Контрольная работа 5

Физическое проектирование БД

Уважаемый, студент!
Вам необходимо:

1. Создать таблицы
2. Создать представления (готовые запросы)
3. Назначить права доступа
4. Создать индексы
5. Разработать стратегии резервного копирования

Этап **физического проектирования** заключается в определении схемы хранения, т.е. физической структуры БД. Схема хранения зависит от той физической структуры, которую поддерживает выбранная СУБД. Физическая структура БД, с одной стороны, должна адекватно отражать логическую структуру БД, а с другой стороны, должна обеспечивать эффективное размещение данных и быстрый доступ к ним. Результаты этого этапа документируются в форме схемы хранения на языке определения данных (DDL, Data Definition Language) выбранной СУБД. Принятые на этом этапе решения оказывают огромное влияние на производительность системы.

Одной из важнейших составляющих проекта базы данных является разработка средств защиты БД. Защита данных имеет два аспекта: защита от сбоев и защита от несанкционированного доступа. Для защиты от сбоев на этапе физического проектирования разрабатывается стратегия резервного копирования. Для защиты от несанкционированного доступа каждому пользователю доступ к данным предоставляется только в соответствии с его правами доступа, набор которых также является составной частью проекта БД.

Пример выполнения заданий (продолжение той же БД, что и в КР2)

Приведём описание схемы БД на DDL.

1. Создание таблиц

1. Отношение Departs (отделы):

```
create table departs (
  d_id  varchar(12)  primary key,
  d_name varchar(100)  not null);
```

2. Отношение Rooms (комнаты):

```
create table rooms (
  d_depart varchar(12)
  references departs(d_id),
  r_room  numeric(4)  not null,
  r_phone varchar(20),
  unique(r_room, r_phone));
```

3. Отношение Posts (должности):

```
create table posts (
  p_post varchar(30)  primary key,
  p_salary numeric(8,2)  not null check(p_salary>=12130));
```

4. Отношение Employees (сотрудники):

```
create table employees (
  e_id  numeric(4)  primary key,
  e_fname varchar(25)  not null,
```

```
  e_lname varchar(30)  not null,
  e_born date   not null,
  e_sex char(1) check(e_sex in ('ж','м')),
  e_pasp char(10)   not null unique,
  e_date date   not null,
  e_given varchar(50)  not null,
  e_inn char(12)   not null unique,
  e_pens char(14)   not null unique,
  e_depart varchar(12)  references departs,
  e_post varchar(30)  references posts,
  e_room numeric(4)  not null,
  e_phone varchar(20)  not null,
  e_login varchar(30),
  foreign key(e_room,e_phone)
  references rooms(r_room,r_phone));
```

(Если внешний ключ ссылается на первичный ключ отношения, его можно не указывать, как в случае ссылок на Departs и Posts).

5. Отношение Edu (образование):

```
create table edu (
  u_id  numeric(4)  references employees,
  u_type  varchar(20)  not null,
  u_spec  varchar(40),
  u_diplom  varchar(15),
  u_year  foat(4) not null,
  check(u_spec  in  ('начальное',  'среднее',  'высшее',  'средне-
  специальное')));
```

6. Отношение AdrTel (адреса-телефоны):

```
create table adrtel (
  a_id  numeric(4)  references employees,
  a_adr  varchar(50),
  a_phone  varchar(30));
```

7. Отношение Clients (заказчики):

```
create table clients (
  c_id  numeric(4)  primary key,
  c_company varchar(40)  not null,
  c_adr  varchar(50)  not null,
  c_person  varchar(50)  not null,
  c_phone  varchar(30));
```

8. Отношение Projects (проекты):

```
create table projects (
  p_id  numeric(6)  not null unique,
  p_title  varchar(100)  not null,
  p_abbr  char(10)   primary key,
  p_depart varchar(12)  references departs,
  p_company numeric(4)  references clients,
  p_chief  numeric(4)  references employees,
  p_begin  date not null,
  p_end  date not null,
  p_finish  date,
```

```
  p_cost  numeric(10) not null check(p_cost>0),
  check (p_end>p_begin),
  check (p_finish is null or p_finish>p_begin));
```

9. Отношение Stages (этапы проектов):

```
create table stages (
  s_pro  char(10)   references projects,
  s_num  numeric(2)  not null,
  s_title  varchar(200)  not null,
  s_begin  date   not null,
  s_end  date   not null,
  s_finish  date,
  s_cost  numeric(10)  not null,
  s_sum  numeric(10)  not null,
  s_form  varchar(100)  not null,
  check (s_cost>0),
  check (s_end>s_begin),
  check (s_finish is null or s_finish>s_begin));
```

10. Отношение Job (участие):

```
create table job (
  j_pro  char(10)   references projects,
  j_emp  numeric(4)  references employees,
  j_role  varchar(20)  not null,
  j_bonus  numeric(2)  not null,
  check(j_bonus>0),
  check (j_role in ('исполнитель', 'консультант')));
```

2. Создание представлений (готовых запросов)

Приведём примеры нескольких готовых запросов (представлений):

1. Список всех текущих проектов (getdate – функция, возвращающая текущую дату, определена в СУБД MySQL; в других системах аналогичная функция может называться по-другому, например, now() в MS Access, sysdate в Oracle и т.д.):

```
create view curr_projects as
  select *
     from projects
     where p_begin<=sysdate and sysdate<=p_end;
```

2. Определение суммы по текущим проектам, полученной на текущую дату:

```
create or alter view summ (title, cost, total) as
  select p_title, p_cost, sum(s_sum)
     from curr_projects, stages
    where p_abbr=s_pro
    group by p_title, p_cost;
```

3. Данные о проектах для руководителя проектов:

```
create or alter view my_projects as
  select *
     from projects p
     where exists (select * from employees e
        where e.e_id=p.p_chief and e.e_login=user);
```

Функция user возвращает имя пользователя, выполняющего текущий запрос. Таким образом, каждый пользователь получит данные только о тех проектах, руководителем

которых является. Используя аналогичный способ, можно ограничить участника проекта данными только о сотрудниках тех проектов, в которых он сам участвует.

4. Данные об участниках проектов для руководителя проектов:

```
create or alter view my_staff as
  select j.*
     from job j
     where exists (select *
        from employees e, projects p
        where e.e_id=p.p_chief and e.e_login=user
        and j.j_pro=p.p_abbr);
```

5. Данные о других участниках проекта:

```
create or alter view my_emps as
  select je.j_pro, e.e_fname, e.e_lname e_name,
  e_depart, e_post, e_phone, e_room
  from employees e, job je
  where e.e_id=je.j_emp and exists (select *
     from job jm, employees m
     where m.e_id=jm.j_emp and
     m.e_login=user and je.j_pro=jm.j_pro);
```

Для того чтобы можно было работать с этими представлениями, соответствующим пользователям нужно назначить права доступа к представлениям. Эти права перечислены в табл. 1.

Таблица 1. Права доступа к представлениям

Представления	Группы пользователей (роли)		
	Руководители организации	Руководители проектов	Участники проектов
Текущие проекты (curr_projects)	S	S	
Сумма по текущим проектам (summ)	S	S	
Проекты для руководителя (my_projects)		SIUD	
Участники проектов для руководителей (my_staff)		SIUD	
Участники проектов (my_emps)			S

3. Назначение прав доступа

Права доступа пользователей предоставляются с помощью команды GRANT. Рассмотрим для примера права сотрудника компании ***ok_user***, который является сотрудником отдела кадров. Права доступа к отношениям Departs и Rooms могут быть описаны следующим образом:

```
grant select, insert, update, delete on departs to ok_user;
grant select, insert, update, delete on rooms to ok_user;
```

Права доступа руководителей проектов (сотрудников, `staff`) к представлению `my_projects` могут быть описаны следующим образом:

```
grant select, insert, update, delete on my_projects to staff;
```

Если сотрудник не является руководителем проекта, он не получит данных через этот запрос и не сможет воспользоваться правами доступа к нему.

Права доступа участников проекта (сотрудников, `staff`) к представлению `my_emps` могут быть описаны следующим образом:

```
grant select on my_emps to staff;
```

Если сотрудник не является участником проекта, он не получит данных через этот запрос и не сможет воспользоваться правами доступа к нему.

4. Создание индексов

Анализ готовых запросов показывает, что для повышения эффективности работы с данными необходимо создать индексы для всех внешних ключей. Приведём примеры создания индексов:

```
create index e_posts on employees(e_post);
create index p_chief on projects(p_chief);
create index e_tel on employees(e_room, e_phone);
```

5. Разработка стратегии резервного копирования

Интенсивность обновления разработанной базы данных низкая, поэтому для обеспечения сохранности вполне достаточно проводить полное резервное копирование БД раз в день (перед окончанием рабочего дня). Для разработанной БД нет необходимости держать сервер включенным круглосуточно, поэтому можно создать соответствующее задание операционной системы, которое будет автоматически запускаться перед выключением сервера.