Проект по Системи за управление на Бази от данни - практикум

База от данни за система за увеселителни паркове

Изготвил: Весела Стоянова

Специалност: Информационни системи

Факултетен номер: 71949

Група: 1

Съдържание:

Част 1. Описание на множествата от същности:

1. Обхват на модела. Дефиниране на задачата:

Базата от данни за система за увеселителни паркове ще съхранява информация за данните в увеселителния парк. Системата разполага с няколко парка, които могат да бъдат в различни градове. В един и същи град **не** може да има повече от един парк. Всеки парк се определя еднозначно по името на парка. Името на парка е името на града.
Системата работи с **посетители**, за които пази информация. Съхраняват се **име, e-mail, телефонен номер, възраст, височина, уникален номер на посетителя**. Те се определят еднозначно по уникален номер на посетителя.
В парковете работят **служители**. Те се определят еднозначно по служебен номер на служителя. За служителите се пази информация за **служебен номер, парк, в който работят, магазин, в който работят, име, телефонен номер и e-mail**. Всеки служител отговаря за определен магазин. Може за един магазин да отговарят няколко служителя, но един служител отговаря за точно един магазин.
Системата се занимава с продажба на **билети**. Билетите могат да бъдат еднодневен, двудневен, седмичен, семеен, комбо и детски. Всеки билет е за определен парк. Цената на билетите е в лева. Всеки билет важи за различен брой посетители.
За всеки тип билет се пази информация за уникален номер на билета, име на парка, за който се отнася, тип на билета, цена и броя на посетителите.
Всеки парк има **атракции**. За всяка атракция се пази информация за **име на парка, в който се намира, име на атракцията, тип, състояние, дължина, височина, скорост, продължителност, минимална възраст, минимална височина, опасности и работно време.**

Типът на атракцията може да бъде бърза, спокойна, водна, тъмна, шумна, страшна, детска.
Състоянието на атракцията може да бъде работещо и неработещо.
Във всеки парк има **магазини за сувенири**. За всеки магазин се пази информация за **парка, в който се намира, име на магазина и състояние.**
За всеки продукт се пази информация за **инвентарен номер, служебен номер на доставчик, име на магазина, в който се намира, име на продукта, тип на продукта и цена на продукта.**
Типът на продукта може да бъде хранителен продукт, дреха и други.
За хранителния продукт се пази информация за **срок на годност**.
За дрехите се пази информация за **размер, пол и цвят**.
За другите се пази информация за **размер на продукта и материала, от който е направен**.
За доставчика се пази информация за **служебен номер, име, имейл и телефонен номер.** Цената на продуктите е в лева.

2. Множества от същности и техните атрибути:

- **Паркове** – име: , адрес;
- **Атракции** – име, име на парка, тип, минимална възраст, продължителност, скорост, опасности, минимална височина, работно време, височина, дължина, състояние;
- **Посетители** – уникален номер, височина, години, име, имейл, телефонен номер;
- **Билети** – уникален номер, име на парка, тип, брой на посетители, цена;
- **Служители** – служебен номер, име на парк, име на магазин, имейл, телефонен номер, име на служителя;
- **Магазини** – име, име на парк, състояние;

- Продукти – инвентарен номер, служебен номер на доставчика, име на магазина, цена, име на продукта, тип на продукта;
  - **Храна** – инвентарен номер, срок на годност;
  - **Дрехи** – инвентарен номер, размер, пол, цвят;
  - **Други** – инвентарен номер, размер, материал
- **Доставчици** – служебен номер, име, имейл, телефонен номер.

3. Домейн на атрибутите:

- **Паркове** – име: низ, адрес: низ;
- **Атракции** – име: низ, име на парка: низ, тип: низ, минимална възраст: цяло положително число, продължителност: цяло положително число, скорост: цяло положително число, опасности: низ, минимална височина: реално положително число, работно време: низ, височина: реално положително число , дължина: реално положително число, състояние: низ;
- **Посетители** – уникален номер: низ, височина: реално положително число, години: цяло положително число, име: низ, имейл: низ, телефонен номер: низ;
- **Билети** – уникален номер: низ, име на парка: низ, тип: низ, брой на посетители: цяло положително число, цена: реално положително число;
- **Служители** – служебен номер: низ, име на парк: низ, име на магазин: низ, имейл: низ, телефонен номер: низ, име на служителя: низ;
- **Магазини** – име: низ, име на парк: низ, състояние: низ;
- **Продукти** – инвентарен номер: низ, служебен номер на доставчика: низ, име на магазина: низ, цена: реално положително число, име на продукта: низ, тип на продукта: низ;
  - **Храна** – инвентарен номер: низ, срок на годност: дата;
  - **Дрехи** – инвентарен номер: низ, размер: цяло положително число, пол: низ, цвят: низ;

➢ **Други** – инвентарен номер: низ, размер: цяло положително число, материал: низ;

- **Доставчици** – служебен номер: низ, име: низ, имейл: низ, телефонен номер: низ.

4. Връзки:

- В един парк работят много служители. Един служител работи точно в един парк.
- Един посетител може да отиде в един парк. В един парк могат да дойдат много посетители.
- Всяка атракция се намира в един парк. В един парк може да има много атракции.
- Всяка атракция е един тип. Може да има няколко атракции от един и същи тип.
- Всеки магазин за сувенири се намира в един парк. В един парк може да има много магазини за сувенири.
- Един служител може да работи в един магазин за сувенири. В един магазин могат да работят много служители.
- Всеки продукт принадлежи на един магазин. В един магазин може да има много продукти.
- Всеки продукт е един тип. От всеки тип може да има много продукти.
- Всеки продукт е доставян от един доставчик. Един доставчик може да доставя много продукти.

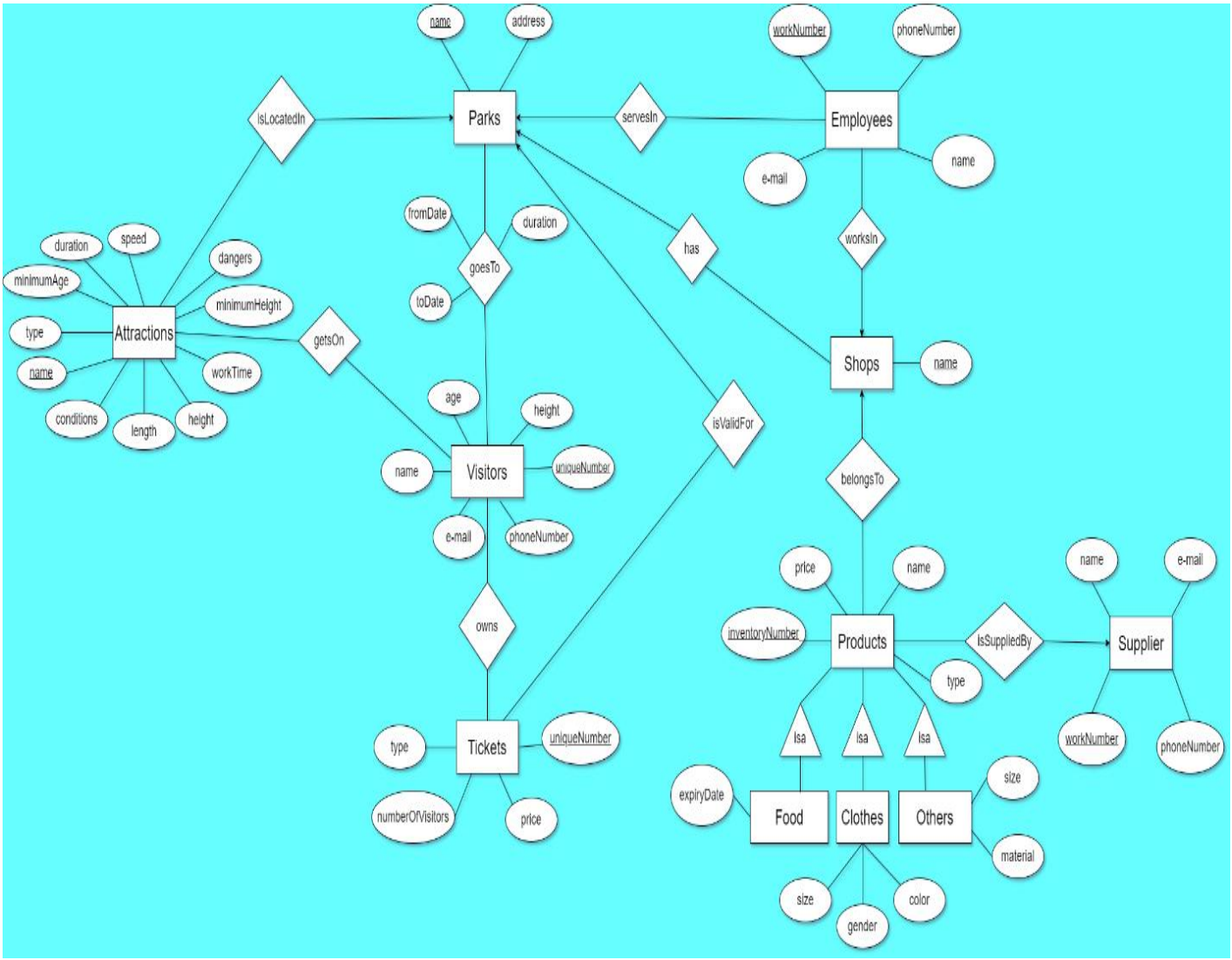
5. Ограничения по единствена стойност, референтна цялостност и друг тип ограничения:

- **Парк** – име: еднозначно определя парка.
- **Посетители** – уникален номер: еднозначно определя посетителя, имейл: проверка за валиден имейл.
- **Служител** – служебен номер: еднозначно определя служителя, имейл: проверка за валиден имейл.
- **Атракция** – име: еднозначно определя атракцията.
- **Магазин за сувенири** – име: еднозначно определя магазина.
- **Доставчик** – служебен номер: еднозначно определя доставчика, имейл: проверка за валиден имейл.
- **Билети** – уникален номер: еднозначно определя билета, имейл: проверка за валиден имейл.
- **Продукти** – инвентарен номер: еднозначно определя продукта.

6. Правила и проверки:

- Проверка за всички реални и цели числа дали са положителни;
- **Атракции** – за тип на атракцията проверка дали е бърза, спокойна, водна, тъмна, шумна, страшна или детска;
- **Атракции** – за състояние проверка дали е отворено или затворено;
- **Билет** – за тип на билета проверка дали е еднодневен, двудневен, седмичен, семеен, комбо или детски;
- **Магазини** – за състояние проверка дали е отворено или затворено;
- **Продукти** – за тип на продукта проверка дали е храна, дреха или друго;
- **Дрехи** – за пол проверка дали е за мъж или за жена.

Част 2. E/R диаграма на модела на БД:



Част 3. Преобразуване от E/R модел към релационен модел:

1. Релационен модел на данните:

Parks (name, address)
Attractions (name, parkName, type, minimumAge, duration, speed, dangers, minimumHeight, workTime, height, length, conditions)
GetsOn (attractionName, visitorsUniqueNumber)
GoesTo (parksName, visitorsUniqueNumber, fromDate, toDate, duration)
Visitors (uniqueNumber, height, age, name, e-mail, phoneNumber)

Owns (visitorsUniqueNumber, ticketsUniqueNumber)
Tickets (uniqueNumber, parkName , type, numberOfVisitors, price)
Employees (workNumber, parkName, shopName, e-mail, phoneNumber, name)
Shops (name, parkName)
Products (inventoryNumber, supplierWorkNumber, price, name, type)
Food (inventoryNumber, expiryDate)
Clothes (inventoryNumber, size, gender, color)
Others (inventoryNumber, size, material)
Supplier (workNumber, name, e-mail, phoneNumber)

2. Релационен модел на данните (релационни схеми):

Схемата на базата от данни се състои от следните релационни схеми:
Parks (name, address)
Attractions (name, parkName, type, minimumAge, duration, speed, dangers, minimumHeight, workTime, height, length, conditions)
GetsOn (attractionName, visitorsUniqueNumber)
GoesTo (parksName, visitorsUniqueNumber, fromDate, toDate, duration)
Visitors (uniqueNumber, height, age, name, e-mail, phoneNumber)
Owns (visitorsUniqueNumber, ticketsUniqueNumber)
Tickets (uniqueNumber, parkName , type, numberOfVisitors, price)
Employees (workNumber, parkName, shopName, e-mail, phoneNumber, name)
Shops (name, parkName)
Products (inventoryNumber, supplierWorkNumber, price, name, type)
Food (inventoryNumber, expiryDate)
Clothes (inventoryNumber, size, gender, color)
Others (inventoryNumber, size, material)
Supplier (workNumber, name, e-mail, phoneNumber)

3. ФЗ и Нормализация:

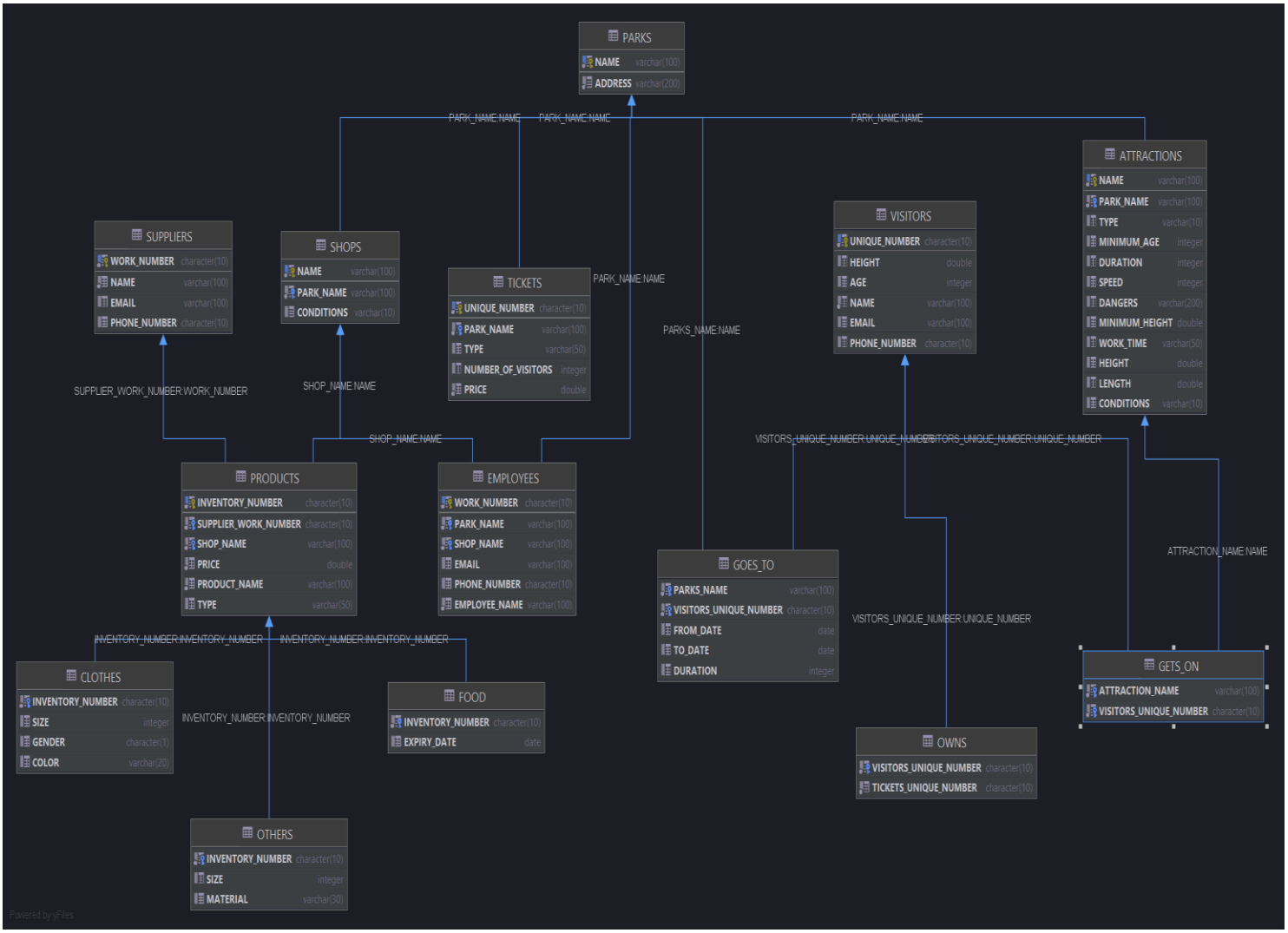
Когато се използва първо ER модел за да се направи дизайн на БД и след това се преобразува до релационни схеми, като релационните схеми, които се получават са в 3NF. Счита се, че ако схемата на БД е в НФБК, то дизайна на БД е добър. Така че, при използването на подхода първо абстрактен модел – ER диаграма, която след това се преобразува до релационни схеми, единствено трябва да проверите за ФЗ в релационните схеми, които нарушават НФБК. Ако намерите такива трябва да декомпозирате съответната релационна схема до две релации.

За схемата на базата от данни по-горе са в сила следните ФЗ:

- **ФЗ – 1:** name -> address (Parks)
- **ФЗ – 2:** name, parkName -> type, minimumAge, duration,, speed, dangers, minimumHeight, worktime, height, length, conditions (Attractions)
- **ФЗ – 3:** parkName, visitorsUniqueNumber -> fromDate, toDate, duration (GoesTo)
- **ФЗ – 4:** uniqueNumber -> height, age, name, e-mail, phoneNumber (Visitors)
- **ФЗ – 5:** uniqueNumber, parkName -> type, numberOfVisitors, price (Tickets)
- **ФЗ – 6:** workNumber, parkName, shopName -> e-mail, phoneNumber, name (Employees)
- **ФЗ – 7:** inventoryNumber, supplierWorkNumber -> price, name, type (Products)
- **ФЗ – 8:** inventoryNumber -> expiryDate (Food)
- **ФЗ – 9:** inventoryNumber -> size, gender, color (Clothes)
- **ФЗ – 10:** inventoryNumber -> size, material (Others)
- **ФЗ – 11:** workNumber -> name, e-mail, phoneNumber (Supplier)

За всички ФЗ на релациите е в сила, че в лявата част се намира суперключ за релацията, следователно всички релации са в НФБК. Не се налага да правим нищо допълнително.

Част 4. Картинка на релационния модел:



Част 5. Описание на функциите:

1. Функция 1:

✓ averageAgeOfVisitors(parkName VARCHAR(100));

✓ Таблична функция;

✓ Функция, която по подадено име на парк връща средната възраст на посетителите в този парк;

```
-- Функция, която по подадено име на парк връща средната възраст на посетителите на този парк
CREATE FUNCTION FN71949.averageAgeOfVisitors(parkName VARCHAR(100))
RETURNS TABLE(
    park_name VARCHAR(100),
    avg_visitors_age INTEGER
    )
RETURN
    SELECT PARK.NAME AS PARK_NAME, AVG(VISITOR.AGE) AS AVG_VISITORS_AGE
    FROM PARKS AS PARK, VISITORS AS VISITOR, GOES_TO AS GOES_TO
    WHERE GOES_TO.PARKS_NAME = PARK.NAME
    AND GOES_TO.VISITORS_UNIQUE_NUMBER = VISITOR.UNIQUE_NUMBER
    AND PARK.NAME = parkName
    GROUP BY PARK.NAME;
```

✓ Извикване на функцията за парк с име Plovdiv;

```
SELECT * FROM TABLE (FN71949.averageAgeOfVisitors('Plovdiv'));
```

✓ Резултат от извикване на функцията за парка с име Plovdiv;

	PARK_NAME	AVG_VISITORS_AGE
1	Plovdiv	26

✓ Извикване на функцията за парк с име Montana;

```
SELECT * FROM TABLE (FN71949.averageAgeOfVisitors('Montana'));
```

✓ Резултат от извикване на функцията за парка с име Montana;

	PARK_NAME	AVG_VISITORS_AGE
1	Montana	17

2. Функция 2:

- ✓ attractionsBasicInfo(parkName VARCHAR(100));
- ✓ Таблична функция;
- ✓ По подадено име на парк връща най-важната информация за атракциите в него – име на атракцията, работното време, типа на атракцията и минималната височина;

```
--функция, която по подадено име на парк връща най-важната информация за атракциите в него
CREATE FUNCTION FN71949.attractionsBasicInfo(parkName VARCHAR(100))
RETURNS TABLE(
    attractionName VARCHAR(100),
    attractionWorkTime VARCHAR(50),
    attractionType VARCHAR(10),
    attractionMinimumHeight DOUBLE
)
RETURN
    SELECT ATTR.NAME AS ATTRACTION_NAME, ATTR.WORK_TIME AS ATTRACTION_WORK_TIME,
           ATTR.TYPE AS ATTRACTION_TYPE, ATTR.MINIMUM_HEIGHT AS ATTRACTION_MIN_HEIGHT
    FROM PARKS AS PARK, ATTRACTIONS AS ATTR
    WHERE PARK.NAME = ATTR.PARK_NAME
    AND PARK_NAME = parkName;
```

- ✓ Извикване на функцията за парка с име Gabrovo;

```
SELECT * FROM TABLE (FN71949.attractionsBasicInfo('Gabrovo'));
```

- ✓ Резултат от извикване на функцията за парка с име Gabrovo;

	ATTRACTIONNAME	ATTRACTIONWORKTIME	ATTRACTIONTYPE	ATTRACTIONMINIMUMHEIGHT
1	Dragon	08:00 - 21:00	FAST	130
2	Jungle Cruise	09:00 - 21:00	CHILD	80
3	House of illusions	09:00 - 21:00	DARK	120

- ✓ Извикване на функцията за парка с име Sofia;

```
SELECT * FROM TABLE (FN71949.attractionsBasicInfo('Sofia'));
```

- ✓ Резултат от извикването на функцията за парка с име Sofia;

	ATTRACTIONNAME	ATTRACTIONWORKTIME	ATTRACTIONTYPE	ATTRACTIONMINIMUMHEIGHT
1	Splash Mountain	08:00 - 20:00	CALM	100
2	Pirates of Caribbean	09:00 - 20:00	FAST	140
3	Tower of power	09:00 - 19:00	WATER	120
4	Pink Star	09:00 - 21:00	FAST	140

3. Функция 3:

✓ attractionVisitorBasicInfo(attractionName VARCHAR(100));

✓ Таблична функция;

✓ По подадено име на атракция връща информация за посетителите, които са се качили на нея – име, години и телефонен номер на посетителя;

```
--функция, която по подадено име на атракция връща информация за посетителите, които са се качили на нея
CREATE FUNCTION FN71949.attractionVisitorBasicInfo(attractionName VARCHAR(100))
RETURNS TABLE(
    attractionName VARCHAR(100),
    visitorName VARCHAR(100),
    visitorAge INTEGER,
    visitorPhoneNumber CHAR(10)
)
RETURN
    SELECT ATTR.NAME AS ATTR_NAME, VISITOR.NAME AS VISITOR_NAME, VISITOR.AGE AS VISITOR_AGE, VISITOR.PHONE_NUMBER AS VISITOR_PHONE_NUMBER
    FROM ATTRACTIONS AS ATTR, VISITORS AS VISITOR, GETS_ON AS GETS
    WHERE ATTR.NAME = GETS.ATTRACTION_NAME
    AND VISITOR.UNIQUE_NUMBER = GETS.VISITORS_UNIQUE_NUMBER
    AND ATTR.NAME = attractionName;
```

✓ Извикване на функцията за атракцията Jungle Snake;

```
SELECT * FROM TABLE (FN71949.attractionVisitorBasicInfo('Jungle Snake'));
```

✓ Резултат от извикване на функцията за атракцията Jungle Snake;

	ATTRACTIONNAME	VISITORNAME	VISITORAGE	VISITORPHONENUMBER
1	Jungle Snake	Christian Ivanov	12	0894223442
2	Jungle Snake	Christina Emilova	22	0884337700
3	Jungle Snake	Zornitsa Ilianova	21	0884227482
4	Jungle Snake	Ivan Stoyanov	17	0881335821

✓ Извикване на функцията за атракцията Pirates of Caribbean;

```
SELECT * FROM TABLE (FN71949.attractionVisitorBasicInfo('Pirates of Caribbean'));
```

✓ Резултат от извикване на функцията за атракцията Pirates of Caribbean;

	ATTRACTIONNAME	VISITORNAME	VISITORAGE	VISITORPHONENUMBER
1	Pirates of Caribbean	Christian Ivanov	12	0894223442
2	Pirates of Caribbean	Kamelia Gerasimova	32	0892524431
3	Pirates of Caribbean	Nikola Kolev	27	0892343472
4	Pirates of Caribbean	Ivan Stoyanov	17	0881335821

Част 6. Описание на тригерите:

1. Тригер 1:

✓triggerUpdateWorkTime

✓ After update тригер;

✓ Записва в отделна таблица информация за промененото работно време на дадена атракция;

```
--Тригер, който в отделна таблица записва информация за промененото работно време на дадена атракция
CREATE TABLE FN71949.ATTR_WORKTIME_CHANGES(
    CHANGETIME TIMESTAMP,
    ATTR_WORKTIME VARCHAR(5000)
)

DROP TABLE FN71949.ATTR_WORKTIME_CHANGES;

CREATE TRIGGER triggerUpdateWorkTime
    AFTER UPDATE OF WORK_TIME ON FN71949.ATTRACTIONS
    REFERENCING OLD AS O NEW AS N
    FOR EACH ROW
        INSERT INTO FN71949.ATTR_WORKTIME_CHANGES
        VALUES (CURRENT_TIMESTAMP, 'Attraction ' || O.NAME || ' changes the work time from ' || O.WORK_TIME || ' to ' || N.WORK_TIME);

SELECT * FROM FN71949.ATTR_WORKTIME_CHANGES;

UPDATE FN71949.ATTRACTIONS
SET WORK_TIME = '08:00 - 20:00'
WHERE NAME = 'Splash Mountain';
```

✓ Резултат от изпълнението на тригера

	CHANGETIME	ATTR_WORKTIME
1	2022-01-14 19:37:35.414603	Attraction Splash Mountainchanges the work time from 09:00 - 20:00 to 08:00 - 20:00

2. Тригер 2:

✓ updatePriceForProduct

✓ After update тригер;

✓ Извиква процедурата productProcedure (IN productName VARCHAR(100), IN productNewPrice DOUBLE);

✓ Записва в отделна таблица информацията за промяна на цената на даден продукт, заедно с неговото име;

```
CREATE TABLE NEW_PRODUCT_PRICE(
    PRODUCT_NAME VARCHAR(100),
    PRODUCT_PRICE DOUBLE
);

DROP TABLE NEW_PRODUCT_PRICE;

CREATE TRIGGER updatePriceForProduct
AFTER UPDATE OF PRICE ON FN71949.PRODUCTS
    REFERENCING OLD AS O NEW AS N
    FOR EACH ROW
    WHEN (O.PRICE > 0)
        CALL FN71949.productProcedure( productName: O.PRODUCT_NAME,  productNewPrice: N.PRICE);

DROP TRIGGER updatePriceForProduct;

SELECT * FROM FN71949.NEW_PRODUCT_PRICE;

CREATE PROCEDURE FN71949.productProcedure(IN productName VARCHAR(100), IN productNewPrice DOUBLE)
RESULT SETS 1
LANGUAGE SQL
SPECIFIC productProcedure
BEGIN
    DECLARE firstCursor CURSOR WITH RETURN FOR SELECT * FROM PRODUCTS WHERE PRODUCTS.PRODUCT_NAME<>productName;
    INSERT INTO NEW_PRODUCT_PRICE VALUES (productName, productNewPrice);
    OPEN firstCursor;
end;

UPDATE FN71949.PRODUCTS
SET PRICE = 40
WHERE PRODUCT_NAME = 'DRESS';
```

✓ Резултат от изпълнението на тригера;

	PRODUCT_NAME	PRODUCT_PRICE
1	DRESS	40
2	DRESS	40

Част 7. Описание на изгледите:

1. Изглед 1:

✓ whereVisitorCanGetsOn(VISITOR_NAME, VISITOR_UNIQUE_NUMBER, VISITOR_AGE, VISITOR_HEIGHT, ATTRACTION_NAME, ATTRACTION_PARK_NAME);

✓ Показва за всеки посетител на кои атракции може да се качва спрямо височината и възрастта му;

```
CREATE VIEW FN71949.whereVisitorCanGetsOn(VISITOR_NAME, VISITOR_UNIQUE_NUMBER, VISITOR_AGE, VISITOR_HEIGHT, ATTRACTION_NAME, ATTRACTION_PARK_NAME
AS
    SELECT DISTINCT (V.NAME), V.UNIQUE_NUMBER,V.AGE, V.HEIGHT, A.NAME, A.PARK_NAME
    FROM VISITORS AS V, ATTRACTIONS AS A
    WHERE V.AGE >= A.MINIMUM_AGE
    AND V.HEIGHT >= A.MINIMUM_HEIGHT;

DROP VIEW FN71949.whereVisitorCanGetsOn;

SELECT * FROM whereVisitorCanGetsOn;
```

✓ Тестване на изгледа;

99 rows — CSV

	VISITOR_NAME	VISITOR_UNIQUE_NUMBER	VISITOR_AGE	VISITOR_HEIGHT	ATTRACTION_NAME	ATTRACTION_PARK_NAME
1	Petar Todorov	0000000001	25	170	Splash Mountain	Sofia
2	Petar Todorov	0000000001	25	170	Pirates of Caribbean	Sofia
3	Petar Todorov	0000000001	25	170	Tower of power	Sofia
4	Petar Todorov	0000000001	25	170	The Giant	Montana
5	Petar Todorov	0000000001	25	170	Peter Pan Flight	Montana
6	Petar Todorov	0000000001	25	170	Indiana Jones Adventure	Montana
7	Petar Todorov	0000000001	25	170	Dragon	Gabrovo
8	Petar Todorov	0000000001	25	170	SuperSplash	Plovdiv
9	Petar Todorov	0000000001	25	170	Silver Star	Plovdiv
10	Petar Todorov	0000000001	25	170	Jungle Snake	Montana
11	Petar Todorov	0000000001	25	170	Pink Star	Sofia
12	Petar Todorov	0000000001	25	170	Golden Star	Pleven
13	Petar Todorov	0000000001	25	170	House of illusions	Gabrovo
14	Christian Ivanov	0000000002	12	120	Splash Mountain	Sofia
15	Christian Ivanov	0000000002	12	120	The Giant	Montana
16	Christian Ivanov	0000000002	12	120	Jungle Snake	Montana
17	Christian Ivanov	0000000002	12	120	House of illusions	Gabrovo
18	Kamelia Gerasimova	0000000004	32	192	Splash Mountain	Sofia
19	Kamelia Gerasimova	0000000004	32	192	Pirates of Caribbean	Sofia
20	Kamelia Gerasimova	0000000004	32	192	Tower of power	Sofia
21	Kamelia Gerasimova	0000000004	32	192	The Giant	Montana

2. Изглед 2:

✓ supplierProducts(SUPPLIER_NAME, SUPPLIER_WORK_NUMBER, PRODUCT_NAME);

✓ Показва, който показва за всеки доставчик какъв продукт е доставил;

```
CREATE VIEW FN71949.supplierProducts(SUPPLIER_NAME, SUPPLIER_WORK_NUMBER, PRODUCT_NAME)
AS
    SELECT S.NAME, S.WORK_NUMBER, P.PRODUCT_NAME
    FROM SUPPLIERS S, PRODUCTS P
    WHERE S.WORK_NUMBER = P.SUPPLIER_WORK_NUMBER;

SELECT * FROM supplierProducts;
```

✓ Тестване на изгледа;

	SUPPLIER_NAME	SUPPLIER_WORK_NUMBER	PRODUCT_NAME
1	Stefan Petrov	0000000001	DRESS
2	Cristian Todorov	0000000003	JEANS
3	Cristian Todorov	0000000003	T-SHIRT
4	Cvetelina Kirilova	0000003004	SWIMWEAR
5	Stefan Petrov	0000000001	DRESS
6	Stefan Petrov	0000000001	CANDIES
7	Cvetelina Kirilova	0000003004	LOLLIPOP
8	Cristian Todorov	0000000003	CHOCOLATE
9	Vasil Georgiev	0001002002	CHOCOLATE
10	Vasil Georgiev	0001002002	SANDWICH
11	Stefan Petrov	0000000001	SANDWICH
12	Stefan Petrov	0000000001	MAGNET
13	Vasil Georgiev	0001002002	MAGNET
14	Cvetelina Kirilova	0000003004	KEYCHAIN
15	Silvana Ivanova	0000000004	KEYCHAIN
16	Vasil Georgiev	0001002002	CARD
17	Cristian Todorov	0000000003	CARD
18	Vasil Georgiev	0001002002	T-shirt
19	Vasil Georgiev	0001002002	T-shirt

Част 8. Описание на процедурите:

1. Процедура с курсор и входни и изходящи параметри:

- ✓ Има за цел да създаде нова таблица, където ще се пази информация за името, годините и имейла на всички посетители, които са се качили на дадена атракция.
- ✓ Името на таблицата се задава като входен параметър, заедно с името на атракцията;
- ✓ Връща като резултат дали успешно е създадена таблицата и броя на добавените редове в нея;

```
CREATE PROCEDURE FN71949.createAttractionInfo(IN p_attractionName VARCHAR(100), IN p_tableName VARCHAR(50),
                                              OUT p_isSuccessfullAdded VARCHAR(150), OUT p_insertedRows VARCHAR(150))
LANGUAGE SQL
SPECIFIC createAttractionInfo
BEGIN
    DECLARE v_countVisitors INTEGER DEFAULT 0;
    DECLARE v_visitorName VARCHAR(100) DEFAULT ' ';
    DECLARE v_visitorAge INTEGER DEFAULT 0;
    DECLARE v_visitorEmail VARCHAR(100) DEFAULT ' ';
    DECLARE at_End INTEGER DEFAULT 0;
    DECLARE not_found CONDITION FOR SQLSTATE '02000';
    DECLARE firstCursor CURSOR FOR
        SELECT V.NAME, V.AGE, V.EMAIL
               FROM VISITORS AS V, GETS_ON AS G, ATTRACTIONS AS A
    WHERE p_attractionName = G.ATTRACTION_NAME
    AND V.UNIQUE_NUMBER = G.VISITORS_UNIQUE_NUMBER
    AND A.NAME = G.ATTRACTION_NAME;
    DECLARE CONTINUE HANDLER FOR not_found
        SET at_end = 1;
    CALL FN71949.createTableByAttractionName(p_tableName);
OPEN firstCursor;
    firstLoop: LOOP
        FETCH firstCursor INTO v_visitorName, v_visitorAge, v_visitorEmail;
        IF at_end = 1 THEN
            LEAVE firstLoop;
        end if;
        CALL FN71949.insertIntoAttractionTable(p_tableName, v_visitorName, v_visitorAge, v_visitorEmail);
    SET v_countVisitors = v_countVisitors + 1;
    end loop;
SET p_isSuccessfullAdded = 'Table with name ' || p_tableName || 'was successful created.';
SET p_insertedRows = v_countVisitors || ' rows were inserted.';
end@

DROP PROCEDURE FN71949.createAttractionInfo@

CALL FN71949.createAttractionInfo('Tower of power', 'INFO_ABOUT_TOWER_OF_POWER', ?, ?)@

SELECT * FROM INFO_ABOUT_TOWER_OF_POWER@
```

- ✓ Тестване на процедурата за атракцията Tower of power;

	NAME	AGE	EMAIL
1	Petar Todorov	25	peter.todorov@gmail.com
2	Kaloian Georgiev	4	kaloian.g@gmail.com
3	Nikola Kolev	27	nikola.kolev@gmail.com

2. Процедура с прихващане на изключения:

- ✓ Има за цел да изведе информация за номер на служител, име на служител и магазин, в който работи служителя;
  - ✓ Ако се появи грешка, то тя да бъде прихваната с UNDO handler-a;
  - ✓ Тоест, ако е намерена null стойност, то да се прекрати обхождането на редовете и да се изведе резултата до тук;

```
CREATE TABLE FN71949.EMPLOYEE_WORKNUMBER_CHANGE(CTIME TIMESTAMP, MESSAGE VARCHAR(5000))@
DROP TABLE FN71949.EMPLOYEE_WORKNUMBER_CHANGE@

CREATE PROCEDURE FN71949.EMPLOYEES_ITERATE()
LANGUAGE SQL
BEGIN ATOMIC
    DECLARE nullValue INTEGER DEFAULT 0;
    DECLARE outOfRange INTEGER DEFAULT 0;

    DECLARE employeeWorkNumber CHAR(10) DEFAULT ' ';
    DECLARE employeeName VARCHAR(100) DEFAULT ' ';
    DECLARE employeeShopName VARCHAR(100) DEFAULT ' ';

    DECLARE nullNotAllowed CONDITION FOR SQLSTATE '22004';
    DECLARE outRange CONDITION FOR SQLSTATE '02000';

    --Курсор, с който щеобходим редовете на таблицата EMPLOYEE
    DECLARE firstCursor CURSOR FOR SELECT WORK_NUMBER, EMPLOYEE_NAME, SHOP_NAME FROM FN71949.EMPLOYEES;

    --Декларирам типовете condition handlers
    DECLARE UNDO HANDLER FOR nullNotAllowed SET nullValue = 1;
    DECLARE CONTINUE HANDLER FOR outRange SET outOfRange = 1;

    OPEN firstCursor;
    firstLoop: LOOP
        FETCH firstCursor INTO employeeWorkNumber, employeeName, employeeShopName;
        IF nullValue = 1 OR outOfRange = 1 THEN LEAVE firstLoop;
        ELSEIF employeeWorkNumber = '0000000001' THEN ITERATE firstLoop;
        END IF;

        INSERT INTO FN71949.EMPLOYEE_WORKNUMBER_CHANGE(CTIME, MESSAGE)
        VALUES(CURRENT_TIMESTAMP, 'WNumber:' || employeeWorkNumber || ' EName:' || employeeName || ' Shop:' || employeeShopName);
    END LOOP;
    CLOSE firstCursor;
END@

DROP PROCEDURE FN71949.EMPLOYEES_ITERATE()@

SELECT * FROM FN71949.EMPLOYEE_WORKNUMBER_CHANGE@
CALL FN71949.EMPLOYEES_ITERATE()@
```

✓ Резултат от изпълнението на процедурата

	CTIME	MESSAGE
1	2022-01-15 18:23:58.706095	WNumber:0000000002 EName:Ivan Petrov Shop:Sofia outskirts
2	2022-01-15 18:23:58.706331	WNumber:0000000003 EName:Cvetelina Stoyanova Shop:Gabrovo ce
3	2022-01-15 18:23:58.706388	WNumber:0000000004 EName:Christina Petrova Shop:Montana cent
4	2022-01-15 18:23:58.706473	WNumber:0000000005 EName:Yana Ivanova Shop:Pleven
5	2022-01-15 18:23:58.70653	WNumber:0000000006 EName:Iliana Georgieva Shop:Plovdiv
6	2022-01-15 18:23:58.706612	WNumber:0000000007 EName:Ilian Ivanov Shop:Plovdiv
7	2022-01-15 18:23:58.706668	WNumber:0000000008 EName:Kiril Kolev Shop:Gabrovo
8	2022-01-15 18:23:58.706754	WNumber:0000000009 EName:Borislava Todorova Shop:Montana
9	2022-01-15 18:23:58.706837	WNumber:0000000010 EName:Dimitar Dimitrov Shop:Sofia

3. Процедура с курсор и while цикъл:

✓ Целта на процедурата е по въведено име на продукт да върне служебния номер на доставчика;

```
CREATE TABLE PRODUCT_SUPPL(
    PRODUCT_NAME VARCHAR(100),
    SUPPL_WORK_NUMBER CHAR(10)
)@

CREATE PROCEDURE FN71949.productSupplier(IN productName VARCHAR(100), OUT supplierWorkNumber CHAR(10))
LANGUAGE SQL
BEGIN
    DECLARE atEnd INTEGER DEFAULT 0;
    DECLARE v_product_name VARCHAR(100);
    DECLARE v_supplier_work_number CHAR(10);

    DECLARE firstCursor CURSOR FOR SELECT SUPPLIER_WORK_NUMBER, PRODUCT_NAME FROM PRODUCTS;

    OPEN firstCursor;

    WHILE atEnd = 0 DO
        FETCH firstCursor INTO v_supplier_work_number, v_product_name;
        IF productName = v_product_name THEN SET atEnd = 1;
        END IF;
    END WHILE;

    SET supplierWorkNumber = v_supplier_work_number;
    INSERT INTO PRODUCT_SUPPL(PRODUCT_NAME, SUPPL_WORK_NUMBER) VALUES (v_product_name, v_supplier_work_number);

    CLOSE firstCursor;
END@
```

✓ Тестване на процедурата;

```
CALL FN71949.productSupplier('DRESS', ?)@
CALL FN71949.productSupplier('MAGNET', ?)@
CALL FN71949.productSupplier('JEANS', ?)@
SELECT * FROM PRODUCT_SUPPL@
```

✓ Резултат от извикването на процедурата;

	PRODUCT_NAME	SUPPL_WORK_NUMBER
1	DRESS	0000000001
2	MAGNET	0000000001
3	JEANS	0000000003

Част 9. Съдържание на проекта:

- ✓ createTables.sql – скрипт за създаване на таблици и ограниченията към тях;
- ✓ dataLoad.sql - скрипт за попълване на таблиците: insert, update, delete;
- ✓ createFunc.sql - скрипт, в който са дефинирани функции и тестове с тях;
- ✓ createTrig.sql - скрипт, в който са дефинирани тригери и тестове с тях;
- ✓ createView.sql - скрипт създаващ изгледи и тестове с тях;
- ✓ createProc.sql - скрипт, в които са дефинирани процедури и тестове с тях;