# Лабораторная работа 3. Объединение данных из разных таблиц. Вложенные подзапросы

## *I. Операция JOIN*

Одна из ключевых возможностей реляционных баз данных – это оператор JOIN в команде SQL, который при выборке данных из БД выполняет операцию объединения строк из двух (или более) связанных таблиц. JOIN – это оператор, а не команда, потому что он используется внутри SQL-запроса SELECT для объединения данных, но сам по себе не применяется.

В реальных проектах SQL-запросы без JOIN встречаются редко, так как данные почти всегда распределены между несколькими таблицами.

Виды JOIN в SQL:

- **INNER JOIN** – возвращает только те строки, у которых есть соответствие в обеих таблицах.
- **LEFT JOIN** (LEFT OUTER JOIN) – возвращает все строки из левой таблицы и совпадающие строки из правой. Если для строки из левой таблицы не нашлось пары в правой таблице, то вместо данных из правой таблицы в результирующей таблице будет NULL, но сама строка из левой таблицы будет присутствовать в ответе.
- **RIGHT JOIN** (RIGHT OUTER JOIN) – аналогично LEFT JOIN, но берет все строки из правой таблицы.
- **FULL JOIN** (FULL OUTER JOIN) – возвращает все строки из обеих таблиц. Если нет совпадения, то подставляется `NULL`.
- **CROSS JOIN** – возвращает декартово произведение двух таблиц (каждая строка первой таблицы объединяется со всеми строками второй).
- **SELF JOIN** – соединение таблицы с самой собой.

Оператор JOIN используют очень часто, особенно в сложных базах данных:

- в отчетах и аналитике (`INNER JOIN`, `LEFT JOIN`);
- в интернет-магазинах (получение товаров с категориями и ценами);
- в системах управления (`JOIN` между пользователями, ролями, заказами и т. д.).

Далее рассмотрим наиболее популярные операторы - INNER JOIN и LEFT JOIN.

## *2. INNER JOIN*

INNER JOIN — это самый распространенный тип объединения таблиц в SQL. Он возвращает только те строки, у которых есть совпадения в обеих таблицах по указанному критерию. Если хотя бы в одной таблице отсутствует соответствующее значение, эта строка не попадёт в итоговый результат. В SQL можно писать просто JOIN вместо INNER JOIN – это одно и то же. По умолчанию, если указано JOIN без дополнительных уточнений (например, LEFT, RIGHT, FULL), то подразумевается именно INNER JOIN.

Как работает INNER JOIN:

- Для каждой строки из первой таблицы система ищет соответствующую строку во второй таблице.
- Если найдено совпадение по указанному критерию (например, идентификатору или другому значению), строки объединяются в один результат.

Если совпадения нет, строка исключается из результата.

Этот тип объединения удобен, когда нужно работать только с полными, связанными данными и исключить те записи, у которых нет соответствий в другой таблице.

Пример. Пусть в БД имеется следующие таблицы с данными:

Таблица Author

| AuthorID | Name |
|---|---|
| 1 | Л. Толстой |
| 2 | Ф. Достоевский |
| 3 | А. Чехов |

Таблица Book

| BookID | Title | Author |
|---|---|---|
| 1 | Война и мир | 1 |
| 2 | Преступление и наказание | 2 |
| 3 | Человек в футляре | 3 |

Запрос:

```
SELECT Author.Name, Book.Title
FROM Author
INNER JOIN Book ON Author.AuthorID = Book.Author;
```

Результат запроса:

| Name | Title |
|---|---|
| Л. Толстой | Война и мир |
| Ф. Достоевский | Преступление и наказание |
| А. Чехов | Человек в футляре |

В запросе INNER JOIN берет только те строки, где есть соответствие `AuthorID = Author` в таблице `Book`. Если бы в таблице `Book` была книга без указанного автора, она бы не попала в результат.

## *3. LEFT JOIN*

"Левая" и "правая" в LEFT JOIN – это условные названия в зависимости от расположения таблицы в SQL-запросе. Левая таблица находится слева от слова JOIN, а правая, соответственно, справа. Если в SQL-запросе используется несколько JOIN-ов**,** то понятия "левая" и "правая" таблицы сохраняются для каждого конкретного JOIN-а**.**

Как работает LEFT JOIN:

- Возвращает все строки из левой таблицы.
- Если для строки из левой таблицы есть совпадение в правой, данные из правой таблицы добавляются в строку результата.
- Если совпадения нет, вместо данных из правой таблицы будет NULL.

LEFT JOIN нужен, когда важно сохранить все данные из левой таблицы, даже если в правой таблице нет совпадений.

Основные случаи использования:

- Найти данные, у которых нет связей в другой таблице. Например, список всех студентов, включая тех, кто ещё не записан на курсы**.**
- Объединить данные, но не терять записи из левой таблицы. Например, список всех сотрудников и их проектов (даже если у некоторых проектов пока нет).
- Анализ данных с учётом отсутствующих значений. Например, список клиентов и их заказов (NULL покажет, у кого ещё нет заказов).

– Создание отчетов с полными списками. Например, список товаров с информацией о продажах (даже если товар ещё не покупали).

То есть LEFT JOIN помогает не терять важные данные, когда правой таблицы может не хватать.

Пример. Допустим, есть две таблицы с данными:

Таблица User

| UserID | Name |
|---|---|
| 1 | Алексей |
| 2 | Мария |
| 3 | Иван |

Таблица Order

| ProductID | UserID | Product |
|---|---|---|
| 1 | 1 | Телефон |
| 2 | 1 | Ноутбук |
| 3 | 3 | Книга |

Запрос:

```
SELECT User.UserID, User.Name, Order.Product
FROM User
LEFT JOIN Order ON User.UserID = Order.UserID;
```

Результат запроса:

| ID | Name | Product |
|---|---|---|
| 1 | Алексей | Телефон |
| 1 | Алексей | Ноутбук |
| 2 | Мария | NULL |
| 3 | Иван | Книга |

– Алексей есть в обоих таблицах → соединились строки.
– Иван тоже есть в обеих таблицах → соединились строки.
– Мария есть в `User`, но в `Order` её нет → для неё поставили `NULL`.

То есть строки из левой таблицы никогда не пропадут, даже если в правой ничего не нашлось.

## *2. Вложенные запросы*

Вложенные запросы (subqueries) – это SQL-запросы внутри другого SQL-запроса.

1. Вложенные запросы могут быть использованы для того, чтобы сначала выбрать одно значение в вложенном запросе, а затем использовать это значение в внешнем запросе для выполнения дальнейших операций.

Пример: Найти название книги и имя ее автора без `JOIN`. Причем у книги только один автор.

```
SELECT Title,
       (SELECT Name FROM Author WHERE AuthorID = Book.Author) AS AuthorName
FROM Book;
```

2. Если вложенный запрос возвращает несколько строк, а внешний запрос ожидает одно значение (например, при использовании оператора =), это может вызвать ошибку или некорректный результат. Поэтому следует использовать оператор IN.

Пример с оператором IN. Причем у книги может быть несколько авторов.

```
SELECT Title
FROM Book
WHERE Author IN (SELECT Author FROM Book WHERE Title = 'Мать');
```

В этом случае подзапрос возвращает несколько авторов, и оператор IN корректно сопоставит их с полем Author, не вызывая ошибок.

3. Вложенные запросы могут помочь если связи между таблицами сложные или JOIN приводит к дублированию строк,

Пример: Найти книги, у которых год публикации раньше самого старого года публикации в таблице. JOIN тут не поможет, потому что мы сравниваем с одной агрегированной величиной.

```
SELECT Title, PublicationYear
FROM Book
WHERE PublicationYear < (SELECT MIN(PublicationYear) FROM Book);
```

Вложенные запросы помогают отфильтровать данные. Это удобно, когда мы фильтруем таблицу на основе условий из другой таблицы.

Пример: Найти все книги, написанные авторами, у которых есть хотя бы одна книга после 2000 года.

```
SELECT Title
FROM Book
WHERE Author IN (SELECT AuthorID FROM Book WHERE PublicationYear > 2000);
```

Итак:

JOIN предпочтительнее, если нужно объединить таблицы, а вложенные запросы полезны, когда нужно вернуть одно значение для фильтрации или сравнения.

Важно помнить, что использование вложенных запросов может стать менее эффективным, если база данных становится очень большой и запросы становятся сложными. В таких случаях использование JOIN может быть быстрее, так как соединение данных может быть выполнено за один проход, а не через несколько запросов.

## *V. Окончательный базовый код программы:*

Программа LAB_BASE_3a.py Объединение таблиц JSON

```
import sqlite3
import os
import sys

# Путь к файлу базы данных SQLite
db_file = 'library.db'


# Если файл базы данных существует, просто подключаемся к нему
# и создаем объект курсор для выполнения SQL-запросов
if  os.path.isfile(db_file):
    conn = sqlite3.connect(db_file)
    cursor = conn.cursor()
else:
    print(f"Файл базы данных {db_file} не существует.")
    sys.exit()  # Завершаем программу



# ********************************
# Использование объединений JOIN
# ********************************

# Объединение имени автора и названия книги (таблицы Author, Book)
print()
print('**********  INNER JOIN  (Author, Book) **********')

cursor.execute('''
    SELECT Author.Name , Book.Title
    FROM Author
    JOIN Book ON Author.AuthorID = Book.Author
''')
results = cursor.fetchall()
# Печать результатов
for result in results:
    print(f'Автор: {result[0]}, Название: {result[1]}')
print()


# Объединение  названия книги, имени автора, жанра и года публикации
# (таблицы Book, Author, Genre)

print('********* INNER JOIN (Author, Book, Genre) *********')
cursor.execute('''
        SELECT Book.Title, Author.Name, Genre.GenreName, Book.PublicationYear
        FROM Book
        JOIN Author ON Book.Author = Author.AuthorID
```

```
        JOIN Genre ON Book.Genre = Genre.GenreID
    ''')
results = cursor.fetchall()

# Печать результатов
for result in results:
    print(f'Название: {result[0]}, Автор: {result[1]}, Жанр: {result[2]}, Год
публикации: {result[3]}')
print()


# подсчет числа экземпляров каждой книги
print('********* INNER JOIN (Book, BookInstance)  COUNT  GROUP BY  *********')
cursor.execute('''
    SELECT b.Title, COUNT(bi.BookInstanceID) AS NumberOfCopies
    FROM Book b
    JOIN BookInstance bi ON b.BookID = bi.Book
    GROUP BY b.BookID
''')
# Получение и вывод результатов
results = cursor.fetchall()
for row in results:
    print(f"Книга: {row[0]}, Количество экземпляров: {row[1]}")
print()


# Поиск книг по заданному жанру (таблицы Book и Genre)
# Использована фильтрация по определенному жанру, например, 'роман'
# Название жанра передается в запрос через переменную genre_name

genre_name = 'роман'

# SQL-запрос с использованием переменной
cursor.execute('''
SELECT Book.Title, Genre.GenreName
FROM Book
JOIN Genre ON Book.Genre = Genre.GenreID
WHERE Genre.GenreName = ?
''', (genre_name,))  # используем переменную в запросе

# Получение и вывод результатов
print('**********  INNER JOIN  ( Book, Genre для genre_name = роман) **********')
results = cursor.fetchall()
if results:
    for row in results:
        print(f'Название книги: {row[0]}, Жанр: {row[1]}')
else:
    print(f'Нет книг жанра "{genre_name}".')
print()
```

```
# Не забудьте закрыть соединение с базой данных, когда закончите работу

conn.close()
```

Результаты работы программы:

```
********** INNER JOIN (Author, Book) **********
Автор: М. Горький, Название: Мать
Автор: М. Горький, Название: На дне
Автор: А. Чехов, Название: Каштанка
Автор: А. Чехов, Название: Вишневый сад
Автор: Т. Шевченко, Название: Гайдамаки
Автор: Л. Толстой, Название: Война и мир
Автор: Т. Шевченко, Название: Катерина

********* INNER JOIN (Author, Book, Genre) *********
Название: Мать, Автор: М. Горький, Жанр: роман, Год публикации: 1996
Название: На дне, Автор: М. Горький, Жанр: драма, Год публикации: 1992
Название: Каштанка, Автор: А. Чехов, Жанр: повесть, Год публикации: 2017
Название: Вишневый сад, Автор: А. Чехов, Жанр: повесть, Год публикации: 2017
Название: Гайдамаки, Автор: Т. Шевченко, Жанр: поэма, Год публикации: 1930
Название: Война и мир, Автор: Л. Толстой, Жанр: роман, Год публикации: 2025
Название: Катерина, Автор: Т. Шевченко, Жанр: поэма, Год публикации: 1950

********* INNER JOIN (Book, BookInstance)  COUNT  GROUP BY  *********
Книга: Мать, Количество экземпляров: 2
Книга: На дне, Количество экземпляров: 2
Книга: Каштанка, Количество экземпляров: 2
Книга: Вишневый сад, Количество экземпляров: 1
Книга: Гайдамаки, Количество экземпляров: 1
Книга: Война и мир, Количество экземпляров: 1

**********  INNER JOIN  ( Book, Genre для genre_name = роман) **********
Название книги: Мать, Жанр: роман
Название книги: Война и мир, Жанр: роман
```

Программа LAB_BASE_3b.py  Вложенные запросы

```
import sqlite3
import os
import sys

# Путь к файлу базы данных SQLite
db_file = 'library.db'


# Если файл базы данных существует, просто подключаемся к нему
# и создаем объект курсор для выполнения SQL-запросов
if  os.path.isfile(db_file):
    conn = sqlite3.connect(db_file)
    cursor = conn.cursor()
else:
    print(f"Файл базы данных {db_file} не существует.")
    sys.exit()  # Завершаем программу



# ******************************
# Использование вложенных запросов
# ******************************

print()
# Объединение имени названия книги и имени автора (таблицы Book, Author )
# При условии что у книги один автор
# Использование оператора "=" в подзапросе
print('********* SELECT (SELECT from Author) from Book *********')
cursor.execute('''
    SELECT Book.Title,
        (SELECT Author.Name
        FROM Author
        WHERE Author.AuthorID = Book.Author)
FROM Book;
''')
# Получаем результаты
results = cursor.fetchall()
# Печать результатов
for i, result in enumerate(results, start=1):  # начинаем нумерацию с 1
  print(f'{i}. Книга: {result[0]}    Автор: {result[1]} ')  # Печатаем номер и
название книги
print()


# Использование оператора IN  в подзапросе для поиска книг,
# у которых есть хотя бы один доступный экземпляр в таблице BookInstance,
# не учитывая количество доступных экземпляров.
# У книги много экземпляров, поэтому использован IN
```

```
status = 'Доступен'  # Значение, которое нужно передать в запрос
print('********* SELECT from Book (SELECT from BookInstance)  *********')
cursor.execute('''
    SELECT Title
    FROM Book
    WHERE BookID IN (SELECT BookID FROM BookInstance WHERE VolumeStatus = ?)
''', (status,))  # Передаем переменную 'status' в запрос

# Получаем результаты
results = cursor.fetchall()

# Печать результатов
if results:
    print("Книги с доступными экземплярами:")
    for i, result in enumerate(results, start=1):  # начинаем нумерацию с 1
        print(f'{i}. {result[0]}')  # Печатаем номер и название книги
else:
    print("Нет книг с доступными экземплярами.")
print()


# Вложенный запрос на выбор книг конкретного автора
print('******* SELECT from Book (SELECT from Author)  *******')
name = 'М. Горький'
cursor.execute('''
    SELECT Title  FROM Book
    WHERE Author = (SELECT AuthorID FROM Author WHERE Name = ?)
''', (name,))  # Передаем имя как параметр через переменную name

# Получаем результаты
results = cursor.fetchall()
# Печать результатов
print(f'Автор книг: {name}')
for i, result in enumerate(results, start=1):  # начинаем нумерацию с 1
  print(f'{i}. {result[0]}')  # Печатаем номер и название книги
print()




# Напишите вложенный запрос, который выводит название жанров и всех книг каждого
жанра.
# Используйте таблицы Book и Genre.
# Пример запроса должен вернуть список жанров и соответствующие им книги.



# Напишите вложенный запрос, который выводит название указанного жанра и всех
книг этого жанра.
# Используйте таблицы Genre и Book.
```

```
# Передайте жанр в запрос как параметр через переменную genre



# Не забудьте закрыть соединение с базой данных, когда закончите работу
conn.close()
```

Результаты работы программы:

```
********* SELECT (SELECT from Author) from Book *********
1. Книга: Мать    Автор: М. Горький
2. Книга: На дне    Автор: М. Горький
3. Книга: Каштанка    Автор: А. Чехов
4. Книга: Вишневый сад    Автор: А. Чехов
5. Книга: Гайдамаки    Автор: Т. Шевченко
6. Книга: Война и мир    Автор: Л. Толстой
7. Книга: Катерина    Автор: Т. Шевченко

********* SELECT from Book (SELECT from BookInstance) *********
Книги с доступными экземплярами:
1. Мать
2. На дне
3. Каштанка
4. Вишневый сад
5. Гайдамаки
6. Война и мир
7. Катерина

******* SELECT from Book (SELECT from Author) *******
Автор книг: М. Горький
1. Мать
2. На дне
```

## *Задание на самостоятельную работу*

Задание на JOIN

- Напишите запрос, который выводит название жанров и всех книг каждого жанра. Используйте таблицы Genre и Book.
- Напишите запрос, который выводит имена авторов и общее количество книг, написанных каждым автором; используйте таблицы Author и Book. Используйте для подсчета книг функцию COUNT.
- Создайте запрос, который выводит название конкретной книги, имя автора и количество экземпляров этой книги. Используйте таблицы Book, Author и BookInstance. Название книги передавайте в запрос через переменную, например title='Мать'.
- Создайте запрос, который выводит названия книг и имена авторов для тех книг, у которых нет ни одного экземпляра в библиотеке. Используйте таблицы Book,

Author и BookInstance. Используйте конструкцию WHERE BookInstance.BookInstanceID IS NULL.

Вложенные запросы

- Напишите вложенный запрос, который выводит название жанров и всех книг каждого жанра. Используйте таблицы Genre и Book. Пример запроса должен вернуть список жанров и соответствующие им книги.
- Напишите вложенный запрос, который выводит название указанного жанра и всех книг этого жанра. Используйте таблицы Genre и Book. Передайте жанр в запрос как параметр через переменную.