Introduction to database systems

General informations

General information

Written exams : No midterm exams

Homeworks (40% of final grade)

- 3 homeworks.
- Homework must be completed within the prescribed period (~ 10 days).
- Each day of overdue time brings a 3% lower grade from homework.
- The homework average must be positive (> 50%).
- Last year's students do not need to this part again if they passed it last year.

1. homework (end of October – beginning of November)

Changing a group is only possible if you find a suitable replacement.

Contact hours: By agreement in advance

Contact: uros.sergas@upr.si

Topics to be covered in tutorials

1. Relational model and SQL
2. Relational algebra
3. Relational calculus
4. QBE
5. Indexing (Tree and Hash)
6. Evaluation and Optimisation
7. Transactions and concurrency control
8. Logical design and normalisation
9. Entity-relational models
10. (Recap before the exam)

Introduction to database systems

Relational model and SQL

Relational data model - terminology

- **Relation** – two-dimensonal table with columns and rows

Name	Gender	Age
Mark	Male	22
Ivo	Male	18
Tadeja	Female	21
Meta	Female	18

relation

Relational data model - terminology

- **Atrribute** – named column of the relation

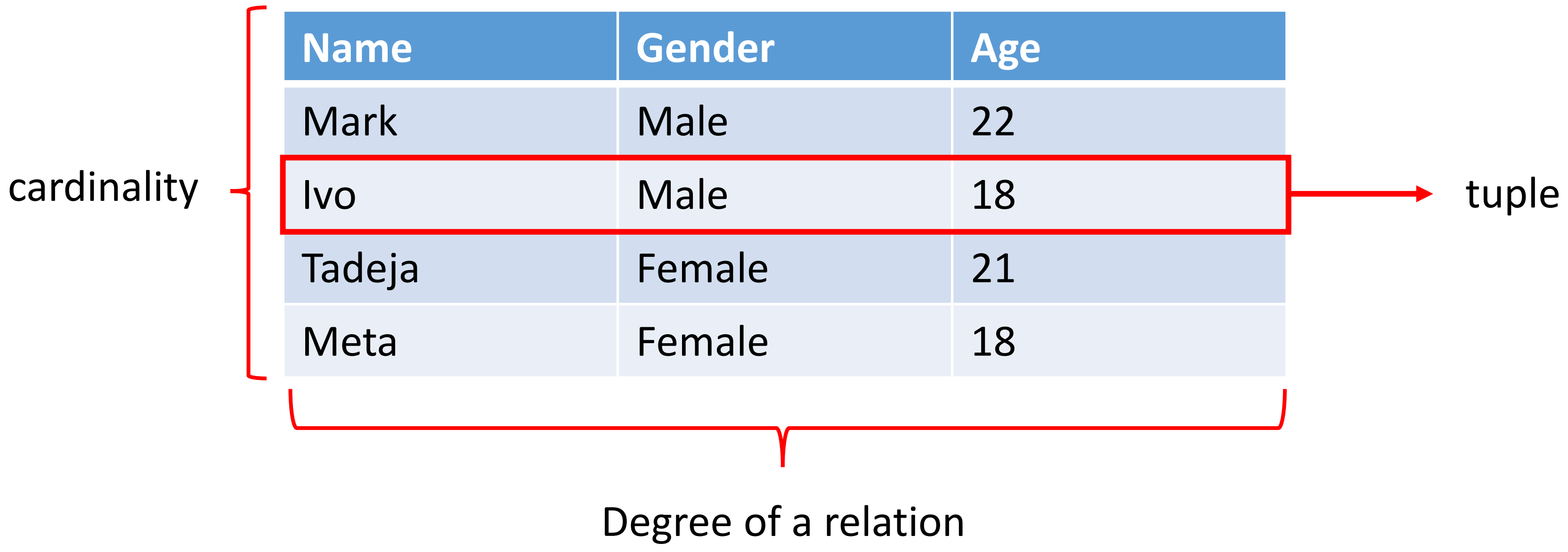
Name	Gender	Age
Mark	Male	22
Ivo	Male	18
Tadeja	Female	21
Meta	Female	18

attribute

- **Domain** – a set of „allowed“ values of one or more attributes
  - Examples: colour: {yellow, red, green, blue}
    date of birth: date

Relational data model - terminology

- **Tuple** – one line in the relation.
- **Cardinality** – number of tuples in a relation.
- **Degree or arity** – number of attributes in a relation.



Name	Gender	Age
Mark	Male	22
Ivo	Male	18
Tadeja	Female	21
Meta	Female	18

Standard SQL types in PostgreSQL database

- PostgreSQL supports the following types (and some others):
  - Int (-2 147 483 648 do 2 147 483 647)
  - Smallint (-32 768 do 32 767)
  - real
  - char(N)
  - varchar(N)
  - date
  - time
  - Timestamp

  …

  More: https://www.postgresql.org/docs/9.5/datatype.html

Groups of SQL commands

- **DDL (Data Definition Language)**
  - Table creation - CREATE (TABLE, USER, VIEW,...)
  - Table alteration (ALTER TABLE)
  - Table deletion (DROP TABLE)
- **DML (Data Manipulation Language)**
  - **Record selection (SELECT)**
  - Record insertion (INSERT)
  - Record alteration (UPDATE)
  - Record deletion (DELETE)

Groups of SQL commands

- **DCL (Data Control Language)**
  - (GRANT)
  - (REVOKE)
- **TPO (Transaction Processing Option)**
  - (COMMIT)
  - (ROLLBACK)
- **Keys**
  - PRIMARY KEY
    - UNIQUE and NOT NULL
  - FOREIGN KEY
    - REFERENCES

DML group - selections

- **SELECT** for inquiry

```
SELECT  [distinct] A1, A2, ..., Ak
FROM      T1, T2, ..., Tn
WHERE     P;
```

select attributes (A_1, A_2, ..., A_k)

from tables (T_1, T_2, ..., T_n)

condition P

DML group - selections

- **SELECT** for inquiry

```
SELECT  [distinct] A1, A2, ..., Ak
FROM     T1, T2, ..., Tn
WHERE    P;
```

select attributes ($A_1, A_2, \ldots, A_k$)

from tables ($T_1, T_2, \ldots, T_n$)

condition P

Selects all the customers from the country "Mexico"

```
SELECT *
FROM Customers
WHERE Country='Mexico';
```

DML group - selections

- **SELECT** for inquiry

```
SELECT   [distinct] A1, A2, ..., Ak
FROM     T1, T2, ..., Tn
WHERE    P
GROUP BY attributes
HAVING   group conditions
ORDER BY attributes, expressions [asc|desc]
LIMIT    number of lines;
```

For selecting all attributes, use *

Number of lines in the output

DML group - selections

- **SELECT** for inquiry

Selects all the customers from the country "Mexico"

```
SELECT   *
FROM     Customers
WHERE    Country = 'Mexico‘;
```

DML group - selections

- **SELECT** for inquiry

Lists the number of customers in each Mexican city

```
SELECT   COUNT(CustomerID), City
FROM     Customers
WHERE    Country = 'Mexico'
GROUP BY City;
```

DML group - selections

- **SELECT** for inquiry

Lists the number of customers in each Mexican city.

```
SELECT   COUNT(CustomerID), City
FROM     Customers
WHERE    Country = 'Mexico'
GROUP BY City
HAVING COUNT(CustomerID) > 5
ORDER BY City DESC;
```

Include only cities with more than 5 customers

Return a sorted descending by the City column

DML group - selections

- **SELECT** for inquiry

Lists the number of customers in each Mexican city.

```
SELECT   COUNT(CustomerID), City
FROM     Customers
WHERE    Country = 'Mexico'
GROUP BY City
HAVING COUNT(CustomerID) > 5
ORDER BY City DESC
LIMIT 5;
```

Include only with more than 5 customers

Return a sorted descending by the City column

Display only top 5

Operators for comparison

=	equals
!= or <>	differs
> and >=	it is greater or it is greater or equal
< and <=	it is lower or it is lower or equal
IN (...)	corresponds to any element of the set
BETWEEN ... AND ...	is among the given values
LIKE ‘...%...‘	corresponds to a pattern of a string of characters
IS NULL	is an unknown value
NOT ...	operator negation

Operators for comparison

IN (...)

list details of all customers that live in one of the following countries: Mexico, Italy, Spain

```
SELECT    *
FROM      Customers
WHERE     Country IN ('Mexico', 'Italy', 'Spain');
```

BETWEEN ... AND ...

list details of all customers aged between 20 – 30

```
SELECT    *
FROM      Customers
WHERE     age BETWEEN 20 AND 30;
```

LIKE '...%...'

list details of all customers whose name begins with K

```
SELECT    *
FROM      Customers
WHERE     name LIKE 'K%';
```

Introduction to database systems

Work environment setup

Work environment setup

Go to postgreSQL: http://www.student.famnit.upr.si/phppgadmin/

Login as:

user:**OPBvaje**
pass:**OPBvaje**

Click on “PostgreSQL” → on the top right click “SQL”

Execute each of these two lines separately.

- CREATE USER "user_name" WITH PASSWORD 'set_password' CREATEDB SUPERUSER;
- CREATE DATABASE "name_database" WITH OWNER "user_name" TEMPLATE opb_19_20;
- Logout and login with a newly created username

Create, Insert, Select

```
CREATE TABLE Cities (
   PostID int,
   Name varchar(255),
   Region varchar(255),
   Population int);
```

We create a table called 'Cities' with atributes CityID, Name, Region in Population

```
INSERT INTO Cities (PostID, Name, Region, Population)
VALUES (6000, 'Koper', 'Primorska', 30000);
```

We input the data about Koper

```
SELECT *
FROM Cities;
```

We return the whole table of 'Cities'

Update, Alter

UPDATE Cities
SET Population = 25000
WHERE PostID = 6000;
	We update the value of population

ALTER TABLE Cities **ADD PRIMARY KEY (PostID);**	We define 'PostID' as the primary key in the table 'Cities'

Integrity constraint – Domain, Entity, Key

INSERT INTO Cities (PostID, Name, Region, Population) VALUES ('Thousand', 'Ljubljana', 'Osrednjeslovenska', 300000);	**We get an error because we inputed a string value instead of an integer** **(Domain constraint)**
INSERT INTO Cities (Name, Region, Population) VALUES ('Ljubljana', 'Osrednjeslovenska', 300000);	We get an error, because we left the ID empty ('PostID' = Null) (Entitety constraint)
INSERT INTO Cities (PostID, Name, Region, Population) VALUES (6000, 'Ljubljana', 'Osrednjeslovenska', 300000);	Duplicate key error (Key constraint)

Integrity constraint – Reference

CREATE TABLE Districts (**DistrictID int NOT NULL,** **PostID int,** **PRIMARY KEY (DistrictID),** **FOREIGN KEY (PostID) REFERENCES Cities(PostID));**	**We create a new table 'Districts' with primary key 'DistrictID' foreign key 'PostID' which is referenced from table 'Cities'**

INSERT INTO Districts (DistrictID, PostID)
VALUES (8, 6000);
	Insert into table 'Districts'

INSERT INTO Districts (DistrictID, PostID)
VALUES (69, 2000);
	This insert does not work because table 'Cities' does not include 'PostID'=2000 which would be referenced

Integrity constraint – Unique

ALTER TABLE Cities ADD UNIQUE (name);	**We add a new constraint on the atribute 'name'. Values have to be unique on each insert**

**INSERT INTO Cities (PostID, Name, Region, Population)
VALUES (1000, 'Koper', 'Osrednjeslovenska', 300000);**
	Does not work because the name 'Koper' is already taken

Primary / Foreign Key

**INSERT INTO Cities (PostID, Name, Region, Population)
VALUES (1000, 'Ljubljana', 'Osrednjeslovenska', 300000);**
	We add an additional row to 'Cities'

**INSERT INTO Districts (DistrictID, PostID)
VALUES (12, 1000), (96,1000), (240, 1000);**
	And one into 'Districts'

**SELECT *
FROM Cities C, Districts D
WHERE C.PostID=D.PostID;**
	We select all atributes from both tables where PostID is matched on both tables

**SELECT C.postid, districtid, name, region
FROM Cities C, Districts D
WHERE C.PostID = D.PostID AND region='Primorska';**
	We select 'PostID', 'DistrictID', 'name' and 'region' where the latter equals 'Primorska'

Delete

DELETE
FROM Cities
WHERE PostID = 6000;
	Does not work because 'PostID' is referenced in table 'DistrictID' as foreign key

DELETE
FROM Districts
WHERE DistrictID = 12;
	We delete a row where DistrictID equals 12

Drop / Cascade

DROP TABLE Cities;	**Does not work because of the reference to the other table. We must use CASCADE**
DROP TABLE Cities CASCADE;	Now the table is deleted. Rows in 'Districts' that were refered from 'Cities' are also deleted.
DROP TABLE Districts;	Now we delete 'Districts' aswell.