N

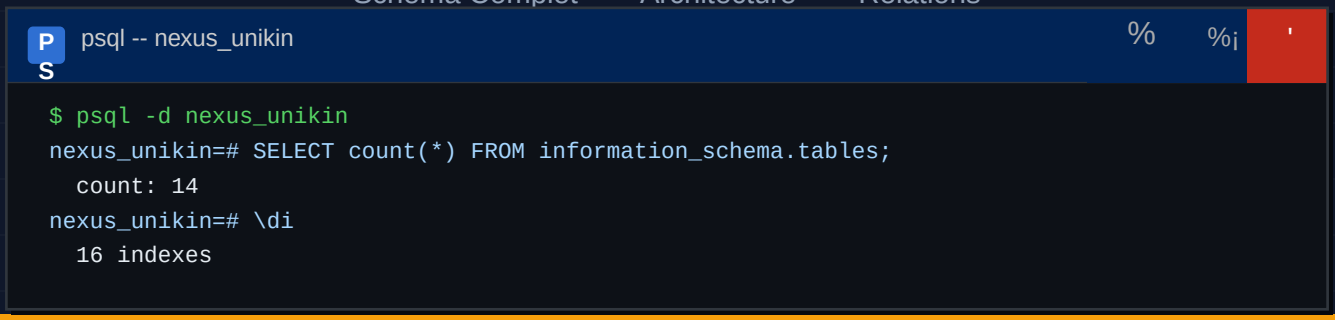
NEXUS
UNIKIN

DOCUMENTATION TECHNIQUE

Base de Données

PostgreSQL

Schéma Complet — Architecture — Relations



Université de Kinshasa

Projet NEXUS — Système de Gestion Universitaire

Version 1.0 | Février 2026 | Confidentiel

Table des Matières

1 Introduction

Vue d'ensemble du système NEXUS UNIKIN

NEXUS UNIKIN est la plateforme intégrée de gestion de l'Université de Kinshasa. Ce document décrit l'intégralité du schéma de base de données PostgreSQL : structure des tables, relations, contraintes, index et triggers.

Spécifications Techniques

SGBD	`PostgreSQL 15+`
Framework	`Next.js 14.1.0 (App Router + TypeScript)`
Accès BD	`pg (node-postgres) — Requêtes SQL natives`
Authentification	`bcryptjs (hash, coût 10) + JWT`
Architecture BD	`14 tables | 16 index | 8 triggers`
Normalisation	`3ème forme normale (3NF)`

Modules Fonctionnels

> Authentification & Rôles
5 rôles : Super Admin, Admin, Enseignant, Étudiant, Employé

> Structure Académique
Hiérarchie Facultés > Départements > Promotions (L1 à D3)

> Gestion des Cours
Crédits, volumes horaires CM/TD/TP, affectation enseignants

> Inscriptions & Notes
Inscription par année, saisie TP/TD/Examen, validation jury

> Présences
Relevé quotidien : Présent, Absent, Retard, Excusé

> Finances
Paiements Mobile Money, Cash, Banque avec reçus

> Notifications
Alertes ciblées (Info, Warning, Success, Error)

2 Architecture

Organisation logique des 14 tables en couches

Couche	Tables	Fonction
Référence	academic_years, users, faculties, departments, promotions	Données structurelles
Profils	admins, teachers, students, employees	Extension 1:1 de users par rôle
Académique	courses, course_schedules, enrollments, grades, attendance	Cours, horaires, notes, présences
Opérationnel	payments, notifications	Finance et communication

PS psql -- \dt (List Tables)

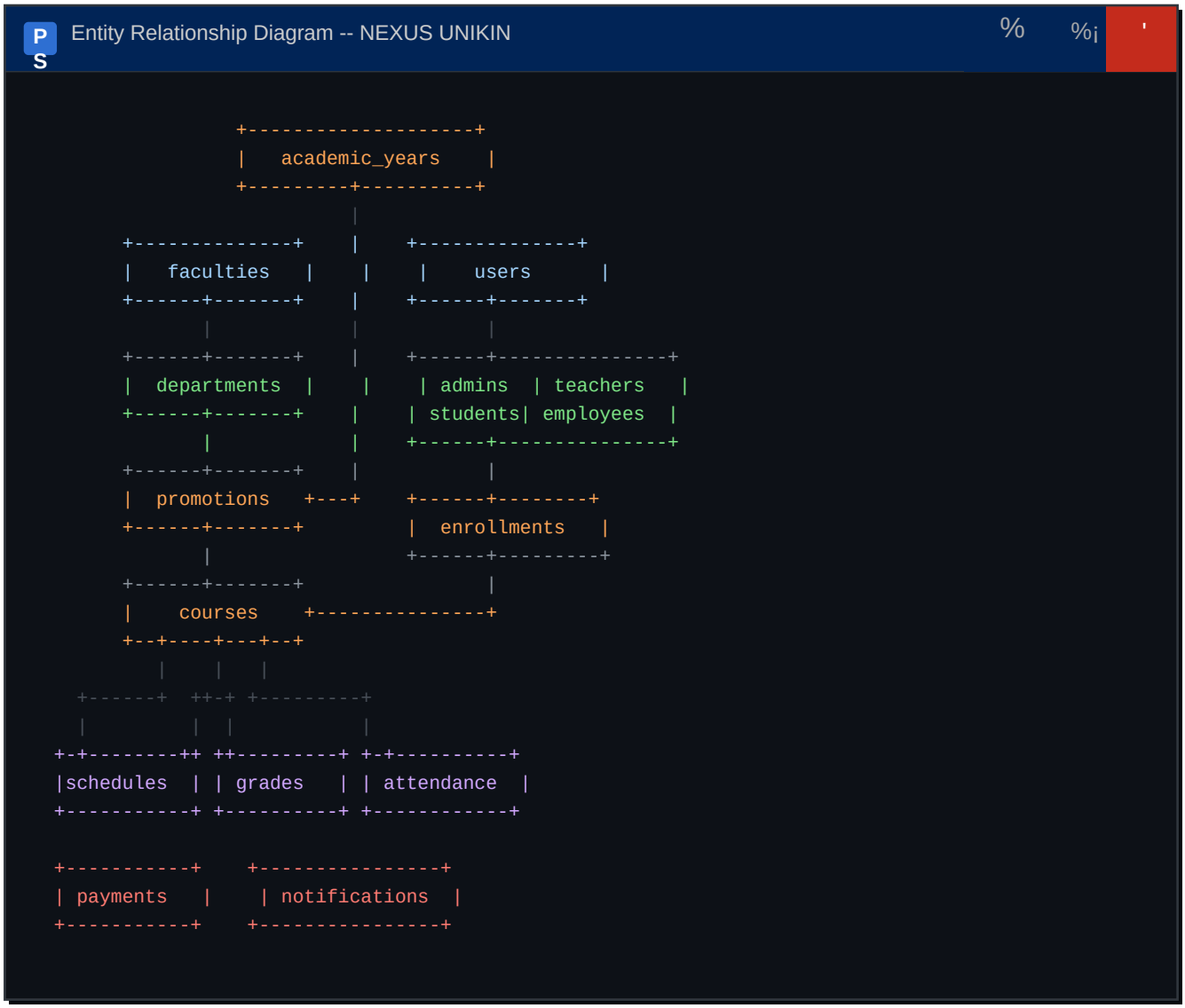
```
$ psql -h localhost -U postgres -d nexus_unikin

nexus_unikin=# \dt
                   List of relations
  Schema  |       Name          |  Type  |   Owner
 ---------+---------------------+--------+----------
  public  | academic_years      | table  | postgres
  public  | admins              | table  | postgres
  public  | attendance          | table  | postgres
  public  | course_schedules    | table  | postgres
  public  | courses             | table  | postgres
  public  | departments         | table  | postgres
  public  | employees           | table  | postgres
  public  | enrollments         | table  | postgres
  public  | faculties           | table  | postgres
  public  | grades              | table  | postgres
  public  | notifications       | table  | postgres
  public  | payments            | table  | postgres
  public  | promotions          | table  | postgres
  public  | students            | table  | postgres
  public  | teachers            | table  | postgres
  public  | users               | table  | postgres
  (16 rows)
```

3 Diagramme Entité-Relation

Visualisation des clés étrangères

Le diagramme ci-dessous illustre les relations entre toutes les tables :

Entity Relationship Diagram -- NEXUS UNIKIN



RELATIONS — Clés Étrangères (FK)

23 lignes

Table Source	Colonne FK	Référence
departments	faculty_id	faculties.id
promotions	department_id	departments.id
promotions	academic_year_id	academic_years.id
admins	user_id	users.id (UNIQUE)
teachers	user_id	users.id (UNIQUE)
teachers	department_id	departments.id
students	user_id	users.id (UNIQUE)
students	promotion_id	promotions.id
employees	user_id	users.id (UNIQUE)
courses	promotion_id	promotions.id
courses	teacher_id	teachers.id
course_schedules	course_id	courses.id
enrollments	student_id	students.id
enrollments	course_id	courses.id
enrollments	academic_year_id	academic_years.id
grades	student_id	students.id
grades	course_id	courses.id
grades	academic_year_id	academic_years.id
attendance	student_id	students.id
attendance	course_id	courses.id
payments	student_id	students.id
payments	academic_year_id	academic_years.id
notifications	user_id	users.id

4 Tables de Référence

Données structurelles de base

academic_years

Périodes académiques. Chaque année a un début, une fin et un indicateur is_current.

academic_years — 7 lignes

Colonne	Type	Contraintes / Description
id	SERIAL	PRIMARY KEY — Auto-incrémenté
name	VARCHAR(20)	UNIQUE NOT NULL — Ex: "2024-2025"
start_date	DATE	NOT NULL — Début de l'année
end_date	DATE	NOT NULL — Fin de l'année
is_current	BOOLEAN	DEFAULT FALSE — Année active
created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP
updated_at	TIMESTAMP	AUTO via trigger

INFO Une seule année peut être marquée is_current = TRUE. Référencée par promotions, enrollments, grades, payments.

PS CREATE TABLE academic_years % %¡ '

```
CREATE TABLE academic_years (
    id          SERIAL PRIMARY KEY,
    name        VARCHAR(20) NOT NULL UNIQUE,
    start_date  DATE NOT NULL,
    end_date    DATE NOT NULL,
    is_current  BOOLEAN DEFAULT FALSE,
    created_at  TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at  TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

users

Table centrale d'authentification. Le rôle détermine le profil associé (admins, teachers, students, employees).

users 13 lignes

Colonne	Type	Contraintes / Description
id	SERIAL	PRIMARY KEY
email	VARCHAR(255)	UNIQUE NOT NULL — Login
password	VARCHAR(255)	NOT NULL — Hash bcrypt (60 chars)
first_name	VARCHAR(100)	NOT NULL — Prénom
last_name	VARCHAR(100)	NOT NULL — Nom de famille
phone	VARCHAR(20)	Nullable — Téléphone (+243...)
address	TEXT	Nullable — Adresse physique
photo_url	TEXT	Nullable — URL photo
role	VARCHAR(20)	CHECK: SUPER_ADMIN|ADMIN|TEACHER|STUDENT|EMPLOYEE
is_active	BOOLEAN	DEFAULT TRUE — Activation
last_login	TIMESTAMP	Nullable — Dernière connexion
created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP
updated_at	TIMESTAMP	AUTO via trigger

INFO Index sur email et rôle. Password hash bcrypt coût 10. La colonne role détermine la table-profil 1:1.

PS CREATE TABLE users % %¡ '

```
CREATE TABLE users (
    id          SERIAL PRIMARY KEY,
    email       VARCHAR(255) NOT NULL UNIQUE,
    password    VARCHAR(255) NOT NULL,
    first_name  VARCHAR(100) NOT NULL,
    last_name   VARCHAR(100) NOT NULL,
    phone       VARCHAR(20),
    address     TEXT,
    photo_url   TEXT,
    role        VARCHAR(20) NOT NULL CHECK (role IN
                ('SUPER_ADMIN','ADMIN','TEACHER','STUDENT','EMPLOYEE')),
    is_active   BOOLEAN DEFAULT TRUE,
    last_login  TIMESTAMP,
    created_at  TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at  TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

faculties

Premier niveau de hiérarchie académique — les grandes facultés.

faculties — 8 lignes

Colonne	Type	Contraintes / Description
id	SERIAL	PRIMARY KEY
code	VARCHAR(10)	UNIQUE NOT NULL — Code court (FSEG, FSI)
name	VARCHAR(255)	NOT NULL — Nom complet
description	TEXT	Nullable
dean_id	INTEGER	FK -> users.id — Doyen
is_active	BOOLEAN	DEFAULT TRUE
created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP
updated_at	TIMESTAMP	AUTO via trigger

PS CREATE TABLE faculties

```
CREATE TABLE faculties (
    id          SERIAL PRIMARY KEY,
    code        VARCHAR(10) NOT NULL UNIQUE,
    name        VARCHAR(255) NOT NULL,
    description TEXT,
    dean_id     INTEGER REFERENCES users(id),
    is_active   BOOLEAN DEFAULT TRUE,
    created_at  TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at  TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

departments

Deuxième niveau — départements au sein des facultés. Suppression en cascade depuis faculties.

departments — 9 lignes

Colonne	Type	Contraintes / Description
id	SERIAL	PRIMARY KEY
code	VARCHAR(10)	UNIQUE NOT NULL — INFO, ECO
name	VARCHAR(255)	NOT NULL
description	TEXT	Nullable
faculty_id	INTEGER	FK -> faculties.id — ON DELETE CASCADE
head_id	INTEGER	FK -> users.id — Chef de département
is_active	BOOLEAN	DEFAULT TRUE
created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP
updated_at	TIMESTAMP	AUTO via trigger

INFO CASCADE : supprimer une faculté supprime tous ses départements.

PS CREATE TABLE departments

```
CREATE TABLE departments (
    id          SERIAL PRIMARY KEY,
    code        VARCHAR(10) NOT NULL UNIQUE,
    name        VARCHAR(255) NOT NULL,
    description TEXT,
    faculty_id  INTEGER NOT NULL REFERENCES faculties(id) ON DELETE CASCADE,
    head_id     INTEGER REFERENCES users(id),
    is_active   BOOLEAN DEFAULT TRUE,
    created_at  TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at  TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

promotions

Troisième niveau — niveaux d'étude (L1 à D3) liés à un département et une année.

promotions — 9 lignes

Colonne	Type	Contraintes / Description
id	SERIAL	PRIMARY KEY
code	VARCHAR(20)	NOT NULL — Code promotion
name	VARCHAR(255)	NOT NULL — Ex: "L1 Informatique"
level	VARCHAR(10)	CHECK: L1|L2|L3|M1|M2|D1|D2|D3
department_id	INTEGER	FK -> departments.id — CASCADE
academic_year_id	INTEGER	FK -> academic_years.id
is_active	BOOLEAN	DEFAULT TRUE
created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP
updated_at	TIMESTAMP	AUTO via trigger

INFO UNIQUE(code, academic_year_id) : une promotion est unique par année académique.

PS CREATE TABLE promotions

```
CREATE TABLE promotions (
    id               SERIAL PRIMARY KEY,
    code             VARCHAR(20) NOT NULL,
    name             VARCHAR(255) NOT NULL,
    level            VARCHAR(10) NOT NULL CHECK
                     (level IN ('L1','L2','L3','M1','M2','D1','D2','D3')),
    department_id    INTEGER NOT NULL
                     REFERENCES departments(id) ON DELETE CASCADE,
    academic_year_id INTEGER NOT NULL REFERENCES academic_years(id),
    is_active        BOOLEAN DEFAULT TRUE,
    UNIQUE(code, academic_year_id)
);
```

5 Tables de Profils

Extension 1:1 de la table users par rôle

INFO Chaque utilisateur possède exactement un profil spécialisé. La relation est 1:1 via user_id UNIQUE.

admins

Profil administrateur — gestion par périmètre (global, faculté, département).

admins 7 lignes

Colonne	Type	Contraintes / Description
id	SERIAL	PRIMARY KEY
user_id	INTEGER	FK -> users.id — UNIQUE, CASCADE
admin_type	VARCHAR(20)	CHECK: SUPER_ADMIN|FACULTY_ADMIN|DEPARTMENT_ADMIN
faculty_id	INTEGER	FK -> faculties.id — NULL si SUPER_ADMIN
department_id	INTEGER	FK -> departments.id — NULL si SUPER/FACULTY
created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP
updated_at	TIMESTAMP	AUTO via trigger

PS CREATE TABLE admins % %¡ '

```
CREATE TABLE admins (
    id            SERIAL PRIMARY KEY,
    user_id       INTEGER NOT NULL UNIQUE
                  REFERENCES users(id) ON DELETE CASCADE,
    admin_type    VARCHAR(20) NOT NULL CHECK (admin_type IN
                  ('SUPER_ADMIN','FACULTY_ADMIN','DEPARTMENT_ADMIN')),
    faculty_id    INTEGER REFERENCES faculties(id),
    department_id INTEGER REFERENCES departments(id)
);
```

teachers

Profil enseignant — grade académique, spécialisation, rattachement.

teachers

10 lignes

Colonne	Type	Contraintes / Description
id	SERIAL	PRIMARY KEY
user_id	INTEGER	FK -> users.id — UNIQUE, CASCADE
matricule	VARCHAR(50)	UNIQUE NOT NULL — PROF-2024-001
grade	VARCHAR(50)	CHECK: ASSISTANT à PROFESSEUR_ORDINAIRE
specialization	VARCHAR(255)	Nullable — Domaine d'expertise
department_id	INTEGER	FK -> departments.id
hire_date	DATE	Nullable — Date embauche
is_permanent	BOOLEAN	DEFAULT TRUE — Permanent vs Vacataire
created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP
updated_at	TIMESTAMP	AUTO via trigger

INFO Grades : ASSISTANT < CHEF_TRAVAUX < PROFESSEUR_ASSOCIE < PROFESSEUR < PROFESSEUR_ORDINAIRE.

PS CREATE TABLE teachers % %¡ '

```
CREATE TABLE teachers (
    id              SERIAL PRIMARY KEY,
    user_id         INTEGER NOT NULL UNIQUE
                    REFERENCES users(id) ON DELETE CASCADE,
    matricule       VARCHAR(50) NOT NULL UNIQUE,
    grade           VARCHAR(50) NOT NULL CHECK (grade IN
                    ('ASSISTANT','CHEF_TRAVAUX','PROFESSEUR_ASSOCIE',
                     'PROFESSEUR','PROFESSEUR_ORDINAIRE')),
    specialization VARCHAR(255),
    department_id  INTEGER REFERENCES departments(id),
    hire_date       DATE,
    is_permanent    BOOLEAN DEFAULT TRUE
);
```

students

Profil étudiant — inscription, paiement, état civil. Table la plus volumineuse.

students 12 lignes

Colonne	Type	Contraintes / Description
id	SERIAL	PRIMARY KEY
user_id	INTEGER	FK -> users.id — UNIQUE, CASCADE
matricule	VARCHAR(50)	UNIQUE NOT NULL — ETU-2024-001
promotion_id	INTEGER	FK -> promotions.id — Niveau actuel
enrollment_date	DATE	NOT NULL — Date inscription
status	VARCHAR(20)	CHECK: ACTIVE|SUSPENDED|GRADUATED|DROPPED
payment_status	VARCHAR(20)	CHECK: PAID|PARTIAL|UNPAID|BLOCKED
birth_date	DATE	Nullable
nationality	VARCHAR(100)	DEFAULT "Congolaise"
gender	VARCHAR(10)	CHECK: M ou F
parent_name	VARCHAR(255)	Nullable — Nom du tuteur
parent_phone	VARCHAR(20)	Nullable — Tél. du tuteur

INFO payment_status = BLOCKED bloque l'accès aux notes. Index sur matricule, promotion_id, status.

PS CREATE TABLE students % %¡ '

```
CREATE TABLE students (
    id              SERIAL PRIMARY KEY,
    user_id         INTEGER NOT NULL UNIQUE
                    REFERENCES users(id) ON DELETE CASCADE,
    matricule       VARCHAR(50) NOT NULL UNIQUE,
    promotion_id    INTEGER NOT NULL REFERENCES promotions(id),
    enrollment_date DATE NOT NULL,
    status          VARCHAR(20) DEFAULT 'ACTIVE' CHECK (status IN
                    ('ACTIVE','SUSPENDED','GRADUATED','DROPPED')),
    payment_status  VARCHAR(20) DEFAULT 'UNPAID' CHECK (payment_status IN
                    ('PAID','PARTIAL','UNPAID','BLOCKED')),
    birth_date      DATE,
    nationality     VARCHAR(100) DEFAULT 'Congolaise',
    gender          VARCHAR(10) CHECK (gender IN ('M','F')),
    parent_name     VARCHAR(255),
    parent_phone    VARCHAR(20)
);
```

employees

Profil employé — personnel administratif et technique (non-enseignant).

employees

8 lignes

Colonne	Type	Contraintes / Description
id	SERIAL	PRIMARY KEY
user_id	INTEGER	FK -> users.id — UNIQUE, CASCADE
matricule	VARCHAR(50)	UNIQUE NOT NULL — EMP-2024-001
position	VARCHAR(255)	NOT NULL — Poste occupé
department	VARCHAR(255)	Nullable — Service (texte libre, non FK)
service	VARCHAR(255)	Nullable — Sous-service
hire_date	DATE	Nullable
contract_type	VARCHAR(20)	CHECK: PERMANENT|CONTRACT|TEMPORARY

PS CREATE TABLE employees % %¡ '

```
CREATE TABLE employees (
    id            SERIAL PRIMARY KEY,
    user_id       INTEGER NOT NULL UNIQUE
                  REFERENCES users(id) ON DELETE CASCADE,
    matricule     VARCHAR(50) NOT NULL UNIQUE,
    position      VARCHAR(255) NOT NULL,
    department    VARCHAR(255),
    service       VARCHAR(255),
    hire_date     DATE,
    contract_type VARCHAR(20) CHECK (contract_type IN
                  ('PERMANENT','CONTRACT','TEMPORARY'))
);
```

6 Tables Académiques

Cours, horaires, inscriptions, notes et présences

courses

Catalogue des cours — crédits, volumes horaires, affectation promotions et enseignants.

courses — 13 lignes

Colonne	Type	Contraintes / Description
id	SERIAL	PRIMARY KEY
code	VARCHAR(20)	NOT NULL — Ex: INFO101
name	VARCHAR(255)	NOT NULL — Intitulé du cours
description	TEXT	Nullable
credits	INTEGER	DEFAULT 3 — Crédits ECTS
hours_cm	INTEGER	DEFAULT 0 — Heures Cours Magistral
hours_td	INTEGER	DEFAULT 0 — Heures Travaux Dirigés
hours_tp	INTEGER	DEFAULT 0 — Heures Travaux Pratiques
promotion_id	INTEGER	FK -> promotions.id — CASCADE
teacher_id	INTEGER	FK -> teachers.id — Enseignant titulaire
semester	INTEGER	CHECK: 1 ou 2
course_type	VARCHAR(20)	CHECK: OBLIGATOIRE|OPTIONNEL|LIBRE
is_active	BOOLEAN	DEFAULT TRUE

INFO UNIQUE(code, promotion_id). Les volumes CM/TD/TP servent au calcul de l'emploi du temps.

PS CREATE TABLE courses

```
CREATE TABLE courses (
    id            SERIAL PRIMARY KEY,
    code          VARCHAR(20) NOT NULL,
    name          VARCHAR(255) NOT NULL,
    credits       INTEGER NOT NULL DEFAULT 3,
    hours_cm      INTEGER DEFAULT 0,
    hours_td      INTEGER DEFAULT 0,
    hours_tp      INTEGER DEFAULT 0,
    promotion_id  INTEGER NOT NULL
                  REFERENCES promotions(id) ON DELETE CASCADE,
    teacher_id    INTEGER REFERENCES teachers(id),
    semester      INTEGER NOT NULL CHECK (semester IN (1, 2)),
    course_type   VARCHAR(20) DEFAULT 'OBLIGATOIRE'
                  CHECK (course_type IN ('OBLIGATOIRE','OPTIONNEL','LIBRE')),
    is_active     BOOLEAN DEFAULT TRUE,
    UNIQUE(code, promotion_id)
);
```

course_schedules

Créneaux horaires hebdomadaires — un cours peut avoir plusieurs créneaux.

course_schedules — 7 lignes

Colonne	Type	Contraintes / Description
id	SERIAL	PRIMARY KEY
course_id	INTEGER	FK -> courses.id — ON DELETE CASCADE
day_of_week	INTEGER	CHECK 0-6 (0=Lundi, 6=Dimanche)
start_time	TIME	NOT NULL — Heure début
end_time	TIME	NOT NULL — Heure fin
room	VARCHAR(50)	Nullable — Salle (ex: "A201")
schedule_type	VARCHAR(10)	CHECK: CM|TD|TP — DEFAULT CM

PS CREATE TABLE course_schedules % %¡ '

```
CREATE TABLE course_schedules (
    id            SERIAL PRIMARY KEY,
    course_id     INTEGER NOT NULL
                  REFERENCES courses(id) ON DELETE CASCADE,
    day_of_week   INTEGER NOT NULL CHECK (day_of_week BETWEEN 0 AND 6),
    start_time    TIME NOT NULL,
    end_time      TIME NOT NULL,
    room          VARCHAR(50),
    schedule_type VARCHAR(10) DEFAULT 'CM'
                  CHECK (schedule_type IN ('CM','TD','TP'))
);
```

enrollments

Inscriptions des étudiants aux cours — une par étudiant, par cours, par année.

enrollments — 6 lignes

Colonne	Type	Contraintes / Description
id	SERIAL	PRIMARY KEY
student_id	INTEGER	FK -> students.id — CASCADE
course_id	INTEGER	FK -> courses.id — CASCADE
academic_year_id	INTEGER	FK -> academic_years.id
enrollment_date	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP
status	VARCHAR(20)	CHECK: ENROLLED|DROPPED|COMPLETED

INFO UNIQUE(student_id, course_id, academic_year_id) — empêche les doublons.

PS CREATE TABLE enrollments % %¡ '

```
CREATE TABLE enrollments (
    id               SERIAL PRIMARY KEY,
    student_id       INTEGER NOT NULL
                     REFERENCES students(id) ON DELETE CASCADE,
    course_id        INTEGER NOT NULL
                     REFERENCES courses(id) ON DELETE CASCADE,
    academic_year_id INTEGER NOT NULL REFERENCES academic_years(id),
    status           VARCHAR(20) DEFAULT 'ENROLLED'
                     CHECK (status IN ('ENROLLED','DROPPED','COMPLETED')),
    UNIQUE(student_id, course_id, academic_year_id)
);
```

grades

Notes des étudiants — composantes TP, TD, Examen et note finale validée par le jury.

grades 13 lignes

Colonne	Type	Contraintes / Description
id	SERIAL	PRIMARY KEY
student_id	INTEGER	FK -> students.id — CASCADE
course_id	INTEGER	FK -> courses.id — CASCADE
academic_year_id	INTEGER	FK -> academic_years.id
tp_score	DECIMAL(5,2)	Nullable — Note TP (0-100)
td_score	DECIMAL(5,2)	Nullable — Note TD (0-100)
exam_score	DECIMAL(5,2)	Nullable — Note Examen (0-100)
final_score	DECIMAL(5,2)	Nullable — Note finale calculée
grade_letter	VARCHAR(2)	Nullable — A, B, C, D, E, F
is_validated	BOOLEAN	DEFAULT FALSE — Verrou jury
validated_by	INTEGER	FK -> users.id — Validateur
validated_at	TIMESTAMP	Nullable — Date validation
remarks	TEXT	Nullable

INFO Formule : final = TP×20% + TD×20% + Exam×60%. is_validated=TRUE verrouille la note.

CREATE TABLE grades

```
CREATE TABLE grades (
    id               SERIAL PRIMARY KEY,
    student_id       INTEGER NOT NULL
                     REFERENCES students(id) ON DELETE CASCADE,
    course_id        INTEGER NOT NULL
                     REFERENCES courses(id) ON DELETE CASCADE,
    academic_year_id INTEGER NOT NULL REFERENCES academic_years(id),
    tp_score         DECIMAL(5,2),
    td_score         DECIMAL(5,2),
    exam_score       DECIMAL(5,2),
    final_score      DECIMAL(5,2),
    grade_letter     VARCHAR(2),
    is_validated     BOOLEAN DEFAULT FALSE,
    validated_by     INTEGER REFERENCES users(id),
    validated_at     TIMESTAMP,
    remarks          TEXT,
    UNIQUE(student_id, course_id, academic_year_id)
);
```

attendance

Relevé de présence quotidien — par étudiant, par cours, par créneau.

attendance — 8 lignes

Colonne	Type	Contraintes / Description
id	SERIAL	PRIMARY KEY
student_id	INTEGER	FK -> students.id — CASCADE
course_id	INTEGER	FK -> courses.id — CASCADE
schedule_id	INTEGER	FK -> course_schedules.id — Créneau
attendance_date	DATE	NOT NULL
status	VARCHAR(20)	CHECK: PRESENT|ABSENT|LATE|EXCUSED
remarks	TEXT	Nullable — Justificatif
recorded_by	INTEGER	FK -> users.id — Qui a enregistré

CREATE TABLE attendance

```
CREATE TABLE attendance (
    id              SERIAL PRIMARY KEY,
    student_id      INTEGER NOT NULL
                    REFERENCES students(id) ON DELETE CASCADE,
    course_id       INTEGER NOT NULL
                    REFERENCES courses(id) ON DELETE CASCADE,
    schedule_id     INTEGER REFERENCES course_schedules(id),
    attendance_date DATE NOT NULL,
    status          VARCHAR(20) DEFAULT 'PRESENT'
                    CHECK (status IN ('PRESENT','ABSENT','LATE','EXCUSED')),
    remarks         TEXT,
    recorded_by     INTEGER REFERENCES users(id)
);
```

7 Tables Opérationnelles

Paiements étudiants et notifications système

payments

Tous les paiements étudiants — montants en USD, méthodes variées, reçus et statuts.

payments — 12 lignes

Colonne	Type	Contraintes / Description
id	SERIAL	PRIMARY KEY
student_id	INTEGER	FK -> students.id — CASCADE
academic_year_id	INTEGER	FK -> academic_years.id
amount	DECIMAL(15,2)	NOT NULL — Montant en USD
payment_type	VARCHAR(50)	CHECK: INSCRIPTION|FRAIS_ACADEMIQUES|MINERVAL|AUTRES
payment_date	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP
payment_method	VARCHAR(20)	CHECK: CASH|BANK|MOBILE_MONEY|CHECK
reference	VARCHAR(100)	Nullable — Réf. transaction
receipt_number	VARCHAR(100)	Nullable — N° reçu officiel
status	VARCHAR(20)	CHECK: PENDING|COMPLETED|CANCELLED|REFUNDED
recorded_by	INTEGER	FK -> users.id — Enregistreur
remarks	TEXT	Nullable

INFO Le receipt_number (REC-2026-XXXXX) est la preuve physique. Lié à academic_year pour le reporting annuel.

CREATE TABLE payments

```
CREATE TABLE payments (
    id              SERIAL PRIMARY KEY,
    student_id      INTEGER NOT NULL
                    REFERENCES students(id) ON DELETE CASCADE,
    academic_year_id INTEGER NOT NULL REFERENCES academic_years(id),
    amount          DECIMAL(15,2) NOT NULL,
    payment_type    VARCHAR(50) NOT NULL CHECK (payment_type IN
                    ('INSCRIPTION','FRAIS_ACADEMIQUES','FRAIS_MINERVAL','AUTRES')),
    payment_date    TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    payment_method  VARCHAR(20) CHECK (payment_method IN
                    ('CASH','BANK','MOBILE_MONEY','CHECK')),
    reference       VARCHAR(100),
    receipt_number  VARCHAR(100),
    status          VARCHAR(20) DEFAULT 'PENDING' CHECK (status IN
                    ('PENDING','COMPLETED','CANCELLED','REFUNDED')),
    recorded_by     INTEGER REFERENCES users(id),
    remarks         TEXT
);
```

notifications

Notifications personnelles pour chaque utilisateur — alertes, messages système.

notifications 9 lignes

Colonne	Type	Contraintes / Description
id	SERIAL	PRIMARY KEY
user_id	INTEGER	FK -> users.id — CASCADE
title	VARCHAR(255)	NOT NULL — Titre
message	TEXT	NOT NULL — Contenu
type	VARCHAR(20)	CHECK: INFO|WARNING|SUCCESS|ERROR
is_read	BOOLEAN	DEFAULT FALSE — Lu / Non lu
link	VARCHAR(255)	Nullable — URL de redirection
created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP
updated_at	TIMESTAMP	AUTO via trigger

CREATE TABLE notifications % %¡ '

```
CREATE TABLE notifications (
    id         SERIAL PRIMARY KEY,
    user_id    INTEGER NOT NULL
               REFERENCES users(id) ON DELETE CASCADE,
    title      VARCHAR(255) NOT NULL,
    message    TEXT NOT NULL,
    type       VARCHAR(20) DEFAULT 'INFO' CHECK (type IN
               ('INFO','WARNING','SUCCESS','ERROR')),
    is_read    BOOLEAN DEFAULT FALSE,
    link       VARCHAR(255),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

8 Index & Performances

16 index pour l'optimisation des requêtes

Les index suivants ciblent les colonnes les plus utilisées dans les WHERE, JOIN et ORDER BY :

INDEX -- Performance Optimization — 16 lignes

Index	Table	Colonne	Usage Principal
idx_users_email	users	email	Login (recherche par email)
idx_users_role	users	role	Filtrage par rôle
idx_students_matricule	students	matricule	Recherche par matricule
idx_students_promotion	students	promotion_id	Liste par promotion
idx_students_status	students	status	Filtrage par statut
idx_teachers_matricule	teachers	matricule	Recherche enseignant
idx_teachers_department	teachers	department_id	Liste par département
idx_courses_promotion	courses	promotion_id	Cours par promotion
idx_courses_teacher	courses	teacher_id	Cours par enseignant
idx_grades_student	grades	student_id	Notes par étudiant
idx_grades_course	grades	course_id	Notes par cours
idx_attendance_student	attendance	student_id	Présences par étudiant
idx_attendance_date	attendance	attendance_date	Tri par date
idx_payments_student	payments	student_id	Paiements par étudiant
idx_notifications_user	notifications	user_id	Notifs par utilisateur
idx_notifications_read	notifications	is_read	Filtrage non lues

psql -- CREATE INDEX (x16)

```
-- INDEX DE PERFORMANCE (16 index)

CREATE INDEX idx_users_email          ON users(email);
CREATE INDEX idx_users_role           ON users(role);
CREATE INDEX idx_students_matricule   ON students(matricule);
CREATE INDEX idx_students_promotion   ON students(promotion_id);
CREATE INDEX idx_students_status      ON students(status);
CREATE INDEX idx_teachers_matricule   ON teachers(matricule);
CREATE INDEX idx_teachers_department  ON teachers(department_id);
CREATE INDEX idx_courses_promotion    ON courses(promotion_id);
CREATE INDEX idx_courses_teacher      ON courses(teacher_id);
CREATE INDEX idx_grades_student       ON grades(student_id);
CREATE INDEX idx_grades_course        ON grades(course_id);
CREATE INDEX idx_attendance_student   ON attendance(student_id);
CREATE INDEX idx_attendance_date      ON attendance(attendance_date);
CREATE INDEX idx_payments_student     ON payments(student_id);
CREATE INDEX idx_notifications_user   ON notifications(user_id);
CREATE INDEX idx_notifications_read   ON notifications(is_read);
```

NOTE

Impact : chaque index réduit le temps de réponse de > 100ms à < 5ms sur 10 000+ enregistrements.

9 Triggers & Fonctions

Automatisation PL/pgSQL

Un trigger BEFORE UPDATE met à jour automatiquement le champ updated_at :

Fonction PL/pgSQL

```
-- FONCTION PL/pgSQL

CREATE OR REPLACE FUNCTION update_updated_at_column()
RETURNS TRIGGER AS $$
BEGIN
    NEW.updated_at = CURRENT_TIMESTAMP;
    RETURN NEW;
END;
$$ LANGUAGE 'plpgsql';
```

TRIGGERS -- BEFORE UPDATE (x8) — 8 lignes

Trigger	Table	Événement	Fonction
update_users_updated_at	users	BEFORE UPDATE	update_updated_at_column()
update_faculties_updated_at	faculties	BEFORE UPDATE	update_updated_at_column()
update_departments_updated_at	departments	BEFORE UPDATE	update_updated_at_column()
update_promotions_updated_at	promotions	BEFORE UPDATE	update_updated_at_column()
update_students_updated_at	students	BEFORE UPDATE	update_updated_at_column()
update_teachers_updated_at	teachers	BEFORE UPDATE	update_updated_at_column()
update_courses_updated_at	courses	BEFORE UPDATE	update_updated_at_column()
update_grades_updated_at	grades	BEFORE UPDATE	update_updated_at_column()

NOTE Le trigger se déclenche automatiquement à chaque UPDATE. Aucune modification côté Next.js n'est nécessaire.

10 Comptes par Défaut

Identifiants de développement

Le système est pré-configuré avec les comptes ci-dessous. En production, tous les mots de passe doivent être changés.

SUPER_ADMIN — 1 lignes

Nom	Email	Mot de passe	Rôle
Jean-Pierre Mbeki	admin@unikin.ac.cd	Admin@2026	SUPER_ADMIN

ENSEIGNANTS (TEACHER) — 4 lignes

Nom	Email	Mot de passe
François Kabongo	prof.kabongo@unikin.ac.cd	Prof@2026
Joseph Mukendi	prof.mukendi@unikin.ac.cd	Prof@2026
Thérèse Lukusa	prof.lukusa@unikin.ac.cd	Prof@2026
Jean-Pierre Kabongo	jeanpierre.kabongo1@unikin.ac.cd	Prof@2026

ÉTUDIANTS (STUDENT) — 8 lignes

Nom	Email	Mot de passe
Chris Ngozulu	2111188338@unikin.ac.cd	Etudiant@2026
Marie Kasongo	etudiant.kasongo@student.unikin.ac.cd	Etudiant@2026
David Tshimanga	etudiant.tshimanga@student.unikin.ac.cd	Etudiant@2026
Grace Mbala	grace.mbala@student.unikin.ac.cd	Etudiant@2026
Josué Kalala	josue.kalala@student.unikin.ac.cd	Etudiant@2026
Esther Lunda	esther.lunda@student.unikin.ac.cd	Etudiant@2026
Samuel Nkulu	samuel.nkulu@student.unikin.ac.cd	Etudiant@2026
Étudiant 2122340977	2122340977@unikin.ac.cd	Etudiant@2026

EMPLOYÉS (EMPLOYEE) — 1 lignes

Nom	Email	Mot de passe
Pierre Mutombo	employe.mutombo@unikin.ac.cd	Employe@2026

SÉCURITÉ Ces mots de passe sont réservés au développement. En production : changer tous les mots de passe, activer le 2FA, configurer les politiques de rotation.

NEXUS UNIKIN

Documentation Base de Données

14 Tables | 16 Index | 8 Triggers

PostgreSQL 15+ | Next.js 14.1.0 | TypeScript