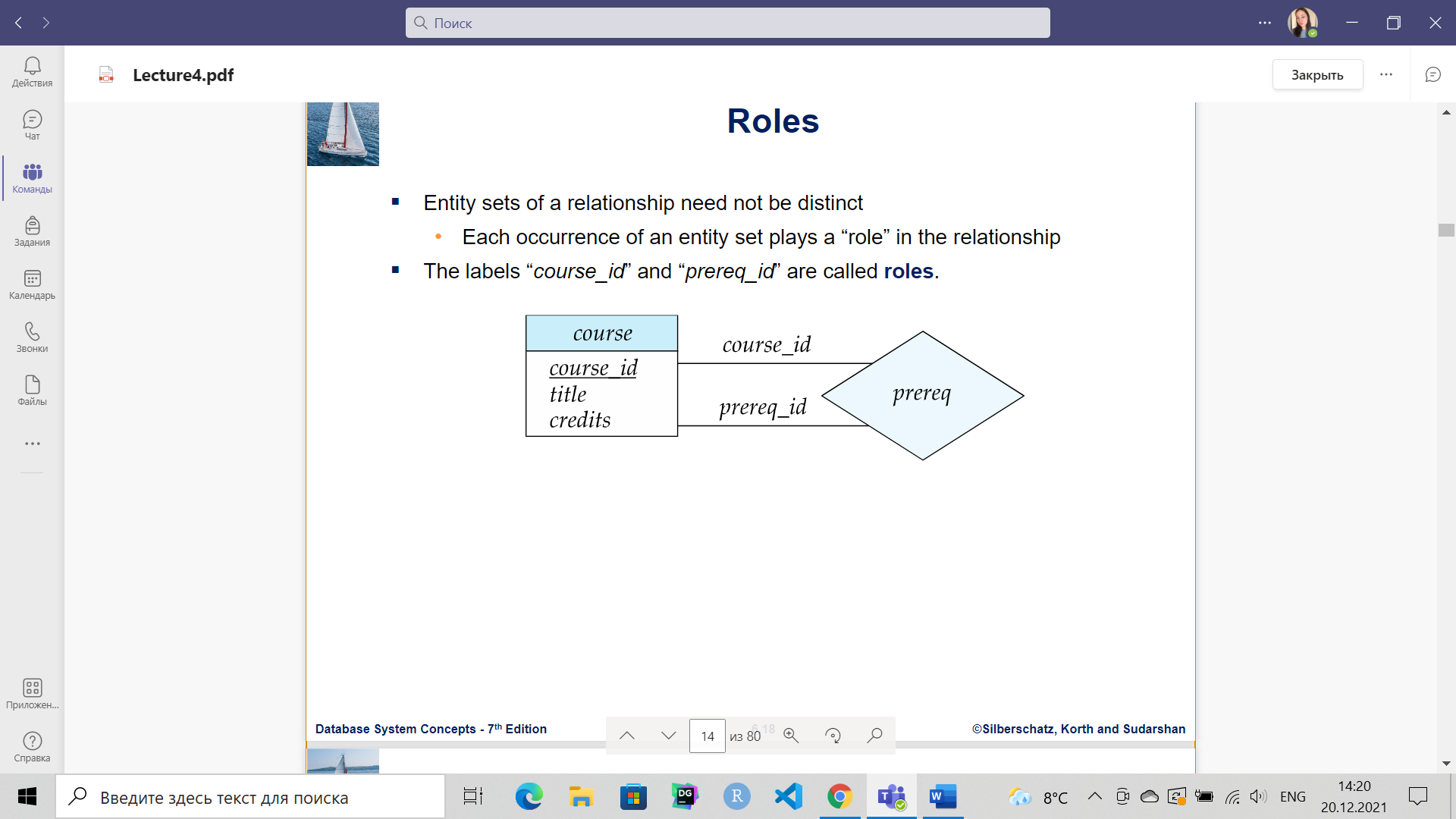
constraints are used to specify rules for data in a table.

**Primary Key** – constraint indicates that a column, or group of columns, can be used as a unique identifier for rows in the table.

•This requires that the values be both unique and not null

**Foreign key** – constraint specifies that the values in a column (or a group of columns) must match the values appearing in some row of another table

Lecture 4

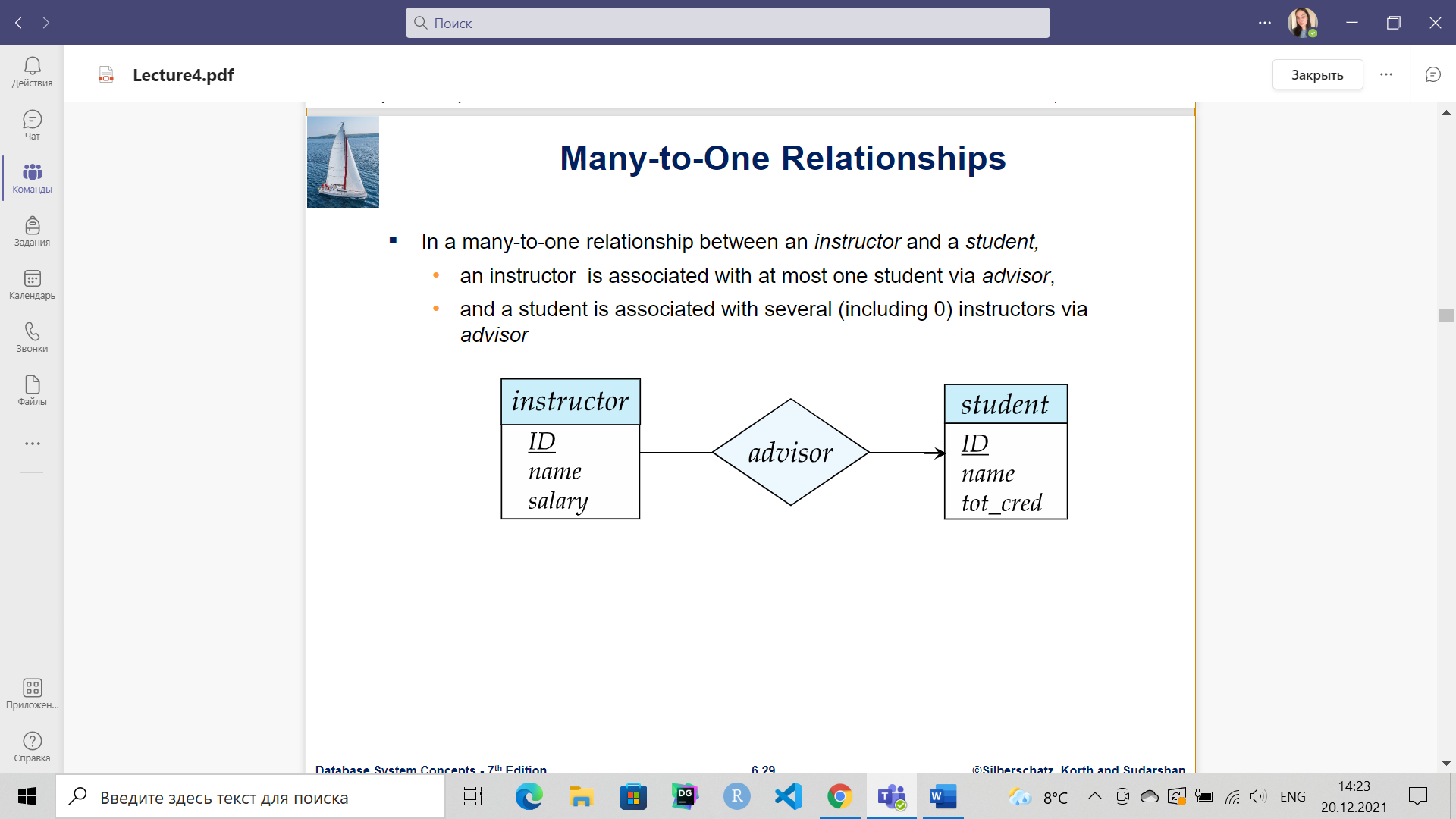


Изображение выглядит как текст

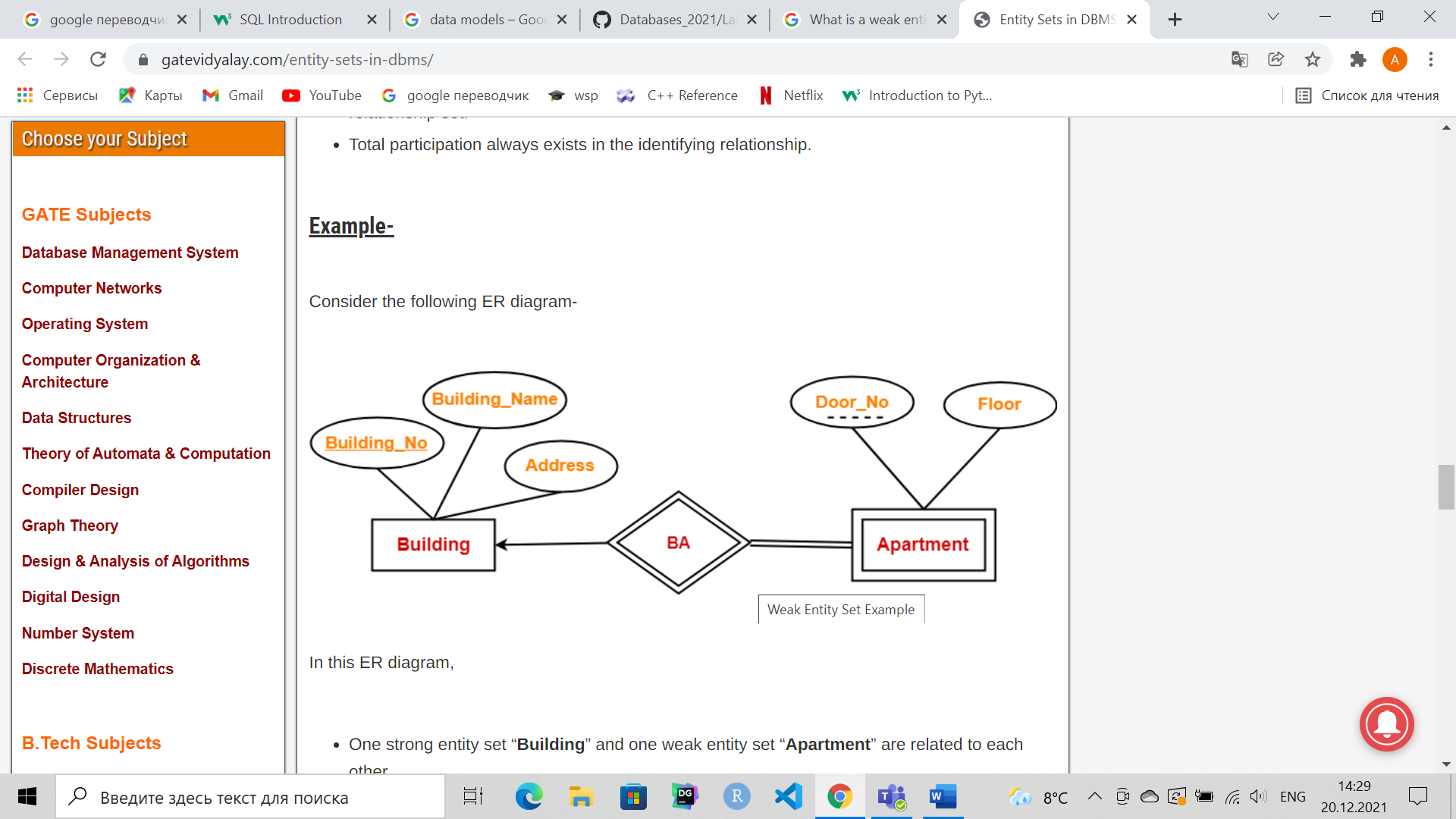
Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

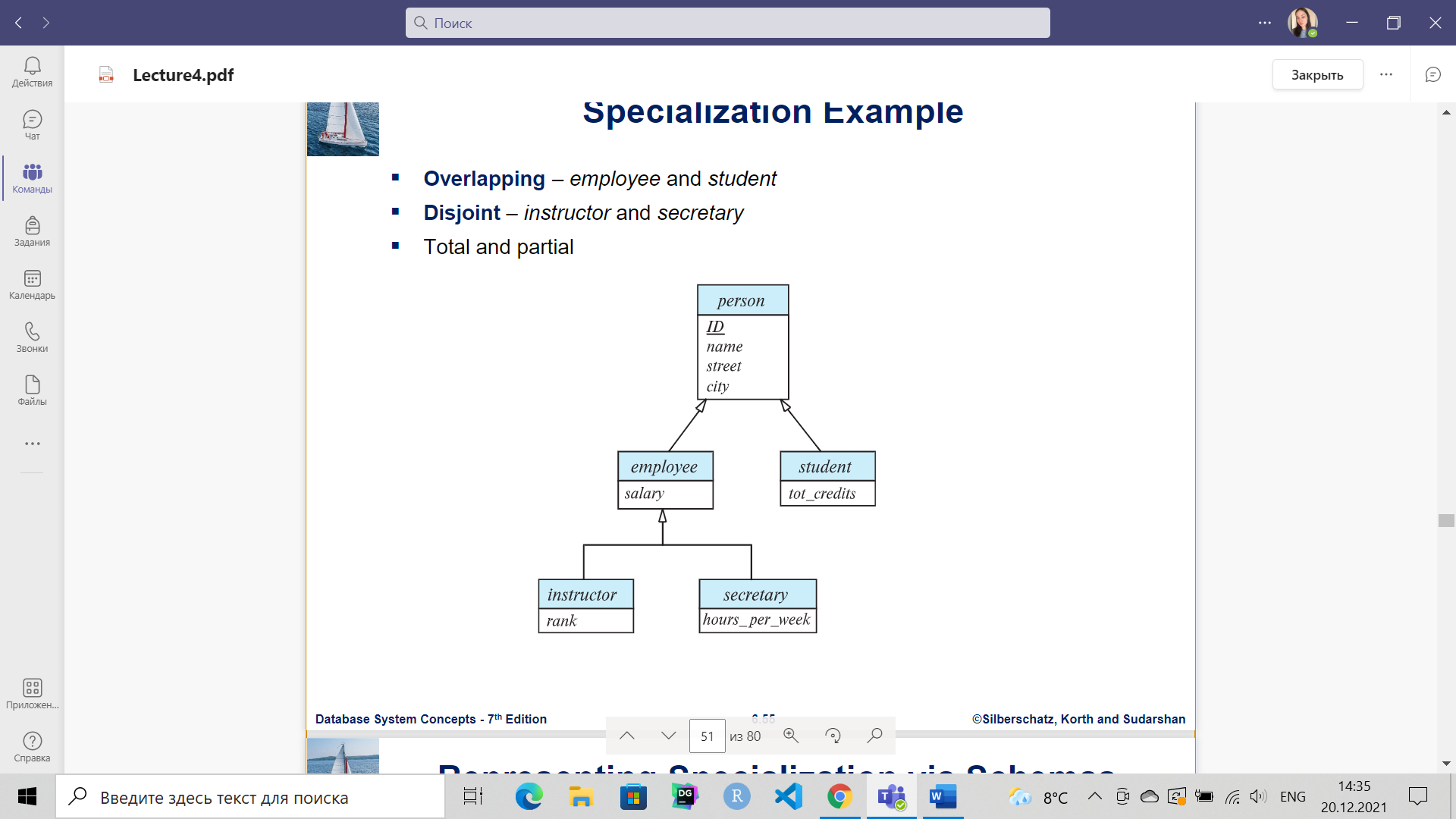
**Weak Entity Sets**: a primary key does not exist for a weak entity set. However, it contains a partial key called as a discriminator. (double rectangle is a weak entity set)



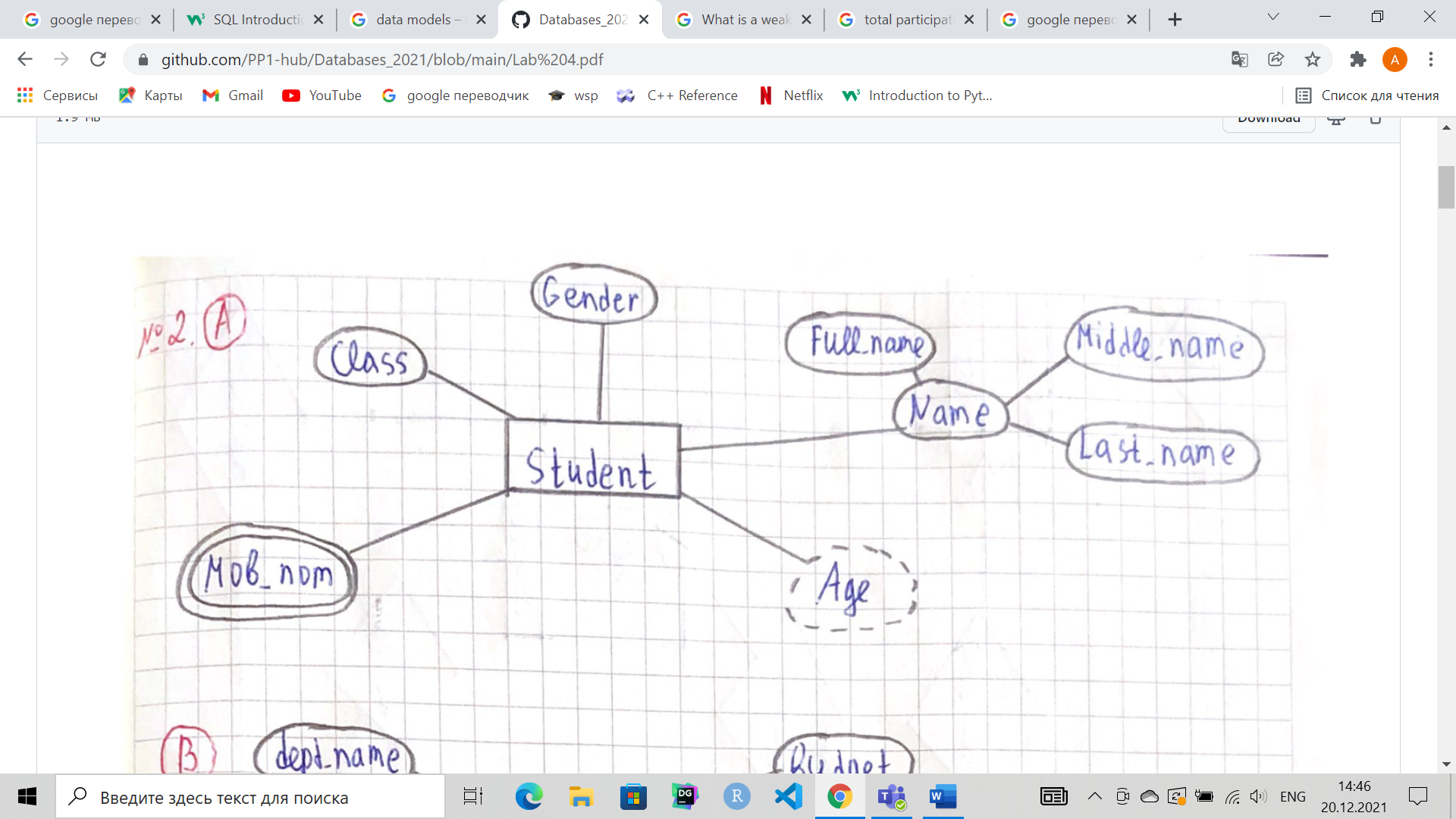
**Total participation (indicated by double line):**

Изображение выглядит как текст

Автоматически созданное описание



**5 attributes:** simple (Class, Gender), composite (Name), derived(Age), multivalued(Mob\_nom)



Lecture 6

**Normal Forms**

**Decomposition** – is the process of breaking down in parts or elements. It breaks the table into multiple tables in a database.

**Lossy decomposition** – There is loss of information as extraneous tuples are added into the relation after natural join of decompositions

**Lossless decomposition** – There is no loss of information as the relation obtained after natural join of decompositions is equivalent to original relation.

* **1st Normal Form**:

Single Valued Attributes: each column should not contain multiple values. (атомарные данные)

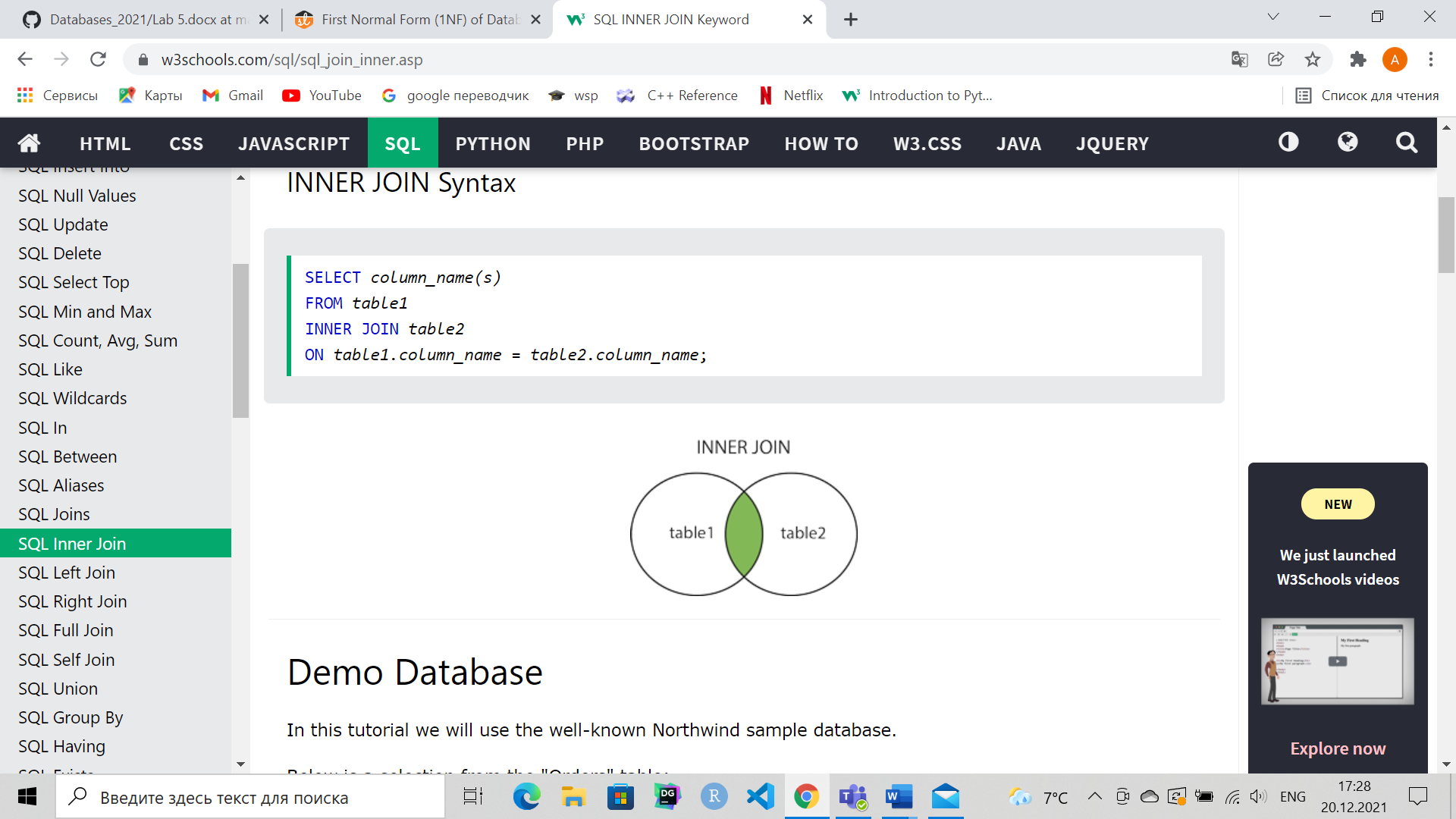
#### Attribute Domain should not change: In each column the values stored must be of the same kind or type.

#### Unique names for Attributes/Column

Lecture 7

**Join operations** – take two relations and return as a result another relation. (Natural join, Inner join, Outer join)

* **Natural join -** matches tuples with the same values for all common attributes, and retains only one copy of each common column
* **Inner join** - takes the intersection of tables



* **Outer join** - result in the union of two tables.

[LEFT JOIN returns only unmatched rows from the left table.](https://mode.com/sql-tutorial/sql-left-join)

[RIGHT JOIN returns only unmatched rows from the right table.](https://mode.com/sql-tutorial/sql-right-join)

[FULL OUTER JOIN returns unmatched rows from both tables.](https://mode.com/sql-tutorial/sql-full-outer-join)

Изображение выглядит как текст

Автоматически созданное описание

**Views** – provides a mechanism to hide certain data from the view of certain users.

create view v as <query expression >

Lecture 8

**Transaction** – consists of a sequence of query and/or update statements and is a “unit” of work

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

**Index** – on an attribute of a relation is a data structure that allows the database system to find those tuples in the relation that have a specified value for that attribute efficiently, without scanning through all the tuples of the relation.

create index <name> on <relation-name> (attribute);

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Lecture 9-10

**Functions and Procedures –** allow “business logic” to be stored in the database and executed from SQL statements.

Trigger – is a statement that is executed automatically by the system as a side effect of a modification to the database.

|  |
| --- |
|  |

CREATE FUNCTION sum(p1 numeric, p2 numeric) RETURNS numeric AS

$$

BEGIN

RETURN p1 + p2;

END;

$$

LANGUAGE plpgsql;

Select sum(5,6)

CREATE table item(

Id serial primary key,

|  |  |
| --- | --- |
|  | name varchar(100), |
|  | price numeric, |
|  | total\_price numeric |
|  |  |
|  | ); |
|  | INSERT INTO item(name, price) VALUES ('AirPods Pro', 170); |
|  |  |
|  | CREATE FUNCTION item\_tax() RETURNS TRIGGER AS |
|  | $$ |
|  | BEGIN |
|  | new.total\_price = new.price + (new.price \* 0.12); |
|  | return new; |
|  |  |
|  | end; |
|  | $$ |
|  | language plpgsql; |
|  |  |
|  | CREATE TRIGGER total\_price\_tax before insert or update on item |
|  | for each row execute procedure item\_tax(); |
|  |  |
|  | INSERT INTO item(name, price) VALUES ('AirPods1' , 150); |
|  | update item set price=200 where id=1; |
|  |  |
|  | select \* |
|  | from item; |