# Автоматизация работы деканата.

В рамках данного цикла лабораторных работ необходимо автоматизировать работу деканата. Для этого в рамках базы данных PostgreSQL необходимо создать объекты в схеме вашего пользователя (логин и пароль пользователя для доступа к базе данных студент должен получить у преподавателя) и написать клиентское приложение на базе компонентов ADO.NET или JDBC.

Под автоматизацией работы деканата понимается ведение статистики получения оценок студентами без анализа временных ограничений на сдачу и необходимых для сдачи предметов. Студенты и преподаватели характеризуются фамилией, именем и отчеством. Предмет – названием. Под оценкой понимается целое число в диапазоне [2..5]. При получении студентом оценки должен учитываться предмет и преподаватель, поставивший оценку. Студенты разделяются по группам. Студент принадлежит одной и только одной группе. Группа характеризуется наименованием. Наименование группы формируется как курс, направление, номер подгруппы, «\_» и год. Например, 3084/1\_2004. Целью разрабатываемой системы автоматизации является организация и разработка средств контроля над успеваемостью студентов. В разрабатываемой системе следует предусмотреть возможность анализа успеваемости, как по группам, так и по предметам, преподавателям и студентам. При построении динамики изменения успеваемостей, результаты следует усреднять в рамках года обучения, вычисляя его из наименований рассматриваемых групп.

**База данных должна удовлетворять следующим требованиям:**

1. Контроль целостности данных, используя механизм связей
2. Операции модификации групп данных и данных в связанных таблицах должны быть выполнены в рамках транзакций.
3. Логика работы приложения должна контролироваться триггерами. В частности:
   * Триггер должен запрещать установку оценки вне диапазона [2..5]
   * Триггер должен контролировать, чтобы вместо студента не оказался преподаватель и наоборот (см. схему данных).
4. Все операции вычисления различных показателей (из требований к клиентскому приложению) должны реализовываться хранимыми процедурами.

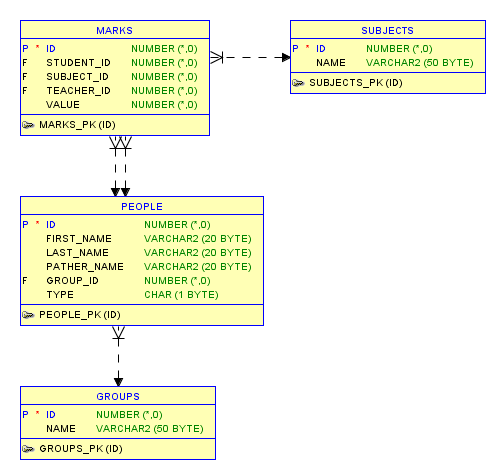
**Требования к клиентскому приложению:**

1. Необходимо реализовать интерфейсы для ввода, модификации и удаления

* Групп, студентов и преподавателей
* Предметов и оценок

1. Должен быть реализован диалог (или группа диалогов) анализа, предоставляющие визуальные средства для расчета среднего балла в зависимости от интервала дат. В качестве фильтров сущностей могут выступать года, студенты, группы, предметы и преподаватели.
2. Результаты анализа необходимо предоставлять в табличной форме и в графическом виде согласно выбору пользователя.

## Работа №1. Создание базы данных.



|  |  |  |
| --- | --- | --- |
| **Имя таблицы** | **Имя колонки** | **Расшифровка** |
| groups |  | Таблица групп |
|  | id | Идентификатор группы |
|  | name | Наименование группы |
| people |  | Таблица студентов и преп. |
|  | id | Идентификатор человека |
|  | first\_name, last\_name, pather\_name | ФИО |
|  | group\_id | Идент. группы (у преп. «\_ » ) |
|  | type | Тип («S» - студ, «P» - преп.) |
| marks |  | Таблица оценок |
|  | id | Идентификатор оценки |
|  | student\_id | Идентификатор студента |
|  | subject\_id | Идентификатор предмета |
|  | teacher\_id | Идентификатор преподавателя |
|  | value | Значение оценки |
| subjects |  | Таблица предметов |
|  | id | Идентификатор предмета |
|  | name | Наименование предмета. |

1. Запустить pgAdmin или psql.
2. Создать соединение, используя логин и пароль.
3. Изменить пароль вашего пользователя на свой, используя команду

ALTER USER *your*\_*user\_name* IDENTIFIED BY *new\_password*;

1. Создать необходимые таблицы.
2. Создать связи между таблицами:

|  |  |  |
| --- | --- | --- |
| **Название** | **Primary Key** | **Foreign Key** |
| fk\_people\_groups | groups.id | people.group\_id |
| fk\_marks\_subjects | subjects.id | marks.subject\_id |
| fk\_marks\_people1 | people.id | marks.student\_id |
| fk\_marks\_people2 | people.id | marks.teacher\_id |

1. Создать Backup базы и запомнить место его расположения.
2. Удалить базу с сервера
3. Восстановить базу из Backup базы
4. Для всех таблиц реализовать автоматическое заполнение первичного ключа при вставке данных (автоинкремент).

### Правила выполнения работ

1. Засчитываются только целиком сделанные работы
2. Преподаватель оставляет за собой право на выдачу дополнительных заданий, в случае возникновения подозрения на стороннюю помощь при выполнении работы.
3. Посещения занятий ОБЯЗАТЕЛЬНО, кроме случаев оговоренных лично с преподавателем.
4. Если студент сдает работу, то он должен знать, как он ее сделал.
5. Все работы можно делать дома, но это не является основанием для пропусков занятий.
6. Перед уходом студент должен удалить с сервера свою базу. Рекомендую не забывать делать Backup в осознанное место, что бы на следующем занятии не пришлось делать работу с начала.
7. К каждому заданию прилагается список операторов, ключевых.

## Работа №2. Язык SQL-DML

### Выборка данных

* однотабличная выборка
  1. Вывести все группы, упорядочив их в обратном порядке по году и в прямом порядке по наименованию.

SELECT \* FROM groups ORDER BY year DESC, name ASC;

* выборка с подзапросами
  1. Вывести число предметов, по которым есть оценки у группы, заданной по наименованию.

SELECT COUNT(DISTINCT subject\_id) as count FROM marks

JOIN people ON marks.student\_id = people.id

JOIN groups ON people.group\_id = groups.id

WHERE groups.name = '5130/1\_2023';

* соединение таблиц (join)
  1. Вывести студентов и наименования предметов, включая предметы, которые не читались студентам и студентов, которые не получили ни одной оценки.

SELECT people.first\_name, people.second\_name, people.pather\_name, subjects.name, marks.value as mark

FROM people

CROSS JOIN subjects

LEFT JOIN marks ON people.id = marks.student\_id AND subjects.id = marks.subject\_id

where people.type = ‘S’ AND marks.value IS NOT NULL

order by people.second\_name ASC;

* 1. Вывести все группы и средний балл студентов каждой.

SELECT groups.name, AVG(value) as average\_mark FROM marks

JOIN people ON marks.student\_id = people.id

JOIN groups ON people.group\_id = groups.id

group by groups.name

order by name

Hints: select, count, join, where, in, exists, order by, group by, having

### Вставка данных

x

Hints: insert, where, in, exists, commit, rollback

### Удаление данных

* удаление по фильтру и удаление из связанных таблиц
  1. Удалить студентов, у которых средний бал ниже заданного

Delete from marks

where student\_id IN(

Select student\_id

from marks

group by student\_id

Having AVG(value) < 4

);

* 1. Удалить заданную группу и студентов, принадлежащих ей

Delete from marks

where student\_id IN(

Select id

from people

where group\_id = (

Select id

from groups

where name = 8060/1\_2022'

)

);

Delete from people

where group\_id = (

Select id

from groups

where name = 8060/1\_2022'

);

Delete from groups where name =’8060/1\_2022';

* удаление в рамках транзакции
  1. Удалить в рамках транзакции группу с самым маленьким средним балом и студентов, принадлежащих ей
  2. то же, что и п1, но, если в удаленной группе читались 3 предмета, которые больше нигде не читались – транзакцию откатить

DO $$

DECLARE

min\_groupID INT;

unique\_subjects\_count INT;

BEGIN

min\_groupID := (Select group\_id

FROM marks

JOIN people ON marks.student\_id = people.id

GROUP BY group\_id

HAVING AVG(value) = (

SELECT MIN(avg\_value)

FROM (

SELECT group\_id , AVG(value) AS avg\_value

FROM marks

JOIN people ON marks.student\_id = people.id

GROUP BY group\_id

) AS subquery

) Limit 1);

unique\_subjects\_count := (SELECT COUNT(DISTINCT subject\_id) FROM marks

join people on marks.student\_id = people.id

WHERE group\_id = min\_groupID AND subject\_id NOT IN (

SELECT DISTINCT subject\_id

FROM marks

join people on marks.student\_id = people.id

WHERE group\_id <> min\_groupID));

IF (unique\_subjects\_count = 3) THEN

ROLLBACK;

RETURN;

END IF;

Delete from marks where student\_id IN (

Select id from people where group\_id = min\_groupID

);

Delete from people where group\_id = min\_groupID;

Delete from groups where id = min\_groupID;

COMMIT;

END $$;

Hints: delete, where, in, exists, commit, rollback

### Модификация данных

* модификация по фильтру
  1. Подменить заданную группу на другую.

UPDATE groups SET name = '9999/1\_2022' WHERE name = '3690/1\_2022';

* модификация в рамках транзакции
  1. Заменить во всех существующих записях заданный предмет на другой и у удалить заданный.
  2. то же, что и п1, но в случае, если преподаватель, читающий удаляемый предмет больше ничего не читает – откатить транзакцию.

DO $$

DECLARE

teacher\_id\_uniq INT;

id\_check INT;

subject TEXT := 'GNU/Linux';

BEGIN

teacher\_id\_uniq := (SELECT DISTINCT teacher\_id FROM marks WHERE subject\_id = (SELECT id FROM subjects WHERE name = subject));

id\_check := (SELECT count(distinct subject\_id) from marks where teacher\_id = teacher\_id\_uniq);

IF (id\_check < 2) THEN

ROLLBACK;

RETURN;

END IF;

DELETE FROM marks WHERE subject\_id = (SELECT id FROM subjects WHERE name = subject);

UPDATE subjects SET name = 'Linux OS' WHERE name = subject;

COMMIT;

END $$;

Hints: update, where, in, exists, commit, rollback

### Дополнительные задания

## Работа №3. Представления, хр. процедуры, триггера и курсоры.

### Представления

1. Создать представление, отображающее всех преподавателей, предметы и средний бал, выставляемый ими, агрегируя по предмета

CREATE VIEW teacher\_subject\_avg AS

SELECT p.second\_name || ' ' || p.first\_name || ' ' || p.pather\_name AS teacher\_name,

s.name AS subject\_name,

AVG(m.value) AS avg\_mark

FROM marks m

JOIN people p ON m.teacher\_id = p.id

JOIN subjects s ON m.subject\_id = s.id

where p.type = 'P'

GROUP BY p.id, s.id;

Select \* from teacher\_subject\_avg;

DROP VIEW teacher\_subject\_avg;

1. Создать представление, отображающее средние баллы, агрегируя оценки по годам

CREATE VIEW year\_avg AS

SELECT g.year AS year,

AVG(m.value) AS avg\_mark

FROM marks m

join people p on m.student\_id = p.id

join groups g on p.group\_id = g.id

GROUP BY year;

Select \* from year\_avg;

drop view year\_avg;

Hints: select, where, count, max, group by, having, like, create view, drop view

### Хранимые процедуры

* без параметров

1. Создать хранимую процедуру, выводящую все предметы, по которым в заданном году были выставлены оценки, и средний балл по каждому из предметов.

CREATE or replace FUNCTION subject\_avg() returns table(subject character varying, avg\_mark numeric)

Language plpgsql AS

$$

BEGIN

return query

SELECT s.name AS subject, AVG(m.value) AS avg\_mark

FROM marks m

JOIN subjects s ON m.subject\_id = s.id

join people p on m.student\_id = p.id

join groups g on p.group\_id = g.id

WHERE g.year = 2023

GROUP BY s.name;

END

$$;

select \* from subject\_avg();

drop function subject\_avg;

* с входными параметрами

1. Создать хранимую процедуру. Входные параметры задают интервал времени. Процедура должна вернуть выборку средних баллов по группам, попавшим в этот интервал.

CREATE or replace function group\_avg(In start\_date int, In end\_date int) returns table(group\_name character varying, avg\_mark numeric)

Language plpgsql AS

$$

BEGIN

return query

SELECT g.name AS group\_name, AVG(m.value) AS avg\_mark

FROM marks m

JOIN people p ON m.student\_id = p.id

JOIN groups g ON p.group\_id = g.id

WHERE g.year BETWEEN start\_date AND end\_date

GROUP BY g.name;

END

$$;

select \* from group\_avg(2022, 2023);

drop function group\_avg;

* с выходными параметрами

1. Создать хранимую процедуру с входным параметром «преподаватель» и выходным параметром – группа, с наименьшим средним баллом у этого преподавателя.

CREATE OR REPLACE PROCEDURE group\_min\_avg\_mark\_by\_teacher(IN teacherID integer, OUT group\_name character varying)

Language plpgsql

As

$$

BEGIN

SELECT g.name INTO group\_name

FROM groups g

JOIN people p ON g.id = p.group\_id

JOIN marks m ON p.id = m.student\_id

WHERE m.teacher\_id = teacherID

GROUP BY g.id

ORDER BY AVG(m.value)

LIMIT 1;

END

$$;

DO

$$

DECLARE group\_name character varying;

BEGIN

Call group\_min\_avg\_mark\_by\_teacher(1, group\_name);

--Raise Notice '%', group\_name;

END

$$;

drop procedure group\_min\_avg\_mark\_by\_teacher;

Hints: select, where, count, max, group by, having, create procedure, drop procedure

### Триггера

* Триггера на вставку

1. Создать триггер, который не позволяет добавить оценку, не попадающую в интервал [2..5].

CREATE OR REPLACE FUNCTION check\_mark\_range\_func() returns trigger

Language plpgsql

as $$

begin

IF NEW.value NOT BETWEEN 2 AND 5 THEN

RAISE SQLSTATE '45000'

USING MESSAGE = 'Оценка должна быть в интервале [2..5]';

END IF;

return new;

end;

$$;

create trigger check\_mark\_range

before insert on marks

for each row

execute procedure check\_mark\_range\_func();

insert into marks values(default, 33, 5, 4, 55);

drop function check\_mark\_range\_func;

drop trigger check\_mark\_range on marks;

* Триггера на модификацию

1. Создать триггер, который не позволяет изменить наименование предмета, если на него есть ссылки

CREATE OR REPLACE FUNCTION check\_subject\_refs\_func() returns trigger

Language plpgsql

as $$

declare

ref\_count integer;

begin

select count(\*) into ref\_count from marks where subject\_id = old.id;

if ref\_count > 0 and new.name <> old.name then

raise sqlstate '45000'

using message = 'Нельзя изменить наименование предмета, на который есть ссылки';

end if;

return new;

end;

$$;

create trigger check\_subject\_references

before update on subjects

for each row

execute procedure check\_subject\_refs\_func();

update subjects SET name = 'Math analysis' WHERE name = 'Math';

* Триггера на удаление

1. Создать триггер, который при удалении предмета, если на него существуют ссылки – откатывает транзакцию.

CREATE OR REPLACE FUNCTION check\_subject\_refs\_count\_func() returns trigger

Language plpgsql

as $$

declare

ref\_count integer;

begin

SELECT COUNT(\*) INTO ref\_count FROM marks WHERE subject\_id = OLD.id;

IF ref\_count > 0 THEN

ROLLBACK;

END IF;

return new;

end;

$$;

create trigger check\_subject\_references\_on\_delete

before delete on subjects

for each row

execute procedure check\_subject\_refs\_count\_func();

delete from subjects where name = 'Math';

Hints: select, where, in, exists, join, commit, rollback, create trigger, drop trigger

### Курсоры

* Хранимая процедура для расчета успеваемости и прироста успеваемости

Хранимая процедура имеет два параметра определяющие анализируемый интервал времени. Результатом работы процедуры должна явится выборка, содержащая средний бал по всем предметам в рассматриваемом интервале времени, и разницу текущего среднего балла и предыдущего.

Алгоритм реализации предлагается следующий. Организуется курсор, перебирающий все года, в которых проводилось обучение, попадающие в заданный интервал времени. Средний бал предыдущего года запоминается в переменной. В теле курсора формируется выборка необходимых данных.

CREATE OR REPLACE FUNCTION get\_avg\_marks(IN start\_date integer, IN end\_date integer) returns TABLE(year integer, average\_mark numeric, diff numeric)

Language plpgsql AS

$$

DECLARE

prev\_avg numeric;

curr\_avg numeric;

curr\_year integer;

prev\_year integer;

diff numeric;

done boolean DEFAULT FALSE;

cur CURSOR FOR

SELECT DISTINCT groups.year FROM marks

join people on marks.student\_id = people.id

join groups on people.group\_id = groups.id

WHERE groups.year BETWEEN start\_date AND end\_date;

BEGIN

drop table if exists results;

create temporary table results(year integer, average\_mark numeric, diff numeric);

OPEN cur;

FETCH cur INTO curr\_year;

prev\_year := curr\_year - 1;

WHILE NOT done LOOP

curr\_avg := (SELECT AVG(value) FROM marks

JOIN people ON marks.student\_id = people.id

JOIN groups ON people.group\_id = groups.id

WHERE groups.year = curr\_year);

prev\_avg := (SELECT AVG(value) FROM marks

JOIN people ON marks.student\_id = people.id

JOIN groups ON people.group\_id = groups.id

WHERE groups.year = prev\_year);

diff := curr\_avg - prev\_avg;

prev\_year := curr\_year;

insert into results values(prev\_year, curr\_avg, diff);

FETCH cur INTO curr\_year;

IF NOT FOUND THEN

done := true;

END IF;

END LOOP;

CLOSE cur;

return query

Select \* from results;

END

$$;

select \* from get\_avg\_marks(2022, 2023);

drop function get\_avg\_marks;

Hints: CURSOR, %NOTFOUND, FETCH