ЗМІСТ

ВСТУП 4
1 ПОСТАНОВКА ЗАДАЧІ 6
2 ПРОЕКТУВАННЯ БАЗИ ДАНИХ 8
3 ВИБІР ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 12
4 СТВОРЕННЯ БАЗИ ДАНИХ 17
4.1 Створення таблиць 17
4.2 Створення представлень 24
4.3 Створення тригерів 26
4.4 Створення збережених процедур (функцій). 32
5 МАНІПУЛЮВАННЯ ДАНИМИ 38
5.1 Оператори відновлення 38
5.2 Оператори вибірки 43
6 СТВОРЕННЯ КОРИСТУВАЧІВ І ПРИЗНАЧЕННЯ ПРАВ ДОСТУПУ 50
ВИСНОВКИ 54
СПИСОК ЛІТЕРАТУРИ 55
ДОДАТОК А СТВОРЕННЯ ПРЕДСТАВЛЕНЬ 57
ДОДАТОК Б СТВОРЕННЯ ТРИГЕРІВ 58
ДОДАТОК В СТВОРЕННЯ ЗБЕРЕЖЕНИХ ПРОЦЕДУР (ФУНКЦІЇ) 70
ДОДАТОК Д ОПЕРАТОРИ ВІДНОВЛЕННЯ 74

ВСТУП

У наш час величезна кількість фірм використовують персональні комп'ютери для збереження і обробки будь-якого виду інформації. Ця інформація міститься в базах даних. Бази даних відіграють важливу роль в світі, що розвивається технологій. Все, з чим ми щодня взаємодіємо в житті, по всій видимості, зафіксовано в який-небудь базі. Робота з базами даних є найважливішим навиком в роботі з комп'ютером, а фахівці даної області стають все більш затребуваними. Головні ідеї нинішньої інформаційної методики базуються на уявленні, відповідно чому інформація повинна бути утворена в бази даних з завданням відображення динамічно мінливого світу і задоволення всіх потреб в інформації у користувачів. Бази даних формуються і працюють під управлінням спеціальних програмних засобів, які називаються системами управління базами даних (СУБД).

База даних - це організована структура, яка призначена для зберігання інформації. У той час, коли відбувався розвиток терміна баз даних, в них зберігалися виключно інформація, проте вже в наші дні багато систем управління базами даних дозволяє розміщувати в своїх структурах і дані, і програмний код, за допомогою якого здійснюється зв'язок з користувачами або з іншими програмно - аппаратними комплексами. При цьому дані мають не суперечити один одному,бути цілісними і не надмірними. База даних створюється для збереження і безпосереднього доступу до інформації, що містить відомості про шуканої предметної області. Ступінь конкретизації даних обумовлюється групою факторів. Перш за все, метою використання інформації з баз даних і складністю інформаційних процесів, існуючих в межах предметної області в конкретних умовах.

У комп'ютері дані бази даних представляється у вигляді таблиці, схожою на електронну таблицю. Назви стовпців, що представляють

заголовки таблиці, називають іменами полів, а самі стовпчики - полями. Дані, які знаходяться в полях, називають значеннями полів.

Самі бази даних - це сховища величезної кількості систематизованої інформації, з якими здійснюються такі операції: зміна, копіювання, видалення, додавання, упорядкування. Накопичення зберігається обсягу інформації, зростання групи користувачів інформаційних систем служать джерелом до великого розвитку найкомфортніших в інтерфейсі і щодо легких для розуміння табличних систем управління базами даних. Створення доступу до інформації бази даних відразу декількох користувачів одночасного, часто знаходяться на далекій відстані від місця зберігання баз даних, а також один від одного, тому і створені на багато користувачів мережеві версії баз даних сформовані на табличній структурі. У них вирішуються проблеми характерні для паралельних процесів, правильності даних, а також отримання несанкціонованого входу.

За останні роки йде спостереження напрямки до ускладнення структури даних. Прості типи інформації, які подаються у вигляді текстових рядків і чисел, не втративши своєї важливості, доповнюються сьогодні великою кількістю документів, які використовують засоби мультимедіа, образів графіки, процедурних або активних даних і великим числом інших істотно ускладнених форм інформації. З цієї причини виникла ціла низка досить витончених систем управління базами даних, що забезпечують нові колекції даних і вміють реалізувати переваги сучасних апаратних технологій. Однією з таких систем управління базами даних називається Microsoft Access, а також Postgres SQL, яка є найпоширенішою повноцінної серверної системою управління базами даних.

1 ПОСТАНОВКА ЗАДАЧІ

Актуальність: технічний прогрес завжди рухав бізнес сферу вперед і ті видавництва, які не встигає за технічним прогресом нерідко залишаються поза справами. В перспективі, можна буде використовуючи цю базу даних, як фундамент, створити інтернет – видавництво.

В наслідок цього - рідко можна зустріти компанії, підприємства, і тим більше корпорації, які не шанують прогресивні технології і можуть конкурувати з сучасними підприємствами, які використовують всі плюси технічного прогресу. Така технологія як бази даних це давно не нова технологія і на багатьох видавництвах є дуже важливим аспектом. СУБД може істотно полегшити роботу видавництва, а найголовніше збільшити швидкість та ефективність роботи.

Створення бази даних щодо роботи видавництва має на меті полегшання керування видавництвом та забезпечення швидкішого обліку кадрів та видань. Ця база даних зберігає інформацію про замовлення кліента та його особисті данні, а також інформацію про співробітників фірми. Також вана містить інформацію про друкарні з якими укладений контракт, інформацію про авторів з якими укладений контракт, а також інформацію про видання.

Для полегшання роботи працівників видавництва існують спеціальні функції, які допоможуть швидше виконувать потрібні дії. Наприклад: додавання інформації не турбуючись над тим, який повинен бути первинний ключ. Особливість даної БД полягає в автоматичному розподілу працівників та друкарень у разі звільнення (розірвання контракту) працивника (з друкарнею), а також автоматичне створення зв’язків між авторами та виданням. Це виключає можливість помилки співпрацівника в цілому.

Дана база даних має такий функціонал:

- ведення замовлень від клієнтів;
- автоматичний розподіл праці;
- зручне додавання інформації;
- автоматичне створення зв’язків між авторами та виданням;
- забезпечити оперативний перегляд списку замовлень ;
- додавання, зміни, пошуку інформації в базі даних;
- ведення бугалтерії та ін.

2 ПРОЕКТУВАННЯ БАЗИ ДАНИХ

Створення бази даних слід починати з її проектування (розробки). У результаті проектування має бути визначена структура бази, тобто склад таблиць, їхня структура та логічні зв'язки. Спочатку потрібно зрозуміти з яких етапів складається проектування бази даних, а саме:

- робота починається з визначення генерального переліку полів, який може нараховувати десятки і сотні позицій;
- відповідно до типу даних, що розміщуються в кожному полі, визначають тип кожного поля;
- розподіляють поля генерального списку по базових таблицях. На першому етапі розподіл здійснюють за функціональною ознакою. Мета - забезпечити одноразове введення даних в одну таблицю по можливості в рамках одного підрозділу, або (ще краще) - на одному робочому місці. На другому етапі розподілу полів здійснюють нормалізацію даних з метою вилучення повторів даних у таблицях бази даних;
- для кожної таблиці визначають ключове поле. Ключовим вибирають поле, дані в якому повторюватись не можуть;
- на наступному етапі визначають зв'язки між таблицями (схему даних). Зв'язки між таблицями організуються на основі спільного поля, причому в одній із таблиць воно обов'язково має бути ключовим;
- "паперовий" етап роботи над технічними пропозиціями закінчується розробкою схеми даних. Якщо схема даних складена правильно, підключити до бази нові таблиці неважко. На цьому етапі завершується попереднє проектування бази даних, і на наступному етапі починається її безпосередня розробка (впровадження).

Таблиця 2.1-Опис сутностей

Ім’я сутності	Зміст
Client	Містить загальну інформацію про клієнтів
Entity	Містить інформацію про клієнтів – юридичних осіб
Individual	Містить інформацію про клієнтів – фізичних осіб
Employee	Містить всі відомості про співробітників видавництва
Booking	Містить інформацію про замовлення
Edition	Містить інформацію про видання для даного замовлення
Author	Містить інформацію аторів
Typography	Містить інформацію друкарні
Edition_Author	Містить зв’язки між авторами та виданнями

Таблиця 2.2-Опис атрибутів

Ім’я сутності	Ім’я атрибутів	Тип даних	Ключ
Client	*client_id telephone fax address	Чисельний Символьний Символьний Address	Первинний
Entity	*client_id telephone fax address name representative	Чисельний Символьний Символьний Address Символьний Символьний	Первинний
Individual	*client_id telephone fax address full name pasport	Чисельний Символьний Символьний Address Символьний Символьний	Первинний
Employee	*passport salary full name post	Символьний Числовий Символьний Символьний	Первинний
Booking	*booking_id *client_id *typography_id booking_date check_mark realization_date *responsible_id	Чисельний Чисельний Чисельний Дата Біт Дата Символьний	Первинний Зовнішній Зовнішній Зовнішній

Продовження таблиці 2.2 – Опис атрибутів

Edition	*edition_id *booking_id page_size printing name edition_type	Чисельний Чисельний Чисельний Чисельний Символьний Символьний	Первинний Зовнішній
Author	*author_id full_name note	Чисельний Символьний Текстовий	Первиний
Typography	*typography_id name address telephone contact_person	Чисельний Символьний Address Символьний Символьний	Первиний
Edition_Author	*edition_id *author_id	Чисельний Чисельний	Первиний, Зовнішній Первиний, Зовнішній

3 ВИБІР ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Вибір системи управління баз даних (СКБД) являє собою складну багатопараметричних завдання і є одним з важливих етапів при розробці додатків баз даних. Обраний програмний продукт повинен задовольняти як поточним, так і майбутнім потребам підприємства, при цьому слід враховувати фінансові витрати на придбання необхідного обладнання, самої системи, розробку необхідного програмного забезпечення на її основі, а також навчання персоналу. Крім того, необхідно переконатися, що нова СУБД здатна принести підприємству реальні вигоди.

Очевидно, найбільшпростий підхід при виборі СУБД заснований на оцінці того, якою мірою існуючісистеми задовольняють основним вимогам створюваного проекту інформаційноїсистеми. Більш складним і дорогим варіантом є створення випробувального проекту на основі кількох СУБД і подальший вибір найбільшгідного із кандидатів. Але і в цьому випадку необхідно обмежувати коломожливих систем, спираючись на певні критерії відбору. Взагалі кажучи, переліквимог до СУБД, використовуваних при аналізі тієї чи іншої інформаційної системи,може змінюватися в залежності від поставлених цілей. Проте можна виділити кілька груп критеріїв:

- моделювання даних;
- особливості архітектури та функціональні можливості;
- контроль роботи системи;
- особливості розробки додатків;
- продуктивність;
- надійність;
- вимоги до робочого середовища;
- змішані критері.

В данній курсовій роботі розроблена база даних за допомогою PostgreSQL. Нижче наведене обгрунтування такого вибору, а також сильні сторони данного СУБД.

PostgreSQL - об'єктно-реляційна система управління базами даних. Є альтернативою як комерційним СУБД (Oracle Database, Microsoft SQL Server, IBM DB2 та інші), так і СУБД з відкритим кодом (MySQL, Firebird, SQLite).

PostgreSQL надає безліч різних можливостей, досить надійна і має хороші характеристики по продуктивності. Вона працює практично на всіх UNIX-платформах, включаючи UNIX-подібні системи, такі як FreeBSD і Linux. Її можна застосовувати на Windows NT Server і Windows 2000 Server, а для розробки годяться навіть такі системи Microsoft для робочих станцій, як ME. Крім того, PostgreSQL вільно поширюється і має відкритий вихідний код. PostgreSQL вигідно відрізняється від багатьох інших СУБД. Вона володіє практично всіма можливостями, які є в інших базах даних (комерційних або Open Source), а також деякими додатковими. Перелік функціональних можливостей PostgreSQL:

- транзакції;
- вкладені запити;
- уявлення;
- посилальна цілісність - зовнішні ключі;
- складні блокування;
- типи, визначені користувачем;
- спадковість;
- правила;
- перевірка сумісності версій.

Починаючи з версії 6.5 PostgreSQL є вельми стійка система, кожна наступна версія проходить процедуру регресивного тестування, що забезпечує стабільність.

В ході експлуатації PostgreSQL проявила себе як СУБД, яка заслуговує довіри. Кожна версія перевіряється дуже ретельно, бета-версії проходять як мінімум місячне тестування. Завдяки численній спільноті користувачів і відкритого доступу до вихідного коду помилки виправляються дуже швидко.

Продуктивність цієї СУБД також зростає від версії до версії, і останні атестації показують, що при певних умовах вона не поступається комерційним продуктам. Деякі системи, що володіють не таким повним набором можливостей, перевершують PostgreSQL в продуктивності, але за рахунок втрати функціональності.

Порівняно до інших проектів з відкритим кодом, такими як Apache, FreeBSD або MySQL, PostgreSQL не контролюється якоюсь однією компанією, її розробка можлива завдяки співпраці багатьох людей та компаній, які хочуть використовувати цю СУБД та впроваджувати у неї найновіші досягнення.

Однією з сильних сторін PostgreSQL є її архітектура. Як і багато комерційних СУБД, PostgreSQL може застосовуватися в середовищі клієнт-сервер, що дає масу переваг як користувачам, так і розробникам.

Основа PostgreSQL становить серверний процес бази даних. Він виконується на одному сервері. (В цій СУБД ще не реалізована технологія високої готовності, як у деяких інших комерційних системах рівня підприємства, які можуть розподіляти навантаження між декількома серверами, домагаючись таким чином додаткової масштабованості і стійкості до зовнішніх впливів).

Доступ з додатків до даних бази здійснюється за допомогою процесу бази даних. Клієнтські програми не можуть отримати доступ до даних самостійно, навіть якщо вони працюють на тому ж комп'ютері, на якому виконується серверний процес.

Такий поділ клієнтів і сервера дозволяє побудувати розподілену систему (Рисунок 3.1, аркуш). Можна відокремити клієнтів від сервера за допомогою мережі і розробляти клієнтські програми в середовищі, зручною для користувача. Наприклад, можна реалізувати базу даних під UNIX і створити клієнтські програми, які будуть працювати в системі Microsoft Windows.

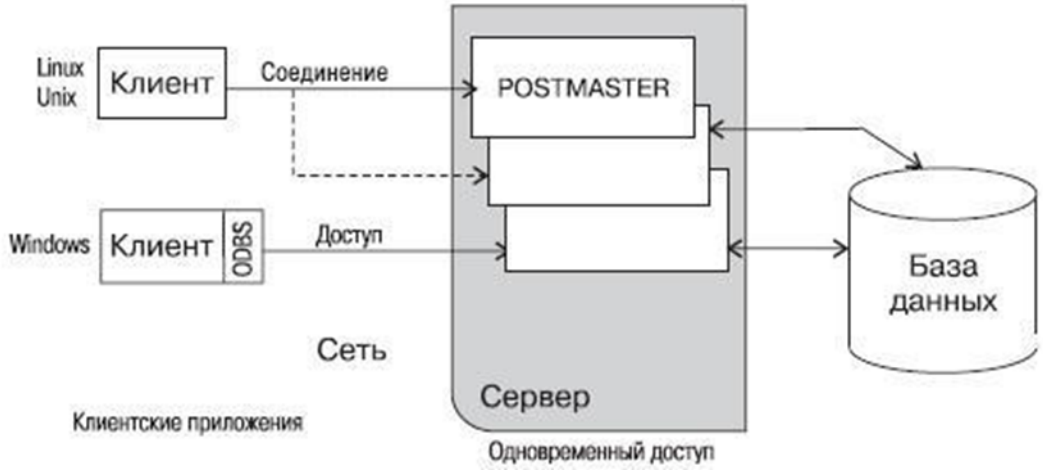

Рисунок 3.1 - Робота типового додатка PostgreSQL

Кілька клієнтів приєднуються до сервера по мережі. PostgreSQL орієнтований на протокол TCP / IP - це може бути локальна мережа або Інтернет. Кожен клієнт з'єднується з основним серверним процесом бази даних (на схемі - Postmaster), який створює новий серверний процес спеціально для обслуговування запитів на доступ до даних конкретного клієнта.

Завдяки тому, що маніпулювання даними зосереджено на сервері, СУБД не доводиться контролювати численних клієнтів, які отримують доступ в спільно використовуваний каталог сервера, і PostgreSQL може підтримувати цілісність даних навіть при одночасному доступі великої кількості користувачів.

Клієнтські програми з'єднуються з базою зі спеціального протоколу PostgreSQL. Однак можна встановити на стороні клієнта програмне забезпечення, яке надає стандартний інтерфейс для роботи потрібної програми, наприклад, за стандартом ODBC або JDBC. Доступність ODBC-

драйвера дозволяє застосовувати PostgreSQL в якості бази даних для багатьох існуючих додатків, включаючи такі продукти Microsoft Office, як Excel і Access.

Архітектура клієнт-сервер робить можливим розподіл праці. Машина-сервер добре підходить для зберігання і управління доступом до великих обсягів даних, вона може використовуватися як надійний репозитарій. Для клієнтів можуть бути розроблені складні графічні додатки. В якості альтернативи можна створити зовнішній інтерфейс на основі Інтернету, який надавав би доступ до даних і повертав результат у вигляді веб-сторінок в стандартний веб-браузер, при цьому не потрібно було б ніякого додаткового клієнтського програмного забезпечення.

Виходячи з фактів наведених вище, я вважаю, что PostgreSQL бесперечно має переваги серед інших СУБД.

4 СТВОРЕННЯ БАЗИ ДАНИХ

4.1 Створення таблиць

Створення послідовності typography_pk

Створення цієї послідовності дозволяє забезпечити унікальність первинного ключа друкарні.

Скрипт:

```
CREATE SEQUENCE typography_pk START 1;
```

Створення послідовності edition_pk

Створення цієї послідовності дозволяє забезпечити унікальність первинного ключа видавництва.

Скрипт:

```
CREATE SEQUENCE edition_pk START 1;
```

Створення послідовності client_pk

Створення цієї послідовності дозволяє забезпечити унікальність первинного ключа клієнта.

Скрипт:

```
CREATE SEQUENCE client_pk START 1;
```

Створення послідовності booking_pk

Створення цієї послідовності дозволяє забезпечити унікальність первинного ключа заказа.

Скрипт:

```
CREATE SEQUENCE booking_pk START 1;
```

Створення послідовності author_pk

Створення цієї послідовності дозволяє забезпечити унікальність первинного ключа автора.

Скрипт:

```
CREATE SEQUENCE author_pk START 1;
```

Створення домену positive

Створення цього домену дозволяє оголошувати атрибути таблиці, які можуть набувати великих значень та за умовою не можуть бути менше або рівні нулю.

Скрипт:

```
CREATE DOMAIN positive INT
    CHECK (value > 0);
```

Створення домену smallpositive

Створення цього домену дозволяє оголошувати атрибути таблиці, які за умовою не можуть бути менше або рівні нулю.

Скрипт:

```
CREATE DOMAIN smallpositive SMALLINT
    CHECK (value > 0);
```

Створення типу даних Address

Створення цього типу дозволяє зручніше зберігати адрес в базі даних та маніпулювати з ним.

Скрипт:

```
CREATE TYPE Address AS(
    City VARCHAR,
    Street VARCHAR,
    House smallpositive,
    Flat smallpositive
);
```

Створення таблиці Client

Створення таблиці Client дозволяє зручно зберігати дані про клієнта, а саме: телефон, факс, адреса, його ідентифікаційний номер.

Скрипт:

```
CREATE TABLE Client(
    client_id POSITIVE,
    address ADDRESS NOT NULL,
    telephone VARCHAR NOT NULL,
    fax VARCHAR,
    PRIMARY KEY(client_id)
);
```

Створення таблиці Individual

Створення таблиці Individual дозволяє уточнити дані про клієнта, а саме що він є фізичною особою, крім даних в таблиці Client додаються ще ПІБ та паспорт.

Скрипт:

```
CREATE TABLE Individual(
    pasport VARCHAR NOT NULL,
    full_name VARCHAR NOT NULL
) INHERITS (client);
```

Створення таблиці Entity

Створення таблиці Entity дозволяє уточнити дані про клієнта, а саме що він є юридичною особою, крім даних в таблиці Client додаються ще назва юридичної установи та її представник.

Скрипт:

```
CREATE TABLE Entity(
    representative VARCHAR NOT NULL,
    name VARCHAR NOT NULL
) INHERITS (client);
```

Створення таблиці Employee

Створення таблиці Employee дозволяє зручно зберігати дані про співробітника, а саме: ПІБ, зарплата (причому вона не може бути від'ємною), паспорт, посада.

Скрипт:

```
CREATE TABLE Employee(
    pasport VARCHAR,
    salary NUMERIC(7,2) NOT NULL,
    full_name VARCHAR NOT NULL,
    post VARCHAR NOT NULL,
    CHECK (salary > 0),
    PRIMARY KEY (pasport)
```

);

Створення таблиці Typography

Створення таблиці Typography дозволяє зручно зберігати дані про друкарню, а саме: назва, адреса, телефон, контактна особа, її ідентифікаційний номер.

Скрипт:

```
CREATE TABLE Typography(
    typography_id POSITIVE,
    name VARCHAR NOT NULL,
    address ADDRESS NOT NULL,
    telephone VARCHAR NOT NULL,
    contact_person VARCHAR,
    PRIMARY KEY (typography_id)
);
```

Створення таблиці Author

Створення таблиці Author дозволяє зручно зберігати дані про авторів, а саме: ПІБ, його інтереси, його ідентифікаційний номер.

Скрипт:

```
CREATE TABLE Author(
    author_id POSITIVE,
    full_name VARCHAR NOT NULL,
    note TEXT,
    PRIMARY KEY (author_id)
);
```

Створення таблиці Booking

Створення таблиці Booking дозволяє зручно зберігати дані про заказ, а саме: ідентифікаційний номер клієнта, ідентифікаційний номер друкарні (яка виконує цей заказ), дата створення заказу, стан заказу, дата виконання заказу, паспорт співробітника (який виконує цей заказ), його ідентифікаційний номер.

Скрипт:

```
CREATE TABLE Booking(
    booking_id POSITIVE,
    client_id POSITIVE NOT NULL,
    typography_id POSITIVE NOT NULL,
    booking_date DATE NOT NULL,
    check_mark BIT(1) NOT NULL,
    realization_date DATE,
    responsible_id VARCHAR NOT NULL,
    PRIMARY KEY (booking_id),
    FOREIGN KEY (client_id) REFERENCES client(client_id)
    ON UPDATE CASCADE ON DELETE CASCADE,
    FOREIGN KEY (typography_id)
    REFERENCES typography(typography_id)
    ON UPDATE CASCADE ON DELETE CASCADE,
    FOREIGN KEY (responsible_id) REFERENCES employee(pasport)
    ON UPDATE CASCADE ON DELETE CASCADE
);
```

Створення таблиці Edition

Створення таблиці Edition дозволяє зручно зберігати дані про видання, а саме: об’єм сторінок, тираж, назва, тип видання, ідентифікаційний номер видання, ідентифікаційний номер заказу.

Скрипт:

```
CREATE TABLE Edition(
    edition_id POSITIVE,
    booking_id POSITIVE,
    page_size SMALLPOSITIVE NOT NULL,
    printing POSITIVE NOT NULL,
    name VARCHAR NOT NULL,
    edition_type VARCHAR NOT NULL,
    PRIMARY KEY (edition_id),
    FOREIGN KEY (booking_id) REFERENCES booking(booking_id)
    ON UPDATE CASCADE ON DELETE CASCADE
);
```

Створення таблиці Edition_Author

Створення таблиці Edition_Author дозволяє встановити зв’язок між виданням та його авторами.

Скрипт:

```
CREATE TABLE Edition_Author(
    edition_id POSITIVE,
    author_id POSITIVE,
    PRIMARY KEY (edition_id, author_id),
    FOREIGN KEY (edition_id) REFERENCES edition(edition_id)
    ON UPDATE CASCADE ON DELETE CASCADE,
    FOREIGN KEY (author_id) REFERENCES author(author_id)
    ON UPDATE CASCADE ON DELETE CASCADE
```

);

4.2 Створення представлень

Створення представлення Individual_Manager (рисунок 4.1).

Створення представлення, яке полегшує взаємодію між клієнтом (фізичною особою) та менеджером. Клієнт може відстежити терміни виконання замовлення, виконавця замовлення, чи правильно вказані його особисті дані. Менеджер може відстежити який у нього клієнт і замовлення, скільки у нього є ще часу для виконання замовлення.

Скрипт:

CREATE VIEW Individual_Manager("Client full name", telephone, fax, address,"Number of Order ", "Booking Date", "Realization Date","Employee full name", post)

AS

SELECT I.full_name, I.telephone, I.fax, I.address, B.booking_id, B.booking_date, B.realization_date, E.full_name, E.post FROM Individual I, Booking B, Employee E

WHERE I.client_id = B.client_id AND B.responsible_id = E.pasport

AND B.check_mark = '0'

ORDER BY B.realization_date;

Client full name character varying	telephone character varying	fax character varying	address address	Number of Order positive	Booking Date date	Realization Date date	Employee full name character varying	post character varying
Maznov P.K.	067 555 45 78	77 68 67	(Odessa,Kanatna,45,5)	12	2018-12-08	2018-12-27	Reshebnikova S.E.	Booking Manager
Luznev A.K.	067 545 56 78	77 68 51	(Slavuta,"Klara Cetkina",3,)	8	2018-12-08	2019-07-05	Nortilova K.L.	General Manager
Dolgov L.N.	067 225 56 78	77 60 56	(Kiev,Shevchenko,5,7)	4	2018-12-08	2019-10-09	Gorbachev M.A.	Manager

Рисунок 4.1 – Представлення Individual_Manager

Створення представлення Entity_Manager аналогічне(Додаток А).

Створення представлення Author_Edition (рисунок 4.2).

Створення представлення, яке полегшує взаємодію між авторами та менеджером. Автор може відстежити яке у нього замовлення, тип замовлення, щоб було зрозуміло як далі працювати. Менеджер може відстежити які автора відповідають за потрібне йому замовлення.

Скрипт:

```
CREATE VIEW Author_Edition("Author full name", "Author Note","Number of Order", "Edition Name", "Edition Type") AS
SELECT A.full_name, A.note, E.booking_id, E.name, E.edition_type
FROM Author A, Edition E, Edition_Author AE
WHERE A.author_id = AE.author_id AND E.edition_id = AE.edition_id AND E.booking_id = ANY(
SELECT booking_id FROM booking WHERE check_mark = '0')
ORDER BY A.full_name;
```

Author full name character varying	Author Note text	Number of Order positive	Edition Name character varying	Edition Type character varying
Avtor D.H.	Promotion	12	Maznov - card	Business card
Avtor D.H.	Promotion	3	Charodijka - card	Business card
Avtor D.H.	Promotion	3	Charodijka - advertising	Poster
Avtor D.H.	Promotion	3	Charodijka - journal	Advertising
Avtor D.H.	Promotion	5	Ferma - journal	Advertising
Borisov F.S.	Computer Science	4	Computer Science	Conspect

Рисунок 4.2 – Представлення Author_Edition

Створення представлення Order_Typography (рисунок 4.3, аркуш 26).

Створення представлення, яке полегшує роботу менеджеру і друкарні. Друкарня може відстежувати які в них замовлення і в разі затримок вони зможуть повідомити менеджерів відповідальних за ці замовлення. У свою чергу менеджер може відстежити які друкарні відповідають за його закази. В разі затримки або зміни терміну виконання замовлення, він зможе зв'язатися з друкарнею для уточнення нових деталей.

Скрипт:

CREATE VIEW Order_Typography("Number of Order","Booking Date", "Realization Date", "Employee full name", post, "Typography name","Typography address", "Typography telephone","Contact person")

AS

SELECT B.booking_id, B.booking_date, B.realization_date, E.full_name, E.post, T.name, T.address, T.telephone, T.contact_person

FROM Typography T, Booking B, Employee E

WHERE T.typography_id = B.typography_id AND B.responsible_id = E.pasport

AND B.check_mark = '0'

ORDER BY B.realization_date;

Number of Order positive	Booking Date date	Realization Date date	Employee full name character varying	post character varying	Typography name character varying	Typography address address	Typography telephone character varying	Contact person character varying
12	2018-12-08	2018-12-27	Reshebnikova S.E.	Booking Manager	Pechatalka	(Odessa,"Dnepropetrovskaja doroga",123,)	097 846 37 57	Imbers D.F.
5	2018-12-08	2019-03-12	Gudneva K.O.	Manager	Good	(Kiev,Obolonskaja,24,)	050 385 28 58	Luckov E.R.
3	2018-12-08	2019-06-30	Glavachev S.D.	SE Manager	Good	(Kiev,Obolonskaja,24,)	050 385 28 58	Luckov E.R.
8	2018-12-08	2019-07-05	Nortilova K.L.	General Manager	Slavut Pechat	(Slavuta,Dokovaja,5,)	028 589 59 12	Trushilov T.O.
1	2018-12-08	2019-09-30	Serpunov I.B.	Manager	Pechatalka	(Odessa,"Dnepropetrovskaja doroga",123,)	097 846 37 57	Imbers D.F.
4	2018-12-08	2019-10-09	Gorbachev M.A.	Manager	Good	(Kiev,Obolonskaja,24,)	050 385 28 58	Luckov E.R.
11	2018-12-08	2019-12-28	Nortilova K.L.	General Manager	Pechatalka	(Odessa,"Dnepropetrovskaja doroga",123,)	097 846 37 57	Imbers D.F.

Рисунок 4.3 - Представлення Order_Typography

4.3 Створення тригерів

Створення тригерної функції check_client.

Створення цієї тригерної функції надає можливість уникати однакових client_id в таблиці client при додаванні нових клієнтів в таблицю individual або в таблицю entity. Також в цьому тригері здійснений механізм роботи з таблицею client, яка є батьком для таблиць individual і entity. Всі дані додані в таблицю individual або в таблицю entity відображаються в таблиці client, але тільки на візуальному рівні - нема на фізичному. Тому при використанні атрибута таблиці client - client_id в інших таблицях виникають проблеми. У цьому тригері прописується механізм додавання

(поновлення) в таблицю(-і) client даних, які були додані (оновлені) в таблиці individual або в таблиці entity.

Скрипт:

```
CREATE OR REPLACE FUNCTION check_client() RETURNS TRIGGER
AS $$
DECLARE client_id client.client_id%TYPE;
BEGIN
 IF (TG_OP = 'UPDATE') THEN
   IF(new.client_id <> old.client_id) THEN
   SELECT C.client_id INTO client_id FROM ONLY Client C
   WHERE C.client_id = new.client_id;
    IF FOUND THEN
    RAISE EXCEPTION 'Duplicate client_id!';
    END IF;
   END IF;
  UPDATE ONLY client SET (client_id, address, telephone, fax) = (new.client_id, new.address, new.telephone, new.fax)
   WHERE client.client_id = old.client_id;
   RAISE NOTICE 'Successfully updated!';
   RETURN NEW;
  END IF;
  SELECT C.client_id INTO client_id FROM ONLY Client C
  WHERE C.client_id = new.client_id;
  IF NOT FOUND THEN
   INSERT INTO client Values(new.client_id, new.address, new.telephone, new.fax);
   RAISE NOTICE 'Successfully added!';
   RETURN NEW;
```

```
    ELSE
    RAISE EXCEPTION 'Duplicate client_id!';
   END IF;
  END;
  $$ LANGUAGE plpgsql;
```

Створення тригера pk_entity на таблицю entity, який перед операцією INSERT або UPDATE виконує функцію check_client.

Скрипт:

```
CREATE TRIGGER pk_entity
BEFORE INSERT OR UPDATE ON entity
FOR EACH ROW EXECUTE PROCEDURE check_client();
```

Створення тригера pk_individual аналогічне (Додаток Б).

Створення тригерної функції check_author.

Створення цієї тригерної функції дозволяє уникнути додавання (зміни) первинного ключа такого ж (на такий же), який вже є в таблиці author. У разі такої помилки буде виведено повідомлення про те що ви ввели ключ, який вже існує.

Скрипт:

```
CREATE OR REPLACE FUNCTION check_author() RETURNS TRIGGER
AS $$
DECLARE author_id author.author_id%TYPE;
BEGIN
  IF (TG_OP = 'UPDATE') THEN
    IF(new.author_id <> old.author_id) THEN
```

```
        SELECT A.author_id INTO author_id FROM ONLY Author A
        WHERE A.author_id = new.author_id;
        IF FOUND THEN
        RAISE EXCEPTION 'Duplicate author_id!';
        END IF;
      END IF;
      RAISE NOTICE 'Author % successfully updated!',old.full_name;
      RETURN NEW;
    END IF;
    SELECT A.author_id INTO author_id FROM author A
    WHERE A.author_id = new.author_id;
    IF NOT FOUND THEN
      RAISE NOTICE 'Author % successfully added!',new.full_name;
      RETURN NEW;
      ELSE
      RAISE EXCEPTION 'Duplicate author_id!';
    END IF;
  END;
  $$ LANGUAGE plpgsql;
```

Створення тригера pk_author на таблицю author, який перед операцією INSERT або UPDATE виконує функцію check_ author.

Скрипт:

```
CREATE TRIGGER pk_author
BEFORE INSERT OR UPDATE
ON author
FOR EACH ROW
EXECUTE PROCEDURE public.check_author();
```

Створення тригерних функції на перевірку та їх тригерів для інших таблиць аналогічне (Додаток Б).

Створення тригерної функції delete_client.

Створення цієї тригерної функції дозволяє коректно видаляти дані в батьківській таблиці client при видаленні даних в дочірніх таблицях individual і entity. У зв'язку з штучним додаванням в таблицю client при додаванні даних в таблицях individual і entity потрібно прописувати і штучне видалення.

Скрипт:

```
CREATE OR REPLACE FUNCTION delete_client() RETURNS TRIGGER
AS $$
BEGIN
   DELETE FROM ONLY client WHERE client_id = old.client_id;
   RETURN OLD;
END;
$$ LANGUAGE plpgsql;
```

Створення тригера delete_entity на таблицю entity, який перед операцією DELETE виконує функцію delete_client.

Скрипт:

```
CREATE TRIGGER delete_entity
BEFORE DELETE ON entity
FOR EACH ROW EXECUTE PROCEDURE delete_client();
```

Створення тригера delete_individual аналогічне (Додаток Б).

Створення тригерної функції delete_employee.

Створення цієї тригерної функції дозволяє в разі звільнення працівника всі замовлення, які були на ньому, розподілити по іншим працівникам.

Скрипт:

```
CREATE OR REPLACE FUNCTION delete_employee() RETURNS TRIGGER
AS $$
DECLARE pasport employee.pasport%TYPE;
BEGIN
SELECT E.pasport INTO pasport FROM employee E
WHERE E.pasport <> old.pasport;

IF FOUND THEN
UPDATE booking SET responsible_id = pasport
WHERE responsible_id = old.pasport;
END IF;

RETURN OLD;
END;
$$ LANGUAGE plpgsql;
```

Створення тригера del_employee на таблицю employee, який перед операцією DELETE виконує функцію delete_employee

Скрипт:

```
CREATE TRIGGER del_employee
BEFORE DELETE ON employee
FOR EACH ROW EXECUTE PROCEDURE public.delete_employee();
```

Створення тригерної функції delete_typography та тригера del_typography аналогічне(Додаток Б).

4.4 Створення збережених процедур (функцій).

Створення функції insert_client.

Створення цієї функції дозволяє полегшити менеджеру заповнення анкети клієнта. Менеджеру не потрібно придумувати унікальний первинний ключ, за нього все зробить програма.

Скрипт:

```
CREATE OR REPLACE FUNCTION insert_client(char(1),City VARCHAR, Street VARCHAR,House int,Flat int,telephone varchar, fax varchar,varchar,varchar) RETURNS INT
AS $$
   DECLARE client_id client.client_id%TYPE;
   DECLARE address client.address%TYPE;
BEGIN
IF (House <= 0 OR Flat <= 0) THEN
RAISE EXCEPTION 'Flat(House) cannot be negative!';
END IF;

IF ($1 <> 'E' AND $1 <> 'I') THEN
RAISE EXCEPTION 'Please, select Entity(E) or Individual(I)!';
END IF;

client_id := nextval('client_pk');
address := ROW(City, Street, House, Flat);
```

```
IF($1 = 'E') THEN
INSERT INTO entity  VALUES (client_id,address,$6,$7,$8,$9);
ELSE
INSERT INTO individual VALUES (client_id,address,$6,$7,$8,$9);
END IF;
RAISE NOTICE 'Client %, successfully added!', $9;

RETURN 1;
END;
$$ LANGUAGE plpgsql;
```

Аналгичні функціі для таблиць typography, author, booking (Додаток В).

Створення функції insert_ edition.

Створення цієї функції дозволяє полегшити роботу менеджеру, тим що створюючи нове видання в функцію передається інформація про авторів цього видання. Функція дивиться, чи є такі автора в таблиці author, якщо є, то функція автоматично створює зв'язок між цим виданням і його авторами в таблиці edition_author. Якщо немає, то функція створює в таблиці author нових авторів і прописує зв'язок між цими авторами і виданням в таблиці edition_author.

Скрипт:

```
CREATE OR REPLACE FUNCTION insert_edition(booking_id int, page_size int, printing int,name varchar, edition_type varchar, authors varchar [][])
RETURNS INT
```

```
AS $$
DECLARE
   rec RECORD;
   i int;
   p positive;
BEGIN

IF (page_size <= 0 OR printing <= 0 OR booking_id <= 0) THEN
RAISE EXCEPTION 'Page_size(printing, booking_id) cannot be negative!';
END IF;

SELECT INTO rec FROM booking B
WHERE B.booking_id = insert_edition.booking_id;
IF NOT FOUND THEN
RAISE EXCEPTION 'booking_id not found!';
END IF;

INSERT INTO edition  VALUES ( nextval('edition_pk'), booking_id, page_size, printing, name, edition_type);
RAISE NOTICE 'Edition %, successfully added!', name;

FOR i IN 1..(cardinality(authors)/2) LOOP

SELECT A.author_id INTO p FROM author A
WHERE A.full_name = insert_edition.authors[i][1]
AND A.note = insert_edition.authors[i][2];
IF NOT FOUND THEN
PERFORM insert_author(authors[i][1],authors[i][2]);
```

```
RAISE NOTICE 'Author %, successfully added!', authors[i][1];
p := currval('author_pk');
END IF;
INSERT INTO edition_author VALUES(currval('edition_pk'),p);

END LOOP;

RETURN 1;
END;
$$ LANGUAGE plpgsql;
```

Створення функції completed.

Створення цієї функції дозволяє менеджеру, передати свої дані і номер замовлення яке він виконав. Після виконання функції, в базі замовлення буде зберігається як виконаний, з поточною датою.

Скрипт:

```
CREATE OR REPLACE FUNCTION completed(booking_id int, responsible_id varchar)
RETURNS INT
AS $$
DECLARE
rec RECORD;
BEGIN

SELECT INTO rec FROM booking B
WHERE B.booking_id = insert_edition.booking_id;
IF NOT FOUND THEN
RAISE EXCEPTION 'booking_id not found!';
```

```
END IF;

SELECT INTO rec FROM employee E
WHERE E.pasport = completed.responsible_id;
IF NOT FOUND THEN
RAISE EXCEPTION 'responsible_id not found!';
END IF;

UPDATE booking SET check_mark = '1', realization_date = current_date
WHERE booking_id = insert_edition.booking_id
AND responsible_id = completed.responsible_id;

RETURN 1;
END;
$$ LANGUAGE plpgsql;
```

Створення функції increase_salary.

Створення цієї функції дозволяє директору зменшувати або збільшувати зарплату менеджера на конкретний відсоток.

Скрипт:

```
CREATE OR REPLACE FUNCTION increase_salary
(responsible_id varchar, percent numeric)
RETURNS INT

AS $$
DECLARE
rec RECORD;
BEGIN
```

```
SELECT INTO rec FROM Employee E
WHERE E.pasport = responsible_id;
IF NOT FOUND THEN
RAISE EXCEPTION 'Employee not found!';
END IF;

UPDATE Employee SET salary = salary + salary * (percent / 100)
WHERE pasport = responsible_id;

RETURN 1;
END;
$$ LANGUAGE plpgsql;
```

5 МАНІПУЛЮВАННЯ ДАНИМИ

5.1 Оператори відновлення

Заповнення клієнтської бази.

Додавання в клієнтську базу фізичної особи Petrov P. N.

Використовується функція insert_client.

Скрипт:

SELECT insert_client('I', 'Odessa', 'Kosmonavtov', 16, 8, '0675555678', '776856', 'СК 1236489', 'Petrov P.N.');

Додавання в клієнтську базу юридичної особи «Prom».

Використовується функція insert_client.

Скрипт:

SELECT insert_client('E', 'Odessa', 'Dumskaja', 1, 1, '0674565678', '778956', 'Kranov O.L.', 'Prom');

Додавання інших фізичних і юридичних осіб аналогічне (Додаток Д).

Для того щоб переглянути вміст таблиці Individual використовується запит:

SELECT *FROM individual;

Результат запиту зображен на рисунку 5.1.

client_id integer	address address	telephone character varying	fax character varying	pasport character varying	full_name character varying
1	(Odessa,Kosmonavtov,16,8)	067 555 56 78	77 68 56	СК 1236489	Petrov P.N.
3	(Odessa,Kanatna,45,5)	067 555 45 78	77 68 67	1234748	Maznov P.K.
5	(Kiev,Shevchenko,5,7)	067 225 56 78	77 60 56	РН 4494830	Dolgov L.N.
7	(Slavuta,Revolucij,111,)	050 554 56 78	45 68 56	3784930	Ivanov P.B.
9	(Slavuta,"Klara Cetkina",3,)	067 545 56 78	77 68 51	КН 4843209	Luznev A.K.

Рисунок 5.1 – Вміст таблиці Individual

Для того щоб переглянути вміст таблиці Entity використовується запит:

SELECT *FROM Entity;

Результат запиту зображен на рисунку 5.2.

client_id integer	address address	telephone character varying	fax character varying	representative character varying	name character varying
2	(Odessa,Dumskaja,1,1)	067 456 56 78	77 89 56	Kranov O.L.	Prom
4	(Kiev,Hreschatik,56,7)	067 555 56 71	74 68 56	Pulnev F.K.	Charodijka
6	(Kiev,Hmelnickogo,13,3)	067 245 56 78	77 68 90	Lomonov G.T.	Ferma
8	(Slavuta,Hmelnickogo,56,7)	067 555 50 78	77 68 89	Rezcov V.N.	Dva Shaga
10	(Odessa,Kanatna,100,7)	067 777 56 78	77 99 56	Rudnev K.G.	Rud

Рисунок 5.2 – Вміст таблиці Entity

Заповнення бази співробітників.

За допомогою оператора INSERT додамо до таблиці Employee наступних осіб: Petrova P.P., Acermanov B.M., Ivanova N.A., Reshebnikova S.E., Wenedrov E.R., Glavachev S.D., Gorbachev M.A., Nortilova K.L., Gudneva K.O., Serpunov I.B.

Скрипт:

```
INSERT INTO employee(pasport, salary, full_name, post)
VALUES
('127718', 12000.5, 'Petrova P.P.', 'Manager'),
('KN 48993', 35000, 'Acermanov B.M.', 'Director'),
('393034', 12000.5, 'Ivanova N.A.', 'Manager'),
('KN 48990', 20000, 'Reshebnikova S.E.', 'Booking Manager'),
('FE 38290', 12000.5, 'Wenedrov E.R.', 'Manager'),
('383902', 10000, 'Glavachev S.D.', 'SE Manager'),
('ER 38291', 12000.5, 'Gorbachev M.A.', 'Manager'),
('284028', 25000, 'Nortilova K.L.', 'General Manager'),
('DE 18340', 12000.5, 'Gudneva K.O.', 'Manager'),
('273920', 12000.5, 'Serpunov I.B.', 'Manager');
```

Для того щоб переглянути вміст таблиці Employee використовується запит:

SELECT *FROM Employee;

Результат запиту зображен на рисунку 5.3.

pasport character varying	salary numeric(7,2)	full_name character varying	post character varying
127718	12000.50	Petrova P.P.	Manager
KN 48993	35000.00	Acermanov B.M.	Director
393034	12000.50	Ivanova N.A.	Manager
KN 48990	20000.00	Reshebnikova S.E.	Booking Manager
FE 38290	12000.50	Wenedrov E.R.	Manager
383902	10000.00	Glavachev S.D.	SE Manager
ER 38291	12000.50	Gorbachev M.A.	Manager
284028	25000.00	Nortilova K.L.	General Manager
DE 18340	12000.50	Gudneva K.O.	Manager
273920	12000.50	Serpunov I.B.	Manager

Рисунок 5.3 - Вміст таблиці Employee

Заповнення бази друкарень.

Додавання в базу друкарень друкарню «Pechatalka».

Використовується функція insert_typography.

Скрипт:

SELECT insert_typography('Pechatalka', 'Odessa', 'Dnepropetrovskaja doroga', 123, NULL, '0978463757', 'Imbers D.F.');

Додавання інших друкарень аналогічне (Додаток Д).

Для того щоб переглянути вміст таблиці Typography використовується запит:

SELECT *FROM Typography;

Результат запиту зображен на рисунку 5.4.

typography_id integer	name character varying	address address	telephone character varying	contact_person character varying
1	Pechatalka	(Odessa,"Dnepropetrovskaja doroga",123,)	097 846 37 57	Imbers D.F.
2	Good	(Kiev,Obolonskaja,24,)	050 385 28 58	Luckov E.R.
3	Slavut Pechat	(Slavuta,Dokovaja,5,)	028 589 59 12	Trushilov T.O.

Рисунок 5.4 - Вміст таблиці Typography

Заповнення бази авторів.

Додавання в базу авторів автора Djonson B. .

Використовується функція insert_author.

Скрипт:

SELECT insert_author('Djonson B.','Philosophy');

Додавання інших авторів аналогічне (Додаток Д).

Для того щоб переглянути вміст таблиці author використовується запит:

SELECT *FROM author;

Результат запиту зображен на рисунку 5.5.

author_id integer	full_name character varying	note text
1	Djonson B.	Philosophy
2	Borisov F.S.	Computer Science
3	Poblajko E.	Math
4	Srednik J.D.	Native language
5	Guliven R.T.	Food Technology
6	Fridsun A.I.	Web Design
7	Bednik D.K.	Food Technology
8	Avtor D.H.	Promotion
9	Nonej J.	AI
10	Sandro S.A.	Food Technology

Рисунок 5.5 - Вміст таблиці author

Заповнення бази замовлень.

Додавання в базу замовлень другого замовлення клієнта з індифікаціїним номером 2.

Використовується функція insert_booking.

Скрипт:

SELECT insert_booking(2, 1, '30 Sep 2019', '273920');

Додавання інших замовлень аналогічне (Додаток Д).

Для того щоб переглянути вміст таблиці booking використовується запит:

SELECT *FROM booking;

Результат запиту зображен на рисунку 5.6.

booking_id integer	client_id integer	typography_id integer	booking_date date	check_mark bit(1)	realization_date date	responsible_id character varying
1	2	1	2018-12-08	0	2019-09-30	273920
3	4	2	2018-12-08	0	2019-06-30	383902
4	5	2	2018-12-08	0	2019-10-09	ER 38291
5	6	2	2018-12-08	0	2019-03-12	DE 18340
8	9	3	2018-12-08	0	2019-07-05	284028
12	3	1	2018-12-08	0	2018-12-27	KN 48990
11	2	1	2018-12-08	0	2019-12-28	284028
2	3	1	2018-12-08	1	2018-12-09	393034
6	7	3	2018-12-08	1	2018-12-09	393034
7	8	3	2018-12-08	1	2018-12-09	KN 48990
10	1	1	2018-12-08	1	2018-12-09	KN 48990
9	10	1	2018-12-08	1	2018-12-09	127718

Рисунок 5.6 - Вміст таблиці booking

Заповнення бази видань.

Додавання в базу видань видання «Prom – advertising #1» для замовлення №1.

Викорустовується функція insert_edition.

Скрипт:

SELECT insert_edition(1, 10, 100, 'Prom – advertising #1', 'Advertising', '{{"Fridsun A.I.", "Web Design"}}');

Додавання інших замовлень аналогічне (Додаток Д).

Для того щоб переглянути вміст таблиці edition використовується запит:

SELECT *FROM edition;

Результат запиту зображен на рисунку 5.7.

edition_id integer	booking_id integer	page_size smallint	printing integer	name character varying	edition_type character varying
1	1	10	100	Prom - advertising #1	Advertising
2	1	1	23	Prom - advertising #2	Poster
3	2	50	57	Cook Book	Book
4	3	1	80	Charodijka - card	Business card
5	3	1	98	Charodijka - advertising	Poster
6	3	24	3455	Charodijka - journal	Advertising
7	4	23	36	Computer Science	Conspect
8	5	20	200	Ferma - journal	Advertising
9	6	30	77	Fourier series	Conspect
10	7	1	30	Dva shaga - card	Business card
11	7	1	85	Dva shaga - advertising	Poster
12	8	587	1	AI	Manual
13	9	1	90	Rud - advertising	Poster
14	10	7	4	National Food	Book
15	11	9	4656	Prom - advertising #3	Journal
16	11	1	365	Prom - card	Business card
17	12	600	1	Philosophy	Book
18	12	1	24	Maznov - card	Business card

Рисунок 5.7 - Вміст таблиці edition

Виконування замовлень

Виконування замовлення №2.

Використовується функція completed.

Скрипт:

```
SELECT completed(2,'393034');
```

Виконування інших замовлень аналогічне (Додаток Д).

5.2 Оператори вибірки

Завантаженість другарень

Щоб рівномірно розподіляти навантаження і ефективно працювати потрібно знати де знаходиться кожна друкарня і скільки на ній зараз замовлень.

Скрипт:

SELECT T.name, T.address, T.telephone, T.contact_person, count(B.typography_id) Count_of_orders FROM typography T, booking B

WHERE B.check_mark = '0' AND T.typography_id = B.typography_id

GROUP BY T.name, T.address, T.telephone, T.contact_person

ORDER BY count(B.typography_id) DESC;

Результат цього запиту зображено на рисунку 5.8.

name character varying	address address	telephone character varying	contact_person character varying	count_of_orders bigint
Good	(Kiev,Obolonskaja,24,)	050 385 28 58	Luckov E.R.	3
Pechatalka	(Odessa,"Dnepropetrovskaja doroga",123,)	097 846 37 57	Imbers D.F.	3
Slavut Pechat	(Slavuta,Dokovaja,5,)	028 589 59 12	Trushilov T.O.	1

Рисунок 5.8 - Завантаженість друкарень

Список поточних невиконаних замовлень і їх відповідальних.

Директорові треба знати, які закази слід раніше виконати та хто за них відповідає.

Скрипт:

SELECT E.full_name, E.post, B.booking_id, B.booking_date, B.realization_date

FROM Employee E, Booking B

WHERE E.pasport = B.responsible_id AND B.check_mark = '0'

GROUP BY E.full_name, E.post, B.booking_id, B.booking_date, B.realization_date

ORDER BY B.realization_date;

Результат цього запиту зображено на рисунку 5.9.

full_name character varying	post character varying	booking_id integer	booking_date date	realization_date date
Reshebnikova S.E.	Booking Manager	12	2018-12-08	2018-12-27
Gudneva K.O.	Manager	5	2018-12-08	2019-03-12
Glavachev S.D.	SE Manager	3	2018-12-08	2019-06-30
Nortilova K.L.	General Manager	8	2018-12-08	2019-07-05
Serpunov I.B.	Manager	1	2018-12-08	2019-09-30
Gorbachev M.A.	Manager	4	2018-12-08	2019-10-09
Nortilova K.L.	General Manager	11	2018-12-08	2019-12-28

Рисунок 5.9 - Список поточних невиконаних замовлень і їх відповідальних

Визначити всі замовлення в Одесі.

Працівникам потрібно знати всі закази в Одесі щоб росподілити їх до найближчих друкарень та щоб представник компанії в Одесі бачив куди яке замовлення доставляти.

Скрипт:

```
SELECT DISTINCT C.address Client_address, C.telephone Client_telephone, B.booking_date, B.realization_date
FROM Client C, Booking B
WHERE C.client_id = B.client_id AND B.typography_id IN(
SELECT typography_id FROM typography
WHERE (address).city = 'Odessa')
ORDER BY B.realization_date;
```

Результат цього запиту зображено на рисунку 5.10.

client_address address	client_telephone character varying	booking_date date	realization_date date
(Odessa,Kanatna,45,5)	067 555 45 78	2018-12-08	2018-12-09
(Odessa,Kanatna,100,7)	067 777 56 78	2018-12-08	2018-12-09
(Odessa,Kosmonavtov,16,8)	067 555 56 78	2018-12-08	2018-12-09
(Odessa,Kanatna,45,5)	067 555 45 78	2018-12-08	2018-12-27
(Odessa,Dumskaja,1,1)	067 456 56 78	2018-12-08	2019-09-30
(Odessa,Dumskaja,1,1)	067 456 56 78	2018-12-08	2019-12-28

Рисунок 5.10 - Всі замовлення в Одесі

Завантаженість авторів.

Щоб рівномірно розподіляти навантаження і ефективно працювати потрібно знати в якій області працює автор та скільки на ньому зараз замовлень.

Скрипт:

```
SELECT full_name, note, count(AE.author_id) "Count of orders" FROM author A, edition_author AE
WHERE A.author_id = AE.author_id AND AE.edition_id = ANY(
SELECT edition_id FROM edition
WHERE booking_id IN(
```

```
SELECT booking_id FROM booking
WHERE check_mark = '0'))
GROUP BY full_name, note
ORDER BY count(AE.author_id);
```

Результат цього запиту зображено на рисунку 5.11.

full_name character varying	note text	Count of orders bigint
Nonej J.	AI	1
Djonson B.	Philosophy	1
Borisov F.S.	Computer Science	1
Fridsun A.I.	Web Design	4
Avtor D.H.	Promotion	5

Рисунок 5.11 - Завантаженісь авторів

Найоб'ємніші замовлення.

Найоб'ємніші замовлення – це ті замовлення в яких добуток тиражу на кількість сторінок більше середнього. Треба знати які саме це замовлення, бо на них треба найбільше часу. Отже знаючи ці замовлення, можна більш ефективніше розподілити працю.

Скрипт:

```
SELECT booking_id, page_size, printing, name, edition_type
FROM edition WHERE (page_size * printing) > (
SELECT AVG(page_size * printing) FROM edition) AND booking_id IN(
SELECT booking_id FROM booking
WHERE check_mark = '0')
ORDER BY (page_size * printing) DESC;
```

Результат цього запиту зображено на рисунку 5.12.

booking_id integer	page_size smallint	printing integer	name character varying	edition_type character varying
3	24	3455	Charodijka - journal	Advertising
11	9	4656	Prom - advertising #3	Journal

Рисунок 5.12 - Найоб'ємніші замовлення

Завантаженість співробітників.

Щоб рівномірно розподіляти навантаження і ефективно працювати потрібно знати скільки на працівнику зараз замовлень.

Скрипт:

```
SELECT full_name, post, responsible_id, count(responsible_id) "Count of orders" FROM booking, employee
WHERE check_mark = '0' AND responsible_id = pasport
GROUP BY full_name, post, responsible_id
ORDER BY count(responsible_id);
```

Результат цього запиту зображено на рисунку 5.13.

full_name character varying	post character varying	responsible_id character varying	Count of orders bigint
Glavachev S.D.	SE Manager	383902	1
Serpunov I.B.	Manager	273920	1
Gorbachev M.A.	Manager	ER 38291	1
Reshebnikova S.E.	Booking Manager	KN 48990	1
Gudneva K.O.	Manager	DE 18340	1
Nortilova K.L.	General Manager	284028	2

Рисунок 5.13 - Завантаженість співробітників

Непрості менеджери.

Директорові треба знати, хто в нього за що відповідає. Як правило, для кожного типу менеджера своя праця. Отже знаючи хто за що відповідає, можна грамотно росподіляти працю, а вони в свою чергу росподіляють її між простими менеджерами.

Скрипт:

```
SELECT pasport, salary, full_name, post FROM employee
WHERE post LIKE '% Manager';
```

Результат цього запиту зображено на рисунку 5.14.

pasport character varying	salary numeric(7,2)	full_name character varying	post character varying
KN 48990	20000.00	Reshebnikova S.E.	Booking Manager
383902	10000.00	Glavachev S.D.	SE Manager
284028	25000.00	Nortilova K.L.	General Manager

Рисунок 5.14 - Непрості менеджери

Кількість заказів клієнта.

Треба знати скільки у клієнта заказів. По – перше, шоб його замовлення виконувалися швидше. По – друге, шоб видати йому карту постійного клієнта зі знижкою.

Скрипт:

```
SELECT address, telephone, fax, count(booking_id) "Count of orders"
FROM client C, booking B
WHERE C.client_id = B.client_id AND check_mark = '0'
GROUP BY address, telephone, fax
ORDER BY count(booking_id) DESC;
```

Результат цього запиту зображено на рисунку 5.15.

address address	telephone character varying	fax character varying	Count of orders bigint
(Odessa,Dumskaja,1,1)	067 456 56 78	77 89 56	4
(Kiev,Hmelnickogo,13,3)	067 245 56 78	77 68 90	2
(Kiev,Hreschatik,56,7)	067 555 56 71	74 68 56	2
(Kiev,Shevchenko,5,7)	067 225 56 78	77 60 56	2
(Odessa,Kanatna,45,5)	067 555 45 78	77 68 67	2
(Slavuta,"Klara Cetkina",3,)	067 545 56 78	77 68 51	2

Рисунок 5.15 - Кількість заказів клієнта

Кількість заказів в кожнії категорії печатної продукції.

Необхідно знати скільки заказів в конкретній категорії печатної продукції, щоб розуміти яких авторів необхідно задіяти. Таким чином, зростає ефективність праці.

Скрипт:

```
SELECT edition_type, count(booking_id) "Count of orders"
FROM edition
WHERE booking_id IN(
SELECT booking_id FROM booking
WHERE check_mark = '0')
GROUP BY edition_type
```

ORDER BY count(booking_id) DESC;

Результат цього запиту зображено на рисунку 5.16.

edition_type character varying	Count of orders bigint
Business card	3
Advertising	3
Poster	2
Conspect	1
Journal	1
Manual	1
Book	1

Рисунок 5.16 - Кількість заказів в кожній категорії печатної продукції

Кількість виконаних заказів кожним працівником.

Директорові необхідно знати хто більше за всіх працює, щоб можна було виписати якусь премію, або підвищити заробітню плату.

Скрипт:

SELECT full_name, post, responsible_id, count(responsible_id) "Count of done orders"

FROM booking, employee

WHERE check_mark = '1' AND responsible_id = pasport

GROUP BY full_name, post, responsible_id

ORDER BY count(responsible_id) DESC;

Результат цього запиту зображено на рисунку 5.17.

full_name character varying	post character varying	responsible_id character varying	Count of done orders bigint
Ivanova N.A.	Manager	393034	2
Reshebnikova S.E.	Booking Manager	KN 48990	2
Petrova P.P.	Manager	127718	1

Рисунок 5.17 - Кількість виконаних заказів кожним працівником

6 СТВОРЕННЯ КОРИСТУВАЧІВ І ПРИЗНАЧЕННЯ ПРАВ ДОСТУПУ

В нашій базі даних необхідно чотири користувача, а саме: менеджер, фізична особа, юридична особа, директор.

Менеджер повинен бути саме користувачем – фізичним об’єктом.

Скрипт створення ролі manager:

```
CREATE ROLE manager WITH LOGIN PASSWORD '11111' VALID UNTIL '2019-12-08';
```

Треба надати менеджеру право на підключення до нашої бази даних.

Скрипт:

```
GRANT CONNECT ON DATABASE "CourseWork" TO manager;
```

Менеджер повинен бачити усі таблиці у базі. Також він має право передавати це право ще комусь.

Скипт дозвілу на перекляд усієї бази:

```
GRANT SELECT ON ALL TABLES IN SCHEMA public TO manager WITH GRANT OPTION;
```

Менеджер може модифікувати усі таблиці, окрім таблиці client та таблиці edition_author, бо вони є допоміжними таблицями для зв’язку інших таблиць один з одним.

Скрипт:

```
GRANT INSERT, UPDATE, DELETE
ON author,booking,edition,employee,entity,individual,typography
TO manager WITH GRANT OPTION;
```

Менеджер не може всталяти нові видання. Для цього є спеціальна функція insert_edition.

Скрипт:

REVOKE INSERT ON edition FROM manager;

Менеджер обов’язково повинен мати дозвіл на виконування усіх функції, бо всі вони якраз и створювалися для співпрацівників, окрім підвищення заробітної плати.

Скрипт:

GRANT EXECUTE

ON ALL FUNCTIONS IN SCHEMA public

TO manager WITH GRANT OPTION;

REVOKE EXECUTE

ON FUNCTION completed(int,varchar)

FROM manager;

Фізична особа повиненна бути саме користувачем – фізичним об’єктом.

Скрипт створення ролі individual:

CREATE ROLE individual WITH LOGIN PASSWORD '22222' VALID UNTIL '2019-12-09';

Треба надати фізичнії особі право на підключення до нашої бази даних.

Скрипт:

GRANT CONNECT ON DATABASE "CourseWork" TO individual;

Фізичнії особі треба бачити:

- хто виконавець замовлення;
- хто автор видання;
- власне видання.

Скрипт:

GRANT SELECT ON individual_manager, author_edition, edition TO individual;

Фізична особа має право змінювати свої дані, якщо вони були некоректно введені або змінені.

Скрипт:

GRANT UPDATE ("Client full name",telephone,fax,address) ON individual_manager

TO individual;

Фізична особа має право змінювати тираж свого видання, а також назву видання.

Скрипт:

GRANT UPDATE (printing, name) ON edition

TO individual;

Юридична особа повиненна бути саме користувачем – фізичним об’єктом.

Скрипт створення ролі entity:

CREATE ROLE entity WITH LOGIN PASSWORD '33333' VALID UNTIL '2019-12-09';

Треба надати юридичнії особі право на підключення до нашої бази даних.

Скрипт:

```
GRANT CONNECT ON DATABASE "CourseWork" TO entity;
```

Юридичнії особі треба бачити:

- хто виконавець замовлення;
- хто автор видання;
- власне видання.

Скрипт:

```
GRANT SELECT ON entity_manager, author_edition, edition TO entity;
```

Юридична особа має право змінювати свої дані, якщо вони були некоректно введені або змінені.

Скрипт:

```
GRANT UPDATE ("Entity name",representative,telephone,fax,address)
ON entity_manager TO entity;
```

Юридична особа має право змінювати тираж свого видання, а також назву видання.

Скрипт:

```
GRANT UPDATE (printing, name) ON edition
TO entity;
```

ВИСНОВКИ

В даній курсовій роботі була створена база даних щодо роботи видавництва, вона має на меті полегшання керування видавництвом та забезпечення швидкішого обліку кадрів та видань. Ця база даних зберігає інформацію про замовлення кліента та його особисті данні, а також інформацію про співробітників фірми. Також вана містить інформацію про друкарні з якими укладений контракт, інформацію про авторів з якими укладений контракт, а також інформацію про видання.

Для полегшання роботи працівників видавництва існують спеціальні функції, які допоможуть швидше виконувать потрібні дії.

Дана база даних має такий функціонал:

- ведення замовлень від клієнтів;
- автоматичний розподіл праці;
- зручне додавання інформації;
- автоматичне створення зв’язків між авторами та виданням;
- забезпечити оперативний перегляд списку замовлень ;
- додавання, зміни, пошуку інформації в базі даних;
- ведення бугалтерії та ін.

Можна буде використовуючи цю базу даних, як фундамент, створити інтернет – видавництво.

СПИСОК ЛІТЕРАТУРИ

1. Малахов Є.В., Блажко О.А., Глава М.Г. Проектування БД та їх реалізація засобами стандартного SQL та PostgreSQL: Навч. посібник для студ. вищих навч. закладів. – О.: ВМВ, 2012.– 248 с.

2. Глава М.Г. Організація баз даних та знань: Конспект лекцій [Електронний ресурс].– Режим доступу: http://library.opu.ua.

3. Глава М.Г. Методичні вказівки до виконання лабораторних робіт з дисципліни „Організація баз даних та знань“ [Електронний ресурс].– Режим доступу: http://library.opu.ua.

4. Вирт Н. Алгоритмы и структуры данных: Перевод с английского [Электронный ресурс].– М.: Мир, 1989. – 360 с.– Режим доступа: https://doc.lagout.org/science/0_Computer%20Science/2_Algorithms/Algorithms%20and%20Data%20Structures%20(RU).pdf

5. Басюк Т. М. Основи інформаційних технологій: навч. посібник / Т.М. Басюк, Н.О. Думанський, О.В. Пасічник; за наук. ред. В.В. Пасічника.– Л. : Новий Світ-2000, 2011.– 390 с. (13 шт)

6. Дейт К. Дж. Введение в системы баз данных: Перевод с английского – 8-е издание [Электронный ресурс].– М.: Вильямс, 2005.– 1318 с.– Режим доступа: ftp://saw.vvsu.ru/books/IT/Access%20BD/Deit.pdf.

7. Системы управления базами данных и знаний: Справочник / А.Н. Наумов, А.М. Вендров, В.К. Иванов и др.; Под ред. А.Н. Наумова.– М. : Финансы и статистика, 1991.– 348 с. (43 шт)

8. Кузнецов С. Д. СУБД (системы управления базами данных) и файловые системы.– М.: Майор, 2001.– 176 с. (30 шт)

9. Інформаційні системи і технології в економіці: Посібник для вузів / В.С. Пономаренко, Р.К. Бутова, І.В. Журавльова та ін.; За ред.В.С.Пономаренка.– К. : Академія, 2002.– 544 с. (24 шт)

10. Грабер М. Введение в SQL: Перевод с английского [Электронный ресурс].– М.: “ЛОРИ”, 1996.– 382 с.– Режим доступа: http://www.e-reading.club/bookreader.php/140820/ Gruber_-_Ponimanie_SQL.pdf

11. Гаврилова Т.А. Базы знаний интеллектуальных систем: учеб.пособие / Т.А. Гаврилова, В.Ф. Хорошевский.– СПб.: Питер, 2001.– 384 с. (31 шт)

12. Берко А. Ю. Системи баз даних та знань. Кн. 1. Організація баз даних та знань: навч. посібник / А.Ю. Берко, О.М. Верес, В.В. Пасічник; за заг. ред. В.В. Пасічника.– Л.: Магнолія 2006, 2008.– 456 с. (1 шт)

13. Джеймс Р. Грофф, Пол Н. Вайнберг, Эндрю Дж. Оппель SQL: Полное руководство: Пер. с англ. – 3-е изд.– Вильямс, 2015. – 961 с.

ДОДАТОК А
СТВОРЕННЯ ПРЕДСТАВЛЕНЬ

Створення представлення Entity_Manager (рисунок А.1, аркуш).

Створення представлення, яке полегшує взаємодію між клієнтом (юридичною особою) та менеджером. Клієнт може відстежити терміни виконання замовлення, виконавця замовлення, чи правильно вказані його особисті дані. Менеджер може відстежити який у нього кл

ієнт і замовлення, скільки у нього є ще часу для виконання замовлення.

Скрипт:

```
CREATE VIEW Entity_Manager("Entity name", representative, telephone, fax, address,"Number of Order ", "Booking Date", "Realization Date","Employee full name", post)
AS
SELECT EN.name, EN.representative, EN.telephone, EN.fax, EN.address, B.booking_id, B.booking_date, B.realization_date, E.full_name, E.post
FROM Entity EN, Booking B, Employee E
WHERE EN.client_id = B.client_id AND B.responsible_id = E.pasport
AND B.check_mark = '0'
ORDER BY B.realization_date;
```

Entity name character varying	representative character varying	telephone character varying	fax character varying	address address	Number of Order positive	Booking Date date	Realization Date date	Employee full name character varying	post character varying
Ferma	Lomonov G.T.	067 245 56 78	77 68 90	(Kiev,Hmelnickogo,13,3)	5	2018-12-08	2019-03-12	Gudneva K.O.	Manager
Charodijka	Pulnev F.K.	067 555 56 71	74 68 56	(Kiev,Hreschatik,56,7)	3	2018-12-08	2019-06-30	Glavachev S.D.	SE Manager
Prom	Kranov O.L.	067 456 56 78	77 89 56	(Odessa,Dumskaja,1,1)	1	2018-12-08	2019-09-30	Serpunov I.B.	Manager
Prom	Kranov O.L.	067 456 56 78	77 89 56	(Odessa,Dumskaja,1,1)	11	2018-12-08	2019-12-28	Nortilova K.L.	General Manager

Рисунок А.1 - Представлення Entity_Manager

ДОДАТОК Б

СТВОРЕННЯ ТРИГЕРІВ

1. Створення тригера pk_individual.

Створення тригера pk_ individual на таблицю individual, який перед операцією INSERT або UPDATE виконує функцію check_client.

Скрипт:

```
CREATE TRIGGER pk_individual
BEFORE INSERT OR UPDATE
ON individual
FOR EACH ROW
EXECUTE PROCEDURE public.check_client();
```

2. Створення тригерної функції check_ employee

Створення цієї тригерної функції дозволяє уникнути додавання (зміни) паспорта співробітника такого ж (на такий же), який вже є в таблиці employee. У разі такої помилки буде виведено повідомлення про те що ви ввели паспорт, який вже існує.

Скрипт:

```
CREATE OR REPLACE FUNCTION check_employee() RETURNS TRIGGER
AS $$
DECLARE pasport employee.pasport%TYPE;
BEGIN
  IF (TG_OP = 'UPDATE') THEN
      IF(new.pasport <> old.pasport) THEN
      SELECT E.pasport INTO pasport FROM employee E
```

```
            WHERE E.pasport = new.pasport;
            IF FOUND THEN
            RAISE EXCEPTION 'Duplicate pasport!';
            END IF;
            END IF;
          RAISE NOTICE 'Employee % successfully updated!',old.full_name;
          RETURN NEW;
        END IF;
        SELECT E.pasport INTO pasport FROM employee E
        WHERE E.pasport = new.pasport;
        IF NOT FOUND THEN
          RAISE NOTICE 'Employee % successfully added!',new.full_name;
          RETURN NEW;
          ELSE
          RAISE EXCEPTION 'Duplicate pasport!';
        END IF;
      END;
      $$ LANGUAGE plpgsql;
```

Створення тригера pk_employee на таблицю employee, який перед операцією INSERT або UPDATE виконує функцію check_employee.

Скрипт:

```
CREATE TRIGGER pk_employee
BEFORE INSERT OR UPDATE
ON employee
FOR EACH ROW
EXECUTE PROCEDURE public.check_employee();
```

3. Створення тригерної функції check_ typography

Створення цієї тригерної функції дозволяє уникнути додавання (зміни) первинного ключа такого ж (на такий же), який вже є в таблиці typography. У разі такої помилки буде виведено повідомлення про те що ви ввели ключ, який вже існує.

Скрипт:

```
CREATE OR REPLACE FUNCTION check_typography() RETURNS TRIGGER
AS $$
DECLARE typography_id typography.typography_id%TYPE;
BEGIN
  IF (TG_OP = 'UPDATE') THEN
    IF(new.typography_id <> old.typography_id) THEN
      SELECT T.typography_id INTO typography_id FROM typography T
      WHERE T.typography_id = new.typography_id;
      IF FOUND THEN
      RAISE EXCEPTION 'Duplicate typography_id!';
      END IF;
    END IF;
    RAISE NOTICE 'Typography % successfully updated!',old.name;
    RETURN NEW;
  END IF;
  SELECT T.typography_id INTO typography_id FROM typography T
  WHERE T.typography_id = new.typography_id;
  IF NOT FOUND THEN
    RAISE NOTICE 'Typography % successfully added!',new.name;
    RETURN NEW;
```

```
    ELSE
        RAISE EXCEPTION 'Duplicate typography_id!';
    END IF;
END;
$$ LANGUAGE plpgsql;
```

Створення тригера pk_typography на таблицю typography, який перед операцією INSERT або UPDATE виконує функцію check_typography.

Скрипт:

```
CREATE TRIGGER pk_typography
BEFORE INSERT OR UPDATE
ON typography
FOR EACH ROW
EXECUTE PROCEDURE public.check_typography();
```

4. Створення тригерної функції check_ booking

Створення цієї тригерної функції дозволяє уникнути:

- додавання (зміни) первинного ключа такого ж (на такий же), який вже є в таблиці booking;
- додавання заказа клієнта, який не існує;
- додавання заказа з друкарнею, яка не існує;
- додавання заказа з виконавцем якого не існує;
- випадку коли дата замовлення пізніше ніж дата виконання.

У разі однієї з цих помилок, буде виведено повідомлення про відповідну помилку.

Скрипт:

```
CREATE OR REPLACE FUNCTION check_booking() RETURNS
TRIGGER
AS $$
DECLARE
    booking_id booking.booking_id%TYPE;
    client_id client.client_id%TYPE;
    typography_id typography.typography_id%TYPE;
    responsible_id employee.pasport%TYPE;
BEGIN

IF(new.booking_date > new.realization_date) THEN
RAISE EXCEPTION 'Date error!';
END IF;

IF((TG_OP = 'UPDATE' AND new.booking_id <> old.booking_id)
 OR TG_OP = 'INSERT') THEN

SELECT B.booking_id INTO booking_id FROM booking B
WHERE B.booking_id = new.booking_id;

IF NOT FOUND THEN

SELECT C.client_id INTO client_id FROM client C
WHERE C.client_id = new.client_id;

IF FOUND THEN

SELECT T.typography_id INTO typography_id FROM typography T
WHERE T.typography_id = new.typography_id;
```

```
IF FOUND THEN

SELECT E.pasport INTO responsible_id FROM employee E
WHERE E.pasport = new.responsible_id;

IF FOUND THEN

IF(TG_OP = 'INSERT') THEN
RAISE NOTICE 'Booking % successfully added!',new.booking_id;
ELSE
RAISE NOTICE 'Booking % successfully updated!',new.booking_id;
END IF;

RETURN NEW;

ELSE
RAISE EXCEPTION 'Non-existent responsible_id!';
END IF;

ELSE
RAISE EXCEPTION 'Non-existent typography_id!';
END IF;

ELSE
RAISE EXCEPTION 'Non-existent client_id!';
END IF;

ELSE
```

```
RAISE EXCEPTION 'Duplicate booking_id!';
END IF;

ELSE

SELECT C.client_id INTO client_id FROM client C
WHERE C.client_id = new.client_id;

IF FOUND THEN

SELECT T.typography_id INTO typography_id FROM typography T
WHERE T.typography_id = new.typography_id;

IF FOUND THEN

SELECT E.pasport INTO responsible_id FROM employee E
WHERE E.pasport = new.responsible_id;

IF FOUND THEN
RAISE NOTICE 'Booking % successfully updated!',new.booking_id;
RETURN NEW;

ELSE
RAISE EXCEPTION 'Non-existent responsible_id!';
END IF;

ELSE
RAISE EXCEPTION 'Non-existent typography_id!';
END IF;
```

```
ELSE
RAISE EXCEPTION 'Non-existent client_id!';
END IF;

END IF;
END;
$$ LANGUAGE plpgsql;
```

Створення тригера test_booking на таблицю booking, який перед операцією INSERT або UPDATE виконує функцію check_booking.

Скрипт:

```
CREATE TRIGGER test_booking
BEFORE INSERT OR UPDATE
ON booking
FOR EACH ROW
EXECUTE PROCEDURE public.check_booking();
```

5. Створення тригерної функції check_edition

Створення цієї тригерної функції дозволяє уникнути додавання (зміни) первинного ключа такого ж (на такий же), який вже є в таблиці edition, та випадку, коли введено замовлення, якого не існує. У разі такої помилки буде виведено повідомлення про те що ви ввели ключ, який вже існує, або про те ща такого замовлення не існує.

Скрипт:

```
CREATE OR REPLACE FUNCTION check_edition() RETURNS
TRIGGER
```

```
AS $$
DECLARE
    booking_id booking.booking_id%TYPE;
    edition_id edition.edition_id%TYPE;
BEGIN
IF((TG_OP = 'UPDATE' AND new.edition_id <> old.edition_id)
 OR TG_OP = 'INSERT') THEN

SELECT E.edition_id INTO edition_id FROM edition E
WHERE E.edition_id = new.edition_id;

IF NOT FOUND THEN
SELECT B.booking_id INTO booking_id FROM booking B
WHERE B.booking_id = new.booking_id;

IF FOUND THEN

IF(TG_OP = 'INSERT') THEN
RAISE NOTICE 'Edition % successfully added!',new.name;
ELSE
RAISE NOTICE 'Edition % successfully updated!',new.name;
END IF;

RETURN NEW;

ELSE
RAISE EXCEPTION 'Non-existent booking_id!';
END IF;
```

```
ELSE
RAISE EXCEPTION 'Duplicate edition_id!';
END IF;

ELSE
SELECT B.booking_id INTO booking_id FROM booking B
WHERE B.booking_id = new.booking_id;

IF FOUND THEN
RAISE NOTICE 'Edition % successfully updated!',new.name;
RETURN NEW;
ELSE
RAISE EXCEPTION 'Non-existent booking_id!';
END IF;

END IF;
END;
$$ LANGUAGE plpgsql;
```

Створення тригера test_edition на таблицю edition, який перед операцією INSERT або UPDATE виконує функцію check_ edition.

Скрипт:

```
CREATE TRIGGER test_edition
BEFORE INSERT OR UPDATE
ON edition
FOR EACH ROW EXECUTE PROCEDURE public.check_edition();
```

6. Створення тригера delete_ individual на таблицю individual, який перед операцією DELETE виконує функцію delete_client.

Скрипт:

```
CREATE TRIGGER delete_individual
BEFORE DELETE ON individual
FOR EACH ROW EXECUTE PROCEDURE public.delete_client();
```

7. Створення тригерної функції delete_typography.

Створення цієї тригерної функції дозволяє у разі розірвання контрактка з друкарнею всі замовлення, які були на нії, розподілити по іншим друкарням.

Скрипт:

```
CREATE OR REPLACE FUNCTION delete_typography() RETURNS TRIGGER
AS $$
DECLARE id_typography typography.typography_id%TYPE;
BEGIN
SELECT T.typography_id INTO id_typography FROM typography T
WHERE T.typography_id <> old.typography_id;

IF FOUND THEN
UPDATE booking SET typography_id = id_typography
WHERE typography_id = old.typography_id;
END IF;

RETURN OLD;
END;
```

$$ LANGUAGE plpgsql;

Створення тригера del_ typography на таблицю typography, який перед операцією DELETE виконує функцію delete_ typography

Скрипт:

```
CREATE TRIGGER del_typography
BEFORE DELETEON typography
FOR EACH ROW EXECUTE PROCEDURE public.delete_typography();
```

ДОДАТОК В

СТВОРЕННЯ ЗБЕРЕЖЕНИХ ПРОЦЕДУР (ФУНКЦІЇ)

1. Створення функції insert_typography.

Створення цієї функції дозволяє полегшити менеджеру заповнення даних про друкарню. Менеджеру не потрібно придумувати унікальний первинний ключ, за нього все зробить програма.

Скрипт:

```
CREATE OR REPLACE FUNCTION insert_typography(name varchar,City VARCHAR, Street VARCHAR,House int,Flat int,telephone varchar,contact_person varchar)
RETURNS INT
AS $$
DECLARE address client.address%TYPE;
BEGIN

IF (House <= 0 OR Flat <= 0) THEN
RAISE EXCEPTION 'Flat(House) cannot be negative!';
END IF;

address := ROW(City, Street, House, Flat);
INSERT INTO typography  VALUES (
nextval('typography_pk'),name,address,telephone,contact_person);
RAISE NOTICE 'Typography %, successfully added!', name;
RETURN 1;
END;
$$ LANGUAGE plpgsql;
```

2. Створення функції insert_ author.

Створення цієї функції дозволяє полегшити менеджеру заповнення даних про авторів. Менеджеру не потрібно придумувати унікальний первинний ключ, за нього все зробить програма.

Скрипт:

```
CREATE OR REPLACE FUNCTION insert_author(full_name varchar, note text) RETURNS INT
AS $$
DECLARE
rec RECORD;

BEGIN

SELECT A.full_name, A.note INTO rec FROM Author A
WHERE A.full_name = insert_author.full_name AND A.note = insert_author.note;

IF FOUND THEN
RAISE EXCEPTION 'This author is already exists!';
END IF;

INSERT INTO author  VALUES (
nextval('author_pk'),full_name, note);
RAISE NOTICE 'Author %, successfully added!', full_name;

RETURN 1;
END;
$$ LANGUAGE plpgsql;
```

3. Створення функції insert_booking.

Створення цієї функції дозволяє полегшити менеджеру оформлення замовлення. Менеджеру не потрібно придумувати унікальний первинний ключ, за нього все зробить програма, а також програма автоматично заповнює атрибут з датою створення заказа та атрибут зі станом заказу.

Скрипт:

```
CREATE OR REPLACE FUNCTION insert_booking(client_id int, typography_id int,realisation_date date, responsible_id varchar)
RETURNS INT

AS $$
DECLARE
rec RECORD;

BEGIN

SELECT INTO rec FROM client C
WHERE C.client_id = insert_booking.client_id;
IF NOT FOUND THEN
RAISE EXCEPTION 'client_id not found!';
END IF;

SELECT INTO rec FROM Employee E
WHERE E.pasport = insert_booking.responsible_id;
IF NOT FOUND THEN
RAISE EXCEPTION 'responsible_id not found!';
END IF;
```

```
SELECT INTO rec FROM Typography T
WHERE T.typography_id = insert_booking.typography_id;
IF NOT FOUND THEN
RAISE EXCEPTION 'typography_id not found!';
END IF;

INSERT INTO booking VALUES (nextval('booking_pk'),client_id, typography_id,current_date,'0', realisation_date, responsible_id);
RAISE NOTICE 'Booking ы%, successfully added!', currval('booking_pk');

RETURN 1;
END;
$$ LANGUAGE plpgsql;
```

ДОДАТОК Д
ОПЕРАТОРИ ВІДНОВЛЕННЯ

1. Заповнення клієнтської бази.

Додавання в клієнтську базу фізичної особи Maznov P.K.

Використовується функція insert_client.

Скрипт:

SELECT insert_client('I', 'Odessa', 'Kanatna', 45, 5, '0675554578', '776867', '1234748', 'Maznov P.K.');

Додавання в клієнтську базу юридичної особи «Charodijka».

Використовується функція insert_client.

Скрипт:

SELECT insert_client('E', 'Kiev', 'Hreschatik', 56 , 7, '0675555671', '746856', 'Pulnev F.K.', 'Charodijka');

Додавання в клієнтську базу фізичної особи Dolgov L.N.

Використовується функція insert_client.

Скрипт:

SELECT insert_client('I', 'Kiev', 'Shevchenko', 5, 7, '0672255678', '776056', 'PH 4494830', 'Dolgov L.N.');

Додавання в клієнтську базу юридичної особи «Ferma».

Використовується функція insert_client.

Скрипт:

SELECT insert_client('E', 'Kiev', 'Hmelnickogo', 13, 3, '0672455678', '776890', 'Lomonov G.T.', 'Ferma');

Додавання в клієнтську базу фізичної особи Ivanov P.B.

Використовується функція insert_client.

Скрипт:

SELECT insert_client('I', 'Slavuta', 'Revolucij', 111, NULL, '0505545678', '456856', '3784930', 'Ivanov P.B.');

Додавання в клієнтську базу юридичної особи «Dva Shaga».

Використовується функція insert_client.

Скрипт:

SELECT insert_client('E', 'Slavuta', 'Hmelnickogo', 56, 7, '0675555078', '776889', 'Rezcov V.N.', 'Dva Shaga');

Додавання в клієнтську базу фізичної особи Luznev A.K.

Використовується функція insert_client.

Скрипт:

SELECT insert_client('I', 'Slavuta', 'Klara Cetkina', 3, NULL, '0675455678', '776851', 'КН 4843209', 'Luznev A.K.');

Додавання в клієнтську базу юридичної особи «Rud».

Використовується функція insert_client.

Скрипт:

SELECT insert_client('E', 'Odessa', 'Kanatna', 100, 7, '0677775678', '779956', 'Rudnev K.G.', 'Rud');

2. Заповнення бази друкарень.

Додавання в базу друкарень друкарню «Good».

Використовується функція insert_typography.

Скрипт:

SELECT insert_typography('Good', 'Kiev', 'Obolonskaja', 24, NULL, '0503852858', 'Luckov E.R.');

Додавання в базу друкарень друкарню «Slavut Pechat».

Використовується функція insert_typography.

Скрипт:

SELECT insert_typography('Slavut Pechat', 'Slavuta', 'Dokovaja', 5, NULL, '0285895912', 'Trushilov T.O.');

3. Заповнення бази авторів.

Додавання в базу авторів автора Borisov F.S. .

Використовується функція insert_author.

Скрипт:

SELECT insert_author('Borisov F.S.','Computer Science');

Додавання в базу авторів автора Poblajko E. .

Використовується функція insert_author.

Скрипт:

SELECT insert_author('Poblajko E.','Math');

Додавання в базу авторів автора Srednik J.D. .

Використовується функція insert_author.

Скрипт:

SELECT insert_author('Srednik J.D.','Native language');

Додавання в базу авторів автора Guliven R.T. .

Використовується функція insert_author.

Скрипт:

SELECT insert_author('Guliven R.T.','Food Technology');

Додавання в базу авторів автора Fridsun A.I. .

Використовується функція insert_author.

Скрипт:

SELECT insert_author('Fridsun A.I.','Web Design');

Додавання в базу авторів автора Bednik D.K. .

Використовується функція insert_author.

Скрипт:

SELECT insert_author('Bednik D.K.','Food Technology');

Додавання в базу авторів автора Avtor D.H. .

Використовується функція insert_author.

Скрипт:

SELECT insert_author('Avtor D.H.','Promotion');

Додавання в базу авторів автора Nonej J. .

Використовується функція insert_author.

Скрипт:

SELECT insert_author('Nonej J.','AI');

Додавання в базу авторів автора Sandro S.A. .

Використовується функція insert_author.

Скрипт:

SELECT insert_author('Sandro S.A.','Food Technology');

4. Заповнення бази замовлень.

Додавання в базу замовлень замовлення клієнта з індифікаціїним номером 3.

Використовується функція insert_booking.

Скрипт:

```
SELECT insert_booking(3,1,'23 Jul 2019','393034');
```

Додавання в базу замовлень замовлення клієнта з індифікаціїним номером 4.

Використовується функція insert_booking.

Скрипт:

```
SELECT insert_booking(4,2,'30 Jun 2019','383902');
```

Додавання в базу замовлень замовлення клієнта з індифікаціїним номером 5.

Використовується функція insert_booking.

Скрипт:

```
SELECT insert_booking(5,2,'09 Oct 2019','ER 38291');
```

Додавання в базу замовлень замовлення клієнта з індифікаціїним номером 6.

Використовується функція insert_booking.

Скрипт:

```
SELECT insert_booking(6,2,'12 Mar 2019','DE 18340');
```

Додавання в базу замовлень замовлення клієнта з індифікаціїним номером 7.

Використовується функція insert_booking.

Скрипт:

SELECT insert_booking(7,3,'30 Apr 2019','393034');

Додавання в базу замовлень замовлення клієнта з індифікаційним номером 8.

Використовується функція insert_booking.

Скрипт:

SELECT insert_booking(8,3,'22 DEC 2018','KN 48990');

Додавання в базу замовлень замовлення клієнта з індифікаційним номером 9.

Використовується функція insert_booking.

Скрипт:

SELECT insert_booking(9,3,'05 Jul 2019','284028');

Додавання в базу замовлень замовлення клієнта з індифікаційним номером 10.

Використовується функція insert_booking.

Скрипт:

SELECT insert_booking(10,1,'30 Oct 2019','127718');

Додавання в базу замовлень другого замовлення клієнта з індифікаційним номером 2.

Використовується функція insert_booking.

Скрипт:

SELECT insert_booking(2,1,'29 DEC 2018','KN 48990');

Додавання в базу замовлень замовлення клієнта з індифікаційним номером 1.

Використовується функція insert_booking.

Скрипт:

SELECT insert_booking(1,1,'28 DEC 2019','284028');

Додавання в базу замовлень другого замовлення клієнта з індифікаціїним номером 3.

Використовується функція insert_booking.

Скрипт:

SELECT insert_booking(3,1,'27 DEC 2018','KN 48990');

5. Заповнення бази видань.

Додавання в базу видань видання «Prom – advertising #2» для замовлення №1.

Використовується функція insert_edition.

Скрипт:

SELECT insert_edition(1, 1, 23, 'Prom - advertising #2', 'Poster', '{{"Fridsun A.I.", "Web Design"}}');

Додавання в базу видань видання «Cook Book» для замовлення №2.

Використовується функція insert_edition.

Скрипт:

SELECT insert_edition(2, 50, 57, 'Cook Book', 'Book', '{{"Guliven R.T.", "Food Technology"}, {"Bednik D.K.", "Food Technology"}, {"Sandro S.A.", "Food Technology"}}');

Додавання в базу видань видання «Charodijka - card» для замовлення №3.

Використовується функція insert_edition.

Скрипт:

SELECT insert_edition(3, 1, 80, 'Charodijka - card', 'Business card', '{{"Avtor D.H.", "Promotion"}}');

Додавання в базу видань видання «Charodijka - advertising» для замовлення №3.

Використовується функція insert_edition.

Скрипт:

SELECT insert_edition(3, 1, 98, 'Charodijka - advertising', 'Poster', '{{"Avtor D.H.", "Promotion"}}');

Додавання в базу видань видання «Charodijka - journal» для замовлення №3.

Використовується функція insert_edition.

Скрипт:

SELECT insert_edition(3, 24, 3455, 'Charodijka - journal', 'Advertising', '{{"Avtor D.H.", "Promotion"}}');

Додавання в базу видань видання «Computer Science» для замовлення №4.

Використовується функція insert_edition.

Скрипт:

SELECT insert_edition(4, 23, 36, 'Computer Science', 'Conspect', '{{"Borisov F.S.", "Computer Science"}}');

Додавання в базу видань видання «Ferma - journal» для замовлення №5.

Використовується функція insert_edition.

Скрипт:

SELECT insert_edition(5, 20, 200, 'Ferma - journal', 'Advertising', '{{"Avtor D.H.", "Promotion"}}');

Додавання в базу видань видання «Fourier series» для замовлення №6.

Використовується функція insert_edition.

Скрипт:

SELECT insert_edition(6, 30, 77, 'Fourier series', 'Conspect', '{{"Poblajko E.", "Math"}}');

Додавання в базу видань видання «Dva shaga - card» для замовлення №7.

Використовується функція insert_edition.

Скрипт:

SELECT insert_edition(7, 1, 30, 'Dva shaga - card', 'Business card', '{{"Avtor D.H.", "Promotion"}}');

Додавання в базу видань видання «Dva shaga - advertising» для замовлення №7.

Використовується функція insert_edition.

Скрипт:

SELECT insert_edition(7, 1, 85, 'Dva shaga - advertising', 'Poster', '{{"Avtor D.H.", "Promotion"}}');

Додавання в базу видань видання «AI» для замовлення №8.

Використовується функція insert_edition.

Скрипт:

SELECT insert_edition(8, 587, 1, 'AI', 'Manual', '{{"Nonej J.", "AI"}}');

Додавання в базу видань видання «Rud - advertising» для замовлення №9.

Використовується функція insert_edition.

Скрипт:

SELECT insert_edition(9, 1, 90, 'Rud - advertising', 'Poster', '{{"Avtor D.H.", "Promotion"}}');

Додавання в базу видань видання «National Food» для замовлення №10.

Використовується функція insert_edition.

Скрипт:

SELECT insert_edition(10, 7, 4, 'National Food', 'Book', '{{"Guliven R.T.","Food Technology"}, {"Bednik D.K.","Food Technology"}, {"Sandro S.A.","Food Technology"}}');

Додавання в базу видань видання «Prom – advertising #3» для замовлення №11.

Використовується функція insert_edition.

Скрипт:

SELECT insert_edition(11, 9, 4656, 'Prom - advertising #3', 'Journal', '{{"Fridsun A.I.", "Web Design"}}');

Додавання в базу видань видання «Prom – card» для замовлення №11.

Використовується функція insert_edition.

Скрипт:

SELECT insert_edition(11, 1, 365, 'Prom - card', 'Business card', '{{"Fridsun A.I.", "Web Design"}}');

Додавання в базу видань видання «Philosophy» для замовлення №12.

Використовується функція insert_edition.

Скрипт:

```
SELECT insert_edition(12, 600, 1, 'Philosophy', 'Book', '{{"Djonson B.", "Philosophy"}}');
```

Додавання в базу видань видання «Maznov - card» для замовлення №12.

Використовується функція insert_edition.

Скрипт:

```
SELECT insert_edition(12, 1, 24, 'Maznov - card', 'Business card', '{{"Avtor D.H.", "Promotion"}}');
```

6. Виконування замовлень

Виконування замовлення №6.

Використовується функція completed.

Скрипт:

```
SELECT completed(6,'393034');
```

Виконування замовлення №7.

Використовується функція completed.

Скрипт:

```
SELECT completed(7,'KN 48990');
```

Виконування замовлення №10.

Використовується функція completed.

Скрипт:

```
SELECT completed(10,'KN 48990');
```

Виконування замовлення №9.

Використовується функція completed.

Скрипт:

```
SELECT completed(9,'127718');
```