-- Data Definition Language

-- --------------------------------------------------------------------------------------------------------------

**Overview**

до момента имахме:SELECT, INSERT, UPDATE, DELETE

когато говорим за дефиниция на схемата, използваме CREATE, ALTER,  DROP

**--**CREATE TABLE <има на таблица> (

<име на колона> <тип на колона> <constraints>,

…

)

-- VARCHAR(\_\_) 1 ... 4096 Characters – трябва винаги да определим колко ще е размера на VARCHAR, ако искаме повече от 4096, се използва TEXT/CLOB

-- TEXT / CLOB (Character Large OBject)

-- DROP TABLE – трие всички данни в таблицата и са загубени завинаги, ако имаме връзки с други таблици(чрез foreign key примерно), ще даде грешка и първо трябва да си оправим данните

**create table**

CREATE TABLE students (

        fn INT PRIMARY KEY,

        name VARCHAR(256),

        dob DATE

);

при създаване на таблица е хубаво да имаме primary key

CREATE TABLE students (

        fn INT CONSTRAINT students\_pk PRIMARY KEY CHECK(fn > 0), // така задаваме име на primary key-а, --NOT NULL, UNIQUE, creates an implicit [index]

        name VARCHAR(256),

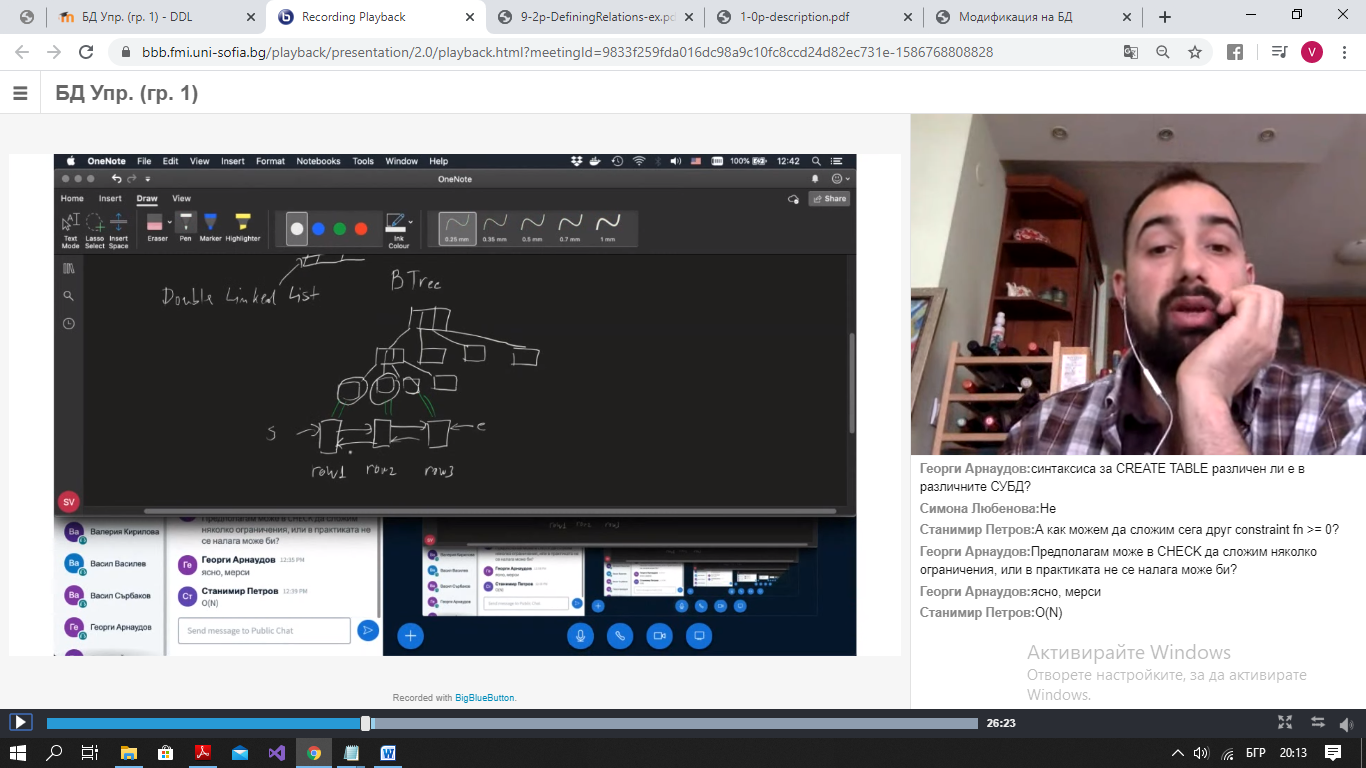
        dob DATE

);

този индекс при primary key-а се създава, за да можем по-лесно да ровим из данните

как работят индексите:

в базата от данни имаме много таблици и всяка една от тях има множество редове. Тези редове не са в таблица вътре в системата, а в накакъв двойно свързан списък. В двойно свързан списък търсим със сложност O(n). Прекалено голяма сложност за мноого дани, за това в базите данни се използват дървета – в общия случай B или B+ дървета. В двоичните делим на по-малки от корена, по-големи от корена. В B дърветата имам такива, които да по-малки от първия, такива които са между първия и втория елемент, такива които са между втория и третия и по-големи от третия. Това навсякъде се повтаря. Листата на дървото са елементите от двойно-свързания списък.



B дървото е върху колоните и ни помага да търсим. Наричаме B дървото индекс. При правенето на праймъри кий, се създава автоматично индекс и за това търсенето по primary key става много бързо.

CREATE TABLE universities (

        id SERIAL PRIMARY KEY, -- MySQL: auto increment, MsSQL: identity, serial автоматично увеличава id-то

        name VARCHAR NOT NULL

);

INSERT INTO universities (name) VALUES ('TU');

INSERT INTO universities (name) VALUES ('PU');

INSERT INTO universities (name) VALUES ('TU');

DELETE FROM universities WHERE id = 4;

INSERT INTO universities (name) VALUES ('NBU');

**check constraints –** в тях можем да реферираме само колоната

различни начини за задаване на ограничения:

CREATE TABLE students (

        fn INT PRIMARY KEY CHECK(fn > 0),

        egn VARCHAR(10) UNIQUE,

        name VARCHAR(256) NOT NULL CHECK(length(name) > 0),

        dob DATE,

        university\_id INT NOT NULL REFERENCES universities(id)

);

предпочита се втория начин при повече ограничения, тъй като се вижда по-добре, също така може да му задаваме име на ограничението и ако при добавяне на стойности изгърми, ще изкара името на ограничението. Хубаво е да са разбити на части, също така са и по-лесни за разбиване.

CREATE TABLE students (

        fn INT NOT NULL,

        egn VARCHAR(10),

        name VARCHAR(256) NOT NULL,

        dob DATE,

year INT DEFAULT 1 – задаване на default стойности

        university\_id INT NOT NULL,

        CONSTRAINT students\_pk PRIMARY KEY (fn),

        CONSTRAINT students\_egn\_uq UNIQUE (egn),

        CONSTRAINT students\_university\_fk FOREIGN KEY (university\_id) REFERENCES universities(id),

        -- CONSTRAINT students\_fn\_name\_ck CHECK(fn > 0 AND length(name) > 0),

        -- Prefer this option:

        CONSTRAINT students\_fn\_ck CHECK(fn > 0),

        CONSTRAINT students\_name\_ck CHECK(length(name) > 0)

);

други ограничения:

-- PRIMARY KEY -- NOT NULL, UNIQUE, creates an implicit [index]

-- CHECK

-- NOT NULL -- values are unique or null, пишат се до колоните

-- UNIQUE – стойностите в колоната са или уникални, или  null

-- FOREIGN KEY

INSERT INTO students (fn, name, dob) VALUES (10020, 'Peter Petrov', '1999-10-24');

INSERT INTO students (fn, name, dob) VALUES (10023, 'Ivan Ivanov', '1999-10-24');

INSERT INTO students (fn, name, dob) VALUES (10024, 'Ivan Petrov', '1999-10-24');

DROP TABLE students;

DATETIME в MySql е аналогично на TIMESTAMP в postgres и GETDATE в mssql

**--ALTER TABLE –** за добавяне и махане на колони от таблици

ALTER TABLE students ADD COLUMN city VARCHAR;

ALTER TABLE students DROP COLUMN updated\_at;

ALTER TABLE students ADD updated\_at TIMESTAMP NOT NULL DEFAULT NOW();

ALTER TABLE students DROP COLUMN city;

-- ALTER TABLE students ADD COLUMN city VARCHAR NOT NULL; -- not working because of null values; трябва да добавим default-ни стойности.

-- Option 1:

ALTER TABLE students ADD COLUMN city VARCHAR;

UPDATE students SET city = 'N/A';

ALTER TABLE students ALTER city SET NOT NULL;

-- Option 2:

ALTER TABLE students ADD COLUMN city VARCHAR NOT NULL DEFAULT 'N/A';

ALTER TABLE students ALTER city DROP DEFAULT; -- remove the default values

ALTER TABLE students ADD CONSTRAINT students\_created\_updated\_ck CHECK (created\_at <= updated\_at);

като е точно 4 символа CHAR(4)

Люко е до 4 VARCHAR(4)

ако е до 4 – VARCHAR(4)

**-- Задача 1**

*-- а) Дефинирайте следните релации:*

-- Product (maker, model, type), където:

--  - модел е низ от точно 4 символа,

--  - производител е низ от точно 1 символ,

--  - тип е низ до 7 символа;

CREATE TABLE product (

        maker CHAR(1),

        model CHAR(4),

        type VARCHAR(7)

);

-- Printer (code, model, price), където:

--  - код е цяло число,

--  - модел е низ от точно 4 символа,

--  - цена с точност до два знака след десетичната запетая;

CREATE TABLE printer (

        code INT,

        model CHAR(4),

        price DECIMAL(6,2) -- xxxx,yy

);

*-- б) Добавете кортежи с примерни данни към новосъздадените релации.*

INSERT INTO product (maker, model, type) VALUES ('A', '1234', 'PRINTER');

INSERT INTO printer (code, model, price) VALUES (101, '1234', 1234.10);

*-- в) Добавете към релацията Printer атрибути:*

--  - type - низ до 6 символа

--                  (забележка: type може да приема стойност 'laser', 'matrix' или 'jet'),

--  - color - низ от точно 1 символ, стойност по подразбиране 'n'

--           (забележка: color може да приема стойност 'y' или 'n').

-- Option 1

ALTER TABLE printer ADD type VARCHAR(6) CHECK(type IN ('laser', 'matrix', 'jet'));

ALTER TABLE printer ADD color VARCHAR(1) NOT NULL DEFAULT 'n' CHECK(color IN ('n', 'y'));

-- ALTER TABLE printer DROP type;

-- ALTER TABLE printer DROP color;

-- Option 2

ALTER TABLE printer ADD type VARCHAR(6);

ALTER TABLE printer ADD color VARCHAR(1) NOT NULL DEFAULT 'n';

ALTER TABLE printer ADD CONSTRAINT printer\_type\_ck CHECK(type IN ('laser', 'matrix', 'jet'));

ALTER TABLE printer ADD CONSTRAINT printer\_color\_ck CHECK(color IN ('n', 'y'));

*-- г) Напишете заявка, която премахва атрибута price от релацията Printer.*

ALTER TABLE printer DROP price;

*-- д) Изтрийте релациите, които сте създали в Задача 1.*

DROP TABLE printer;

DROP TABLE product;