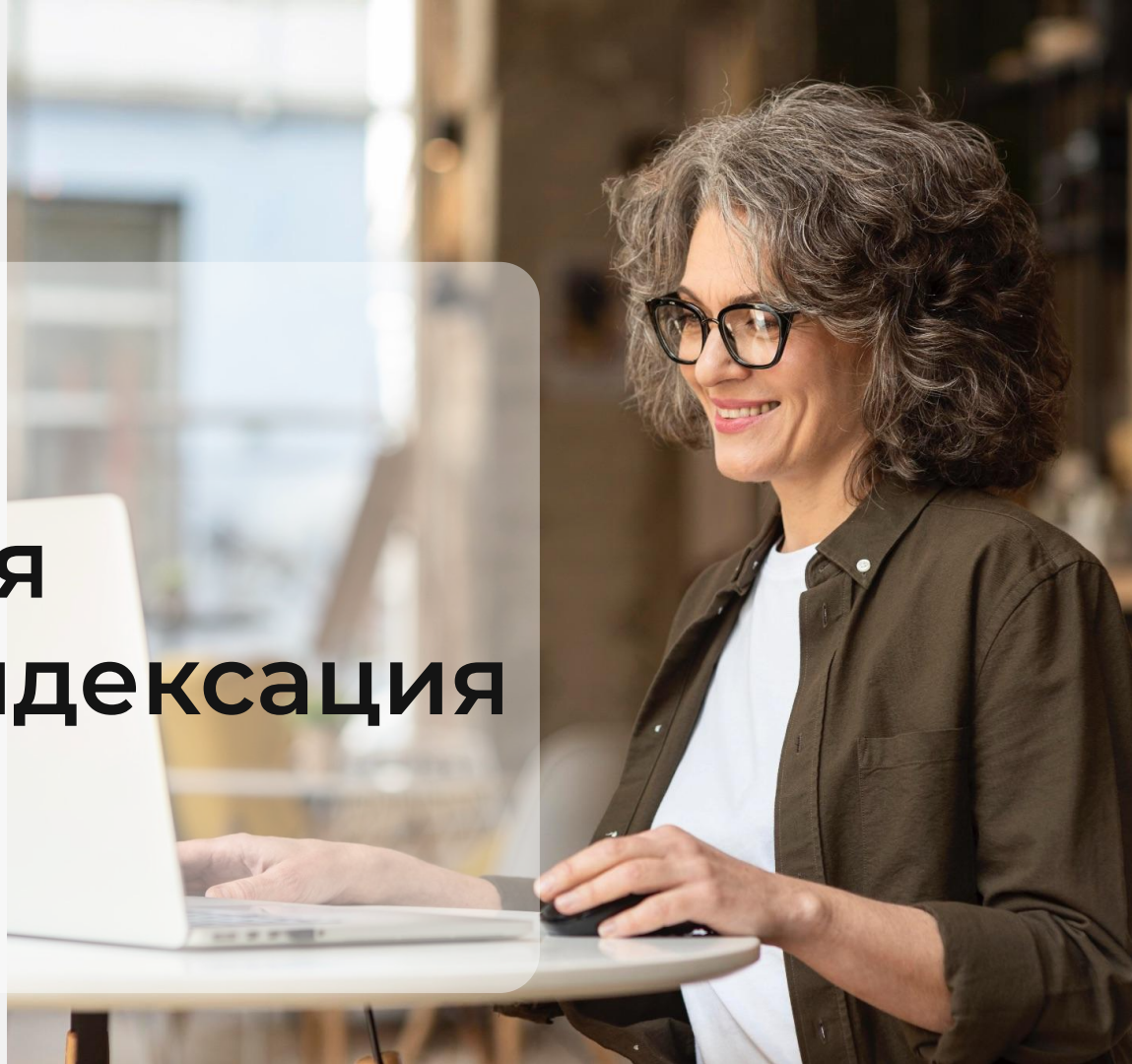


Node.JS

Оптимизация запросов. Индексация



Преподаватель

Портрет

Имя Фамилия

Текущая должность

Количество лет опыта

Какой у Вас опыт - ключевые кейсы

Самые яркие проекты

Дополнительная информация по вашему усмотрению

Корпоративный e-mail

Социальные сети (по желанию)

Важно

- 

Камера должна быть включена на протяжении всего занятия
- 

В течение занятия вопросы задавать в чате или когда преподаватель спрашивает, есть ли у Вас вопросы
- 

Вести себя уважительно и этично по отношению к остальным участникам занятия
- 

Организационные вопросы по обучению решаются с кураторами, а не на тематических занятиях
- 

Во время занятия будут интерактивные задания, будьте готовы включить камеру или демонстрацию экрана по просьбе преподавателя

Повторение



Оконные функции смещения и выбора



Общий синтаксис



Основные оконные функции смещения и выбора

План занятия

- Оптимизация запросов
- Типы индексов в MySQL
- Как индексы ускоряют запросы
- Минусы индексации
- Анализ запросов с помощью EXPLAIN

База данных с доступом на запись

hostname: ich-edit.edu.itcareerhub.de

MYSQL_USER: ich1

MYSQL_PASSWORD: ich1_password_ilovedbs



ОСНОВНОЙ БЛОК





Оптимизация запросов



Индексация

Это способ организации данных, который ускоряет выполнение запросов.



Индекс

Это структура данных (чаще всего дерево), которая помогает быстрее находить нужные записи в таблице без необходимости полного сканирования всей таблицы.



ВОПРОСЫ





Типы индексов в MySQL

Типы индексов в MySQL

PRIMARY KEY

UNIQUE INDEX

INDEX (или обычный
индекс)

FULLTEXT INDEX

COMPOSITE INDEX
(составной индекс)



PRIMARY KEY

Это индекс, который автоматически создается для поля, объявленного как первичный ключ. Этот индекс гарантирует уникальность значений и быстрый поиск по ключу

Пример для PRIMARY KEY

```
CREATE TABLE employees (  
    id INT PRIMARY KEY,  
    name VARCHAR(100),  
    age INT  
);
```



UNIQUE INDEX

Это индекс, который гарантирует уникальность значений в одном или нескольких столбцах

Пример для UNIQUE INDEX

```
CREATE UNIQUE INDEX idx_employee_email ON employees (email);
```



INDEX (или обычный индекс)

Этот индекс используется для ускорения поиска по одному или нескольким полям. Индекс не накладывает ограничений на уникальность значений

Пример для INDEX (или обычный индекс)

```
CREATE INDEX idx_employee_name ON employees (name);
```



FULLTEXT INDEX

Этот тип индекса используется для полнотекстового поиска по текстовым полям

Пример для FULLTEXT INDEX

```
CREATE FULLTEXT INDEX idx_article_text ON articles (content);
```



COMPOSITE INDEX (составной индекс)

Это индекс, созданный на несколько столбцов одновременно. Он полезен для запросов, которые используют сразу несколько условий по разным полям

Пример для COMPOSITE INDEX (составной индекс)

```
CREATE INDEX idx_employee_name_age ON employees (name, age);
```



ВОПРОСЫ





Как индексы ускоряют запросы

Как индексы ускоряют запросы



Индексы помогают MySQL быстрее находить строки, соответствующие условиям запроса, уменьшая количество данных, которые нужно просмотреть.

Работа без индекса



Пример

```
SELECT * FROM employees WHERE age = 30;
```

Пояснение

MySQL будет просматривать каждую строку в таблице, чтобы найти все записи, где age = 30

Работа с индексом



Пример

```
CREATE INDEX idx_employee_age ON
employees (age);
```

```
SELECT * FROM employees WHERE age = 30;
```

Пояснение

С индексом MySQL будет использовать структуру индекса, чтобы быстро найти строки, соответствующие возрасту 30.

Пример работы с индексами

Рабочая таблица

```
CREATE TABLE employees (  
    id INT PRIMARY KEY AUTO_INCREMENT,  
    name VARCHAR(100),  
    age INT,  
    salary INT,  
    department_id INT  
);
```

Пример работы с индексами

1. В таблице много записей
2. Вы часто выполняете запросы на поиск сотрудников по полю age

Имеет смысл создать индекс на это поле

```
CREATE INDEX idx_age ON employees (age);
```

Теперь при выполнении запроса MySQL будет использовать индекс для быстрого доступа к нужным строкам

```
SELECT * FROM employees WHERE age = 30;
```



ВОПРОСЫ





Минусы индексации

Недостатки

Использование
дискового пространства

Замедление операций
вставки, обновления и
удаления

Неэффективность для
очень маленьких таблиц

Составные индексы



Если запросы часто содержат несколько условий на разные поля, можно создать составной индекс для этих полей

Пример с составным индексом

```
CREATE INDEX idx_department_age_salary ON employees (department_id, age, salary);
```

В этом примере запросы, использующие одновременно столбцы department_id, age и salary, будут работать быстрее

Важно!



Не индексируйте все подряд. Индексация всех столбцов подряд может замедлить операции вставки и обновления. Создавайте индексы только на те поля, которые действительно необходимы для ускорения поиска.

Использование индексов для сортировки и группировки



Индексы могут помочь ускорить запросы с `ORDER BY` и `GROUP BY`, если индекс включает столбцы, по которым производится сортировка или группировка.

Использование индексов для сортировки и группировки



Пример

```
CREATE INDEX idx_name_age ON employees
(name, age);
```

Запрос

```
SELECT * FROM employees ORDER BY name,
age;
```



ВОПРОСЫ





Анализ запросов с помощью EXPLAIN



EXPLAIN

Это команда, которая позволяет увидеть, как MySQL будет выполнять запрос.

Использование EXPLAIN



Пример

```
EXPLAIN SELECT * FROM employees WHERE  
age = 30;
```

Вывод покажет

- Какой тип сканирования будет использоваться (например, полный скан таблицы или использование индекса).
- Оценку количества строк, которые должны быть прочитаны.
- Какой индекс будет использоваться (если применимо).



ВОПРОСЫ





ПРАКТИЧЕСКАЯ РАБОТА



Таблица students

1. Создайте таблицу students с такими столбцами: id (INT), name (VARCHAR), age (INT), grade (DECIMAL).
2. Заполните таблицу несколькими строками.
3. Создайте индекс на столбец age, чтобы ускорить поиск по возрасту.
4. Напишите запрос, который выбирает всех студентов определенного возраста.
5. Просмотрите план выполнения запроса с помощью команды EXPLAIN.



ВОПРОСЫ



Заключение

