Index Report

This report provides B-tree index recommendations for the **Meeting Management System** project's database models.

1. Index: `Student.name`

- **Index Type**: B-tree Index
- **Supported Queries**:
  - Search students by name:

    ```
    students = Student.objects.filter(name__icontains='Tianzhi')
    ```

  - Sort student list by name:

    ```
    students = Student.objects.order_by('name')
    ```

- **Optimization Reasons**:
  - **Search Optimization**: Increases the speed of name-based queries, enhancing the user experience when searching and filtering students.

2. Index: `Club.name`

- **Index Type**: B-tree Index
- **Supported Queries**:
  - Search clubs by name:

    ```
    clubs = Club.objects.filter(name__icontains='Science')
    ```

  - Sort club list by name:

    ```
    clubs = Club.objects.order_by('name')
    ```

- **Optimization Reasons**:
  - **Query Efficiency**: Accelerates name-based search and sorting operations, especially when managing and displaying club lists.

3. Index: `Club.address`

- **Index Type**: B-tree Index

- **Supported Queries**:
  - Search clubs by address:

```
clubs = Club.objects.filter(address__icontains='**')
```

  - Perform geographical location queries by address:

```
clubs = Club.objects.filter(address__icontains='**')
```

- **Optimization Reasons**:
  - **Geographical Query Optimization**: Enhances the performance of address-based queries, supporting location-related feature requirements.

4. Index: `Room.building` and `Room.number`

- **Index Type**: Composite Index (`building`, `number`)
- **Supported Queries**:
  - Find room by building and number:

```
room = Room.objects.get(building='Science', number='101')
```

  - Find rooms by building:

```
rooms = Room.objects.filter(building='Science')
```

- **Optimization Reasons**:
  - **Unique Identification**: Ensures each room is uniquely identified within a building.
  - **Query Efficiency**: Speeds up lookups based on building and room number, especially when scheduling meetings to quickly locate rooms.

5. Index: `Meeting.date`

- **Index Type**: B-tree Index
- **Supported Queries**:
  - Filter meetings by date:

```
meetings = Meeting.objects.filter(date='2024-12-01')
```

  - Sort meetings by date:

```
meetings = Meeting.objects.order_by('date')
```

  - Count meetings on a specific date:

```
meeting_count = Meeting.objects.filter(date='2024-12-01').count()
```

- **Optimization Reasons**:
  - **Date Query Optimization**: Improves the performance of date-based filtering and sorting, supporting the generation of date-classified reports and views.

6. Index: `Meeting(date, club)`

- **Index Type**: Composite Index (`date`, `club`)
- **Supported Queries**:
  - Filter meetings by date and club:

```
meetings = Meeting.objects.filter(date__gte='2024-11-01', date__lte='2024-11-15', club=club_instance)
```

- **Optimization Reasons**:
  - **Multi-Condition Query Optimization**: Enhances the performance of filtering based on both date and club, supporting complex reporting and statistical analysis needs.

7. Index: `Meeting.duration`

- **Index Type**: B-tree Index
- **Supported Queries**:
  - Find meetings with duration exceeding a specific value:

```
meetings = Meeting.objects.filter(duration__gte=timedelta(hours=2))
```

- **Optimization Reasons**:
  - **Duration Query Optimization**: Increases the efficiency of duration-based queries, supporting time management and resource allocation functionalities.

8. Index: `Meeting.invited_count` and `Meeting.accepted_count`

- **Index Type**: Composite Index (`invited_count`, `accepted_count`)
- **Supported Queries**:

- Generate statistical reports on invited and accepted counts:

```
report = Meeting.objects.aggregate(
    average_invited=Avg('invited_count'),
    average_accepted=Avg('accepted_count')
)
```

- **Optimization Reasons**:
  - **Report Generation Optimization**: Accelerates aggregation calculations based on invited and accepted counts, enhancing the performance and responsiveness of report generation.

9. Index: `MeetingOrganizer.meeting` and `MeetingOrganizer.student`

- **Index Type**: Composite Index (`meeting`, `student`)
- **Supported Queries**:
  - Find organizers for a specific meeting:

```
organizers = MeetingOrganizer.objects.filter(meeting=meeting_instance)
```

  - Find meetings organized by a specific student:

```
meetings = MeetingOrganizer.objects.filter(student=student_instance)
```

- **Optimization Reasons**:
  - **Query Efficiency**: Speeds up joint queries involving both meeting and student fields, enhancing the performance of organizer management.
