SIXKUL Database Technical Documentation

Sistem Informasi Ekstrakurikuler - Database Schema & Implementation

> ***Document Version***: *1.0*
> ***Last Updated***: *2025-12-20*
> ***Database Type***: *PostgreSQL (Supabase Cloud-Hosted)*
> ***ORM***: *Prisma Client*

Table of Contents

Overview

SIXKUL is a High School Extracurricular Management System designed to manage students, advisors (Pembina), extracurricular activities, enrollments, schedules, attendance, and announcements. The system integrates with **Clerk** for authentication and user management.

Key Statistics

- **Total Models**: 11
- **Total Enums**: 5
- **Authentication Provider**: Clerk
- **Database Provider**: PostgreSQL (Supabase)
- **ORM**: Prisma
- **Generated Client Location**: `src/generated/prisma`

Database Schema Architecture

Technology Stack

```
graph LR
    A[Next.js 16 App Router] --> B[Prisma ORM]
    B --> C[PostgreSQL Database]
    A --> D[Clerk Auth]
    D -.sync.-> C

    style A fill:#0070f3
    style B fill:#2d3748
    style C fill:#336791
    style D fill:#6c47ff
```

Schema Configuration

```
generator client {
  provider = "prisma-client-js"
  output   = "../src/generated/prisma"
}

datasource db {
  provider = "postgresql"
}
```

Enumerations

The database uses 5 enums to enforce data integrity and standardize status values.

1. UserRole

Defines the three user types in the system.

```
enum UserRole {
  ADMIN      // System administrators
  PEMBINA    // Extracurricular advisors/coaches
  SISWA      // Students
}
```

Usage: Determines access control and feature availability across the application.

---

2. ExtracurricularStatus

Tracks the operational status of extracurricular activities.

```
enum ExtracurricularStatus {
  ACTIVE     // Currently accepting enrollments and conducting sessions
  INACTIVE   // Not accepting new enrollments; legacy/archived
}
```

Business Logic: Only `ACTIVE` extracurriculars are displayed in student enrollment flows.

---

3. EnrollmentStatus

Manages the lifecycle of a student's participation in an extracurricular.

```
enum EnrollmentStatus {
  PENDING    // Application submitted, awaiting Pembina approval
  ACTIVE     // Currently participating
  REJECTED   // Application denied
  ALUMNI     // Completed participation (graduated/left)
```

```
  CANCELLED  // Student withdrew enrollment
}
```

State Transitions:

- `PENDING` → `ACTIVE` (approved by Pembina)
- `PENDING` → `REJECTED` (denied by Pembina)
- `ACTIVE` → `ALUMNI` (academic year completion)
- `ACTIVE` → `CANCELLED` (student withdrawal)

---

4. AttendanceStatus

Records student attendance with 5 possible states.

```
enum AttendanceStatus {
  PRESENT     // Attended the session
  SICK        // Absent due to illness (with permission)
  PERMISSION  // Absent with permission (izin)
  ALPHA       // Absent without permission (unexcused)
  LATE        // Arrived late but attended
}
```

Statistics Impact:

- `PRESENT` and `LATE`: Count toward participation
- `SICK` and `PERMISSION`: Excused absences
- `ALPHA`: Unexcused absence (affects performance metrics)

---

5. NotificationType

Categorizes notifications for filtering and routing.

```
enum NotificationType {
  ANNOUNCEMENT    // New announcement posted
  ATTENDANCE      // Attendance recorded/updated
  SCHEDULE        // Schedule change or new session
  ENROLLMENT      // Enrollment status change
}
```

UI Integration: Used for notification icon selection and navigation routing.

Data Models

The database consists of 11 interconnected models forming a comprehensive extracurricular management system.

Model Index

Model	Purpose	Key Relations

`User`	Central authentication entity linked to Clerk	`StudentProfile`, `PembinaProfile`, `Notification`, `Announcement`
`StudentProfile`	Student-specific data (one-to-one with User)	`User`, `Enrollment`, `StudentPreferences`
`PembinaProfile`	Pembina-specific data (one-to-one with User)	`User`, `Extracurricular`
`Extracurricular`	Extracurricular activity/club	`PembinaProfile`, `Enrollment`, `Schedule`, `Session`, `Announcement`
`Enrollment`	Student ↔ Extracurricular junction	`StudentProfile`, `Extracurricular`, `Attendance`, `Notification`
`Schedule`	Recurring weekly schedule template	`Extracurricular`, `Session`
`Session`	Concrete calendar-dated event	`Extracurricular`, `Schedule`, `Attendance`
`Attendance`	Attendance tracking per enrollment per date	`Enrollment`, `Session`
`Announcement`	Messages from Pembina to students	`Extracurricular`, `User` (author)
`Notification`	User notifications	`User`, `Enrollment` (context)
`StudentPreferences`	Student UI/notification preferences	`StudentProfile`

---

1. User Model

Purpose: Central user entity synchronized with Clerk authentication.

```
model User {
  id            String   @id @default(cuid())
  clerk_id      String   @unique  // Clerk user ID (e.g., "user_2abc123...")
  username      String   @unique  // Username for display and auth
  email         String?           // Optional, synced from Clerk
  full_name     String
  role          UserRole
  avatar_url    String?
  created_at    DateTime @default(now())
  updated_at    DateTime @updatedAt

  // Relations
  studentProfile StudentProfile?
  pembinaProfile PembinaProfile?
  notifications  Notification[]
  announcements  Announcement[]

  @@map("users")
}
```

Key Features

- **Clerk Integration**: `clerk_id` is the primary link to Clerk's authentication system
- **Role-Based Access**: `role` enum determines permissions and UI visibility
- **Polymorphic Profiles**: User can have either `StudentProfile` OR `PembinaProfile` (one-to-one)
- **Optional Email**: Email may not exist for all users (synced from Clerk when available)

Current Seeded Data

- **Total Users**: 13
  - 3 ADMIN users
  - 5 PEMBINA users
  - 5 SISWA users

Sample User Records:

Role	Username	Email	Full Name	Clerk ID
ADMIN	admin_sixkul	admin@sixkul.sch.id	Administrator SIXKUL	`user_xxx...`
PEMBINA	pembina_budi	pembina@sixkul.sch.id	Budi Santoso	`user_xxx...`
SISWA	student_siti	student@sixkul.sch.id	Siti Nurhaliza	`user_xxx...`

---

2. StudentProfile Model

Purpose: Student-specific profile data (one-to-one with User).

```
model StudentProfile {
  id            String  @id @default(cuid())
  user_id       String  @unique
  nis           String  @unique // Nomor Induk Siswa (Student ID Number)
  class_name    String  // e.g., "XII IPA 1"
  major         String  // e.g., "IPA", "IPS"
  phone_number  String?
  academic_year String? // e.g., "2024/2025" — exclusive source for Tahun Akademik

  // Relations
  user        User                @relation(fields: [user_id], references: [id], onDelete:
Cascade)
  enrollments Enrollment[]
  preferences StudentPreferences?

  @@map("student_profiles")
}
```

Key Features

- **Unique NIS**: Each student has a unique student identification number
- **Academic Classification**: `class_name` (e.g., "XII IPA 1") and `major` (IPA/IPS)
- **Academic Year Tracking**: `academic_year` field tracks the cohort (e.g., "2024/2025")
- **Cascade Deletion**: Deleting a User automatically deletes the StudentProfile

Current Seeded Data

5 Student Profiles:

NIS	Student Name	Class	Major	Phone Number
2024001	Siti Nurhaliza	XII IPA 1	IPA	+62812345679001
2024002	Andi Firmansyah	XII IPA 2	IPA	+62812345679002
2024003	Maya Sari	XI IPS 1	IPS	+62812345679003
2024004	Rizki Ramadhan	XI IPA 1	IPA	+62812345679004
2024005	Putri Ayu	X IPA 1	IPA	+62812345679005

---

3. PembinaProfile Model

Purpose: Pembina (advisor/coach) specific profile data.

```
model PembinaProfile {
  id           String  @id @default(cuid())
  user_id      String  @unique
  nip          String  @unique // Nomor Induk Pegawai (Employee ID Number)
  expertise    String? // Bidang keahlian (area of expertise)
  phone_number String?

  // Relations
  user             User              @relation(fields: [user_id], references: [id],
onDelete: Cascade)
  extracurriculars Extracurricular[]

  @@map("pembina_profiles")
}
```

Key Features

- **Unique NIP**: Each Pembina has a unique employee identification number
- **Expertise Field**: Describes the Pembina's area of specialization
- **One-to-Many Extracurriculars**: A Pembina can supervise multiple extracurricular activities

Current Seeded Data

5 Pembina Profiles:

NIP	Name	Expertise	Phone
197801152006041001	Budi Santoso	Teknologi Informasi & Olahraga	+62812345678901
198503212008042002	Dewi Lestari	Seni & Musik	+62812345678902
198007102009031003	Agus Prasetyo	Olahraga & Bela Diri	+62812345678903
199001302012042004	Rina Wijaya	Bahasa & Jurnalistik	+62812345678904
198805152015031005	Hendra Kusuma	Sains & Robotik	+62812345678905

---

4. Extracurricular Model

Purpose: Core entity representing extracurricular activities/clubs.

```
model Extracurricular {
  id          String                @id @default(cuid())
  name        String
  category    String // e.g., "Olahraga", "Seni", "Akademik"
  description String?
  logo_url    String?
  status      ExtracurricularStatus  @default(ACTIVE)
  pembina_id  String
  created_at  DateTime              @default(now())
  updated_at  DateTime              @updatedAt

  // Relations
  pembina       PembinaProfile @relation(fields: [pembina_id], references: [id], onDelete:
Restrict)
  enrollments   Enrollment[]
  schedules     Schedule[]
  sessions      Session[]
  announcements Announcement[]

  @@map("extracurriculars")
}
```

Key Features

- **Categorization**: Activities grouped by category (Olahraga, Seni, Teknologi, etc.)
- **Status Management**: `ACTIVE` or `INACTIVE` status controls visibility
- **Pembina Assignment**: Each extracurricular must have an assigned Pembina
- **Restrict Deletion**: Cannot delete Pembina if they have assigned extracurriculars

Current Seeded Data

5 Extracurricular Activities:

Name	Category	Status	Pembina	Description
Olahraga Bola Basket	Olahraga	ACTIVE	Budi Santoso	Kegiatan ekstrakurikuler basket untuk mengembangkan kemampuan olahraga dan kerja sama tim.
Robotik	Teknologi	ACTIVE	Budi Santoso	Mempelajari dan mengembangkan robot sederhana untuk kompetisi tingkat nasional.
Pemrograman dan Pengembangan Video Game	Teknologi	ACTIVE	Budi Santoso	Belajar membuat game menggunakan Unity, Unreal Engine, dan Godot.

Tenis Meja	Olahraga	ACTIVE	Budi Santoso	Ekstrakurikuler tenis meja untuk meningkatkan refleks dan konsentrasi.
Drum Band	Seni	ACTIVE	Budi Santoso	Marching band sekolah yang tampil di berbagai acara dan kompetisi.

> *[!NOTE] All seeded extracurriculars are assigned to the same Pembina (Budi Santoso) for testing purposes.*

---

5. Enrollment Model

Purpose: Junction table managing many-to-many relationship between Students and Extracurriculars.

```
model Enrollment {
  id                String           @id @default(cuid())
  student_id        String
  extracurricular_id String
  status            EnrollmentStatus @default(PENDING)
  joined_at         DateTime         @default(now())
  academic_year     String // e.g., "2024/2025"
  updated_at        DateTime         @updatedAt

  // Relations
  student         StudentProfile  @relation(fields: [student_id], references: [id],
onDelete: Cascade)
  extracurricular Extracurricular @relation(fields: [extracurricular_id], references: [id],
onDelete: Cascade)
  attendances     Attendance[]
  notifications   Notification[]

  // Prevent duplicate enrollments
  @@unique([student_id, extracurricular_id])
  @@map("enrollments")
}
```

Key Features

- **Unique Constraint**: Prevents duplicate enrollments (one student per extracurricular)
- **Status Tracking**: Lifecycle management through `EnrollmentStatus` enum
- **Academic Year Scoped**: Enrollments tied to specific academic years
- **Cascade Deletion**: Deleting student or extracurricular removes enrollments

Current Seeded Data

3 Enrollments (all for student "Siti Nurhaliza"):

Student	Extracurricular	Status	Academic Year	Joined At
Siti Nurhaliza	Olahraga Bola Basket	ACTIVE	2024/2025	Current timestamp

Siti Nurhaliza	Robotik	ACTIVE	2024/2025	Current timestamp
Siti Nurhaliza	Pemrograman dan Pengembangan Video Game	ACTIVE	2024/2025	Current timestamp

---

6. Schedule Model

Purpose: Recurring weekly schedule template for extracurricular activities.

```
model Schedule {
  id                String @id @default(cuid())
  extracurricular_id String
  day_of_week       String // e.g., "MONDAY", "TUESDAY"
  start_time        String // e.g., "14:00"
  end_time          String // e.g., "16:00"
  location          String // e.g., "Lapangan Basket", "Ruang Musik"

  // Relations
  extracurricular Extracurricular @relation(fields: [extracurricular_id], references: [id],
onDelete: Cascade)
  sessions        Session[]

  @@map("schedules")
}
```

Key Features

- **Recurring Templates**: Defines weekly recurring patterns (not concrete calendar dates)
- **Flexible Time Format**: Times stored as strings (HH:MM format)
- **Session Generation**: Used as a template to automatically generate `Session` records

Current Seeded Data

3 Schedule Templates:

Extracurricular	Day	Time	Location
Olahraga Bola Basket	SENIN	15:00 - 17:00	Lapangan Basket Sekolah
Robotik	RABU	14:00 - 16:00	Lab Komputer
Pemrograman dan Pengembangan Video Game	KAMIS	15:00 - 17:30	Lab Multimedia

---

7. Session Model

Purpose: Concrete calendar-dated extracurricular events/meetings.

```
model Session {
  id                String   @id @default(cuid())
  extracurricular_id String
  schedule_id       String?  // Optional: parent Schedule template
```

```
  date               DateTime // Calendar date of the session
  start_time         String   // e.g., "14:00"
  end_time           String   // e.g., "16:00"
  location           String
  notes              String?
  is_cancelled       Boolean  @default(false)
  created_at         DateTime @default(now())

  // Relations
  extracurricular Extracurricular @relation(fields: [extracurricular_id], references: [id],
onDelete: Cascade)
  schedule        Schedule?       @relation(fields: [schedule_id], references: [id],
onDelete: SetNull)
  attendances     Attendance[]

  @@map("sessions")
}
```

Key Features

- **Calendar-Specific**: Each session has a concrete `date` (not just day-of-week)
- **Template Link**: Optional `schedule_id` links back to the recurring template
- **Cancellation Support**: `is_cancelled` flag allows marking sessions as cancelled
- **Attendance Tracking**: Sessions can have multiple attendance records

Current Seeded Data

12 Sessions (generated for next 4 weeks from 3 schedule templates):

```
Sessions generated:
- Basket (SENIN): 4 sessions over 4 consecutive Mondays
- Robotik (RABU): 4 sessions over 4 consecutive Wednesdays
- Game Dev (KAMIS): 4 sessions over 4 consecutive Thursdays
```

Generation Algorithm:

```
// For each schedule template
for (let week = 0; week < 4; week++) {
  // Calculate next occurrence of target weekday
  // Create session with:
  //   - date: calculated calendar date
  //   - time/location: copied from template
  //   - schedule_id: reference to template
}
```

---

8. Attendance Model

Purpose: Track student attendance for each enrollment on specific dates.

```
model Attendance {
  id            String          @id @default(cuid())
  enrollment_id String
```

```
  session_id    String?          // Optional link to specific Session (concrete event)
  date          DateTime
  status        AttendanceStatus
  notes         String?
  created_at    DateTime         @default(now())

  // Relations
  enrollment Enrollment @relation(fields: [enrollment_id], references: [id], onDelete:
Cascade)
  session    Session?   @relation(fields: [session_id], references: [id], onDelete:
SetNull)

  // Prevent duplicate attendance records for same day
  @@unique([enrollment_id, date])
  @@map("attendances")
}
```

Key Features

- **Unique Per Day**: One attendance record per enrollment per date
- **Session Link**: Optional reference to specific `Session` (concrete event)
- **Status Tracking**: 5 attendance statuses (PRESENT, SICK, PERMISSION, ALPHA, LATE)
- **Notes Field**: Additional context for absences or late arrivals

Current Seeded Data

30 Attendance Records (10 records per enrollment × 3 enrollments):

Distribution Pattern:

- **70%** PRESENT
- **15%** SICK
- **10%** PERMISSION
- **5%** ALPHA

Sample Data:

```
For each of the 3 enrollments (Basket, Robotik, Game Dev):
  - 10 weekly attendance records (past 10 weeks)
  - Randomly distributed statuses based on above percentages
  - Notes added for non-PRESENT statuses
```

---

9. Announcement Model

Purpose: Messages/announcements from Pembina to extracurricular members.

```
model Announcement {
  id                 String   @id @default(cuid())
  extracurricular_id String
  author_id          String
  title              String
  content            String
  created_at         DateTime @default(now())
```

```
  // Relations
  extracurricular Extracurricular @relation(fields: [extracurricular_id], references: [id],
onDelete: Cascade)
  author          User            @relation(fields: [author_id], references: [id],
onDelete: Cascade)

  @@map("announcements")
}
```

Key Features

- **Scoped to Extracurricular**: Each announcement belongs to one extracurricular
- **Author Tracking**: Links to the User who created the announcement
- **Timestamp**: `created_at` for chronological ordering

Current Seeded Data

4 Announcements:

Extracurricular	Title	Created At	Author
Basket	Perubahan Jadwal Latihan Basket	Today	Budi Santoso
Robotik	Latihan Robotik Diliburkan	2 days ago	Budi Santoso
Game Dev	Selamat Datang di Ekstrakurikuler Game Development!	5 days ago	Budi Santoso
Basket	Pengumpulan Seragam Tim	1 week ago	Budi Santoso

Content Examples:

```
1. "Latihan minggu ini dipindahkan ke hari Kamis karena ada renovasi lapangan."
2. "Latihan robotik diliburkan minggu ini karena pembina berhalangan."
3. "Halo semua! Selamat bergabung di ekstrakurikuler Game Development. Persiapkan laptop
kalian!"
4. "Seragam tim basket sudah tersedia. Harap dikumpulkan ukuran masing-masing."
```

---

10. Notification Model

Purpose: System-generated notifications for users.

```
model Notification {
  id            String           @id @default(cuid())
  user_id       String
  enrollment_id String?          // Context anchor - links to enrollment for navigation
  type          NotificationType // Notification category
  title         String
  message       String
```

```
  is_read       Boolean          @default(false)
  created_at    DateTime         @default(now())

  // Relations
  user       User        @relation(fields: [user_id], references: [id], onDelete: Cascade)
  enrollment Enrollment? @relation(fields: [enrollment_id], references: [id], onDelete:
SetNull)

  @@map("notifications")
}
```

Key Features

- **Typed Notifications**: `NotificationType` enum for categorization
- **Context Linking**: Optional `enrollment_id` provides navigation context
- **Read Status**: `is_read` boolean for unread badge display
- **Cascade Deletion**: Notifications deleted when user is deleted

Current Seeded Data

0 Notifications (none seeded by default)

> *[!NOTE] Notifications are typically generated programmatically when events occur (e.g., new announcement, attendance recorded).*

---

11. StudentPreferences Model

Purpose: Student-specific UI and notification preferences.

```
model StudentPreferences {
  id                     String   @id @default(cuid())
  student_id             String   @unique

  // Appearance (NO language - Indonesian only)
  theme                  String   @default("system") // light | dark | system

  // Notifications
  notify_announcements    Boolean  @default(true)
  notify_schedule_changes Boolean  @default(true)
  notify_attendance       Boolean  @default(true)

  // Schedule Preferences
  schedule_default_view   String   @default("date") // date | extracurricular
  schedule_range_days     Int      @default(7)

  created_at             DateTime @default(now())
  updated_at             DateTime @updatedAt

  // Relations
  student StudentProfile @relation(fields: [student_id], references: [id], onDelete:
Cascade)
```

```
  @@map("student_preferences")
}
```

Key Features

- **Theme Settings**: Supports light/dark/system themes
- **Notification Toggles**: Fine-grained control over notification types
- **Schedule Display**: Customizable default view and date range
- **No Language Option**: Application is Indonesian-only

Current Seeded Data

0 Preferences (none seeded; created on-demand when student first visits settings)

Default Values:

- Theme: `system`
- All notifications: `true`
- Schedule view: `date`
- Schedule range: `7 days`

Entity Relationships

Entity Relationship Diagram

```
erDiagram
    User ||--o| StudentProfile : "has"
    User ||--o| PembinaProfile : "has"
    User ||--o{ Notification : "receives"
    User ||--o{ Announcement : "authors"

    StudentProfile ||--o| StudentPreferences : "has"
    StudentProfile ||--o{ Enrollment : "enrolls in"

    PembinaProfile ||--o{ Extracurricular : "supervises"

    Extracurricular ||--o{ Enrollment : "has"
    Extracurricular ||--o{ Schedule : "has"
    Extracurricular ||--o{ Session : "conducts"
    Extracurricular ||--o{ Announcement : "publishes"

    Enrollment ||--o{ Attendance : "tracks"
    Enrollment ||--o{ Notification : "triggers"

    Schedule ||--o{ Session : "generates"

    Session ||--o{ Attendance : "records"
```

Relationship Summary

Parent Model	Relationship Type	Child Model	Cascade Behavior

User	One-to-One	StudentProfile	Cascade Delete
User	One-to-One	PembinaProfile	Cascade Delete
User	One-to-Many	Notification	Cascade Delete
User	One-to-Many	Announcement	Cascade Delete
StudentProfile	One-to-One	StudentPreferences	Cascade Delete
StudentProfile	One-to-Many	Enrollment	Cascade Delete
PembinaProfile	One-to-Many	Extracurricular	Restrict Delete
Extracurricular	One-to-Many	Enrollment	Cascade Delete
Extracurricular	One-to-Many	Schedule	Cascade Delete
Extracurricular	One-to-Many	Session	Cascade Delete
Extracurricular	One-to-Many	Announcement	Cascade Delete
Enrollment	One-to-Many	Attendance	Cascade Delete
Enrollment	One-to-Many	Notification	SetNull on Delete
Schedule	One-to-Many	Session	SetNull on Delete
Session	One-to-Many	Attendance	SetNull on Delete

Key Relationship Patterns

1. User Polymorphism

```
User (role: SISWA) ──1:1──> StudentProfile
User (role: PEMBINA) ──1:1──> PembinaProfile
User (role: ADMIN) ──(no profile)
```

- **Constraint**: A User can have EITHER StudentProfile OR PembinaProfile, not both
- **Implementation**: Enforced by application logic, not database constraints

2. Many-to-Many (Student ↔ Extracurricular)

```
StudentProfile ──M:N──> Extracurricular
         ↓
      Enrollment (junction table with metadata)
```

- **Junction Table**: `Enrollment` with additional fields (status, academic_year)
- **Unique Constraint**: One enrollment per student per extracurricular

3. Template-Instance Pattern (Schedule → Session)

```
Schedule (template) ──1:M──> Session (concrete instances)
```

- **Schedule**: Recurring weekly pattern (e.g., "Every Monday 15:00")
- **Session**: Concrete calendar events (e.g., "2025-12-23 15:00")

- **Deletion**: Deleting Schedule sets `session.schedule_id` to NULL

Current Database State

Population Summary

```
📊 Database Population Statistics (Seeded Data)
================================================

Users:                 13
  - ADMIN:              3
  - PEMBINA:            5
  - SISWA:              5

Profiles:              10
  - StudentProfile:     5
  - PembinaProfile:     5

Extracurriculars:       5
  - Category: Olahraga  2
  - Category: Teknologi 2
  - Category: Seni      1

Schedules:              3
Sessions:              12 (4 weeks × 3 schedules)
Enrollments:            3
Attendance Records:    30 (10 × 3 enrollments)
Announcements:          4
Notifications:          0
StudentPreferences:     0
```

Data Distribution

Users by Role

```
ADMIN    ████████ 23.1%
PEMBINA  ████████████ 38.5%
SISWA    ████████████ 38.5%
```

Enrollments by Status

```
ACTIVE   ████████████████████████████ 100% (3/3)
PENDING  ▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒   0% (0/3)
```

Attendance by Status

```
PRESENT    ████████████████████ ~70% (21/30)
SICK       ████                 ~15% (4-5/30)
PERMISSION ██                   ~10% (3/30)
```

```
ALPHA     ▉                    ~5% (1-2/30)
LATE      ▒                     0% (0/30)
```

Seed Data Implementation

Seeding Architecture

The seed script (`prisma/seed.ts`) performs a **two-phase creation process**:

1. **Clerk Phase**: Creates users via Clerk Backend API
2. **Prisma Phase**: Creates database records with Clerk IDs

```
sequenceDiagram
    participant Seed as Seed Script
    participant Clerk as Clerk API
    participant DB as PostgreSQL

    Seed->>Clerk: Create User (email, password, metadata)
    Clerk-->>Seed: Return clerk_id
    Seed->>DB: Create User record (clerk_id, username, role)
    Seed->>DB: Create Profile (StudentProfile/PembinaProfile)
    Seed->>DB: Create Extracurriculars
    Seed->>DB: Create Schedules
    Seed->>DB: Generate Sessions from Schedules
    Seed->>DB: Create Enrollments
    Seed->>DB: Create Attendance Records
    Seed->>DB: Create Announcements
```

Seeding Steps

Step 1: Clean Existing Data

```
// Order matters! Delete children before parents
await prisma.attendance.deleteMany();
await prisma.session.deleteMany();
await prisma.announcement.deleteMany();
await prisma.schedule.deleteMany();
await prisma.enrollment.deleteMany();
await prisma.extracurricular.deleteMany();
await prisma.studentProfile.deleteMany();
await prisma.pembinaProfile.deleteMany();
await prisma.notification.deleteMany();
await prisma.user.deleteMany();
```

Step 2: Create Users via Clerk

```
const clerkUser = await clerkClient.users.createUser({
  emailAddress: [userData.email],
  password: userData.password,
```

```
  username: userData.username,
  firstName: userData.firstName,
  lastName: userData.lastName,
  publicMetadata: {
    role: userData.role,  // Store role in Clerk metadata
  },
});
```

Conflict Handling: If username exists, appends timestamp suffix

Step 3: Create Prisma User + Profile

```
if (userData.role === "SISWA" && userData.profileData) {
  prismaUser = await prisma.user.create({
    data: {
      clerk_id: clerkId,
      username: userData.username,
      email: userData.email,
      full_name: `${userData.firstName} ${userData.lastName}`,
      role: userData.role,
      studentProfile: {
        create: {
          nis: userData.profileData.nis!,
          class_name: userData.profileData.class_name!,
          major: userData.profileData.major!,
          phone_number: userData.profileData.phone_number,
        },
      },
    },
    include: { studentProfile: true },
  });
}
```

Step 4: Generate Sessions from Schedules

```
const dayNameToIndex: Record<string, number> = {
  SENIN: 1,   // Monday
  RABU: 3,    // Wednesday
  KAMIS: 4,   // Thursday
  // ... etc
};

for (const schedule of schedules) {
  const targetDayIndex = dayNameToIndex[schedule.day_of_week.toUpperCase()];

  // Generate 4 weekly sessions
  for (let week = 0; week < 4; week++) {
    const sessionDate = calculateNextOccurrence(targetDayIndex, week);

    sessionsToCreate.push({
```

```
      extracurricular_id: schedule.extracurricular_id,
      schedule_id: schedule.id,
      date: sessionDate,
      start_time: schedule.start_time,
      end_time: schedule.end_time,
      location: schedule.location,
    });
  }
}
```

Step 5: Create Attendance with Random Distribution

```
for (const enrollment of enrollments) {
  for (let i = 1; i <= 10; i++) {
    const date = new Date(today);
    date.setDate(date.getDate() - i * 7);  // Weekly sessions

    // Weighted random selection
    const rand = Math.random();
    let status: AttendanceStatus;
    if (rand < 0.70) status = "PRESENT";       // 70%
    else if (rand < 0.85) status = "SICK";     // 15%
    else if (rand < 0.95) status = "PERMISSION"; // 10%
    else status = "ALPHA";                     // 5%

    attendanceData.push({
      enrollment_id: enrollment.id,
      date: date,
      status: status,
    });
  }
}
```

Login Credentials

All seeded users use the same password for testing:

```
Password: rtx5070ti16gb
```

Sample Accounts:

Role	Email	Password
ADMIN	admin@sixkul.sch.id	rtx5070ti16gb
PEMBINA	pembina@sixkul.sch.id	rtx5070ti16gb
SISWA	student@sixkul.sch.id	rtx5070ti16gb

> *[!CAUTION] **Security Warning**: These are development/testing credentials. Never use simple passwords in production!*

Technical Implementation Details

Prisma Client Generation

```
# Generate Prisma Client
npx prisma generate

# Output location
src/generated/prisma/
```

Database Migrations

```
# Create migration
npx prisma migrate dev --name migration_name

# Deploy to production
npx prisma migrate deploy

# Reset database (development only!)
npx prisma migrate reset
```

Running the Seed Script

```
# Run seed script
npx prisma db seed

# Or directly
npx tsx prisma/seed.ts
```

Prerequisites:

- `CLERK_SECRET_KEY` must be set in `.env`
- Database connection string configured

Clerk Integration

Environment Variables

```
# Clerk Authentication
NEXT_PUBLIC_CLERK_PUBLISHABLE_KEY=pk_test_...
CLERK_SECRET_KEY=sk_test_...

# Clerk Routes
NEXT_PUBLIC_CLERK_SIGN_IN_URL=/sign-in
NEXT_PUBLIC_CLERK_SIGN_UP_URL=/sign-up
NEXT_PUBLIC_CLERK_AFTER_SIGN_IN_URL=/dashboard
NEXT_PUBLIC_CLERK_AFTER_SIGN_UP_URL=/dashboard
```

User Synchronization

Flow: Clerk Webhook → API Route → Prisma Database

1. User signs up/updates in Clerk
2. Clerk sends webhook to `/api/webhooks/clerk`
3. API route syncs Clerk user to Database
4. Creates/updates User record with `clerk_id`

Key Fields Synced:

- `clerk_id` : Primary link (unique identifier)
- `email` : Synced from Clerk email addresses
- `full_name` : Constructed from `firstName + lastName`
- `avatar_url` : Synced from Clerk profile image
- `role` : Retrieved from Clerk `publicMetadata.role`

Database Indexing Strategy

Automatic Indexes (created by Prisma):

- All `@id` fields (primary keys)
- All `@unique` fields
- All foreign key fields

Critical Indexes:

```
// Unique indexes
clerk_id      @unique   // Fast Clerk → Prisma user lookup
username      @unique   // Unique username enforcement
nis           @unique   // Student ID lookup
nip           @unique   // Employee ID lookup

// Composite unique index
@@unique([student_id, extracurricular_id])  // Enrollment uniqueness
@@unique([enrollment_id, date])             // Daily attendance uniqueness
```

Query Optimization Tips

1. Use `include` for Relations

```
// ✅ Good: Single query with join
const user = await prisma.user.findUnique({
  where: { clerk_id: clerkId },
  include: { studentProfile: true },
});

// ❌ Bad: Two separate queries
const user = await prisma.user.findUnique({ where: { clerk_id: clerkId } });
const profile = await prisma.studentProfile.findUnique({ where: { user_id: user.id } });
```

2. Use `select` to Minimize Data Transfer

```
// ✅ Good: Only fetch needed fields
const users = await prisma.user.findMany({
  select: {
    id: true,
    full_name: true,
    email: true,
  },
});

// ❌ Bad: Fetches all fields
const users = await prisma.user.findMany();
```

3. Batch Operations with `createMany`

```
// ✅ Good: Single transaction
await prisma.session.createMany({
  data: sessionsToCreate,  // Array of 12 sessions
});

// ❌ Bad: 12 separate transactions
for (const session of sessionsToCreate) {
  await prisma.session.create({ data: session });
}
```

Appendix

Key Files

File	Purpose
prisma/schema.prisma	Database schema definition
prisma/seed.ts	Database seeding script with Clerk integration
src/generated/prisma/	Generated Prisma Client

Useful Prisma Commands

```
# View database in Prisma Studio
npx prisma studio

# Validate schema
npx prisma validate

# Format schema file
npx prisma format

# Pull schema from existing database
npx prisma db pull
```

```
# Push schema to database (dev only!)
npx prisma db push
```

Database Connection

Provider: Supabase (Cloud-hosted PostgreSQL)

Connection String Format:

```
postgresql://[user]:[password]@[host]:[port]/[database]?pgbouncer=true
```

End of Documentation

This document provides a comprehensive overview of the SIXKUL database schema and current technical implementation. For questions or updates, please refer to the source schema file or contact the development team.