**ФОРМУЛИРОВКА ЗАДАНИЯ**

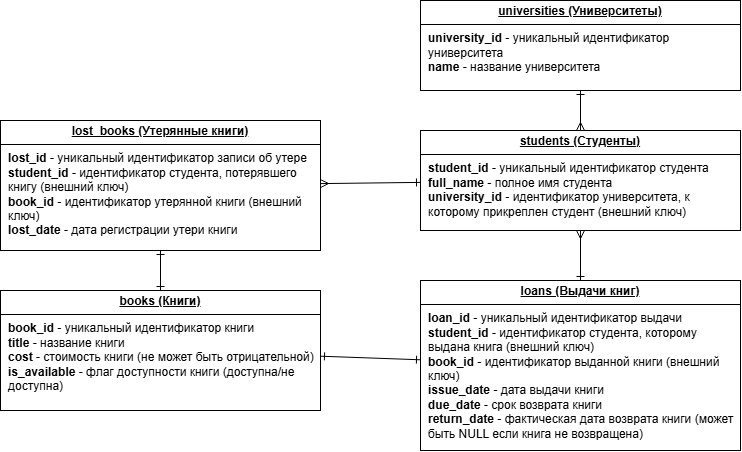
Библиотека Почитайкин

В библиотеке выдач книг студентам ВУЗов ведётся на основе регистров о выдаче с указанием срока возврата, стоимости книг. При возврате книги указывается дата. Если книга утеряна, то студент оплачивает её стоимость.

На указанную дату по указанным ВУЗам составить список студентов, не вернувших книги в срок (экспортировать в MS Excel).

За период сформировать отчёт по ВУЗам согласно общему количеству выданных, невозвращённых и утерянных книг.

**ТАБЛИЦЫ ОПИСАНИЕ БД**



1. universities (Университеты)

university\_id - уникальный идентификатор университета

name - название университета

1. students (Студенты)

student\_id - уникальный идентификатор студента

full\_name - полное имя студента

university\_id - идентификатор университета, к которому прикреплен студент (внешний ключ)

1. books (Книги)

book\_id - уникальный идентификатор книги

title - название книги

cost - стоимость книги (не может быть отрицательной)

is\_available - флаг доступности книги (доступна/не доступна)

1. loans (Выдачи книг)

loan\_id - уникальный идентификатор выдачи

student\_id - идентификатор студента, которому выдана книга (внешний ключ)

book\_id - идентификатор выданной книги (внешний ключ)

issue\_date - дата выдачи книги

due\_date - срок возврата книги

return\_date - фактическая дата возврата книги (может быть NULL если книга не возвращена)

1. lost\_books (Утерянные книги)

lost\_id - уникальный идентификатор записи об утере

student\_id - идентификатор студента, потерявшего книгу (внешний ключ)

book\_id - идентификатор утерянной книги (внешний ключ)

lost\_date - дата регистрации утери книги

**-- СОЗДАНИЕ ТАБЛИЦ**

CREATE TABLE universities (

university\_id SERIAL PRIMARY KEY,

name VARCHAR(100) NOT NULL

);

CREATE TABLE students (

student\_id SERIAL PRIMARY KEY,

full\_name VARCHAR(100) NOT NULL,

university\_id INTEGER NOT NULL REFERENCES universities(university\_id) ON DELETE CASCADE

);

CREATE TABLE books (

book\_id SERIAL PRIMARY KEY,

title VARCHAR(150) NOT NULL,

cost NUMERIC(10, 2) NOT NULL CHECK (cost >= 0),

is\_available BOOLEAN NOT NULL DEFAULT TRUE

);

CREATE TABLE loans (

loan\_id SERIAL PRIMARY KEY,

student\_id INTEGER NOT NULL REFERENCES students(student\_id) ON DELETE CASCADE,

book\_id INTEGER NOT NULL REFERENCES books(book\_id),

issue\_date DATE NOT NULL,

due\_date DATE NOT NULL,

return\_date DATE,

CHECK (return\_date IS NULL OR return\_date >= issue\_date)

);

CREATE TABLE lost\_books (

lost\_id SERIAL PRIMARY KEY,

student\_id INTEGER NOT NULL REFERENCES students(student\_id) ON DELETE CASCADE,

book\_id INTEGER NOT NULL REFERENCES books(book\_id),

lost\_date DATE NOT NULL

);

-- Заполнение таблицы universities

INSERT INTO universities (name) VALUES

('Московский государственный университет'),

('Санкт-Петербургский государственный университет'),

('Новосибирский государственный университет'),

('Казанский федеральный университет'),

('Уральский федеральный университет');

-- Заполнение таблицы students

INSERT INTO students (full\_name, university\_id) VALUES

('Иванов Иван Иванович', 1),

('Петров Петр Петрович', 1),

('Сидорова Анна Михайловна', 2),

('Кузнецов Дмитрий Сергеевич', 3),

('Смирнова Елена Владимировна', 2),

('Васильев Алексей Николаевич', 4),

('Павлова Ольга Дмитриевна', 5),

('Николаев Сергей Иванович', 1),

('Федорова Мария Петровна', 3),

('Алексеев Андрей Васильевич', 5);

-- Заполнение таблицы books

INSERT INTO books (title, cost, is\_available) VALUES

('Введение в алгоритмы', 2500.00, TRUE),

('Современные операционные системы', 1800.50, TRUE),

('Искусство программирования', 3200.75, TRUE),

('Чистый код', 1200.00, TRUE),

('Дизайн баз данных', 1500.25, TRUE),

('Компьютерные сети', 2100.00, TRUE),

('Глубокое обучение', 2800.50, TRUE),

('Паттерны проектирования', 1350.75, TRUE),

('Теория вероятностей', 950.00, TRUE),

('Линейная алгебра', 1100.25, TRUE);

-- Заполнение таблицы loans

INSERT INTO loans (student\_id, book\_id, issue\_date, due\_date, return\_date) VALUES

(1, 1, '2023-01-10', '2023-02-10', '2023-02-08'),

(2, 3, '2023-01-15', '2023-02-15', '2023-02-14'),

(3, 2, '2023-02-01', '2023-03-01', NULL),

(4, 5, '2023-02-10', '2023-03-10', '2023-03-05'),

(5, 4, '2023-02-15', '2023-03-15', NULL),

(6, 7, '2023-03-01', '2023-04-01', '2023-03-25'),

(7, 6, '2023-03-10', '2023-04-10', NULL),

(8, 9, '2023-03-15', '2023-04-15', '2023-04-10'),

(9, 8, '2023-04-01', '2023-05-01', NULL),

(10, 10, '2023-04-10', '2023-05-10', '2023-05-05');

-- Заполнение таблицы lost\_books

INSERT INTO lost\_books (student\_id, book\_id, lost\_date) VALUES

(2, 3, '2023-02-14'),

(6, 7, '2023-03-25'),

(10, 10, '2023-05-05');

**ТРИГЕРЫ ДЛЯ РАБОТЫ С БД**

-- После добавления новой записи в таблицу loans, помечает книгу как недоступную (is\_available = FALSE) в таблице books

Записывает стоимость книги в запись о выдаче (в поле book\_cost)

CREATE OR REPLACE FUNCTION mark\_book\_unavailable()

RETURNS TRIGGER AS $$

BEGIN

-- Пометить книгу как недоступную

UPDATE books SET is\_available = FALSE WHERE book\_id = NEW.book\_id;

-- Опционно: зафиксировать стоимость книги в записи выдачи

NEW.book\_cost := (SELECT cost FROM books WHERE book\_id = NEW.book\_id);

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER after\_issue\_mark\_unavail

AFTER INSERT ON loans

FOR EACH ROW

EXECUTE FUNCTION mark\_book\_unavailable();

-- после обновления поля return\_date в таблице loans (когда книгу возвращают), проверяет, что return\_date не NULL, помечает книгу как доступную (is\_available = TRUE) в таблице books

CREATE OR REPLACE FUNCTION mark\_book\_available()

RETURNS TRIGGER AS $$

BEGIN

IF NEW.return\_date IS NOT NULL THEN

UPDATE books SET is\_available = TRUE WHERE book\_id = NEW.book\_id;

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER after\_return\_mark\_avail

AFTER UPDATE OF return\_date ON loans

FOR EACH ROW

WHEN (NEW.return\_date IS NOT NULL)

EXECUTE FUNCTION mark\_book\_available();

-- После добавления новой записи в таблицу lost\_books (когда книга считается утерянной), помечает книгу как недоступную навсегда (is\_available = FALSE)

CREATE OR REPLACE FUNCTION handle\_book\_loss()

RETURNS TRIGGER AS $$

DECLARE

cost\_numeric NUMERIC;

BEGIN

-- Пометить книгу недоступной навсегда (утеряна)

UPDATE books SET is\_available = FALSE WHERE book\_id = NEW.book\_id;

-- Если штраф не указан, подставить стоимость книги как сумму штрафа

IF NEW.fine\_amount IS NULL THEN

SELECT cost INTO cost\_numeric FROM books WHERE book\_id = NEW.book\_id;

NEW.fine\_amount := cost\_numeric;

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER after\_loss\_mark\_unavail

AFTER INSERT ON lost\_books

FOR EACH ROW

EXECUTE FUNCTION handle\_book\_loss();

**ПРЕДСТАВЛЕНИЯ ПО ТАБЛИЦАМ ДЛЯ DATAGRIDVIEW**

-- Представление для таблицы universities

CREATE OR REPLACE VIEW "Университеты" AS

SELECT

university\_id AS "ID университета",

name AS "Название университета"

FROM universities;

-- Представление для таблицы students с заменой university\_id на название университета

CREATE OR REPLACE VIEW "Студенты" AS

SELECT

s.student\_id AS "ID студента",

s.full\_name AS "ФИО студента",

u.name AS "Университет"

FROM students s

JOIN universities u ON s.university\_id = u.university\_id;

-- Представление для таблицы books

CREATE OR REPLACE VIEW "Книги" AS

SELECT

book\_id AS "ID книги",

title AS "Название книги",

cost AS "Стоимость",

is\_available AS "Доступна"

FROM books;

-- Представление для таблицы loans с человеко-читаемыми данными

-- МБ ИСПРАВИТЬ: УДАЛИТЬ ВЬЮ, СДЕЛАТЬ ПРОВЕРКУ НА СУЩЕСТВОВАНИЕ КНИГИ В УТЕРЯНЫХ, СДЕЛАТЬ CASE СОСТОЯНИЕ «УТЕРЯНО» ОТДЕЛЬНО

CREATE OR REPLACE VIEW "Выдачи книг" AS

SELECT

l.loan\_id AS "ID выдачи",

s.full\_name AS "Студент",

u.name AS "Университет студента",

b.title AS "Книга",

l.issue\_date AS "Дата выдачи",

l.due\_date AS "Срок возврата",

l.return\_date AS "Фактическая дата возврата",

CASE

WHEN l.return\_date IS NULL AND CURRENT\_DATE > l.due\_date THEN 'Просрочено/Утеряно'

WHEN l.return\_date IS NULL THEN 'На руках'

ELSE 'Возвращено'

END AS "Статус"

FROM loans l

JOIN students s ON l.student\_id = s.student\_id

JOIN universities u ON s.university\_id = u.university\_id

JOIN books b ON l.book\_id = b.book\_id;

-- Представление для таблицы lost\_books с человеко-читаемыми данными

CREATE OR REPLACE VIEW "Утерянные книги" AS

SELECT

lb.lost\_id AS "ID утери",

s.full\_name AS "Студент",

u.name AS "Университет студента",

b.title AS "Книга",

b.cost AS "Стоимость книги",

lb.lost\_date AS "Дата утери"

FROM lost\_books lb

JOIN students s ON lb.student\_id = s.student\_id

JOIN universities u ON s.university\_id = u.university\_id

JOIN books b ON lb.book\_id = b.book\_id;

**ПРЕДСТАВЛЕНИЯ, СДЕЛАННЫЕ ПО ПРИКОЛУ, МОЖНО БУДЕТ ИХ ДОБАВИТЬ ПО ТРЕБОВАНИЮ РАСКОМЕНТИВ КОД В VIEWDATA**

private void ViewData\_Load(object sender, EventArgs e)

{

Dictionary<string, string> tableNames = new Dictionary<string, string>

{

//{"Текущие выданные книги", "Текущие выданные книги"},

//{"Статистика по университетам", "Статистика по университетам"}

-- Дополнительное полезное представление: текущие выданные книги

CREATE OR REPLACE VIEW "Текущие выданные книги" AS

SELECT

s.full\_name AS "Студент",

u.name AS "Университет",

b.title AS "Книга",

l.issue\_date AS "Дата выдачи",

l.due\_date AS "Срок возврата",

(l.due\_date - CURRENT\_DATE) AS "Дней до возврата"

FROM loans l

JOIN students s ON l.student\_id = s.student\_id

JOIN universities u ON s.university\_id = u.university\_id

JOIN books b ON l.book\_id = b.book\_id

WHERE l.return\_date IS NULL;

-- Дополнительное полезное представление: статистика по университетам

CREATE OR REPLACE VIEW "Статистика по университетам" AS

SELECT

u.name AS "Университет",

COUNT(DISTINCT s.student\_id) AS "Количество студентов",

COUNT(DISTINCT l.loan\_id) AS "Всего выдач книг",

COUNT(DISTINCT CASE WHEN l.return\_date IS NULL THEN l.loan\_id END) AS "Текущие выдачи",

COUNT(DISTINCT lb.lost\_id) AS "Утерянные книги"

FROM universities u

LEFT JOIN students s ON u.university\_id = s.university\_id

LEFT JOIN loans l ON s.student\_id = l.student\_id

LEFT JOIN lost\_books lb ON s.student\_id = lb.student\_id

GROUP BY u.name;