Отчет по лабораторной работе №3.

Вариант 0900909889899. Раевский Григорий, Р3121.

1. Текст задания

В редких и трудно предвидимых случаях Шут буквально вверх дном переворачивал город какой-нибудь своей проделкой, которая могла быть не более чем тонко задуманной дурацкой шуткой или же рассчитанным выпадом против популярного в данный момент убеждения, а то и всего образа жизни. Принимая все это во внимание, можно было утверждать, что титул <шут> оказался в высшей степени удачным. В свое время, еще когда существовали короли и их дворы, шуты решали именно такие задачи и преследовали те же цели.

1. Функциональные зависимоти
2. Таблица cities:

id -> city, country\_id

1. Таблица countries:

id -> name, average\_height, average\_weight

1. Таблица human:  
    id -> gender, date\_of\_birth, weight, height, originlocationid, name, surname
2. Таблица royal\_court:

court\_id -> king\_or\_queen, royal\_court\_name, locationid, king\_id

1. Таблица skills:

humanid, skill -> (составной ключ)

1. Таблица statuses:

status\_id -> human\_id, social\_status, date\_of\_start, date\_of\_end

1. Таблица subordinate\_connection:  
    subordinate\_id, superior\_id -> (составной ключ)
2. Таблица court\_members:

court\_id, member\_id -> (составной ключ)

1. Проверка на принадлежность к 3NF:
2. Все таблицы имеют первичные ключи. Их атрибуты так же являются атомарными, значит все таблицы находятся в 1NF.
3. Проверим на принадлежность к 2NF:
   1. cities: все атрибуты зависят только от PK
   2. countries: все атрибуты зависят только от PK
   3. human: все атрибуты зависят только от PK
   4. royal\_court: все атрибуты зависят только от PK
   5. skills: атрибут skill зависит от humanid
   6. statuses: все атрибуты зависят только от PK
   7. subordinate\_connection: таблица-связка, не содержит других атрибутов
   8. court\_members: таблица-связка, не содержит других атрибутов
4. Так как здесь нет транзитивных зависимостей и база данных уже находится в 2NF, то эта база данных так же находится и в 3NF.
5. Для преобразования исходной базы данных к 3Nf были созданы таблицы-связки, связь королевства и подданых была вынесена в отдельную таблицу(так же как и связь подданых для конкретного человека) для получения атомарности.
6. Проверка на пренадлежность к BCNF: BCNF требует, чтобы дла каждой функц зависимости A -> B B является суперключем(что выполняется для каждой таблица(см функц зависимости)). В таблицах, где нет никаких атрибутов кроме ключа(skills, subordinate\_connection, court\_members) нет и функц зависимостей, а значит, они соответствуют требованиям BCNF.
7. В моей базе данных могут быть полезны:
8. Объединение таблиц cities и countries, чтобы упростить получение страны для каждого города.
9. Указание города рождения для каждого человека в самой таблице human. Так же в таблицу human можно добавить атрибут skill для упрощения получения информации.
10. Информацию о подданых можно хранить в таблице royal\_court в виде массива.
11. Функция и триггер соответственно:
12. Данная функция считает средний вес для каждой страны(он записывается в таблице countries):

CREATE or replace function update\_aver\_weight()

RETURNS TRIGGER AS $$

BEGIN

UPDATE countries

SET aver\_weight = (

SELECT SUM(h.weight) / COUNT(h.id)

FROM human h

JOIN cities c ON h.originlocationid = c.id

WHERE c.country\_id = countries.id

);

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

2) Триггер для этой функции:

CREATE TRIGGER update\_aver\_weight\_trigger

AFTER INSERT OR UPDATE OR DELETE ON human

FOR EACH ROW

EXECUTE FUNCTION update\_aver\_weight();

1. Дополнительные задания:
2. На таблицу статусов повесить триггер на изменения. Для таблицы, содержащей все изменения в таблице статусов (в ней дата операции и операция, содержимое данных до операции) написать функцию, которая будет автоматически переносить старые данные в эту таблицу.
3. Написать функцию, которая получает имя короля и возвращает список всех его подданных с их статусами.
4. Решение:
5. CREATE OR REPLACE FUNCTION statuses\_alter()

RETURNS TRIGGER AS

$$

DECLARE

operation text;

old\_data text;

BEGIN

IF TG\_OP = 'INSERT' THEN

operation := 'INSERT';

ELSIF TG\_OP = 'DELETE' THEN

operation := 'DELETE';

ELSIF TG\_OP = 'UPDATE' THEN

operation := 'UPDATE';

END IF;

old\_data := 'status\_id: ' || OLD.status\_id || ', ' ||

'human\_id: ' || OLD.human\_id || ', ' ||

'social\_status: ' || OLD.social\_status || ', ' ||

'date\_of\_start: ' || OLD.date\_of\_start || ', ' ||

'date\_of\_end: ' || OLD.date\_of\_end;

INSERT INTO statuses\_modification (operation\_date,operation,old\_data)

VALUES (CURRENT\_DATE, operation, old\_data);

RETURN NEW;

END;

$$

LANGUAGE plpgsql;

CREATE TRIGGER statuses\_alter\_trigger

AFTER INSERT OR UPDATE OR DELETE ON statuses

FOR EACH ROW

EXECUTE FUNCTION statuses\_alter();

1. CREATE OR REPLACE FUNCTION get\_subordinates\_by\_king\_name(king\_name text)

RETURNS TABLE (name text, surname text, status text) AS

$$

DECLARE

king\_id INTEGER;

king\_exists BOOLEAN;

BEGIN

SELECT EXISTS (

SELECT 1

FROM human

WHERE human.name = get\_subordinates\_by\_king\_name.king\_name

AND human.id IN (

SELECT royal\_court.king\_id

FROM royal\_court

)

) INTO king\_exists;

IF NOT king\_exists THEN

RAISE EXCEPTION 'Это не король: %', king\_name;

END IF;

SELECT id INTO king\_id

FROM human

WHERE human.name = get\_subordinates\_by\_king\_name.king\_name;

RETURN QUERY

SELECT h.name, h.surname, s.social\_status

FROM subordinate\_connection sc

JOIN human h ON sc.subordinate\_id = h.id

JOIN statuses s ON sc.subordinate\_id = s.human\_id

WHERE sc.superior\_id = king\_id;

RETURN;

END;

$$

LANGUAGE plpgsql;

1. Представление, которое возвращает количество записей в таблицах, у которой больше 6 атрибутов.

CREATE OR REPLACE FUNCTION count\_rows(tbl text) RETURNS integer AS $$

DECLARE result integer;

BEGIN

EXECUTE 'SELECT count(1) FROM ' || tbl INTO result;

RETURN result;

END

$$ LANGUAGE plpgsql;

CREATE OR REPLACE VIEW table\_row\_counts AS

SELECT table\_name, count\_rows(table\_name) as row\_count

FROM information\_schema.columns

WHERE table\_schema = 'public'

GROUP BY table\_name

HAVING COUNT(\*) >= 6;