Кафедра ЦТ
Институт информационных технологий
РТУ МИРЭА

Дисциплина «Разработка баз данных»

Практическая работа №2. Многотабличные запросы и теоретико-множественные операции.

Постановка задачи: основываясь на индивидуальной схеме данных, составьте необходимые запросы:

Задание 1: демонстрация различных типов соединений.

1. Запрос с **INNER JOIN**: подсчитайте количество связанных записей между таблицами *(например, «сколько лекарств у каждого производителя?»)*
2. Запрос с **LEFT JOIN**: проанализируйте наличие или отсутствие связей *(например, «сколько лекарств у каждого производителя, включая тех, у кого лекарств нет?»)*
3. Запрос с **RIGHT JOIN** и **WHERE**... **IS NULL** (паттерн «анти-соединение»): найдите и подсчитайте записи без связей *(например, «сколько лекарств не имеют производителя в базе?»)*
4. Запрос с **FULL JOIN**: получите общую статистику – сколько всего связанных записей, и сколько записей без связей.
5. Запрос с **CROSS JOIN**: сформировать декартово произведение всех записей одной таблицы со всеми записями другой, создав тем самым все возможные комбинации строк между ними.

(продолжение на следующем слайде)

Практическая работа №2. Многотабличные запросы и теоретико-множественные операции.

Постановка задачи: основываясь на индивидуальной схеме данных, составьте необходимые запросы:

Задание 2: применение теоретико-множественных операторов.

На основе индивидуальной схемы данных составить и выполнить три запроса, демонстрирующих практическое применение операторов **UNION**, **INTERSECT** и **EXCEPT**.

1. **UNION**: составить единый список из данных двух разных таблиц (столбцы должны быть совместимы по типу).
2. **INTERSECT**: найти общие записи, которые удовлетворяют двум разным условиям или находятся в двух разных наборах данных.
3. **EXCEPT**: найти записи, которые присутствуют в одном наборе данных, но отсутствуют в другом.

(начало на предыдущем слайде)

1. Основы соединения таблиц (JOIN)

Изучаемый ранее процесс нормализации приводит к **разделению** **данных** на несколько логических сущностей (**таблиц**).

Операторы **JOIN** являются фундаментальным механизмом SQL, который позволяет «**собирать**» эти разделённые данные обратно в **единое осмысленное представление**, для дальнейшего анализа и формирования отчетов.

1. Основы соединения таблиц (JOIN)

Для извлечения данных из **нескольких таблиц** используется оператор **JOIN**.

Общий синтаксис:

```
SELECT
    table1.column,
    table2.column
FROM
    table1
[INNER | LEFT | RIGHT | FULL] JOIN
    table2
ON
   table1.common_column =
   table2.common_column
[WHERE condition];
```

Краткое описание:

SELECT – указывает столбцы из обеих таблиц, которые нужно выбрать. **Крайне рекомендуется** указывать **таблицу** *(например, table1.column)*, чтобы избежать неоднозначности.

FROM – указывает первую («**левую**») таблицу для соединения.

JOIN – указывает вторую («**правую**») таблицу, которую нужно присоединить. Тип соединения (**INNER**, **LEFT** и т.д.) определяет, как именно будут объединены строки.

ON – указывает условие соединения. Это ключевая часть, которая определяет, по какому **общему полю** таблицы связаны друг с другом.

1. Основы соединения таблиц (JOIN)

Также существует оператор **CROSS JOIN** *(декартово произведение)*:

Общий синтаксис:

```
SELECT
    table1.column,
    table2.column
FROM
    table1
CROSS JOIN
    table2;
```

Краткое описание:

CROSS JOIN: создаёт декартово произведение – каждая строка из первой таблицы соединяется с каждой строкой из второй.

ВАЖНО: для **CROSS JOIN** не используется предложение **ON**.

1. Основы соединения таблиц (JOIN)

Тип соединения	Краткое описание	Основной сценарий использования
INNER JOIN	Возвращает только те строки, для которых найдено совпадение в **обеих таблицах**.	**Получение строго связанных данных** (например, заказы и их клиенты).
LEFT JOIN	Возвращает **все строки из левой** таблицы и только **совпадающие из правой**.	**Получение полного списка чего-либо с дополнением** (например, все клиенты и их заказы, даже если заказов нет).
RIGHT JOIN	Возвращает **все строки из правой** таблицы и только **совпадающие из левой**.	**Когда фокус запроса на «правой» таблице.** Например, "показать ВСЕ лекарства и добавить информацию об их производителях, если она есть".
FULL JOIN	Возвращает **все строки из обеих таблиц**, соединяя их там, где это возможно.	**Полное сведение двух списков без потери данных** (например, все сотрудники и все отделы, даже если в отделе нет сотрудников).
CROSS JOIN	Возвращает декартово произведение – **все возможные комбинации строк** из обеих таблиц.	**Генерация тестовых данных или комбинаторных наборов** (например, все размеры для всех цветов товара).

1. Основы соединения таблиц (JOIN)

Псевдонимы (**aliases**) дают временные, короткие имена таблицам или столбцам в запросе, за счёт чего повышается читаемость и уменьшается длина кода.

Псевдонимы абсолютно **необходимы** при использовании «**само-соединений**» (**SELF-JOIN**), когда таблица ссылается **сама на себя**.

```
-- Этот код синтаксически невозможен без псевдонимов 'e' и 'm'
SELECT
    ...
FROM
    pharmacists AS e  -- Таблица в роли "сотрудников"
LEFT JOIN
    pharmacists AS m  -- Та же таблица в роли "менеджеров"
ON
    e.manager_id = m.pharmacist_id;
```

2. Продвинутые техники и паттерны соединений

Для получения комплексных отчетов часто требуется **соединять более двух таблиц**.

Чтобы решить эту задачу, СУБД **последовательно** выполняет **соединения**:

- Соединяются **первые две** таблицы, и их результат рассматривается **как одна** временная, **виртуальная таблица**.
- Затем эта виртуальная таблица **соединяется со следующей таблицей**, формируя новую **виртуальную таблицу**. ***При необходимости, этот шаг повторяется*.**

Ограничений на количество соединяемых таблиц **нет**.

2. Продвинутые техники и паттерны соединений

Теоретико-множественные операторы (**UNION**, **INTERSECT**, **EXCEPT**)

Эти операторы предназначены для **объединения результирующих наборов** от двух или более **независимых** **SELECT-запросов** в одну **итоговую таблицу**.

В отличие от **JOIN**, который «**склеивает**» таблицы **по горизонтали** (добавляя столбцы), эти операторы «**склеивают**» результаты **по вертикали** (добавляя **строки**).

2. Продвинутые техники и паттерны соединений

Теоретико-множественные операторы:

- Оператор **UNION** **объединяет** результаты двух запросов в один набор данных, при этом **удаляя** все **дублирующиеся строки**.
- Оператор **UNION ALL** работает быстрее, так как он также **объединяет** результаты, но **сохраняет** **все дубликаты**.
- Оператор **INTERSECT** *(пересечение)* **возвращает** только те строки, которые присутствуют в результатах **обоих запросов**. Порядок запросов не имеет значения.
- Оператор **EXCEPT** *(разность)* **возвращает** строки из **первого запроса**, которых **нет во втором**. Для этого оператора **порядок** **запросов** **критически важен**.

2. Продвинутые техники и паттерны соединений

Общие правила и **принципы** работы теоретико-множественных операторов:

- **Работа с результатами**: операторы применяются **не к самим таблицам**, а **к данным**, которые были возвращены каждым **отдельным** SELECT-запросом.
- **Совместимость столбцов**: это главное правило. **Все** SELECT-запросы в конструкции должны возвращать **одинаковое количество** столбцов, и **типы данных** этих столбцов должны быть **совместимы** *(например, нельзя объединить число со строкой)*.
- **Имена столбцов**: итоговая таблица получает **имена столбцов** из **первого** SELECT-запроса.
- **Удаление дубликатов**: **по умолчанию**, операторы UNION, INTERSECT и EXCEPT **удаляют дубликаты** из итогового набора данных, аналогично работе SELECT DISTINCT. **Для сохранения дубликатов** при объединении используется **UNION ALL**.

Спасибо за внимание