Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Высшая школа программной инженерии

Отчет по заданиям практикума СУБД

по дисциплине: «Базы данных»

Вариант №34

Выполнил
студент гр. в3530904/00030 Дружинин Н.В.

Руководитель
Старший преподаватель Прокофьев О.В.

«22»апреля 2022г.

Санкт-Петербург
2022

Содержание

Описание предметной области

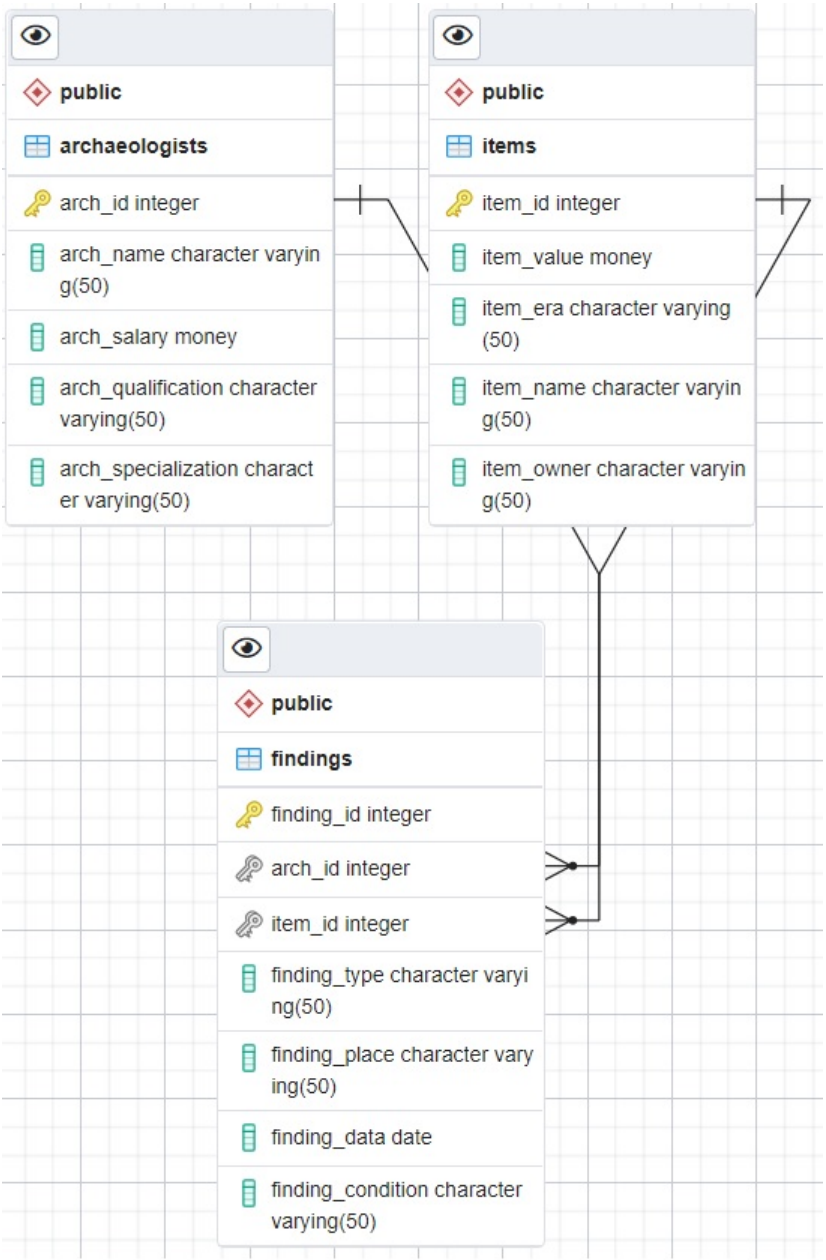
Рис. 1: Археология

1 Создание БД для приложения

1.1 Проектирование схемы базы данных

Постановка задачи:
Практическое задание связано с проектированием схемы базы данных для работы приложения. Каждый индивидуальный вариант содержит предметную область, из которой должна быть проектируемая база данных. Задачей студента является решить, для чего будет использоваться создаваемая база данных, и, исходя из этого, построить её концептуальную схему. Результатом данного практического задания является схема базы данных (в виде ER-диаграммы, содержащей таблицы и связи между ними, без уточнения типов столбцов). При сдаче задания студент должен обосновать соответствие созданной схемы поставленной задаче.

Решение:



1.2 Создание и заполнение таблиц

Постановка задачи:
Практическое задание заключается в подготовке SQL-скрипта для создания таблиц согласно схе-

ме, полученной в предыдущем задании (с уточнением типов столбцов). Необходимо определить первичные и внешние ключи, а также декларативные ограничения целостности (возможность принимать неопределенное значение, уникальные ключи, проверочные ограничения и т. д.). Таблицы следует создавать в отдельной базе данных. Кроме того, нужно подготовить данные для заполнения созданных таблиц. Объем подготовленных данных должен составлять не менее 10 экземпляров для каждой из стержневых (неподчиняющиеся основные сущности) сущностей и 15 экземпляров для каждой из ассоциативных (подчиненные многие-ко-многим). На основе этих данных необходимо создать SQL-скрипт для вставки соответствующих строк в таблицы БД.
Решение:

1. SQL-скрипт создания БД:

```
        create database archeology;
```

2. SQL-скрипт создания таблиц:

```
        create table archaeologists (
            arch_id serial primary key,
            arch_name varchar(50) NOT NULL,
            arch_salary money NOT NULL,
            arch_qualification varchar(50) NOT NULL,
            arch_specialization varchar(50) NOT NULL
        );

        create table items (
            item_id serial primary key,
            item_value money NOT NULL,
            item_era varchar(50) NOT NULL,
            item_name varchar(50) NOT NULL,
            item_owner varchar(50) NOT NULL
        );

        create table findings (
            finding_id serial primary key,
            foreign key (arch_id) references archaeologists (arch_id),
            foreign key (item_id) references items (item_id),
            finding_type varchar(50) NOT NULL,
            finding_place varchar(50) NOT NULL,
            finding_data date NOT NULL,
            finding_condition varchar(50) NOT NULL
        );
```

3. SQL-скрипт заполнения таблиц:

```
insert into archaeologists(arch_name, arch_salary, arch_qualification,
    arch_specialization) values
    ("Patrik", 5050, "Professor", "Field archaeology"),
    ("Sarah", 4440, "Associate", "Field archaeology"),
    ("Conrod", 3350, "Assistant", "Classical archaeology"),
    ("Nikolas", 8900, "Professor", "Prehistoric archaeology"),
    ("James", 5050, "Professor", "Field archaeology"),
    ("Franchesko", 4440, "Associate", "Underwater archaeology"),
    ("Simon", 3350, "Assistant", "Classical archaeology"),
    ("Jaggermeister", 7000, "Professor", "Landscape archaeology"),
    ("Amadeo", 2500, "Intern", "Classical archaeology"),
    ("Nemo", 1902, "Intern", "Zooarchaeology");

insert into items (item_value, item_era, item_name, item_owner) values
    (1000, "100BC", "Pharaon tomb", "Pharaon"),
    (2000, "323BC", "Ancient temple", "Nobleman"),
    (1500, "100AD", "Golden helmet", "Warrior"),
    (1500, "150AD", "Ancient weapon", "Warrior"),
    (3000, "600AD", "Treasure", "King"),
    (4000, "Mesozoic era", "Dinosaurs ones", "Animal"),
    (3500, "79 AD", "The lost city", "n/a"),
```

```
    (5000, "Cenozoic era", "Fish bones", "Fish"),
    (8000, "300BC", "Ancient monument", "Nobleman"),
    (7000,  "Paleozoic era", "Ancient insect", "Insect");

insert into findings (arch_id, item_id, finding_type, finding_place,
    finding_data, finding_condition) values
    (4, 1, "Tomb", "Egypt", 1924-4-11, "average"),
    (7, 2, "Temple", "Italy", 1814-03-09, "average"),
    (9, 4, "Equipment", "Egypt", 1541-05-13, "bad"),
    (2, 5, "Treasure", "UK", 2009-5-6, "good"),
    (8, 8, "Remains of animal", "Philippines", 2007-2-15, "bad"),
    (6, 7, "Ruins", "Italy", 1709-10-10,  "average"),
    (1, 9, "Monument", "Spain", 1868-01-08, "average"),
    (8, 6, "Remains of animal", "Egypt", 1799-06-15, "bad"),
    (3, 9, "Monument", "Spain", 1926-1-6, "good"),
    (5, 1, "Cemetery", "Spain", 1939-3-18, "bad "),
    (4, 1, "Tomb", "Italy", 1804-11-04, "good"),
    (10, 2, "Temple", "Mexico", 1814-03-09, "average"),
    (4, 4, "Equipment", "Russia", 1771-05-13, "good "),
    (10, 10, "Remains of animal", "Philippines", 1991-02-02,"good"),
    (3, 10, "Ruins", "China", 1739-11-10, "bad"),
    (4, 9, "Monument", "Sweden", 1831-05-08, "good"),
    (8, 6, "Remains of animal", "Egypt", 1799-06-15, "bad"),
    (7, 9, "Monument", "Spain", 1926-01-06, "good"),
    (5, 3, "Equipment", "UK", 1905-09-13, "bad"),
    (1, 1, "Cemetery", "Spain", 1939-03-18, "bad"),
    (2, 5, "Treasure", "Ukraine", 2022-02-24, "bad"),
    (2, 4, "Ruins", "Egypt", 2022-05-12, "good");
```

1.3 Операторы

Постановка задачи:
Нужно подготовить 3-4 выборки, которые имеют осмысленное значение для предметной области, и также составить для них SQL-скрипты. Сформулировать 3-4 запроса на изменение и удаление из базы данных. Запросы должны быть сформулированы в терминах предметной области. Составить SQL-скрипты для выполнения этих запросов. **Решение:**

1. Отобразить имена археологов в алфавитном порядке, которые нашли свои находки в Италии и оклад которых выше 2000.

```
select arch_name, arch_salary
from archaeologists
where arch_id in (select arch_id
     from findings
     where finding_place='Italy'
     ) and arch_salary > 2000::money
order by arch_name;
```

	arch_name character varying (50)	arch_salary money
1	Franchesko	$4,440.00
2	Nikolas	$8,900.00
3	Simon	$3,350.00

2. Отобразить имена таких археологов, которые нашли 2 находки.

```sql
select arch_id, arch_name
from archaeologists
where arch_id in (select distinct arch_id
          from (select arch_id,
          count(arch_id) over (partition by arch_id) as count_
          from findings) as ss
          where count_ = 2);
```

	arch_id [PK] integer	arch_name character varying (50)
1	1	Patrik
2	3	Conrod
3	5	James
4	7	Simon
5	10	Jaggermeister

3. Отобразить имена, оклад и квалификацию археологов, а так же их ранг в соответствии с квалификацией и максимальный оклад. Использовать оконные функции.

```sql
select arch_name, arch_salary, arch_qualification,
rank() over (partition by arch_qualification order by arch_salary) as rank
    ,
max(arch_salary) over () as max_salary
from archaeologists;
```

	arch_name character varying (50)	arch_salary money	arch_qualification character varying (50)	rank bigint	max_salary money
1	Conrod	$3,350.00	Assistant	1	$8,900.00
2	Simon	$3,350.00	Assistant	1	$8,900.00
3	Franchesko	$4,440.00	Associate	1	$8,900.00
4	Sarah	$4,440.00	Associate	1	$8,900.00
5	Nemo	$1,902.00	Intern	1	$8,900.00
6	Amadeo	$2,500.00	Intern	2	$8,900.00
7	James	$5,050.00	Professor	1	$8,900.00
8	Patrik	$5,050.00	Professor	1	$8,900.00
9	Jaggermeister	$7,000.00	Professor	3	$8,900.00
10	Nikolas	$8,900.00	Professor	4	$8,900.00

4. Запрос на изменение базы данных.

```sql
update findings
set finding_place = 'Ukraine'
where finding_place = 'Ukrane';

```

```sql
select *
from findings;
```

5. Запрос на удаление из баз данных.

```sql
delete from archaeologists
where arch_salary < 1000::money;

select *
from archaeologists;
```

1.4 Контроль целостности данных

Постановка задачи:
Необходимо подготовить SQL-скрипты для проверки наличия аномалий (потерянных изменений, грязных чтений, неповторяющихся чтений, фантомов) при параллельном исполнении транзакций на различных уровнях изолированности SQL/92 (READ UNCOMMITTED, READ COMMITTED, REPEATABLE READ, SERIALIZABLE). Подготовленные скрипты должны работать с одной из таблиц, созданных в практическом задании №2.1. Для проверки наличия аномалий потребуются два параллельных сеанса, операторы в которых выполняются пошагово:

- Установить в обоих сеансах уровень изоляции READ UNCOMMITTED. Выполнить сценарии проверки наличия аномалий потерянных изменений и грязных чтений.
- Установить в обоих сеансах уровень изоляции READ COMMITTED. Выполнить сценарии проверки наличия аномалий грязных чтений и неповторяющихся чтений.
- Установить в обоих сеансах уровень изоляции REPEATABLE READ. Выполнить сценарии проверки наличия аномалий неповторяющихся чтений и фантомов.
- Установить в обоих сеансах уровень изоляции SERIALIZABLE. Выполнить сценарий проверки наличия фантомов.

Необходимо составить скрипт для создания триггера, а также подготовить несколько запросов для проверки и демонстрации его полезных свойств:

- Изменение данных для сохранения целостности.
- Проверка транзакций и их откат в случае нарушения целостности.

Решение:

Уровень изоляции	«Грязное» чтение	Неповторяемое чтение	Фантомное чтение	Аномалия сериализации
Read uncommited (Чтение незафиксированных данных)	Допускается, но не в PG	Возможно	Возможно	Возможно
Read committed (Чтение зафиксированных данных)	Невозможно	Возможно	Возможно	Возможно
Repeatable read (Повторяемое чтение)	Невозможно	Невозможно	Допускается, но не в PG	Возможно
Serializable (Сериализуемость)	Невозможно	Невозможно	Невозможно	Невозможно

Рис. 2: Уровни изоляции транзакций

SQL-скрипт триггера:

```sql
create or replace function my_func()
returns trigger as $$
  begin
    if (new.finding_data > now()) then
      raise exception '% incorrect date', new.finding_data;
    end if;

    if (new.finding_condition != 'average' or
       new.finding_condition != 'good' or
```

```
10          new.finding_condition != 'bad')
11      then
12          raise exception '% incorrect finding condition', new.finding_condition;
13      end if;
14    end;
15  $$ LANGUAGE plpgsql;
16
17  create trigger test_trigger
18  before insert or update on findings
19  for each row
20  execute function my_func();
```

2 Роли, права, хранимые процедуры

2.1 Управление доступом

Постановка задачи:
Целью практического задания является освоение работы с представлениями и другими способами управления доступом. При выполнении задания необходимо:

- Создать пользователя test и выдать ему доступ к базе данных.
- Составить и выполнить скрипты присвоения новому пользователю прав доступа к таблицам, созданным в практическом задании №2.1. При этом права доступа к различным таблицам должны быть различными, а именно:
  - По крайней мере, для одной таблицы новому пользователю присваиваются права SELECT, INSERT, UPDATE в полном объеме.
  - По крайней мере, для одной таблицы новому пользователю присваиваются права SELECT и UPDATE только избранных столбцов.
  - По крайней мере, для одной таблицы новому пользователю присваивается только право SELECT.
- Составить SQL-скрипты для создания нескольких представлений, которые позволяли бы упростить манипуляции с данными или позволяли бы ограничить доступ к данным, предоставляя только необходимую информацию.
- Присвоить новому пользователю право доступа (SELECT) к одному из представлений
- Создать стандартную роль уровня базы данных, присвоить ей право доступа (UPDATE на некоторые столбцы) к одному из представлений, назначить новому пользователю созданную роль.
- Выполнить от имени нового пользователя некоторые выборки из таблиц и представлений. Убедиться в правильности контроля прав доступа.
- Выполнить от имени нового пользователя операторы изменения таблиц с ограниченными правами доступа. Убедиться в правильности контроля прав доступа

Решение:

1. SQL-скрипты создания роли и предоставления ей прав доступа к таблицам:

```
create role test;

grant update, select, insert on archaeologists to test;
grant select(arch_id), update(finding_data) on findings to test;
grant select on items to test;
```

2. SQL-скрипты проверки прав роли:

```
delete from archaeologists
where arch_salary < 1000::money;

select * from findings;

delete from items
where item_owner = "Warrior";
```

```
ERROR:  permission denied for table archaeologists
SQL state: 42501
ERROR:  permission denied for table findings
SQL state: 42501
ERROR:  permission denied for table items
SQL state: 42501
```

3. SQL-скрипт представления (археологи, которые обнаружили находки до 1900-го года (включить сами находки и даты)):

```sql
drop view if exists old_findings;

create view old_findings as
select arch_name, item_name, finding_data from archaeologists
inner join findings on archaeologists.arch_id = findings.arch_id
inner join items on findings.item_id = items.item_id
where finding_data < '1900-01-01';
```

	arch_name character varying (50)	item_name character varying (50)	finding_data date
1	Simon	Ancient temple	1814-03-09
2	Amadeo	Ancient weapon	1541-05-13
3	Franchesko	The lost city	1709-10-10
4	Patrik	Ancient monument	1868-01-08
5	Nemo	Dinosaurs ones	1799-06-15
6	Nikolas	Pharaon tomb	1804-11-04
7	Jaggermeister	Ancient temple	1814-03-09
8	Nikolas	Ancient weapon	1771-05-13
9	Conrod	Ancient insect	1739-11-10
10	Nikolas	Ancient monument	1831-05-08
11	Nemo	Dinosaurs ones	1799-06-15

4. Вывод макрокоманды psql /dp (флаги привилегий пользователей):

```
archeology=> \dp
                                    Access privileges
 Schema |            Name            |   Type   |     Access privileges     | Column privileges | Policies
--------+----------------------------+----------+---------------------------+-------------------+----------
 public | archaeologists             | table    | postgres=arwdDxt/postgres+|                   |
        |                            |          | test=arw/postgres         |                   |
 public | archaeologists_arch_id_seq | sequence |                           |                   |
 public | findings                   | table    |                           | arch_id:         +|
        |                            |          |                           |   test=r/postgres+|
        |                            |          |                           | finding_data:    +|
        |                            |          |                           |   test=w/postgres |
 public | findings_finding_id_seq    | sequence |                           |                   |
 public | items                      | table    | postgres=arwdDxt/postgres+|                   |
        |                            |          | test=r/postgres           |                   |
 public | items_item_id_seq          | sequence |                           |                   |
 public | old_findings               | view     | postgres=arwdDxt/postgres+|                   |
        |                            |          | test=r/postgres           |                   |
(7 rows)
```

2.2 Функции и язык PL/pgSQL

Постановка задачи:

Практическое задание посвящено упрощению работы с помощью создания и использования функций. При выполнении задания необходимо:

- Составить SQL-скрипты для создания нескольких (2-3) функций, упрощающих работу с данными.
- Продемонстрировать полученные знания о возможностях языка PL/pgSQL. В скриптах должны использоваться:
  - Циклы.
  - Ветвления.
  - Переменные.

- Курсоры.
- Исключения.

- Обосновать преимущества механизма функций перед механизмом представлений.

Решение:

- SQL-скрипт функции с циклом:

```
create or replace function get_all_even_id() returns setof archaeologists
    as
$body$
declare
  r archaeologists%rowtype;
begin
  for r in
    select * from archaeologists where tmp = 0
  loop
    return next r;
  end loop;
  return;
end;
$body$
language plpgsql;

select * from get_all_even_id();
```

	arch_id integer	arch_name character varying (50)	arch_salary money	arch_qualification character varying (50)	arch_specialization character varying (50)
1	2	Sarah	$4,440.00	Associate	Field archaeology
2	4	Nikolas	$8,900.00	Professor	Prehistoric archaeology
3	6	Franchesko	$4,440.00	Associate	Underwater archaeology
4	10	Jaggermeister	$7,000.00	Professor	Landscape archaeology
5	8	Nemo	$1,902.00	Intern	Zooarchaeology

- SQL-скрипт функции с условием:

```
create or replace function raise_salary(key_ int, summ money) returns void
    as
$$
begin
  loop
    update archaeologists set arch_salary = arch_salary + summ where
        arch_id = key_;

    if not found then
      raise notice 'Key is not found!';
    end if;

    return;
  end loop;
end;
$$
language plpgsql;
```

- SQL-скрипт функции-фильтра (выдает перечень объектов в зависимости от даты находки и нашедшего археолога)

```
create or replace function get_objects(d date, n varchar(50))
returns table(year date, founder varchar(50), item varchar(50)) as $$
  select finding_data, arch_name, item_name from findings
  inner join archaeologists on archaeologists.arch_id = findings.arch_id
  inner join items on findings.item_id = items.item_id
  where finding_data = $1 and arch_name = $2;
```

```
7 $$ language sql;
8
9 select get_objects('1924-11-04', 'Nikolas');
```

	get_objects record
1	(1924-11-04,Nikolas,"Pharaon tomb")

3 Вывод

В результате проделанной работы были сформированы следующие навыки и знания:

- проектирование схемы данных;
- манипулирование данными;
- ограничение целостности;
- механизмы поддержки целостности (транзакции и триггеры);
- настройки прав доступа;
- работа с процедурами и функциями.

4 Приложение (dump)

```
--
-- PostgreSQL database dump
--

-- Dumped from database version 14.1
-- Dumped by pg_dump version 14.1

-- Started on 2022-05-26 17:57:46

SET statement_timeout = 0;
SET lock_timeout = 0;
SET idle_in_transaction_session_timeout = 0;
SET client_encoding = 'UTF8';
SET standard_conforming_strings = on;
SELECT pg_catalog.set_config('search_path', '', false);
SET check_function_bodies = false;
SET xmloption = content;
SET client_min_messages = warning;
SET row_security = off;

SET default_tablespace = '';

SET default_table_access_method = heap;

--
-- TOC entry 210 (class 1259 OID 32927)
-- Name: archaeologists; Type: TABLE; Schema: public; Owner: postgres
--

CREATE TABLE public.archaeologists (
    arch_id integer NOT NULL,
    arch_name character varying(50) NOT NULL,
    arch_salary money NOT NULL,
    arch_qualification character varying(50) NOT NULL,
    arch_specialization character varying(50) NOT NULL
);


ALTER TABLE public.archaeologists OWNER TO postgres;

--
-- TOC entry 217 (class 1255 OID 32985)
-- Name: get_all_even_id(); Type: FUNCTION; Schema: public; Owner: postgres
--

CREATE FUNCTION public.get_all_even_id() RETURNS SETOF public.archaeologists
    LANGUAGE plpgsql
    AS $$
declare
  r archaeologists%rowtype;
begin
  for r in
    select * from archaeologists where arch_id % 2 = 0
  loop
    return next r;
  end loop;
  return;
end;
$$;


ALTER FUNCTION public.get_all_even_id() OWNER TO postgres;

--
```

```
-- TOC entry 216 (class 1255 OID 32958)
-- Name: my_func(); Type: FUNCTION; Schema: public; Owner: postgres
--

CREATE FUNCTION public.my_func() RETURNS trigger
    LANGUAGE plpgsql
    AS $$
  begin
    if (new.finding_data > now()) then
      raise exception '% incorrect date', new.finding_data;
    end if;

    if (new.finding_condition != 'average' or
       new.finding_condition != 'good' or
       new.finding_condition != 'bad') then
        raise exception '% incorrect finding condition', new.finding_condition;
    end if;
  end;
$$;


ALTER FUNCTION public.my_func() OWNER TO postgres;

--
-- TOC entry 218 (class 1255 OID 32989)
-- Name: raise_salary(integer, money); Type: FUNCTION; Schema: public; Owner:
    postgres
--

CREATE FUNCTION public.raise_salary(key_ integer, summ money) RETURNS void
    LANGUAGE plpgsql
    AS $$
begin
  loop
    update archaeologists set arch_salary = arch_salary + summ where arch_id =
        key_;

    if found then
      return;
    end if;

          raise notice 'Key is not found!';
    return;
  end loop;
end;
$$;


ALTER FUNCTION public.raise_salary(key_ integer, summ money) OWNER TO postgres;

--
-- TOC entry 209 (class 1259 OID 32926)
-- Name: archaeologists_arch_id_seq; Type: SEQUENCE; Schema: public; Owner:
    postgres
--

CREATE SEQUENCE public.archaeologists_arch_id_seq
    AS integer
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;


ALTER TABLE public.archaeologists_arch_id_seq OWNER TO postgres;
```

```
128
129 --
130 -- TOC entry 3345 (class 0 OID 0)
131 -- Dependencies: 209
132 -- Name: archaeologists_arch_id_seq; Type: SEQUENCE OWNED BY; Schema: public;
    Owner: postgres
133 --
134
135 ALTER SEQUENCE public.archaeologists_arch_id_seq OWNED BY public.archaeologists
    .arch_id;
136
137
138 --
139 -- TOC entry 214 (class 1259 OID 32941)
140 -- Name: findings; Type: TABLE; Schema: public; Owner: postgres
141 --
142
143 CREATE TABLE public.findings (
144     finding_id integer NOT NULL,
145     arch_id integer,
146     item_id integer,
147     finding_type character varying(50) NOT NULL,
148     finding_place character varying(50) NOT NULL,
149     finding_data date NOT NULL,
150     finding_condition character varying(50) NOT NULL
151 );
152
153
154 ALTER TABLE public.findings OWNER TO postgres;
155
156 --
157 -- TOC entry 213 (class 1259 OID 32940)
158 -- Name: findings_finding_id_seq; Type: SEQUENCE; Schema: public; Owner:
    postgres
159 --
160
161 CREATE SEQUENCE public.findings_finding_id_seq
162     AS integer
163     START WITH 1
164     INCREMENT BY 1
165     NO MINVALUE
166     NO MAXVALUE
167     CACHE 1;
168
169
170 ALTER TABLE public.findings_finding_id_seq OWNER TO postgres;
171
172 --
173 -- TOC entry 3348 (class 0 OID 0)
174 -- Dependencies: 213
175 -- Name: findings_finding_id_seq; Type: SEQUENCE OWNED BY; Schema: public;
    Owner: postgres
176 --
177
178 ALTER SEQUENCE public.findings_finding_id_seq OWNED BY public.findings.
    finding_id;
179
180
181 --
182 -- TOC entry 212 (class 1259 OID 32934)
183 -- Name: items; Type: TABLE; Schema: public; Owner: postgres
184 --
185
186 CREATE TABLE public.items (
187     item_id integer NOT NULL,
188     item_value money NOT NULL,
```

```
    item_era character varying(50) NOT NULL,
    item_name character varying(50) NOT NULL,
    item_owner character varying(50) NOT NULL
);


ALTER TABLE public.items OWNER TO postgres;

--
-- TOC entry 211 (class 1259 OID 32933)
-- Name: items_item_id_seq; Type: SEQUENCE; Schema: public; Owner: postgres
--

CREATE SEQUENCE public.items_item_id_seq
    AS integer
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;


ALTER TABLE public.items_item_id_seq OWNER TO postgres;

--
-- TOC entry 3350 (class 0 OID 0)
-- Dependencies: 211
-- Name: items_item_id_seq; Type: SEQUENCE OWNED BY; Schema: public; Owner:
    postgres
--

ALTER SEQUENCE public.items_item_id_seq OWNED BY public.items.item_id;


--
-- TOC entry 215 (class 1259 OID 32981)
-- Name: old_findings; Type: VIEW; Schema: public; Owner: postgres
--

CREATE VIEW public.old_findings AS
 SELECT archaeologists.arch_name,
    items.item_name,
    findings.finding_data
   FROM ((public.archaeologists
     JOIN public.findings ON ((archaeologists.arch_id = findings.arch_id)))
     JOIN public.items ON ((findings.item_id = items.item_id)))
  WHERE (findings.finding_data < '1900-01-01'::date);


ALTER TABLE public.old_findings OWNER TO postgres;

--
-- TOC entry 3181 (class 2604 OID 32930)
-- Name: archaeologists arch_id; Type: DEFAULT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.archaeologists ALTER COLUMN arch_id SET DEFAULT nextval
    ('public.archaeologists_arch_id_seq'::regclass);


--
-- TOC entry 3183 (class 2604 OID 32944)
-- Name: findings finding_id; Type: DEFAULT; Schema: public; Owner: postgres
--

```

```
252 ALTER TABLE ONLY public.findings ALTER COLUMN finding_id SET DEFAULT nextval('
    public.findings_finding_id_seq'::regclass);
253
254
255 --
256 -- TOC entry 3182 (class 2604 OID 32937)
257 -- Name: items item_id; Type: DEFAULT; Schema: public; Owner: postgres
258 --
259
260 ALTER TABLE ONLY public.items ALTER COLUMN item_id SET DEFAULT nextval('public.
    items_item_id_seq'::regclass);
261
262
263 --
264 -- TOC entry 3334 (class 0 OID 32927)
265 -- Dependencies: 210
266 -- Data for Name: archaeologists; Type: TABLE DATA; Schema: public; Owner:
    postgres
267 --
268
269 COPY public.archaeologists (arch_id, arch_name, arch_salary, arch_qualification
    , arch_specialization) FROM stdin;
270 2 Sarah $4,440.00 Associate Field archaeology
271 3 Conrod  $3,350.00 Assistant Classical archaeology
272 4 Nikolas $8,900.00 Professor Prehistoric archaeology
273 5 James $5,050.00 Professor Field archaeology
274 6 Franchesko  $4,440.00 Associate Underwater archaeology
275 7 Simon $3,350.00 Assistant Classical archaeology
276 10  Jaggermeister $7,000.00 Professor Landscape archaeology
277 9 Amadeo  $2,500.00 Intern  Classical archaeology
278 8 Nemo  $1,902.00 Intern  Zooarchaeology
279 1 Patrik  $5,100.00 Professor Field archaeology
280 \.
281
282
283 --
284 -- TOC entry 3338 (class 0 OID 32941)
285 -- Dependencies: 214
286 -- Data for Name: findings; Type: TABLE DATA; Schema: public; Owner: postgres
287 --
288
289 COPY public.findings (finding_id, arch_id, item_id, finding_type, finding_place
    , finding_data, finding_condition) FROM stdin;
290 1 4 1 Tomb  Egypt 1924-11-04  average
291 2 7 2 Temple  Italy 1814-03-09  average
292 3 9 4 Equipment Egypt 1541-05-13  bad
293 4 2 5 Treasure  UK  2009-06-05  good
294 5 8 8 Remains of animal Philippines 2007-02-15  bad
295 6 6 7 Ruins Italy 1709-10-10  average
296 7 1 9 Monument  Spain 1868-01-08  average
297 8 8 6 Remains of animal Egypt 1799-06-15  bad
298 9 3 9 Monument  Spain 1926-06-01  good
299 10  5 1 Cemetery  Spain 1939-03-18  bad
300 11  4 1 Tomb  Italy 1804-11-04  good
301 12  10  2 Temple  Mexico  1814-03-09  average
302 13  4 4 Equipment Russia  1771-05-13  good
303 15  10  10  Remains of animal Philippines 1991-02-02  good
304 16  3 10  Ruins China 1739-11-10  bad
305 17  4 9 Monument  Sweden  1831-05-08  good
306 18  8 6 Remains of animal Egypt 1799-06-15  bad
307 19  7 9 Monument  Spain 1926-06-01  good
308 20  5 3 Equipment UK  1905-09-13  bad
309 21  1 1 Cemetery  Spain 1939-03-18  bad
310 14  2 5 Treasure  Ukraine 2022-02-24  bad
311 22  2 4 Ruins Egypt 2022-05-12  good
312 \.
```

```
313
314
315 --
316 -- TOC entry 3336 (class 0 OID 32934)
317 -- Dependencies: 212
318 -- Data for Name: items; Type: TABLE DATA; Schema: public; Owner: postgres
319 --
320
321 COPY public.items (item_id, item_value, item_era, item_name, item_owner) FROM
        stdin;
322 1 $1,000.00 100BC Pharaon tomb  Pharaon
323 2 $2,000.00 323BC Ancient temple  Nobleman
324 3 $1,500.00 100AD Golden helmet Warrior
325 4 $1,500.00 150AD Ancient weapon  Warrior
326 5 $3,000.00 600AD Treasure  King
327 6 $4,000.00 Mesozoic era  Dinosaurs ones  Animal
328 7 $3,500.00 79 AD The lost city n/a
329 8 $5,000.00 Cenozoic era  Fish bones  Fish
330 9 $8,000.00 300BC Ancient monument  Nobleman
331 10  $7,000.00 Paleozoic era Ancient insect  Insect
332 \.
333
334
335 --
336 -- TOC entry 3352 (class 0 OID 0)
337 -- Dependencies: 209
338 -- Name: archaeologists_arch_id_seq; Type: SEQUENCE SET; Schema: public; Owner:
        postgres
339 --
340
341 SELECT pg_catalog.setval('public.archaeologists_arch_id_seq', 10, true);
342
343
344 --
345 -- TOC entry 3353 (class 0 OID 0)
346 -- Dependencies: 213
347 -- Name: findings_finding_id_seq; Type: SEQUENCE SET; Schema: public; Owner:
        postgres
348 --
349
350 SELECT pg_catalog.setval('public.findings_finding_id_seq', 24, true);
351
352
353 --
354 -- TOC entry 3354 (class 0 OID 0)
355 -- Dependencies: 211
356 -- Name: items_item_id_seq; Type: SEQUENCE SET; Schema: public; Owner: postgres
357 --
358
359 SELECT pg_catalog.setval('public.items_item_id_seq', 10, true);
360
361
362 --
363 -- TOC entry 3185 (class 2606 OID 32932)
364 -- Name: archaeologists archaeologists_pkey; Type: CONSTRAINT; Schema: public;
        Owner: postgres
365 --
366
367 ALTER TABLE ONLY public.archaeologists
368     ADD CONSTRAINT archaeologists_pkey PRIMARY KEY (arch_id);
369
370
371 --
372 -- TOC entry 3189 (class 2606 OID 32946)
373 -- Name: findings findings_pkey; Type: CONSTRAINT; Schema: public; Owner:
        postgres
```

```
--

ALTER TABLE ONLY public.findings
    ADD CONSTRAINT findings_pkey PRIMARY KEY (finding_id);


--
-- TOC entry 3187 (class 2606 OID 32939)
-- Name: items items_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY public.items
    ADD CONSTRAINT items_pkey PRIMARY KEY (item_id);


--
-- TOC entry 3192 (class 2620 OID 32959)
-- Name: findings test_trigger; Type: TRIGGER; Schema: public; Owner: postgres
--

CREATE TRIGGER test_trigger BEFORE INSERT OR UPDATE ON public.findings FOR EACH
    ROW EXECUTE FUNCTION public.my_func();


--
-- TOC entry 3190 (class 2606 OID 32947)
-- Name: findings findings_arch_id_fkey; Type: FK CONSTRAINT; Schema: public;
    Owner: postgres
--

ALTER TABLE ONLY public.findings
    ADD CONSTRAINT findings_arch_id_fkey FOREIGN KEY (arch_id) REFERENCES
        public.archaeologists(arch_id);


--
-- TOC entry 3191 (class 2606 OID 32952)
-- Name: findings findings_item_id_fkey; Type: FK CONSTRAINT; Schema: public;
    Owner: postgres
--

ALTER TABLE ONLY public.findings
    ADD CONSTRAINT findings_item_id_fkey FOREIGN KEY (item_id) REFERENCES
        public.items(item_id);


--
-- TOC entry 3344 (class 0 OID 0)
-- Dependencies: 210
-- Name: TABLE archaeologists; Type: ACL; Schema: public; Owner: postgres
--

GRANT SELECT,INSERT,UPDATE ON TABLE public.archaeologists TO test;


--
-- TOC entry 3346 (class 0 OID 0)
-- Dependencies: 214
-- Name: COLUMN findings.arch_id; Type: ACL; Schema: public; Owner: postgres
--

GRANT SELECT(arch_id) ON TABLE public.findings TO test;


--
-- TOC entry 3347 (class 0 OID 0)
```

```
435 -- Dependencies: 214
436 -- Name: COLUMN findings.finding_data; Type: ACL; Schema: public; Owner:
    postgres
437 --
438
439 GRANT UPDATE(finding_data) ON TABLE public.findings TO test;
440
441
442 --
443 -- TOC entry 3349 (class 0 OID 0)
444 -- Dependencies: 212
445 -- Name: TABLE items; Type: ACL; Schema: public; Owner: postgres
446 --
447
448 GRANT SELECT ON TABLE public.items TO test;
449
450
451 --
452 -- TOC entry 3351 (class 0 OID 0)
453 -- Dependencies: 215
454 -- Name: TABLE old_findings; Type: ACL; Schema: public; Owner: postgres
455 --
456
457 GRANT SELECT ON TABLE public.old_findings TO test;
458
459
460 -- Completed on 2022-05-26 17:57:47
461
462 --
463 -- PostgreSQL database dump complete
464 --
```