МИНОБРНАУКИ РОССИИ

РГУ НЕФТИ И ГАЗА (НИУ) ИМЕНИ И.М. ГУБКИНА

Факультет Автоматики и вычислительной техники
Кафедра Автоматизированных систем управления

Оценка комиссии: _____ Рейтинг: _____
Подписи членов комиссии:

_____ Папилина Т.М.
(подпись) (фамилия, имя, отчество)
_____ Волков Д.А.
(подпись) (фамилия, имя, отчество)

_____
(дата)

КУРСОВАЯ РАБОТА

по дисциплине Базы данных

на тему Проектирование реляционной базы данных.

«К ЗАЩИТЕ»

ВЫПОЛНИЛ:
Студент группы АС-21-04
(номер группы)

Доцент, к.т.н., Папилина Т.М.
(должность, ученая степень; фамилия, и.о.)

_____
(подпись)

_____
(дата)

Лунев Елисей Сергеевич
(фамилия, имя, отчество)

_____
(подпись)

_____
(дата)

Москва, 20 23

МИНОБРНАУКИ РОССИИ

РГУ НЕФТИ И ГАЗА (НИУ) ИМЕНИ И.М. ГУБКИНА

Факультет Автоматики и вычислительной техники
Кафедра Автоматизированных систем управления

ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ

по дисциплине Базы данных

на тему Проектирование реляционной базы данных

ДАНО студенту Лунев Елисей Сергеевич (фамилия, имя, отчество в дательном падеже) группы АС-21-04 (номер группы)

Содержание работы:

1. Концептуальное проектирование
2. Логическое проектирование
3. Физическое проектирование
4. Создание представлений для работы с БД

Исходные данные для выполнения работы:

1.
2.

Рекомендуемая литература:

1. Базы данных [Электронный ресурс] : учебное пособие / . — Саратов : Научная книга, 2012. — 158 c. — Режим доступа: http://www.iprbookshop.ru/6261.html

Графическая часть:

1. ER-диаграмма базы данных

Требования к представлению результатов

✓	Электронная версия
	Бумажный вариант и электронный образ документа

Руководитель: _____ (уч.степень) _____ (должность) _____ (подпись) _____ (фамилия, имя, отчество)

Задание принял к исполнению: студент _____ (подпись) Лунев Елисей Сергеевич (фамилия, имя, отчество)

Оглавление

Введение

Целью данной работы является проектирование реляционной базы данных, через прохождение 4 последовательных этапов: концептуальное проектирование, логическое проектирование, физическое проектирование и создание представлений.

В качестве тематики для проектирования я выбрал магазин одежды.

Для выполнения данной работы мы будем использовать реляционную СУБД – PostgreSQL, в частности инструмент для визуального проектирования баз данных – PgAdmin4.

В результате выполнения задания мы ожидаем получить функционирующую реляционную базу данных отелей.

PostgreSQL — это объектно-реляционная система управления базами данных (ORDBMS), наиболее развитая из открытых СУБД в мире. Имеет открытый исходный код и является альтернативой коммерческим базам данных.

База данных – это совокупность взаимосвязанных, хранящихся вместе данных при наличии такой минимальной избыточности, которая допускает их использование оптимальным образом для одного или нескольких приложений.

СУБД – Система Управления Базами Данных – программа поддержки интегрированной совокупности данных, предназначенная для создания, ведения и использования базы данных многими пользователями.

Реляционная база данных – организация данных в виде двухмерных таблиц отношений, где каждая таблица обладает свойствами:

- все столбцы однородны;
- каждый столбец имеет уникальное имя;
- отсутствуют одинаковые строки;
- порядок строк и столбцов неважен.

Концептуальное проектирование

Концептуальное проектирование заключается в создании абстрактной (концептуальной) модели базы данных. Эта модель данных создается на основе информации, записанной в спецификациях требований пользователей.

Концептуальное проектирование базы данных абсолютно не зависит от таких подробностей ее реализации, как тип выбранной целевой СУБД, набор создаваемых прикладных программ, используемые языки программирования,тип выбранной вычислительной платформы, а также от любых других особенностей физической реализации [1].

На данном этапе была выбрана тема для проектируемой базы данных:Магазин-цех одежды. Были выделены следующие сущности: Работники, покупатели, заказы, товары, цех, поставщик.

Сущности и принадлежащие им атрибуты:

- Работники: id, full_name, date_birth, job_title, salary;
- Покупатели: id, buyer_name, login;
- Заказы: id, count_order, id_product;
- Товары: id, price, amount, id_cloth;
- Цех: id, textile_count, fittings_clout, id_cloth;
- Поставщик: id, name, countmetr, price.

Между собой данные сущности связанны следующим образом:

- Работники-заказы один ко многим, один работник может быть ответственным за множество заказов
- Работники- цех, один ко многим, один работник может быть ответственным за множество цехов
- Заказы- продукты многие ко многим, в заказе может быть несколько одинаковых продуктов, один продукт может быть, в нескольких заказах
- Заказы- покупатели, один ко многим, один покупатель может заказать множество заказов, но каждый заказ индивидуален и не может быть одновременно у двух покупателей
- Цех- поставщик- многие ко многим, множество цехов могут быть зависимы от множества поставщиков и наоборот
- Все эти сущности связаны дополнительной таблицей и внешними

ключами

Логическое проектирование

На втором этапе – в логическом проектировании – создается схема базы данных, основанная на концептуальной модели данных, спроектированной на первом этапе. Учитываются особенности выбранной модели данных, в данномслучае – реляционной модели данных. Создаем схему отношений сущностей, которые мы выявили на первом этапе, с указание их первичных ключей.

Этот этап целиком осуществляем с помощью инструмента – PostgreSQL. В этой программе строим схему нашей базы данных.

Первичный ключ – поле, или группа полей, которые позволяютоднозначно определить каждую запись.

Первичными ключами у каждой сущности стали:

- Работники: id;
- Покупатели: id;
- Заказы: id;
- Товары: id;
- Цех: id;
- Поставщик: id.

Внешний ключ – поле, по которому происходит связь двух сущностей.Внешние ключи у сущностей:

- Заказы-продукты – id_orders, id_products;
- Заказы-покупатели –id_buyers, id_orders;
- Заказы-работники –id_employees, id_orders;
- Цех-поставщики- id_workshop, id_supply;
- Цех-работники –id_employees, id_workshop.

Результат второго этапа представлен на – ER-диаграмме базы данных (*Рис.1*).

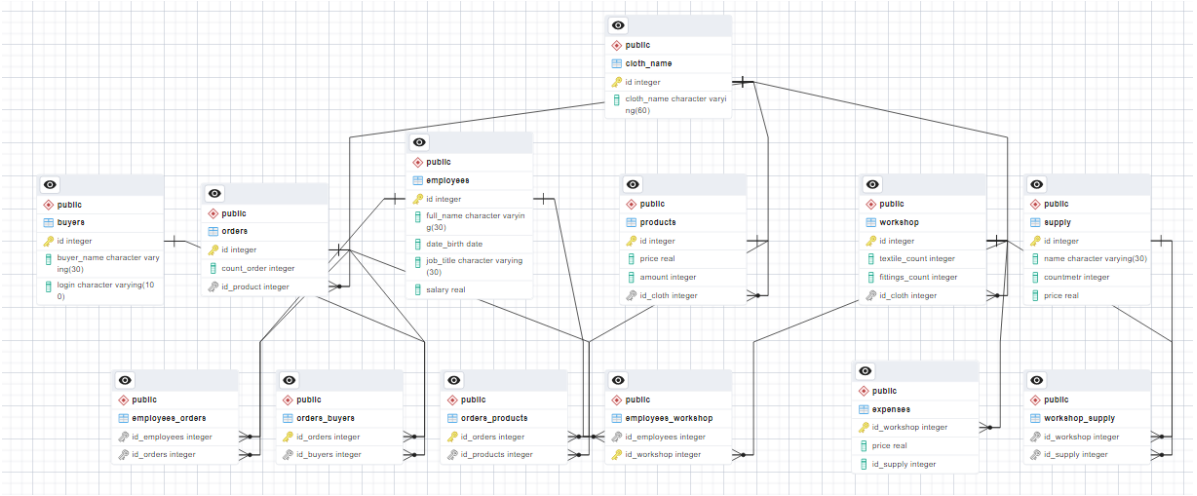

Рис. 1 EER-диаграмма базы данных

Физическое проектирование

Физическое проектирование базы данных - процесс подготовки описания реализации базы данных на вторичных запоминающих устройствах; на этом этапе рассматриваются основные отношения, организация файлов и индексов, предназначенных для обеспечения эффективного доступа к данным, а также связанные с этим ограничения целостности.

Под целостностью базы данных подразумевается – соответствие имеющейся в базе данных информации ее внутренней логике, структуре и всем явно заданным правилам. Каждое правило, налагающее некоторое ограничение на возможное состояние базы данных, называется ограничением целостности.

Основной целью физического проектирования базы данных является описание способа физической реализации логического проекта базы данных.

В случае реляционной модели данных под этим подразумеваетсяследующее:

- создание набора реляционных таблиц и ограничений для них на основе информации, представленной в глобальной логической модели данных;
- определение конкретных структур хранения данных и методов доступа кним, обеспечивающих оптимальную производительность СУБД;
- разработка средств защиты создаваемой системы.

Задача этого этапа – создание скрипта на языке SQL, а также обеспечение целостности связей между сущностями в базе данных.

Результатом третьего этапа является скрипт на языке SQL, представляющий собой физическое воплощение проектируемой базы данных.

Листинг программы представлен в приложении 1.

Создание представлений для работы с БД

Представление или view – это виртуальная таблица, созданная запросом, которая обычно объединяет несколько таблиц.

В ходе данной работы было создано 5 представлений:

1. Первое представление связывает покупателя и работника, который за него отвечает. Листинг программы и её вывод представлен на рисунках 2 и 3.

	string_agg text	buyer_name character varying (30)
1	Griffith Love	Hedda Decker
2	Griffith Love	Ivanov Ivan
3	Beau Barnett	Krivenko Danil
4	Beau Barnett	Lesley Mcclain

Рис 2 Вывод 1 представления

```
SELECT string_agg(e.full_name::text, ','::text)
AS string_agg,
   b.buyer_name
  FROM employees e
    JOIN employees_orders eo
    ON e.id = eo.id_employees
    JOIN orders o ON o.id = eo.id_orders
    JOIN orders_buyers ob ON o.id = ob.id_orders
    JOIN buyers b ON b.id = ob.id_buyers
 GROUP BY b.buyer_name;
```

Рис 3 Листинг 1 представления

2. Второе представление показывает, сколько приобрели вещей для каждой категории. Листинг программы и её вывод представлен на рисунках 3 и 4.

	cloth_name character varying (60)	count_order integer	price real	sum bigint
1	tshirt	1	1500	15
2	hoodie	1	3999	6
3	tshirt	3	1500	15
4	hoodie	3	3999	6
5	tshirt	2	1500	15
6	hoodie	2	3999	6
7	tshirt	4	1500	15
8	tshirt	5	1500	15

Рис 4. Вывод 2 представления

```
SELECT cn.cloth_name,
   o.count_order,
   p.price,
   sum(o.count_order) OVER
   (PARTITION BY cn.cloth_name) AS sum
  FROM orders o
    JOIN orders_products op ON op.id_orders = o.id
    JOIN products p ON op.id_products = p.id
    JOIN cloth_name cn ON cn.id = p.id_cloth
 GROUP BY cn.cloth_name, o.count_order, p.price
 ORDER BY (sum(o.count_order));
```

Рис 4. Листинг 2 представления

3. Третье представление показывает, сколько и какой ткани/фурнитуры надо заказать. Листинг программы и её вывод представлен на рисунках 5 и 6.

	id integer	textile_count integer	fittings_count integer	cloth_name character varying (60)	name character varying (30)	count bigint
1	1	60	20	hoodie	трикотаж	1
2	2	30	10	tshirt	флис	1
3	1	60	20	hoodie	фурнитура	2
4	2	30	10	tshirt	фурнитура	2

Рис 5. Вывод 3 представления

```
SELECT w.id,
   w.textile_count,
   w.fittings_count,
   cn.cloth_name,
   s.name,
   COUNT(*) OVER (PARTITION BY s.name)
  FROM workshop w
    JOIN workshop_supply ws ON ws.id_workshop = w.id
    JOIN supply s ON s.id = ws.id_supply
    JOIN cloth_name cn ON cn.id = w.id_cloth
 GROUP BY w.id, w.textile_count, w.fittings_count, cn.cloth_name, s.name
 ORDER BY (count(*)) DESC;
```

Рис 6. Листинг 3 представления

4. Четвёртое представление показывает, какой работник отвечает за какой цех, сколько ткани необходимо заказать. Листинг программы и её вывод представлен на рисунках 7 и 8.

	id integer	buyer_name character varying (30)	count_order integer	cloth_name character varying (60)	price real	sum real
1	2	Krivenko Danil	2	tshirt	1500	9498
2	1	Ivanov Ivan	4	tshirt	1500	3000
3	4	Hedda Decker	1	tshirt	1500	1500
4	6	Ryder Chase	5	tshirt	1500	5499
5	5	Victor Beard	3	tshirt	1500	1500
6	1	Ivanov Ivan	2	tshirt	1500	3000
7	2	Krivenko Danil	2	hoodie	3999	9498
8	2	Krivenko Danil	1	hoodie	3999	9498
9	7	Alice Rowe	1	hoodie	3999	3999
10	3	Lesley Mcclain	3	hoodie	3999	3999
11	6	Ryder Chase	2	hoodie	3999	5499

Рис 9. Вывод 4 представления

```sql
SELECT b.id,
    b.buyer_name,
    o.count_order,
    cn.cloth_name,
    p.price,
    sum(p.price) OVER (PARTITION BY b.buyer_name) AS sum
   FROM buyers b
     JOIN orders_buyers ob ON b.id = ob.id_buyers
     JOIN orders o ON o.id = ob.id_orders
     JOIN orders_products op ON op.id_orders = o.id
     JOIN products p ON p.id = op.id_products
     JOIN cloth_name cn ON p.id_cloth = cn.id
  GROUP BY b.id, b.buyer_name, o.count_order, cn.cloth_name, p.price
  ORDER BY (sum(p.price));
```

Рис 8. Листинг 4 представления

5. Пятое представление связывает сущности заказы и продукты. Показано на рисунках 9 и 10.

	cloth_name character varying (60)	count_order integer	price real	sum bigint
1	tshirt	1	1500	15
2	hoodie	1	3999	6
3	tshirt	3	1500	15
4	hoodie	3	3999	6
5	tshirt	2	1500	15
6	hoodie	2	3999	6
7	tshirt	4	1500	15
8	tshirt	5	1500	15

Рис 9. Вывод 5 представления

```
SELECT cn.cloth_name,
   o.count_order,
   p.price,
   sum(o.count_order) OVER (PARTITION BY
   cn.cloth_name) AS sum
  FROM orders o
    JOIN orders_products op ON op.id_orders = o.id
    JOIN products p ON op.id_products = p.id
    JOIN cloth_name cn ON cn.id = p.id_cloth
 GROUP BY cn.cloth_name, o.count_order, p.price
 ORDER BY (sum(o.count_order));
```

Рис 10. Листинг 5 представления

Заключение

В результате выполнения курсовой работы была создана реляционная база данных по тематике Магазина одежды. Были последовательно пройдены три основных этапов проектирования баз данных: концептуальное проектирование, логическое проектирование и физическое проектирование. Также, в ходе работы были созданы представления, которые могут быть полезны пользователям для работы с базой данных.

Мы ознакомились с процессом проектированием базы данных на языке PostgreSQL, а также изучили инструмент для визуального проектирования баз данных – PgAdmin4.

Список литературы

1. Базы данных [Электронный ресурс] : учебное пособие / . — Саратов : Научная книга, 2012. — 158 с. — Режим доступа: http://www.iprbookshop.ru/6261.html
2. Мартишин, С.А., Симонов В.Л., Храпченко М.В.. Проектирование и реализация баз данных в СУБД MySQL с использованием MySQL Workbench: Методы и средства проектирования информационных систем и технологий - М.: Форум, 2018. - 61 с.
3. Документация PostgreSQL https://postgrespro.ru/docs/postgresql/9.6/

Приложение 1

```
CREATE TABLE IF NOT EXISTS public.buyers
(
    id integer NOT NULL,
    buyer_name character varying(30) COLLATE pg_catalog."default" NOT NULL,
    login character varying(100) COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT buyers_pkey PRIMARY KEY (id)
);

CREATE TABLE IF NOT EXISTS public.cloth_name
(
    id integer NOT NULL,
    cloth_name character varying(60) COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT cloth_name_pkey PRIMARY KEY (id)
);

CREATE TABLE IF NOT EXISTS public.employees
(
    id integer NOT NULL,
    full_name character varying(30) COLLATE pg_catalog."default" NOT NULL,
    date_birth date NOT NULL,
    job_title character varying(30) COLLATE pg_catalog."default" NOT NULL,
    salary real NOT NULL,
    CONSTRAINT employees_pkey PRIMARY KEY (id)
);

CREATE TABLE IF NOT EXISTS public.employees_orders
(
    id_employees integer NOT NULL,
    id_orders integer NOT NULL
);

CREATE TABLE IF NOT EXISTS public.employees_workshop
(
    id_employees integer NOT NULL,
    id_workshop integer NOT NULL,
    CONSTRAINT employees_workshop_pkey PRIMARY KEY (id_workshop)
);

CREATE TABLE IF NOT EXISTS public.expenses
(
    id_workshop integer NOT NULL,
    price real NOT NULL,
    id_supply integer NOT NULL,
    CONSTRAINT expenses_pkey PRIMARY KEY (id_workshop)
);

CREATE TABLE IF NOT EXISTS public.orders
(
    id integer NOT NULL,
    count_order integer NOT NULL,
    id_product integer,
    CONSTRAINT orders_pkey PRIMARY KEY (id)
);
```

```sql
CREATE TABLE IF NOT EXISTS public.orders_buyers
(
    id_orders integer NOT NULL,
    id_buyers integer NOT NULL,
    CONSTRAINT fkey_orders PRIMARY KEY (id_orders)
);

CREATE TABLE IF NOT EXISTS public.orders_products
(
    id_orders integer NOT NULL,
    id_products integer NOT NULL,
    CONSTRAINT orders_products_pkey PRIMARY KEY (id_orders)
);

CREATE TABLE IF NOT EXISTS public.products
(
    id integer NOT NULL,
    price real NOT NULL,
    amount integer NOT NULL,
    id_cloth integer,
    CONSTRAINT products_pkey PRIMARY KEY (id)
);

CREATE TABLE IF NOT EXISTS public.supply
(
    id integer NOT NULL,
    name character varying(30) COLLATE pg_catalog."default" NOT NULL,
    countmetr integer NOT NULL,
    price real NOT NULL,
    CONSTRAINT supply_pkey PRIMARY KEY (id)
);

CREATE TABLE IF NOT EXISTS public.workshop
(
    id integer NOT NULL,
    textile_count integer NOT NULL,
    fittings_count integer NOT NULL,
    id_cloth integer,
    CONSTRAINT workshop_pkey PRIMARY KEY (id)
);
```

```sql
CREATE TABLE IF NOT EXISTS public.workshop_supply
(
    id_workshop integer,
    id_supply integer
);

ALTER TABLE IF EXISTS public.employees_orders
    ADD CONSTRAINT empord_id_employees_fkey FOREIGN KEY (id_employees)
    REFERENCES public.employees (id) MATCH SIMPLE
    ON UPDATE CASCADE
    ON DELETE NO ACTION;


ALTER TABLE IF EXISTS public.employees_orders
    ADD CONSTRAINT empord_id_orders_fkey FOREIGN KEY (id_orders)
    REFERENCES public.orders (id) MATCH SIMPLE
    ON UPDATE CASCADE
    ON DELETE CASCADE;
CREATE INDEX IF NOT EXISTS "fki_F"
    ON public.employees_orders(id_orders);


ALTER TABLE IF EXISTS public.employees_workshop
    ADD CONSTRAINT employees_workshop_id_employees_fkey FOREIGN KEY (id_employees)
    REFERENCES public.employees (id) MATCH SIMPLE
    ON UPDATE CASCADE
    ON DELETE NO ACTION;


ALTER TABLE IF EXISTS public.employees_workshop
    ADD CONSTRAINT employees_workshop_id_workshop_fkey FOREIGN KEY (id_workshop)
    REFERENCES public.workshop (id) MATCH SIMPLE
    ON UPDATE CASCADE
    ON DELETE CASCADE;
CREATE INDEX IF NOT EXISTS employees_workshop_pkey
    ON public.employees_workshop(id_workshop);

ALTER TABLE IF EXISTS public.expenses
    ADD CONSTRAINT id_workshop_fkey FOREIGN KEY (id_workshop)
    REFERENCES public.workshop (id) MATCH SIMPLE
    ON UPDATE CASCADE
    ON DELETE CASCADE;
CREATE INDEX IF NOT EXISTS expenses_pkey
    ON public.expenses(id_workshop);


ALTER TABLE IF EXISTS public.orders
    ADD CONSTRAINT orders_id_cloth_fkey FOREIGN KEY (id_product)
    REFERENCES public.cloth_name (id) MATCH SIMPLE
    ON UPDATE CASCADE
    ON DELETE SET NULL;
```

```sql
ALTER TABLE IF EXISTS public.orders_buyers
    ADD CONSTRAINT orders_buyers_id_buyers_fkey FOREIGN KEY (id_buyers)
    REFERENCES public.buyers (id) MATCH SIMPLE
    ON UPDATE CASCADE
    ON DELETE CASCADE;


ALTER TABLE IF EXISTS public.orders_buyers
    ADD CONSTRAINT orders_buyers_idorders_fkey FOREIGN KEY (id_orders)
    REFERENCES public.orders (id) MATCH SIMPLE
    ON UPDATE CASCADE
    ON DELETE CASCADE;
CREATE INDEX IF NOT EXISTS fkey_orders
    ON public.orders_buyers(id_orders);
```

```sql
ALTER TABLE IF EXISTS public.orders_products
    ADD CONSTRAINT orders_products_id_orders_fkey FOREIGN KEY (id_orders)
    REFERENCES public.orders (id) MATCH SIMPLE
    ON UPDATE CASCADE
    ON DELETE CASCADE;
CREATE INDEX IF NOT EXISTS orders_products_pkey
    ON public.orders_products(id_orders);


ALTER TABLE IF EXISTS public.orders_products
    ADD CONSTRAINT orders_products_id_products_fkey FOREIGN KEY (id_products)
    REFERENCES public.products (id) MATCH SIMPLE
    ON UPDATE CASCADE
    ON DELETE NO ACTION;


ALTER TABLE IF EXISTS public.products
    ADD CONSTRAINT products_id_cloth_fkey FOREIGN KEY (id_cloth)
    REFERENCES public.cloth_name (id) MATCH SIMPLE
    ON UPDATE CASCADE
    ON DELETE SET NULL;
```

```
ALTER TABLE IF EXISTS public.workshop
    ADD CONSTRAINT workshop_id_cloth_fkey FOREIGN KEY (id_cloth)
    REFERENCES public.cloth_name (id) MATCH SIMPLE
    ON UPDATE CASCADE
    ON DELETE SET NULL;


ALTER TABLE IF EXISTS public.workshop_supply
    ADD CONSTRAINT id_supply_fkey FOREIGN KEY (id_supply)
    REFERENCES public.supply (id) MATCH SIMPLE
    ON UPDATE CASCADE
    ON DELETE CASCADE;


ALTER TABLE IF EXISTS public.workshop_supply
    ADD CONSTRAINT id_workshop_fkey FOREIGN KEY (id_workshop)
    REFERENCES public.workshop (id) MATCH SIMPLE
    ON UPDATE CASCADE
    ON DELETE CASCADE;

END;
```

Листинг программы