Въведение в моделирането на бази данни

Филип Андонов

6 юли 2022 г.

Съдържание

- Подходът бази от данни
- Основни понятия
- Модели
- Обект-взаимодействие

Два основни подхода за управление на данните

- Традиционен (файлово-базиран) подход
- Подход "бази от данни"

Два основни подхода за управление на данните

Недостатъците на файловия подход са:

- Дублиране/Излишество на данни (redundancy)
- Опасност от нарушаване цялостността на данните (data integrity)
- Зависимост между данни и програми
- Връзките между данните във файловете са задължение на потребителя

Два основни подхода за управление на данните

- Трудно се осъществява едновременен достъп на няколко потребителя до едни и същи данни, както и тяхната защита
- Структурата на файловете зависи от езика, на който е програмирано приложението
- За създаването на нови справки трябва да се дописва програмен код

Защо да използваме база?

Организирането на данните в база с помощта на някаква СУБД позволява

- комбинирането на данни от различни файлове
- избягване на повторения
- работа с точни и непротиворечиви данни (налагат се ограничения за цялостност)
- обработване на транзакции
- управление на едновременния достъп от СУБД
- защита на базата
- независимост на данните

Защо да използваме база?

В най-основна форма базата е компютърният еквивалент на организиран списък с информация където всяка информационна единица се среща еднократно! Организираният списък е подреден в таблица, с полета и записи – flat database.
Примери за такива бази са телефонен указател, пациентите на общо практикуващите лекари, служителите в дадена организация и др.

База от данни

База от данни (БД) е множество от свързани данни, натрупвани, систематизирани и обработвани за нуждите на дадена дейност.

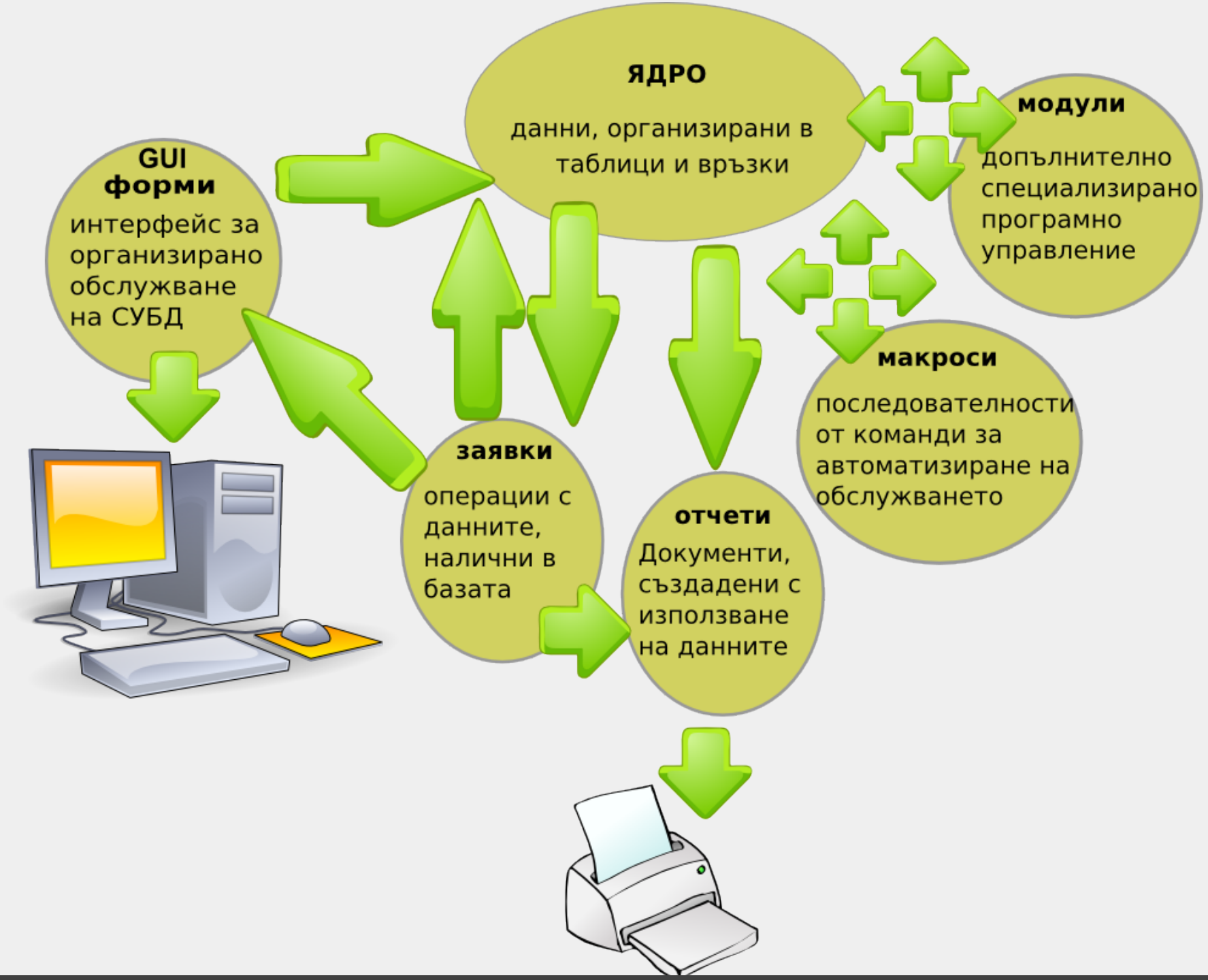
Данна

Данната е факт, произтичащ от реалността, който може да се представи като стойност.

Основни понятия

Свойства на понятието “база от данни”

- Устойчиви/интегрирани данни
- Предметна област – обекти, атрибути и връзки
- Базите се проектират с определена цел


ЯДРО
данни, организирани в таблици и връзки
модули
допълнително специализирано програмно управление
GUI форми
интерфейс за организирано обслужване на СУБД
макроси
последователности от команди за автоматизиране на обслужването
заявки
операции с данните, налични в базата
отчети
Документи, създадени с използване на данните

Критерии

Да представя добре реалността. БД трябва да представлява достоверен образ на реалната система - източник на данни и на това, което се случва в нея, както и да може винаги да генерира информация, съответстваща на актуалното състояние на източника.

Критерии

Да няма повторения. Данните, записани в базата, не трябва да имат повторения както от гледна точка на стриктното отразяване на състава, организацията и семантиката на реалния източник, така и по отношение на физическото им записване.

Критерии

Да спазва независимост на данните от обработката им. Данните са организирани като образ на реалния източник, а програмите за тяхното обработване трябва да са съобразени с така получената структура на данните.

Критерии

Да гарантира сигурност и защита на достъпа до данните.

Критерии

Да осигурява производителност приложенията и заявките.

Модели

Базите данни се изграждат на две нива на моделиране, осигурени съответно от:

- семантични модели
- математически модели

Модели

Прилагането на тези модели позволява да се представи реалният процес – източник на данните като съответстващ му модел на данните.

Модели

По-голямата част от системите за управление на БД са базирани на математически модел, наречен Релационен модел (РМ) на данните, в основата на който стои релационната алгебра.

Релационен модел

- е математически модел с основни “единици” са релациите (таблиците).
- с данните се оперира, като се прилагат операциите на така наречената релационна алгебра.

Релационен модел

“Релация” е множество, чийто елементи са n-орки. Графично елементите на релацията могат да се представят като редове в таблица. Всички елементи-редове са с еднакъв брой позиции (n).

Данните се организират и разпределят в релации, наричани още “таблици” (Tables). Те образуват “ядрото” с данни на БД.

Градивните елементи на базата данн са таблици. Данните се съхраняват единствено в ядрото на базата, в таблиците.

При моделирането на ядрото се определя:

- По какъв начин интересуващите ни характеристики на реалността да се организират в различни таблици.
- Как тези таблици са обвързани и с какви връзки, така че системата от таблици и връзки да описва съществуващото в реалността.

Обект-взаимодействие

Моделът “Обект-Взаимодействие”:

- се прилага в началния етап на проектиране е семантичен модел позволяващ изграждането на формална схема на интересуващото ни реално явление
- получената схема служи за основа при конструиране на базата от данни и осигурява връзката между концептуалното равнище на базата и реалния свят

Обект-взаимодействие

- Моделът ER позволява изграждането на абстрактен образ на реални системи от обекти при прилагане на логически разсъждения за свойствата и взаимната обвързаност на обектите
- Получава се схематично изображение на моделираната реална система, наречено ER- схема, по която лесно се проследяват обектите и взаимодействията
- Моделът ER е замислен при спазване на тясна връзка с естествения език, тъй като той е основно средство за изказ на смисъл

Обект-взаимодействие

Базови термини в модела ER са:

- Свойство
- Обект
- Взаимодействие
- Кардиналност

Свойство

Свойство е елементарна данна, която има смисъл и може да се използва като отделна единица (автономно).

Обект

Конкретен или абстрактен обект, който съществува самостоятелно и независимо, който представлява интерес за дейността на процеса, който се моделира.
Обектът се характеризира с множество Свойства. Всяка конкретна единична проява (екземпляр) на обекта притежава конкретни стойности за всяко от свойствата.

идентификатор

Специално свойство със задачата да има различна стойност за всеки екземпляр, за да могат те да се различават. Не могат да съществуват два различни екземпляра от Обекта, които имат еднакви стойности за свойството-идентификатор.

Обект-взаимодействие

Взаимодействие

Семантична връзка, обвързваща множество Обекти, която представлява интерес за дейността на процеса, който се моделира.

Обект-взаимодействие

Едно Взаимодействие може да притежава Свойства.

Обект-взаимодействие

Взаимодействието не може да притежава собствен идентификатор. Взаимодействието съществува вследствие на съществуването на Обектите, които свързва. То се идентифицира чрез конкатенация на техните идентификатори.

ляво	дясно	описание	пример
1	1	едно към едно	Хора и тяхното ЕГН
0..1	1	опционно от една страна едно към едно	дата на получаване на първо висше образование
0..*	0..*	опционно от двете страни много към много	човек и книга
1	1..*	едно към едно	човек и дата на раждане

Таблица: Кардиналност

Домейн

Множествата от възможни стойности за характеристиките. В релацията, данните се съхраняват в атрибути с определен тип. Типът на атрибута трябва да е подбран така, че да отразява естеството на данните и да позволява записване на данните от домейна.

Термини

В програмните езици

- клас
- обект
- атрибут
- тип

В базите данни

- таблица
- екземпляр, кортеж
- атрибут, поле
- тип

Ограничения на таблицата

- Таблицата не може да съдържа еднакви редове (n-орки).
- Редовете в таблицата не са наредени.
- Стойностите в рамките на даден ред не са наредени.
- Всяка таблица има уникално име.
- Всеки атрибут носи уникално име в рамките на дадена таблица.

Ключ

множество от атрибути, служещи за съхраняване на данните, по които се отличават всички n-орки в една таблица.

Ограничение на ключа (Единственост на ключовете) - една стойност на ключ съществува само веднъж в дадена таблица.
Често се налага ключът да се добави към съществуващите атрибути на обекта, дори да бъде “конструиран” специално за дадената информационна система.

Внимание!

Всяка таблица трябва да има ключ..

Всички стойности на ключове в една таблица са ненулеви.

Обект-взаимодействие

Организацията на данните в една релация се дефинира чрез така наречените “нормални форми”.

Първа нормална форма

Една таблица е в първа нормална форма, ако тя има ключ и всички нейни атрибути са атомарни.

clientnum	name	phonenumbers
3555	Иван Петров	+359 888 123456 +359 898 405820

Един клиент може да има няколко телефонни номера, затова те трябва да бъдат изнесени в отделна таблица.
Имената трябва да са разделени в отделни колони, за да можем да третираме отделните елементи на имената по отделно.

Функционална зависимост между атрибути

Между атрибутите A и B на една таблица съществува функционална зависимост ако на всяка стойност на A винаги съответства точно една определена стойност на B. Атрибутът B е в пряка функционална зависимост от атрибута A, когато за всички стойности на двата атрибута е в сила, че на една стойност от атрибута A съответства точно една стойност от атрибута B.

Втора нормална форма

Една таблица е във втора нормална форма, ако тя е в първа нормална форма всички атрибути, които не са ключ, зависят изцяло от ключа.

Purchases			
client_id	product_id	quantity	client_address

Какво не е наред тук? Атрибутът адрес е свръхопределен. Докато количеството зависи от клиента И от продукта, то адресът зависи само от клиента.
client_id, product_id -> quantity
client_id ->client_address

За да получим втора нормална форма, трябва да направим следните промени.

Purchases		
client_id	product_id	quantity

Clients			
client_id	...	...	client_address

Трета нормална форма

Една таблица е в трета нормална форма, ако тя е във втора нормална форма и няма функционални зависимости измежду останалите атрибути.

1. Планиране на таблици – за всеки обект се предвижда по една таблица.
2. Избор на типове данни – това е формат, който дефинира вида на данните във всяко поле.
3. Посочване на първичен ключ
4. Посочване на общите полета - първичен и външен ключ.

5. Свързване на таблиците - релации
   - Едно-към-едно
     Най-често това се моделира като допълнителен атрибут (колона) в същата таблица.
   - Едно-към-много
     Това се моделира като в подчинената таблица (от страната на "много") се добавя атрибут (колона), който съдържа ключа на кореспондиращия запис в другата таблица. Този атрибут се нарича "външен ключ". Записите от страна на родителската таблица не знае за (потенциално многобройните) си деца.

5. Свързване на таблиците - релации
   - Много-към много
     Моделира се чрез трета таблица. Тази таблица обикновено изразява действие (купувам, записвам, наемам...) и се нарича таблица на взаимодействие. Тя описва измежду всички възможни двойки, получени от един елемент от едната и един елемент на другата таблица, кои от тях наистина са в това взаимодействие. Тази таблица няма собствен ключ. Вместо това, нейният ключ е съставен, като се състои от идентификаторите (ключовете) на таблиците, чийто екземпляри свързва. Ако взаимодействието между два обекта може да се случва повече от веднъж, то към ключа се добавя и колона или колони за отбелязване на времето. Самото взаимодействие също може да има атрибути, ако те зависят наистина и от ключовете на двете свързани таблици.

6. Поддържане на цялостност на връзките – правила за контрол на изтриването и промяната на данните в две свързани таблици.
   Кога се прилагат:
   - Общото поле е първичен ключ на таблица
   - Свързаните полета имат един и същ формат
   - Таблиците са от една и съща база

Ограничения:

1. Преди добавяне на запис в свързана таблица, в първичната таблица трябва да съществува съответен запис.
2. Стойността на първичния ключ в първичната таблица не може да се променя, ако в свързаната таблица има въведени съответни записи.
3. Запис в първичната таблица не може за бъде изтрит, ако в свързаната има съответстващи му.

SELECT заявка

Служи за извличане на данни от една или повече таблици Има следния базов синтаксис:
SELECT колона1, колона2, . . . колонаN FROM таблица
WHERE условие

Заявки

Условието може да е всякакъв израз, който връща булева стойност

- $=$
- $>=$
- $<=$
- $<>$

Заявки

Прости логически изрази могат да се комбинират с логическите оператора:

- OR
- AND
- NOT

Заявки

Пример
"select * from customer where LastName like "%Jo%

INSERT заявка

Служи за добавяне на нов ред в една таблица. Има следния базов синтаксис:

INSERT INTO таблица (колона1, колона2, . . . колонаN)
VALUES (стойност1, стойност2, . . . , стойностN)

CREATE TABLE заявка

Служи за създаване на нова таблица. Има следния базов синтаксис:
CREATE TABLE таблица(Име на колона1 тип на колона 1 модификатор, Име на колона2, тип на колона 2, модификатор, ... Име на колонаN, тип на колона N, модификатор)

Заявки

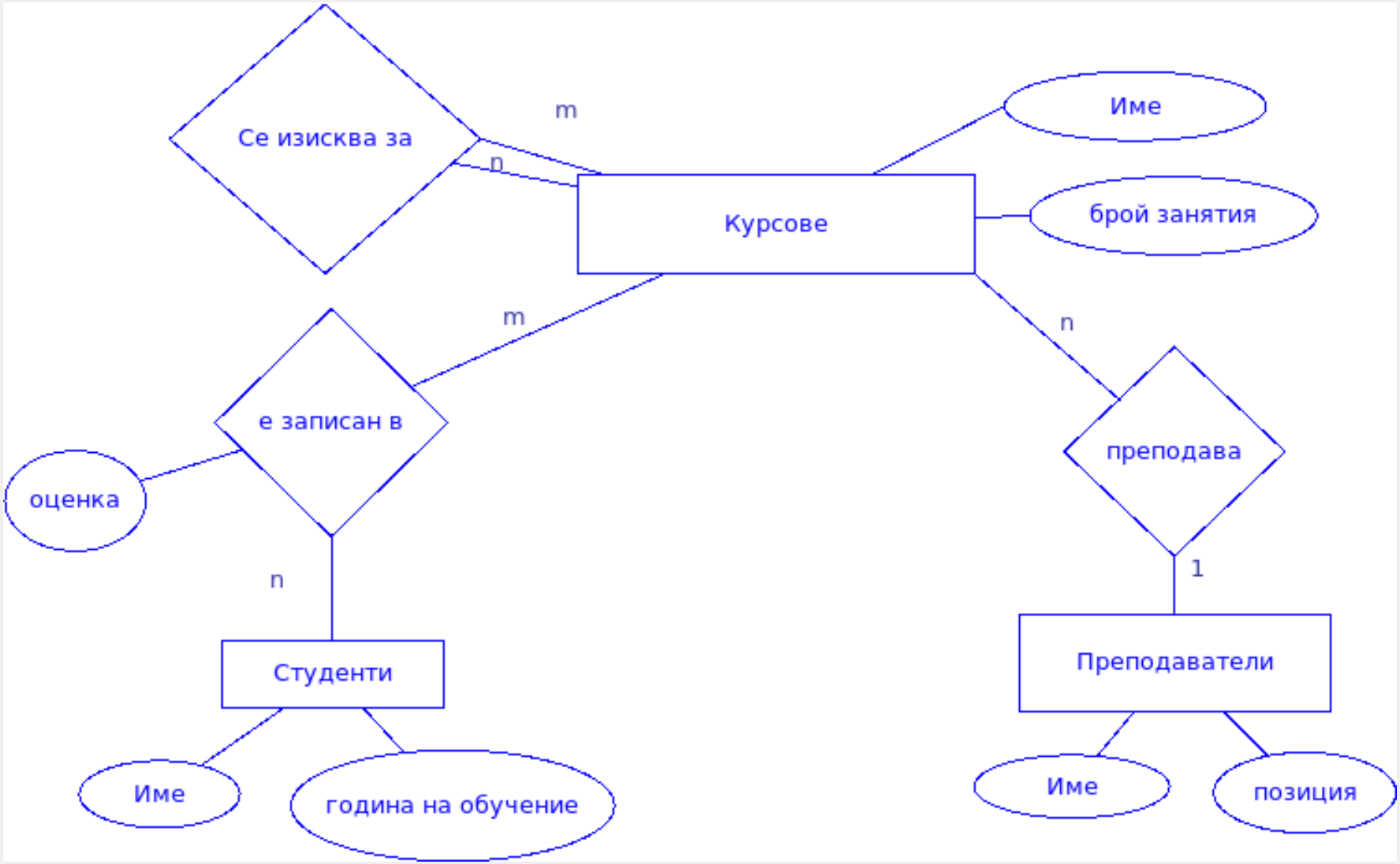
Командата "CREATE TABLE"се използва за създаване на нова таблица в SQLite база данни. Следните атрибути се задават при извикването на CREATE TABLE:

- името на новата таблица.
- базата данни, в която се създава новата таблица.
- Името на всяка колона от таблицата.
- Типа на всяка колона от таблицата.
- Стойност по подразбиране на всяка колона в таблицата.
- Незадължително ПЪРВИЧЕН КЛЮЧ за таблицата (PRIMARY KEY).
- Набор от ограничения за всяка колона. SQLite поддържа UNIQUE, NOT NULL, CHECK и FOREIGN KEY.

Заявки

Примерна заявка:
CREATE TABLE food (id TEXT PRIMARY KEY, desc TEXT, water FLOAT, kcal FLOAT, protein FLOAT, fat FLOAT, ash FLOAT, carbs FLOAT, fiber FLOAT, sugar FLOAT)

Задача


Се изисква за
m
n
Име
Курсове
брой занятия
m
n
е записан в
преподава
оценка
n
1
Студенти
Преподаватели
Име
година на обучение
Име
позиция

Задача

Проектирайте база данни по дадената по-горе диаграма Обект-Взаимодействие.

Благодаря за вниманието!

Въпроси?