

IMPLEMENTASI NORMALISASI DAN KONTROL KEAMANAN PADA BASIS DATA SISTEM INFORMASI AKADEMIK BERBASIS MYSQL

¹Reza Asriano Maulana, ²Halimil Fathi, ³Sukrina Herman

^{1,2,3}Teknologi Rekayasa Perangkat Lunak, Politeknik Engineering Indorama, ^{1,2,3}Indonesia

¹e-mail: rzaeid@proton.me

Abstrak

Sistem Informasi Akademik (SIKAD) mengelola data sensitif perguruan tinggi yang rentan terhadap manipulasi dan serangan SQL Injection. Skema pengamanan yang umumnya bertumpu pada lapisan aplikasi seringkali menyisakan kerentanan apabila validasi aplikasi berhasil ditembus (bypass). Penelitian ini bertujuan mengimplementasikan normalisasi basis data dan kontrol keamanan pada skema SIKAD berbasis MySQL. Keamanan diwujudkan dengan mengintegrasikan struktur relasional bentuk normal ketiga (3NF) dan mekanisme validasi internal langsung di tingkat Database Management System (DBMS). Metode penelitian menggunakan Rapid Application Development (RAD) dengan empat tahapan utama. Implementasi teknis dilakukan pada MySQL DBMS menggunakan storage engine InnoDB, batasan kolom (CHECK constraint, NOT NULL, UNIQUE, FOREIGN KEY referensial), dan 10 aturan bisnis dinamis berbasis trigger untuk tiga proses: penjadwalan kelas, transaksi Kartu Rencana Studi (KRS), dan penilaian hasil studi. Pengujian dilakukan pada 13 skenario (7 skenario trigger, 3 skenario CHECK constraint, 3 skenario SQL Injection/SQLmap) menggunakan kueri terparameterisasi (prepared statements) berbasis PHP PDO, simulasi penetrasi manual, pemindaian SQLmap --level=3 --risk=2, serta General Query Log sebagai audit logging. Hasil pengujian menunjukkan 9 skenario penolakan (skenario 1-2, 4-10) ditolak DBMS secara mandiri dengan SQLSTATE 45000 atau pelanggaran CHECK; 1 skenario konversi otomatis (skenario 3: 3,75 menjadi AB) berhasil; 1 skenario penetrasi manual (skenario 11) diproses sebagai literal dan mengembalikan kosong pada endpoint aman; pemindaian kontrol rentan (skenario 12) terdeteksi vulnerable; pemindaian endpoint aman (skenario 13) tidak ditemukan parameter injectable. Seluruh aktivitas kueri tercatat pada audit logging. Dalam batas 13 skenario yang diuji, skema tidak terdeteksi rentan pada parameter yang diuji untuk penjadwalan, KRS, dan penilaian.

Kata kunci: SIKAD, keamanan basis data, normalisasi 3NF, trigger, prepared statements

Abstract

Academic Information System (SIKAD) manages sensitive university data vulnerable to manipulation and SQL Injection attacks. Application-layer security often leaves vulnerabilities if validation is bypassed. This study implements database normalization and security controls on a MySQL-based SIKAD schema by integrating third normal form (3NF) structure and internal validation at the DBMS level. Using Rapid Application Development (RAD) with four main phases, implementation uses InnoDB storage engine, column constraints (CHECK constraint, NOT NULL, UNIQUE, referential FOREIGN KEY), and 10 trigger-based rules for class scheduling, study plan (KRS) transactions, and grade assessment. Testing covers 13 scenarios (7 trigger, 3 CHECK constraint, 3 SQL Injection/SQLmap) using PHP PDO parameterized queries (prepared statements), manual penetration, SQLmap scanning with --level=3 --risk=2, and General Query Log as audit logging. Results show 9 rejection scenarios (1-2, 4-10) independently rejected with SQLSTATE 45000 or CHECK violation; 1 auto-conversion (scenario 3: 3.75 to AB) succeeded; 1 manual test (scenario 11) processed as literal with empty result on the secure endpoint; vulnerable control scan (scenario 12) detected as vulnerable; secure endpoint scan (scenario 13) with no injectable parameter. All query activities were logged. Within the 13 tested scenarios, the schema was not found vulnerable on the tested parameters for scheduling, KRS, and grading.

Keywords: SIKAD, database security, 3NF normalization, trigger, prepared statements

Implementasi Normalisasi Dan Kontrol Keamanan Pada Basis Data Sistem Informasi Akademik Berbasis Mysql,
Reza Asriano Maulana, Halimil Fathi, Sukrina Herman

©2026 LPPM Politeknik Engineering Indorama. Hak Cipta Dilindungi

Artikel ini merupakan artikel akses terbuka yang didistribusikan berdasarkan ketentuan lisensi CC BY.

(<https://creativecommons.org/licenses/by/4.0/>)

DOI: 10.61713/jrt.v6i2.377

1. PENDAHULUAN

Sistem Informasi Akademik (SIKAD) merupakan infrastruktur teknologi yang krusial untuk mengelola data akademik di perguruan tinggi [1]. Mengingat SIKAD menyimpan data sensitif seperti data pribadi mahasiswa, jadwal kuliah, hingga rekapitulasi nilai, kebocoran atau manipulasi pada basis data ini dapat merusak kredibilitas institusi secara serius [2]. Walaupun pengamanan umumnya berfokus pada lapisan aplikasi, kerentanan terhadap serangan SQL Injection menunjukkan bahwa proteksi tambahan di tingkat basis data tetap diperlukan.

Dalam operasionalnya, SIKAD mengintegrasikan tiga proses bisnis akademik utama: penjadwalan kelas, pengisian Kartu Rencana Studi (KRS), dan penilaian hasil studi [3], [4]. Ketiga proses ini menghasilkan data transaksi yang sangat sensitif dan harus dijaga validitasnya dari kesalahan entri maupun akses tidak sah guna menjamin kelangsungan proses akademik [5]. Penelitian sebelumnya telah banyak membahas pembersihan redundansi data melalui normalisasi basis data [6]. Beberapa studi juga menguji celah keamanan secara pasif menggunakan alat pemindai kerentanan atau audit log transaksi [7], [5], [15]. Namun, terdapat celah penelitian di mana studi-studi tersebut umumnya memisahkan antara perancangan struktur relasional dengan penegakan aturan bisnis secara aktif pada level basis data. Akibatnya, validasi data masih bertumpu pada lapisan aplikasi, sehingga jika lapisan aplikasi berhasil ditembus (bypass), basis data tidak memiliki filter internal untuk menolak kueri manipulasi data ilegal [8], [9], [10].

Untuk mengatasi keterbatasan tersebut, penelitian ini merancang skema basis data SIKAD berbasis MySQL dengan menerapkan mekanisme validasi internal langsung di tingkat DBMS [12], [14]. Kebaruan penelitian ini terletak pada penggabungan struktur relasional 3NF dengan otomatisasi batasan integritas serta trigger untuk menegakkan aturan bisnis penjadwalan, KRS, dan penilaian di sisi basis data. Metode Rapid Application Development (RAD) digunakan untuk menyelaraskan desain basis data dengan kebutuhan operasional pengguna secara iteratif [4]. Artikel ini berkontribusi dalam menyediakan arsitektur pertahanan berlapis (defense in depth) yang menjamin integritas data tetap terjaga sekalipun lapisan aplikasi berhasil ditembus.

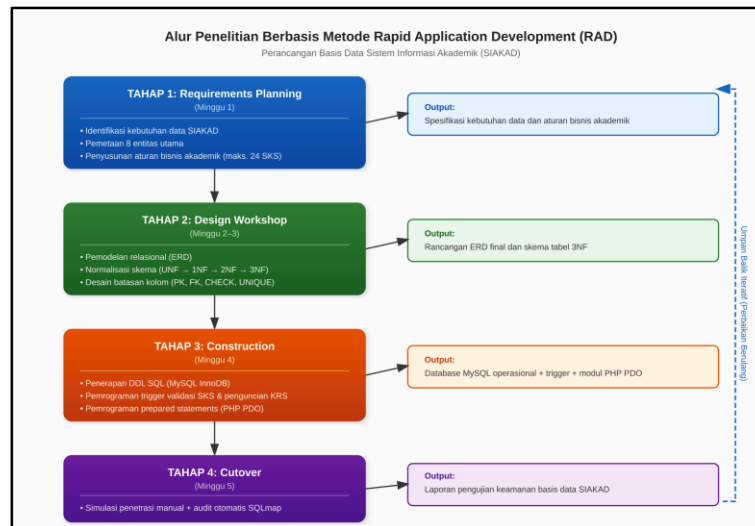
Penelitian ini bertujuan untuk merancang, mengimplementasikan, dan menguji skema basis data SIKAD yang terstruktur secara optimal serta dilengkapi dengan mekanisme validasi internal untuk mencegah manipulasi data ilegal pada proses penjadwalan, pengisian KRS, dan penilaian. Adapun batasan masalah dalam penelitian ini adalah sebagai berikut:

1. Penelitian berfokus pada perancangan skema basis data dan pengujian kontrol keamanan di tingkat DBMS, bukan pengembangan aplikasi SIKAD secara utuh.
2. DBMS yang digunakan dibatasi pada MySQL dengan storage engine InnoDB [14].
3. Kontrol keamanan yang diimplementasikan dan diuji mencakup normalisasi 3NF, batasan integritas kolom (CHECK, NOT NULL, UNIQUE, FOREIGN KEY), trigger (untuk penjadwalan, KRS, dan penilaian), prepared statements melalui PHP PDO [11], [13], serta General Query Log sebagai mekanisme audit logging bawaan MySQL. Mekanisme lain seperti enkripsi kolom (column-level encryption), stored procedures, dan views tidak termasuk dalam lingkup penelitian.
4. Proses bisnis akademik yang menjadi lingkup pengujian mencakup penjadwalan kelas, pengisian KRS, dan penilaian hasil studi. Proses bisnis lain seperti pendaftaran mahasiswa baru dan manajemen kurikulum berada di luar lingkup.
5. Pengujian SQL Injection dilakukan melalui perbandingan antara endpoint tanpa prepared statements (rentan) dan endpoint dengan prepared statements (aman) menggunakan SQLmap. Pengujian terhadap jenis serangan lain seperti cross-site scripting (XSS) dan cross-site request forgery (CSRF) tidak dibahas.
6. Hak akses basis data untuk pengguna aplikasi diterapkan menggunakan prinsip least privilege, namun mekanisme otorisasi berbasis peran (role-based access control) di tingkat aplikasi tidak dibahas.

2. METODE PENELITIAN

2.1. Alur Penelitian Berbasis *Rapid Application Development (RAD)*

Penelitian ini dilaksanakan menggunakan kerangka kerja Rapid Application Development (RAD) yang terdiri dari empat tahapan utama: Requirements Planning, Design Workshop, Construction, dan Cutover [4]. Penggunaan RAD memungkinkan perancangan, pengujian, dan perbaikan skema keamanan basis data secara iteratif berdasarkan kebutuhan operasional perguruan tinggi. Alur penelitian berbasis RAD pada penelitian ini ditunjukkan pada Gambar 1.



Gambar 1. Net Alur penelitian berbasis metode RAD pada perancangan basis data SIADKAD.

Gambar 1. menunjukkan alur penelitian perancangan basis data Sistem Informasi Akademik (SIADKAD) menggunakan metode Rapid Application Development (RAD), yang terdiri dari empat tahap, yaitu *Requirements Planning*, *Design Workshop*, *Construction*, dan *Cutover*, beserta keluaran pada setiap tahap.

2.2. Tahap 1 *Requirements Planning*

Tahap awal dimulai dengan mengidentifikasi kebutuhan data transaksi pada SIADKAD melalui observasi sistem berjalan. Pemetaan dilakukan terhadap delapan entitas data utama yang wajib dikelola, yaitu program studi, mahasiswa, dosen, mata kuliah, kelas, KRS, Detail KRS, dan akun pengguna [3], [4], [12]. Dari pemetaan ini, disusun aturan akademik yang krusial untuk diamankan, seperti pembatasan beban SKS maksimal 24 SKS per semester dan pembatasan hak akses mahasiswa terhadap data nilai. Output dari tahap ini berupa spesifikasi kebutuhan data dan aturan bisnis akademik [1].

2.3. Tahap 2 *Design Workshop*

Pada tahap ini, dilakukan pemodelan data secara teoritis dan desain arsitektur relasional:

1. Pemodelan Relasional. Menyusun diagram hubungan entitas (Entity Relationship Diagram atau ERD) untuk menggambarkan struktur data SIADKAD.
2. Normalisasi Skema. Melakukan dekomposisi tabel dari tahap Unnormalized Form (UNF) hingga bentuk normal ketiga (3NF) guna menghilangkan redundansi dan potensi anomali data [6].
3. Desain Batasan Kolom. Merancang batasan integritas statis di MySQL (seperti primary key, foreign key, CHECK constraint, unique, dan not null) untuk menjaga keabsahan isian kolom secara otomatis [5], [14].

Output pada tahap ini adalah rancangan ERD final dan skema tabel 3NF yang siap diimplementasikan.

2.4. Tahap 3 Construction

Setelah rancangan disetujui, dilakukan konstruksi teknis secara langsung di MySQL:

1. Penerapan Sintaks DDL. Menulis dan mengeksekusi skrip Data Definition Language (DDL) SQL untuk membangun struktur tabel fisik menggunakan storage engine InnoDB guna menjamin integritas transaksi [7], [14]. Struktur program_studi mencakup jenjang D3, D4, S1, S2, S3 agar mendukung kurikulum TRPL D4; batasan CHECK akun relasional yang tidak dapat diterapkan karena ERROR 3823 (HY000) pada MySQL 8.0 dipindahkan ke trigger validasi akun.
2. Pemrograman Basis Data (trigger). Menulis 10 kode trigger: trg_validasi_max_sks dan trg_validasi_max_sks_update untuk validasi SKS dan penguncian KRS, trg_penjadwalan_bentrok untuk cegah bentrok jadwal dosen, trg_auto_konversi_nilai untuk konversi nilai skala IPB, tiga trigger sinkronisasi total_sks (insert, update, delete), trg_validasi_delete_krs untuk proteksi hapus KRS APPROVED, serta trg_akun_validasi_insert dan trg_akun_validasi_update untuk validasi relasi akun menggantikan CHECK yang terlimitasi ERROR 3823.
3. Pemrograman prepared statements. Membangun kueri penarikan data KRS menggunakan PHP PDO terparameterisasi native (ATTR_EMULATE_PREPARES=false) via db-connect-pdo.php sebagai mekanisme pertahanan terhadap SQL Injection [2], [13]. Endpoint rentan kontrol menggunakan mysqli string concatenation (search_vulnerable.php) disiapkan sebagai pembanding.

Output dari tahap ini berupa basis data operasional MySQL yang dilengkapi dengan 10 trigger keamanan serta modul koneksi aplikasi PDO dan mysqli.

2.5. Tahap 4 Cutover

Tahap akhir difokuskan pada pengujian, evaluasi, dan aktivasi mekanisme pemantauan untuk menguji ketahanan pada skenario yang dirancang:

1. Simulasi Penetrasi Manual. Mensimulasikan input karakter khusus (seperti kutip tunggal dan perintah OR/AND) pada form aplikasi untuk menguji apakah kueri ilegal dapat lolos ke basis data [8], [10].
2. Audit Celah Keamanan Otomatis. Menjalankan pemindaian otomatis menggunakan SQLmap untuk menguji kekuatan prepared statements terhadap ancaman SQL Injection [9], [11].
3. Aktivasi Audit Logging. Mengaktifkan General Query Log MySQL dengan output ke tabel (log_output = 'TABLE') untuk mencatat seluruh aktivitas kueri secara otomatis sebagai mekanisme pemantauan (monitoring) dan audit jejak aktivitas basis data.

Rancangan pengujian dibagi empat kelompok. (a) Constraint: uji domain CHECK/UNIQUE dengan input di luar domain, kriteria lolos jika transaksi dibatalkan. (b) Trigger: uji bentrok jadwal dosen, batas 24 SKS/penguncian KRS, dan konversi nilai, kriteria lolos jika DBMS mengembalikan SQLSTATE 45000 atau nilai terkonversi otomatis. (c) Prepared statement: uji payload kutip tunggal/OR pada endpoint aman versus kontrol, kriteria lolos jika endpoint aman mengembalikan kosong sedangkan kontrol mengembalikan data. (d) SQLmap dan audit logging: pindai dengan opsi --level=3 --risk=2, kriteria lolos jika endpoint aman tidak injectable; seluruh kueri harus tercatat di mysql.general_log. Realisasi 13 skenario dirangkum pada Tabel 2.

Seluruh pengujian dilakukan pada MySQL versi 8.0, PHP 8.5.4 (kompatibel 8.1/8.2) dengan ekstensi PDO dalam mode native prepared statements (ATTR_EMULATE_PREPARES=false) dan ekstensi php-mysql, sistem operasi Ubuntu Server 26.04 LTS (Resolute Raccoon, rilis 23 April 2026), dan SQLmap 1.10.4#stable (kompatibel 1.7.x/1.8.x) untuk menjamin aspek replikabilitas (reproducibility) penelitian

3. HASIL DAN PEMBAHASAN

3.1. Kebutuhan Data Sistem Akademik

Untuk mendukung proses operasional pada Sistem Informasi Akademik (SIKAD) dasar, dirancang delapan kebutuhan data utama berikut:

*Implementasi Normalisasi Dan Kontrol Keamanan Pada Basis Data Sistem Informasi Akademik Berbasis Mysql,
Reza Asriano Maulana, Halimil Fathi, Sukrina Herman*

©2026 LPPM Politeknik Enjinering Indorama. Hak Cipta Dilindungi

Artikel ini merupakan artikel akses terbuka yang didistribusikan berdasarkan ketentuan lisensi CC BY.

(<https://creativecommons.org/licenses/by/4.0/>)

DOI: 10.61713/jrt.v6i2.377.

1. Data Program Studi. Menyimpan identitas program studi, jenjang pendidikan, dan kode prodi.
2. Data Mahasiswa. Menyimpan data pribadi mahasiswa, nomor induk mahasiswa (NIM), angkatan, serta program studi yang diambil.
3. Data Dosen. Menyimpan identitas dosen, nomor induk dosen nasional (NIDN), nama, dan kontak.
4. Data Mata Kuliah. Menyimpan daftar mata kuliah, kode mata kuliah, beban SKS, dan semester penawaran.
5. Data Kelas. Menyimpan jadwal kelas akademik yang dibuka per semester, termasuk hari, jam, kapasitas, dosen pengampu, dan mata kuliah terkait.
6. Data KRS. Menyimpan lembar Kartu Rencana Studi (KRS) utama per mahasiswa per semester dan status persetujuannya.
7. Data Detail KRS. Menyimpan daftar kelas yang diambil oleh mahasiswa di dalam KRS-nya beserta nilai hasil studinya.
8. Data Akun Pengguna. Menyimpan kredensial login (username dan password hash) demi keamanan akses.

Pemilihan delapan entitas tersebut merepresentasikan struktur minimal dari aktivitas akademik perguruan tinggi. Pemisahan entitas kelas dari mata kuliah sengaja dilakukan untuk membedakan antara kurikulum akademik yang bersifat statis (mata kuliah) dengan penawaran kelas perkuliahan yang dinamis secara semesteran dengan melibatkan dosen pengampu.

3.2 Aturan Bisnis Sistem Akademik

Aturan bisnis (business rules) yang ditetapkan untuk membatasi dan mengarahkan perilaku operasional basis data SIAKAD ini meliputi:

1. Aturan Identitas dan Relasi 1. Setiap mahasiswa hanya memiliki satu Nomor Induk Mahasiswa (NIM) yang bersifat unik secara nasional. 2. Setiap dosen memiliki satu Nomor Induk Dosen Nasional (NIDN) yang unik sebagai identitas utama pengajaran. 3. Setiap mata kuliah diidentifikasi secara unik menggunakan Kode Mata Kuliah. 4. Semua kolom identitas penting (seperti nama, NIM, kode MK, NIDN) bersifat wajib dan tidak boleh kosong (NOT NULL).
2. Aturan Penjadwalan 5. Setiap kelas harus memiliki hari, jam mulai, dan jam selesai yang valid. 6. Jam selesai harus lebih lambat dari jam mulai ($\text{jam_selesai} > \text{jam_mulai}$). 7. Dosen tidak boleh mengajar dua kelas berbeda pada hari dan jam yang sama di tahun ajaran dan semester yang sama.
3. Aturan Penilaian 8. Nilai huruf (nilai_huruf) diisi secara otomatis berdasarkan nilai angka (nilai_angka) menggunakan skala konversi IPB: A (4.0), AB (3.5), B (3.0), BC (2.5), C (2.0), D (1.0), E (0.0). 9. Input nilai hanya boleh dilakukan pada KRS yang sudah berstatus APPROVED.
4. Aturan KRS dan SKS 10. Mahasiswa hanya dapat mengambil mata kuliah di kelas yang aktif ditawarkan pada semester berjalan melalui pengisian KRS. 11. Satu lembar KRS dapat memiliki banyak baris Detail KRS sesuai dengan jumlah kelas mata kuliah yang diambil. 12. Batas maksimal pengambilan beban studi per mahasiswa dibatasi maksimal 24 SKS dalam satu semester.

3.3. ERD (Entity Relationship Diagram)

3.3.1. Deskripsi Relasi

Hubungan kardinalitas antar entitas dalam basis data SIAKAD dirancang sebagai berikut:

1. program_studi 1 — N mahasiswa: Satu program studi dapat mendidik banyak mahasiswa (One-to-Many).
2. program_studi 1 — N mata_kuliah: Satu program studi dapat memiliki banyak mata kuliah (One-to-Many).
3. mata_kuliah 1 — N kelas: Satu mata kuliah dapat dibuka menjadi beberapa kelas (One-to-Many).
4. dosen 1 — N kelas: Satu dosen dapat mengajar di banyak kelas (One-to-Many).

Implementasi Normalisasi Dan Kontrol Keamanan Pada Basis Data Sistem Informasi Akademik Berbasis Mysql, Reza Asriano Maulana, Halimil Fathi, Sukrina Herman

©2026 LPPM Politeknik Enjinering Indorama. Hak Cipta Dilindungi

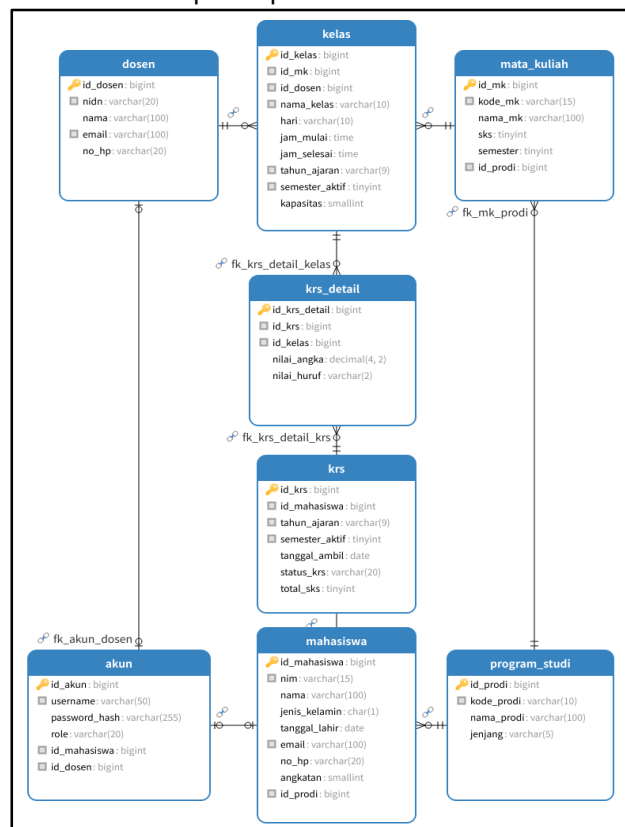
Artikel ini merupakan artikel akses terbuka yang didistribusikan berdasarkan ketentuan lisensi CC BY.

(<https://creativecommons.org/licenses/by/4.0/>)

DOI: 10.61713/jrt.v6i2.377.

5. mahasiswa 1 — N krs: Satu mahasiswa dapat mengajukan KRS di setiap semester yang berbeda (One-to-Many).
6. krs 1 — N krs_detail: Satu KRS memuat banyak detail pengambilan kelas (One-to-Many).
7. kelas 1 — N krs_detail: Satu kelas dapat dipilih oleh banyak mahasiswa di lembar Detail KRS (One-to-Many).
8. mahasiswa 1 — 1 akun: Satu mahasiswa dapat memiliki satu akun pengguna (One-to-One).
9. dosen 1 — 1 akun: Satu dosen dapat memiliki satu akun pengguna (One-to-One).

Hubungan relasional ini dirancang untuk meminimalkan redundansi data. Sebagai contoh, hubungan many-to-many antara mahasiswa dan kelas diselesaikan melalui entitas KRS dan krs_detail guna mencatat riwayat rencana studi mahasiswa secara historis per semester secara terstruktur. Rincian visual ERD ditampilkan pada Gambar 2.



Gambar 2. Entity Relationship Diagram

Gambar 2 menunjukkan diagram relasi basis data (ERD) SIAKAD yang menghubungkan tabel dosen, mahasiswa, kelas, mata kuliah, KRS, detail KRS, akun, dan program studi untuk mendukung pengelolaan data akademik secara terstruktur dan terintegrasi.

3.4. Rancangan Tabel dan Relasi

1. program_studi. Menyimpan informasi program studi dengan primary key pengganti (id_prodi).
2. mahasiswa. Terhubung ke tabel program_studi melalui foreign key id_prodi.
3. dosen. Berisi NIDN dan data profil dosen pengajar.
4. mata_kuliah. Terhubung ke program_studi untuk pemetaan kurikulum prodi.
5. kelas. Menghubungkan mata_kuliah dan dosen untuk jadwal pengajaran di semester aktif, dilengkapi kolom hari dan jam kuliah untuk mendukung validasi penjadwalan.

6. KRS. Menyimpan lembar KRS utama per mahasiswa per tahun ajaran dan semester aktif, dilengkapi kolom total_sks untuk kalkulasi akumulasi SKS.
7. krs_detail. Tabel penghubung many-to-many antara krs dan kelas, menyimpan rekam nilai mahasiswa.
8. akun. Menyimpan kredensial login pengguna (username, password hash, dan role) dengan relasi opsional ke tabel mahasiswa atau dosen.

3.5. DDL SQL (MySQL Syntax)

Berikut adalah struktur DDL SQL yang disesuaikan secara khusus untuk RDBMS MySQL. Untuk efisiensi, tiga tabel yang paling kompleks secara keamanan ditampilkan dalam bentuk kode penuh, sedangkan lima tabel sisanya diringkas dalam bentuk tabel deskriptif.

3.5.1. Kode DDL Kelas

Tabel kelas merupakan entitas kunci untuk modul penjadwalan. Penambahan kolom hari, jam_mulai, dan jam_selesai memungkinkan validasi bentrok jadwal dosen secara langsung di tingkat DBMS dapat dilihat pada Tabel 1..

Tabel 1. Source Code DDL Tabel Kelas

```
CREATE TABLE kelas (
  id_kelas BIGINT AUTO_INCREMENT PRIMARY KEY,
  id_mk BIGINT NOT NULL,
  id_dosen BIGINT NOT NULL,
  nama_kelas VARCHAR(10) NOT NULL,
  hari VARCHAR(10) NOT NULL,
  jam_mulai TIME NOT NULL,
  jam_selesai TIME NOT NULL,
  tahun_ajaran VARCHAR(9) NOT NULL,
  semester_aktif TINYINT NOT NULL,
  kapasitas SMALLINT NOT NULL,
  CONSTRAINT chk_kelas_semester CHECK (semester_aktif IN (1, 2)),
  CONSTRAINT chk_kelas_jam CHECK (jam_selesai > jam_mulai),
  CONSTRAINT chk_kelas_kapasitas CHECK (kapasitas > 0),
  CONSTRAINT fk_kelas_mk FOREIGN KEY (id_mk)
    REFERENCES mata_kuliah (id_mk)
    ON UPDATE CASCADE ON DELETE RESTRICT,
  CONSTRAINT fk_kelas_dosen FOREIGN KEY (id_dosen)
    REFERENCES dosen (id_dosen)
    ON UPDATE CASCADE ON DELETE RESTRICT,
  CONSTRAINT uq_kelas UNIQUE (id_mk, id_dosen, nama_kelas, tahun_ajaran, semester_aktif)
) ENGINE=InnoDB;
```

Tabel 1 menunjukkan kode SQL DDL untuk membuat tabel kelas pada basis data SIAKAD, yang mencakup definisi atribut, kunci utama, kunci tamu, batasan validasi data, serta aturan keunikan untuk menjaga konsistensi data kelas.

3.5.2. Kode DDL KRS Detail

Tabel krs_detail merupakan entitas transaksi utama yang menjadi studi kasus pengujian validasi KRS dan penilaian otomatis. Untuk kueri yang digunakan pada tabel KRS Detail dapat dilihat pada Tabel 2.

Tabel 2. Source Code DDL Table KRS Detail

```
CREATE TABLE krs_detail (
```

```

id_krs_detail BIGINT AUTO_INCREMENT PRIMARY KEY,
id_krs BIGINT NOT NULL,
id_kelas BIGINT NOT NULL,
nilai_angka DECIMAL(4,2) NULL,
nilai_huruf VARCHAR(2) NULL,
CONSTRAINT chk_detail_nilai_angka CHECK (nilai_angka BETWEEN 0.00 AND 4.00),
CONSTRAINT chk_detail_nilai_huruf CHECK (nilai_huruf IN ('A', 'AB', 'B', 'BC', 'C', 'D', 'E')),
CONSTRAINT fk_krs_detail_krs FOREIGN KEY (id_krs)
REFERENCES krs (id_krs)
ON UPDATE CASCADE ON DELETE CASCADE,
CONSTRAINT fk_krs_detail_kelas FOREIGN KEY (id_kelas)
REFERENCES kelas (id_kelas)
ON UPDATE CASCADE ON DELETE RESTRICT,
CONSTRAINT uq_krs_kelas UNIQUE (id_krs, id_kelas)
) ENGINE=InnoDB;

```

Tabel 2 menunjukkan kode SQL DDL untuk membuat tabel krs_detail yang menyimpan nilai angka dan huruf mahasiswa, dilengkapi kunci utama, kunci tamu, batasan validasi nilai, serta aturan keunikan untuk menjaga integritas data akademik.

3.5.3. Ringkasan Skema Lima Tabel Lainnya

Ringkasan kolom dan constraint pada lima tabel pendukung yang tidak ditampilkan dalam bentuk kode penuh disajikan pada Tabel 3.

Tabel 3. Ringkasan Skema Lima Tabel Pendukung Skema SIAKAD

Tabel	Kolom Utama	Constraint Utama
program_studi	id_prodi (PK), kode_prodi (UNIQUE), nama_prodi, jenjang	CHECK jenjang IN ('D3','D4','S1','S2','S3')
mahasiswa	id_mahasiswa (PK), nim (UNIQUE), nama, jenis_kelamin, angkatan, id_prodi (FK)	CHECK gender, CHECK angkatan, FK ke program_studi
dosen	id_dosen (PK), nidn (UNIQUE), nama, email	UNIQUE nidn
mata_kuliah	id_mk (PK), kode_mk (UNIQUE), nama_mk, sks, semester, id_prodi (FK)	CHECK sks BETWEEN 1 AND 20 (20 untuk Magang Industri), CHECK semester BETWEEN 1 AND 14
krs	id_krs (PK), id_mahasiswa (FK), tahun_ajaran, semester_aktif, status_krs, total_sks	CHECK status_krs IN ('DRAFT','SUBMITTED','APPROVED'), UNIQUE (id_mahasiswa, tahun_ajaran, semester_aktif)

3.6. Trigger untuk Aturan Bisnis Dinamis (MySQL Syntax)

Aturan bisnis dinamis diimplementasikan menggunakan trigger MySQL yang dieksekusi secara otomatis sebelum atau sesudah operasi DML pada tabel krs_detail dan kelas. Tiga trigger representatif ditampilkan dalam bentuk kode penuh, sedangkan sisanya dijelaskan secara naratif.

3.6.1. Trigger Validasi Maksimal SKS

Trigger BEFORE INSERT pada tabel krs_detail memvalidasi bahwa penambahan kelas baru tidak melampaui batas 24 SKS dan memastikan status KRS masih dalam kondisi DRAFT. Validasi dibaca dari kolom total_sks di tabel krs untuk menghindari mutating table (Error 1442). Query yang digunakan untuk Trigger Validasi Maksimal SKS dapat dilihat pada Tabel 4.

Tabel 4. Source Code Trigger Validasi SKS

```
DELIMITER //
CREATE TRIGGER trg_validasi_max_sks
BEFORE INSERT ON krs_detail
FOR EACH ROW
BEGIN
    DECLARE sks_krs_sekarang TINYINT;
    DECLARE sks_kelas_baru TINYINT;
    DECLARE status_krs_sekarang VARCHAR(20);
    DECLARE tahun_ajaran_krs VARCHAR(9);
    DECLARE semester_aktif_krs TINYINT;
    DECLARE tahun_ajaran_kelas VARCHAR(9);
    DECLARE semester_aktif_kelas TINYINT;
    SELECT status_krs, tahun_ajaran, semester_aktif, total_sks
    INTO status_krs_sekarang, tahun_ajaran_krs, semester_aktif_krs, sks_krs_sekarang
    FROM krs WHERE id_krs = NEW.id_krs;
    IF status_krs_sekarang <> 'DRAFT' THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Kesalahan: KRS sudah dikunci dan tidak dapat ditambah kelas!';
    END IF;
    SELECT tahun_ajaran, semester_aktif, mk.sks
    INTO tahun_ajaran_kelas, semester_aktif_kelas, sks_kelas_baru
    FROM kelas k JOIN mata_kuliah mk ON k.id_mk = mk.id_mk
    WHERE k.id_kelas = NEW.id_kelas;
    IF (tahun_ajaran_kelas <> tahun_ajaran_krs) OR (semester_aktif_kelas <> semester_aktif_krs)
    THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Kesalahan: Kelas tidak sesuai tahun ajaran/semester lembar KRS!';
    END IF;
    IF (sks_krs_sekarang + sks_kelas_baru) > 24 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Kesalahan: Total SKS melebihi batas maksimal 24 SKS!';
    END IF;
END //
```

Tabel 4 menunjukkan kode SQL trigger untuk memvalidasi batas maksimal 24 SKS, kesesuaian tahun ajaran dan semester, serta status KRS sebelum mahasiswa menambahkan kelas.

3.7. Analisis Penerapan Batasan Keamanan Skema Tabel

Aspek-aspek perancangan basis data yang aman pada skema MySQL SIAKAD dikelompokkan ke dalam lima kategori analisis:

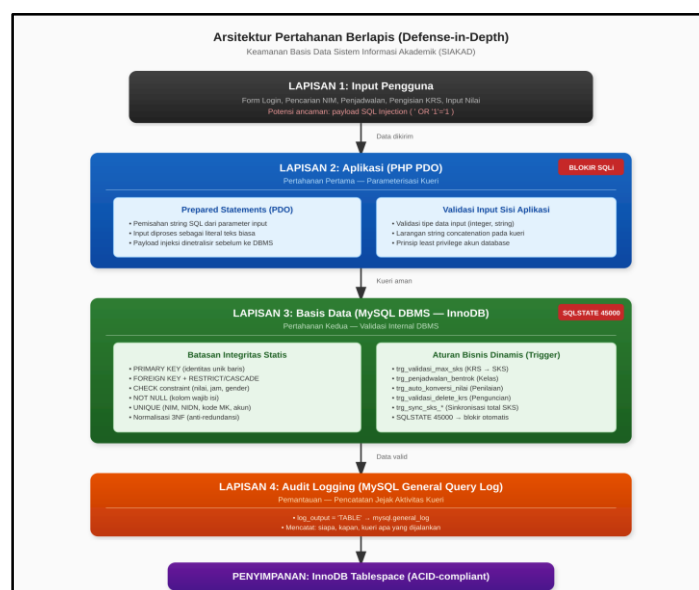
1. Struktur Identitas dan Kunci Utama (Primary & Surrogate Keys). Menetapkan kunci utama numerik otomatis (BIGINT AUTO_INCREMENT) untuk mempercepat indeksasi data,

- serta menggunakan surrogate key guna memisahkan data identitas sistem dari kunci bisnis alami.
2. Integritas Relasional (Foreign Keys & InnoDB). Menghubungkan entitas secara ketat menggunakan kunci tamu (Foreign Key) dengan parameter ON UPDATE CASCADE dan ON DELETE RESTRICT, memecah relasi many-to-many melalui krs_detail, serta mengaktifkan storage engine InnoDB untuk mendukung transaksi ACID-compliant.
 3. Validasi Domain Kolom (CHECK, NOT NULL, UNIQUE). Menerapkan CHECK constraint untuk membatasi domain isian yang sah (gender, angkatan, SKS, jam, status KRS), mewajibkan kolom penting terisi (NOT NULL), dan menjamin keunikan identitas (UNIQUE). Penambahan CHECK (jam_selesai > jam_mulai) memastikan validitas data penjadwalan.
 4. Otomatisasi Aturan Bisnis. Trigger BEFORE INSERT pada kelas mencegah bentrok jadwal dosen. Trigger BEFORE UPDATE pada krs_detail mengotomatiskan konversi nilai angka ke huruf. Trigger pada krs_detail menegakkan pembatasan SKS dan status penguncian KRS.
 5. Pencatatan Jejak Aktivitas (Audit Logging). MySQL menyediakan mekanisme General Query Log dengan output berupa tabel yang mencatat seluruh aktivitas kueri secara otomatis tanpa memerlukan tabel audit manual. Aktivasi dan analisis lebih lanjut dibahas pada Subbab 3.11.

3.8. Contoh Implementasi Keamanan (PHP PDO MySQL)

Sebagai pertahanan berlapis (defense in depth) di sisi aplikasi, integrasi ke basis data dilakukan menggunakan teknik prepared statements via ekstensi PHP Data Objects (PDO) native (ATTR_EMULATE_PREPARES=false) untuk menetralkan serangan SQL Injection [11], [13]. Implementasi aman menggunakan db-connect-pdo.php dan search_safe.php dengan prepare(:nim) dan execute(['nim'=>...]), sementara kontrol rentan search_vulnerable.php menggunakan mysqli string concatenation sebagai pembanding. Melalui metode ini, string kueri SQL dipisahkan secara tegas dari nilai parameter input pengguna. Ketika kueri dikirim ke DBMS MySQL, parameter input dibaca murni sebagai konstanta literal teks dan bukan sebagai bagian dari instruksi kueri yang dapat dieksekusi. Pengujian manual curl -G --data-urlencode "nim=' OR '1'='1" menunjukkan payload diproses sebagai nilai NIM literal dan mengembalikan kosong pada endpoint aman, sementara endpoint rentan mengembalikan seluruh data (skenario 11, Tabel 2). Pada endpoint dan payload yang diuji, tidak ditemukan parameter rentan pada endpoint aman (skenario 13), sedangkan endpoint kontrol terdeteksi rentan (skenario 12).

Konsep pertahanan berlapis ini membagi tanggung jawab keamanan secara sistemik: prepared statements mengamankan jalur masuk data pada lapisan aplikasi (kode PHP), sementara trigger dan constraints menegakkan integritas dan aturan bisnis akademik secara absolut pada lapisan basis data (DBMS MySQL). Sinergi ini memastikan bahwa apabila salah satu lapisan pengamanan berhasil di-bypass atau ditembus, lapisan lainnya tetap aktif memblokir akses ilegal atau manipulasi data yang tidak sah. Untuk mengatasi kegagalan ERROR 3823 (HY000) akibat CHECK pada kolom FK CASCADE, validasi akun dipindah ke trigger trg_akun_validasi_insert/update, dan untuk mengatasi ERROR 1046 (3D000) ditambahkan USE siakad_db; sebelum kueri monitoring. Konsep detail dari arsitektur pertahanan berlapis ini disajikan pada Gambar 6.



Gambar 3. Arsitektur Pertahanan Berlapis (Defense-in-Depth) Keamanan Basis Data.

Gambar 3 menunjukkan arsitektur pertahanan berlapis (Defense-in-Depth) pada basis data SIKAD melalui empat lapisan keamanan, yaitu validasi input, proteksi aplikasi, pengamanan basis data, dan audit aktivitas untuk mencegah serta mendeteksi potensi ancaman keamanan.

3.9. Pengujian Validasi Keamanan dan Integritas Data

Untuk memvalidasi keandalan mekanisme validasi internal langsung di tingkat DBMS yang dirancang, dilakukan serangkaian pengujian fungsionalitas dan keamanan. Pengujian ini bertujuan menguji apakah skema basis data yang dinormalisasi, dikombinasikan dengan batasan integritas statis (*CHECK constraint*) dan aturan bisnis dinamis (*trigger*), mampu menolak manipulasi data yang tidak sah pada skenario yang diuji. Pengujian juga memverifikasi efektivitas *prepared statements* terhadap ancaman *SQL Injection* pada *endpoint* yang diuji, serta menguji mekanisme penjadwalan dan penilaian otomatis. Tabel 5 merangkum skenario, input data, hasil yang diharapkan, hasil riil, pengujian, dan keterangan.

Tabel 5. Skenario Pengujian Validasi Keamanan Basis Data.

No	Skenario Uji	Input Data / Aksi SQL	Hasil yang Diharapkan	Hasil Riil	Pengujian	Keterangan
1	Validasi Bentrok Jadwal Dosen (<i>trigger BEFORE INSERT</i>)	Insert kelas baru dengan dosen yang sudah mengajar pada hari dan jam yang sama di semester yang sama.	Transaksi ditolak oleh <i>trigger</i> dan memicu <i>error</i> kode 45000.	MySQL menolak kueri: <i>"Dosen memiliki jadwal bentrok pada hari dan jam yang sama!"</i>		Lolos (Aman)
2	Validasi Jam Kelas (<i>CHECK constraint</i>)	Insert kelas dengan jam_mulai = '10:00' dan jam_selesai = '08:00'.	Transaksi dibatalkan karena jam_selesai <= jam_mulai.	MySQL menolak kueri: <i>Check constraint 'chk_kelas_jam' is violated.</i>		Lolos (Aman)

3	Auto-Konversi Nilai Angka ke Huruf (<i>trigger BEFORE UPDATE</i>)	Update nilai_angka pada krs_detail dari NULL menjadi 3.75 pada KRS berstatus APPROVED.	nilai_huruf secara otomatis diisi menjadi 'AB' oleh <i>trigger</i> .	MySQL berhasil mengupdate data; nilai_huruf bernilai 'AB'.	Lolos (Aman)
4	Blokir Input Nilai pada KRS DRAFT (<i>trigger BEFORE UPDATE</i>)	Update nilai_angka pada krs_detail yang terkait dengan KRS berstatus DRAFT.	Transaksi ditolak oleh <i>trigger</i> karena KRS belum disetujui.	MySQL menolak kueri: " <i>Input nilai hanya diperbolehkan pada KRS yang sudah disetujui!</i> "	Lolos (Aman)
5	Validasi Batas Maksimal SKS Baru (<i>trigger BEFORE INSERT</i>)	Insert data KRS <i>Detail</i> baru sebesar 6 SKS pada mahasiswa yang telah memiliki beban 20 SKS.	Transaksi dibatalkan secara otomatis oleh <i>trigger</i> dan memicu <i>error</i> kode 45000.	MySQL menolak kueri: " <i>Total SKS melebihi batas maksimal 24 SKS!</i> "	Lolos (Aman)
6	Validasi Batas Maksimal SKS Update (<i>trigger BEFORE UPDATE</i>)	Mengganti kelas pada KRS <i>Detail</i> dari kelas berbobot 3 SKS ke kelas berbobot 6 SKS pada KRS yang telah terisi 20 SKS.	Transaksi pembaruan data ditolak dan memicu <i>error</i> kode 45000.	MySQL menolak kueri: " <i>Total SKS melebihi batas maksimal 24 SKS!</i> "	Lolos (Aman)
7	Proteksi Modifikasi KRS APPROVED (<i>trigger BEFORE INSERT/UPDATE</i>)	<i>Insert/update</i> baris kelas pada lembar KRS yang status_krs bernilai 'APPROVED'.	Transaksi ditolak karena KRS sudah dikunci.	MySQL menolak: " <i>KRS sudah dikunci dan tidak dapat ditambah kelas!</i> "	Lolos (Aman)
8	Proteksi Penghapusan KRS APPROVED (<i>trigger BEFORE DELETE</i>)	<i>Delete</i> baris detail KRS pada lembar KRS yang status_krs bernilai 'APPROVED'.	Transaksi penghapusan ditolak.	MySQL menolak: " <i>KRS sudah dikunci dan kelas tidak dapat dihapus!</i> "	Lolos (Aman)
9	Validasi <i>CHECK constraint</i> (Domain Nilai Huruf)	Insert nilai huruf 'F' ke tabel krs_detail.	Transaksi dibatalkan.	MySQL menolak: <i>Check constraint 'chk_detail_nilai_huruf' is violated.</i>	Lolos (Aman)

10	Validasi <i>CHECK constraint</i> (Domain Gender)	Insert data mahasiswa dengan jenis_kelamin = 'X'.	Transaksi dibatalkan.	MySQL menolak: <i>Check constraint 'chk_mhs_gender' is violated.</i>	Lolos (Aman)
11	Uji Penetrasi <i>SQL Injection (prepared statements)</i>	Input <i>payload</i> ' OR '1'='1 via curl -G --data-urlencode + jq . pada search_safe.php.	Input sebagai teks konstanta <i>literal</i> {"jumlah":0}.	PHP PDO memproses input sebagai <i>literal</i> dan mengembalikan data kosong; kontrol rentan search_vulnerable.php mengembalikan {"jumlah":5} (semua mahasiswa).	Lolos (Aman)
12	Pemindaian SQLmap (Endpoint Rentan)	SQLmap pada <i>endpoint</i> tanpa <i>prepared statements</i> , opsi --level=3 --risk=2 --flush-session atau hapus ~/.local/share/sqlmap/output/127.0.0.1/.	SQLmap mendeteksi parameter rentan.	SQLmap melaporkan parameter rentan (<i>vulnerable</i>) boolean-based blind dan berhasil mengekstrak data (dump 5 mahasiswa minimal / 79 full).	Sesuai Ekspektasi
13	Pemindaian SQLmap (Endpoint Aman)	SQLmap pada <i>endpoint</i> dengan <i>prepared statements</i> PDO native, opsi --level=3 --risk=2 --flush-session.	Seluruh parameter tidak rentan.	SQLmap melaporkan all tested parameters do not appear to be injectable.	Lolos (Aman)

Sebagaimana ditunjukkan pada Gambar 4, penolakan bentrok jadwal (skenario 1, Tabel 2) tercatat pada log pengujian.

```

=== DATA PERSIAPAN BERHASIL DIINSERT ===
id_prodi_test id_mk1 id_mk2 id_dosen1 id_dosen2
2 11 12 3 4
INFO
--- SKENARIO UJI 1: Insert Normal (Bukan Bentrok) ---
id_kelas hari jam_mulai jam_selesai id_dosen
11 senin 08:00:00 10:00:00 3
12 selasa 08:00:00 10:00:00 3
INFO
--- SKENARIO UJI 2: Insert Dengan Bentrok Jadwal Dosen (HARUS DITOLAK - Gambar 3) ---
ERROR 1644 (45000) at line 70: Kesalahan Penjadwalan: Dosen memiliki jadwal bentrok pada hari dan jam yang sama!
INFO
--- SKENARIO UJI 3: Insert Jam Tidak Valid (HARUS DITOLAK) ---
ERROR 1644 (45000) at line 81: Kesalahan Penjadwalan: Jam selesai harus lebih lambat dari jam mulai!
INFO
--- SKENARIO UJI 4: Jam Sama Hari Berbeda (HARUS BERHASIL) ---
INFO
Kelas Kamis berhasil (tidak bentrok):
id_kelas hari jam_mulai
13 kamis 08:00:00
INFO
--- SKENARIO UJI 5: Semester Berbeda (HARUS BERHASIL) ---
INFO
Kelas semester 2 berhasil (tidak bentrok):
id_kelas semester_aktif hari
11 1 senin
14 2 senin
INFO
=== VERIFIKASI AKHIR PENJADWALAN ===
total_kelas_test
4

```

Gambar 4. Hasil pengujian trigger penjadwalan: MySQL menolak penambahan kelas karena dosen memiliki jadwal bentrok.

Sebagaimana ditunjukkan pada Gambar 5, penolakan SKS >24 (skenario 5, Tabel 2) memicu error 45000.

```
INFO
=== DATA PERSIAPAN KRS BERHASIL DIINSERT ===
id_mahasiswa id_krs k1 k8
5 5 16 23
INFO
--- SKENARIO UJI 1: Insert Kelas Normal (3 SKS) ---
total_sks
3
INFO
--- SKENARIO UJI 2: Insert Kelas Sampai 21 SKS ---
total_21_expected
21
INFO
--- SKENARIO UJI 3: Insert Mencapai 24 SKS (HARUS BERHASIL) ---
total_24_expected
24
INFO
--- SKENARIO UJI 4: Insert Melebihi 24 SKS (HARUS DITOLAK - Gambar 5) ---
ERROR 1644 (45000) at line 145: Kesalahan: Total SKS melebihi batas maksimal 24 SKS!
```

Gambar 5. Hasil pengujian trigger validasi SKS: MySQL menolak transaksi KRS yang melebihi batas 24 SKS.

Sebagaimana ditunjukkan pada Gambar 6, modifikasi KRS APPROVED (skenario 7, Tabel 2) ditolak.

```
INFO
--- SKENARIO UJI 5: Setujui KRS ---
status_krs total_sks
APPROVED 24
INFO
--- SKENARIO UJI 6: Insert Kelas Setelah KRS APPROVED (HARUS DITOLAK - Gambar 6) ---
ERROR 1644 (45000) at line 165: Kesalahan: KRS sudah dikunci dan tidak dapat ditambah kelas!
```

Gambar 6. Hasil pengujian proteksi KRS berstatus APPROVED: MySQL menolak modifikasi pada KRS yang telah dikunci.

Sebagaimana ditunjukkan pada Gambar 7, payload pada endpoint aman dikembalikan kosong (skenario 11, Tabel 2).

```
ezasiakad-server:~/database-security-practice/temp$ curl -s -G --data-urlencode "nim=' OR '1'='1" http://127.0.0.1:8080/search_safe.php | jq .
{
  "jumlah": 0,
  "data": []
}
```

Gambar 7. Hasil pengujian SQL Injection melalui PHP PDO: input payload diproses sebagai literal dan mengembalikan 0 data.

Sebagaimana ditunjukkan pada Gambar 8, pemindaian endpoint aman tidak menemukan parameter injectable (skenario 13, Tabel 2).

```
[16:33:12] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (SLEEP)'
[16:33:12] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (SLEEP - comment)'
[16:33:12] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (query SLEEP - comment)'
[16:33:12] [INFO] testing 'MySQL < 5.0.12 AND time-based blind (BENCHMARK)'
[16:33:12] [INFO] testing 'MySQL > 5.0.12 AND time-based blind (heavy query)'
[16:33:12] [INFO] testing 'MySQL >= 5.0.12 RLIKE time-based blind'
[16:33:12] [INFO] testing 'MySQL >= 5.0.12 RLIKE time-based blind (query SLEEP)'
[16:33:12] [INFO] testing 'MySQL AND time-based blind (SLEEP)'
[16:33:12] [INFO] testing 'PostgreSQL >= 8.1 AND time-based blind'
[16:33:12] [INFO] testing 'PostgreSQL AND time-based blind (heavy query)'
[16:33:12] [INFO] testing 'Microsoft SQL Server/Sybase time-based blind (IF)'
[16:33:12] [INFO] testing 'Microsoft SQL Server/Sybase AND time-based blind (heavy query)'
[16:33:12] [INFO] testing 'Oracle AND time-based blind'
[16:33:12] [INFO] testing 'Oracle AND time-based blind (heavy query)'
[16:33:12] [INFO] testing 'IBM DB2 AND time-based blind (heavy query)'
[16:33:12] [INFO] testing 'SQLite > 2.8 AND time-based blind (heavy query)'
[16:33:12] [INFO] testing 'Informix AND time-based blind (heavy query)'
[16:33:12] [INFO] testing 'MySQL >= 5.1 time-based blind (heavy query) - PROCEDURE ANALYSE (EXTRACTVALUE)'
[16:33:12] [INFO] testing 'MySQL >= 5.0.12 time-based blind - Parameter replace'
[16:33:12] [INFO] testing 'MySQL >= 5.0.12 time-based blind - Parameter replace (subtraction)'
[16:33:12] [INFO] testing 'PostgreSQL >= 8.1 time-based blind - Parameter replace'
[16:33:12] [INFO] testing 'Oracle time-based blind - Parameter replace (DBMS_LOCK.SLEEP)'
[16:33:12] [INFO] testing 'Oracle time-based blind - Parameter replace (DBMS_PIPE.RECEIVE_MESSAGE)'
[16:33:12] [INFO] testing 'MySQL >= 5.0.12 time-based blind - ORDER BY, GROUP BY clause'
[16:33:12] [INFO] testing 'PostgreSQL >= 8.1 time-based blind - ORDER BY, GROUP BY clause'
[16:33:12] [INFO] testing 'Oracle time-based blind - ORDER BY, GROUP BY clause (DBMS_LOCK.SLEEP)'
[16:33:12] [INFO] testing 'Oracle time-based blind - ORDER BY, GROUP BY clause (DBMS_PIPE.RECEIVE_MESSAGE)'
[16:33:12] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
[16:33:12] [INFO] testing 'Generic UNION query (random number) - 1 to 10 columns'
[16:33:12] [INFO] testing 'MySQL UNION query (NULL) - 1 to 10 columns'
[16:33:12] [INFO] testing 'MySQL UNION query (random number) - 1 to 10 columns'
[16:33:14] [WARNING] parameter 'Referer' does not seem to be injectable
[16:33:14] [CRITICAL] all tested parameters do not appear to be injectable. Try to increase values for '--level'/'--risk' options if you wish to perform more tests. If you suspect that there is some kind of protection mechanism involved (e.g. WAF) maybe you could try to use option '--tamper' (e.g. '--tamper=space comment') and/or switch '--random-agent'
```

Gambar 8. Hasil pemindaian SQLmap pada endpoint aman: seluruh parameter tidak rentan terhadap injeksi SQL.

Sebagaimana ditunjukkan pada Gambar 9, pemindaian kontrol rentan menemukan parameter vulnerable (skenario 12, Tabel 2).

```

[16:34:03] [INFO] testing 'MySQL >= 5.0.12 stacked queries (query SLEEP)'
[16:34:03] [INFO] testing 'MySQL < 5.0.12 stacked queries (BENCHMARK - comment)'
[16:34:03] [INFO] testing 'MySQL < 5.0.12 stacked queries (BENCHMARK)'
[16:34:03] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)'
[16:34:13] [INFO] GET parameter 'nid' appears to be 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)' injectable
[16:34:13] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
[16:34:13] [INFO] automatically extending ranges for UNION query injection technique tests as there is at least one other (potential) technique found
[16:34:13] [INFO] 'ORDER BY' technique appears to be usable. This should reduce the time needed to find the right number of query columns. Automatically extending the range for current UNION query injection technique test
[16:34:13] [INFO] Target URL appears to have 5 columns in query
[16:34:13] [WARNING] reflective value(s) found and filtering out
[16:34:13] [INFO] GET parameter 'nid' is 'Generic UNION query (NULL) - 1 to 20 columns' injectable
[16:34:13] [INFO] GET parameter 'nid' is vulnerable. Do you want to keep testing the others (if any)? (y/N) N
sqlmap identified the following injection point(s) with a total of 110 HTTP(s) requests:
--
Parameter: nid (GET)
Type: boolean-based blind
Title: AND boolean-based blind - WHERE or HAVING clause (subquery - comment)
Payload: nid=TEST001' AND 1124=(SELECT (CASE WHEN (1124=1124) THEN 1124 ELSE (SELECT 9477 UNION SELECT 9824) END))-- hdsE
Type: time-based blind
Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
Payload: nid=TEST001' AND (SELECT 9070 FROM (SELECT(SLEEP(5)))IDyG)-- Vdai
Type: UNION query
Title: Generic UNION query (NULL) - 5 columns
Payload: nid=TEST001' UNION ALL SELECT CONCAT(0x716a6a7071,0x564e726c6f754e4c415974c4d745a6d78487248776e96d714849414d4b46659614137161625a,0x716b627a71),NULL,NULL,NULL,NULL,--
[16:34:13] [INFO] the back-end DBMS is MySQL
web application technology: PHP 8.5.4
back-end DBMS: MySQL >= 5.0.12
[16:34:13] [WARNING] HTTP error codes detected during run:
500 (Internal Server Error) - 60 times
[16:34:13] [INFO] fetched data logged to text files under '/home/reza/.local/share/sqlmap/output/127.0.0.1'
[*] ending @ 16:34:13 /2026-07-15/

```

Gambar 9. Hasil pemindaian SQLmap pada endpoint rentan: parameter terdeteksi rentan dan data berhasil diekstrak.

Secara keseluruhan, hasil pengujian menunjukkan bahwa kombinasi batasan integritas statis dan dinamis di dalam DBMS berhasil menggantikan atau memperkuat peran validasi di tingkat aplikasi. Kueri manipulasi ilegal yang berhasil melewati filter aplikasi (bypass) secara otomatis diblokir di tingkat basis data sebelum terjadi perubahan fisik pada penyimpanan data.

3.10. Analisis Kinerja dan Latensi Trigger

Penerapan trigger aktif di tingkat basis data menjamin integritas secara mutlak, namun memicu proses komputasi tambahan (overhead) berupa kueri internal untuk validasi relasional setiap modifikasi dieksekusi 13. Pada trg_validasi_max_sks, setiap operasi INSERT ke krs_detail memicu dua SELECT internal untuk memeriksa penguncian KRS dan akumulasi total_sks.

Dalam penelitian ini, pengukuran latensi dengan benchmark terkontrol belum dilakukan; pengujian difokuskan pada kebenaran fungsional, yaitu penolakan transaksi ilegal dengan SQLSTATE '45000' pada Tabel 2. Sebagai konteks, Andrijevic dkk. 13 menunjukkan bahwa komunikasi PHP–MySQL dengan prepared statements memiliki trade-off latensi tergantung kompleksitas kueri, sehingga penambahan validasi di DBMS diprediksi menghasilkan overhead pada orde milidetik. Untuk beban transaksi akademik normal, overhead tersebut diprediksi masih berada dalam batas toleransi.

Sebagai pengembangan lebih lanjut, disarankan pengukuran presisi menggunakan hrtime(true) atau microtime(true) dengan skenario DROP TRIGGER versus CREATE TRIGGER beberapa ratus iterasi, dilengkapi deviasi standar dan uji beban konkuren.

3.11. Audit Logging Bawaan MySQL DBMS

Sebagai lapisan pemantauan (monitoring) dalam arsitektur pertahanan berlapis, MySQL menyediakan mekanisme General Query Log yang dapat diaktifkan untuk mencatat seluruh aktivitas kueri secara otomatis. Berbeda dengan pendekatan tabel audit manual yang dibangun menggunakan trigger, mekanisme ini beroperasi langsung pada tingkat DBMS sehingga tidak menambah beban operasional pada tabel bisnis.

3.11.1. Struktur Tabel mysql.general_log

Struktur logis tabel mysql.general_log yang terbentuk setelah pengaturan log_output = TABLE terdiri atas enam kolom yang mencatat waktu kejadian, identitas pengguna, serta isi kueri. Rincian struktur tersebut ditampilkan pada Tabel 6.

Tabel 6. Struktur Tabel mysql.general_log untuk Audit Logging.

Kolom	Tipe	Keterangan
event_time	TIMESTAMP	Waktu kejadian kueri
user_host	TEXT	User dan host yang terhubung
thread_id	BIGINT	ID thread koneksi
server_id	INT	ID server MySQL
command_type	VARCHAR	Jenis perintah (Query, Execute, dll.)

argument

TEXT

Teks kueri SQL yang dijalankan

3.11.2. Contoh Kueri Monitoring

Setelah General Query Log diaktifkan, aktivitas kueri pada basis data SIAKAD dapat dipantau melalui kueri berikut. Agar pengujian tidak memicu ERROR 1046 (3D000) No database selected seperti yang terjadi pada percobaan awal, ditambahkan USE siakad_db; sebelum kueri COUNT(*) FROM mahasiswa/kelas. Untuk memastikan seluruh skenario uji tetap tercatat meskipun trigger menolak dengan ERROR 45000, pengujian CLI dijalankan dengan opsi --force serta log disimpan ke temp/gambar*.log.

USE			siakad_db;
SELECT	event_time,	user_host,	argument
FROM			mysql.general_log
WHERE	argument	LIKE	'%krs_detail%'
ORDER	BY	event_time	DESC
LIMIT 20;			

Queri di atas menampilkan 20 aktivitas kueri terakhir yang melibatkan tabel krs_detail, memungkinkan administrator untuk mendeteksi pola akses yang mencurigakan secara real-time. Format -G --data-urlencode digunakan pada curl untuk menghindari curl: (3) URL rejected: Malformed input akibat spasi pada payload 'OR '1'='1, serta diformat dengan jq . untuk jeda enter agar prompt tidak menempel pada JSON {"jumlah":0,"data":[]}

3.11.3. Bukti Implementasi

Screenshot hasil eksekusi kueri monitoring terhadap tabel mysql.general_log ditampilkan pada Gambar 13, yang menunjukkan bahwa seluruh aktivitas kueri pada basis data SIAKAD berhasil dicatat oleh mekanisme audit bawaan MySQL.



Gambar 10. Screenshot audit logging MySQL: Aktivitas kueri pada basis data SIAKAD yang tercatat otomatis di tabel mysql.general_log.

4. KESIMPULAN

Tujuan penelitian untuk merancang, mengimplementasikan, dan menguji skema basis data SIAKAD yang terstruktur optimal dengan mekanisme validasi internal telah tercapai dengan baik. Melalui penerapan struktur normalisasi bentuk 3NF yang dipadukan dengan batasan integritas kolom statis (CHECK, NOT NULL, UNIQUE, FOREIGN KEY), basis data terbukti mampu menolak transaksi data yang tidak sah secara mandiri jika terjadi kegagalan validasi di tingkat aplikasi.

Secara spesifik, mekanisme trigger berhasil diterapkan pada tiga proses bisnis akademik utama. Untuk penjadwalan, trigger trg_penjadwalan_bentrok mencegah dosen memiliki jadwal bentrok pada hari dan jam yang sama. Untuk pengisian KRS, trigger trg_validasi_max_sks dan trigger pendukung lainnya menegakkan batas maksimal 24 SKS dan status penguncian KRS secara real-time. Untuk penilaian, trigger trg_auto_konversi_nilai mengotomatiskan konversi nilai

Implementasi Normalisasi Dan Kontrol Keamanan Pada Basis Data Sistem Informasi Akademik Berbasis MySQL,
Reza Asriano Maulana, Halimil Fathi, Sukrina Herman

©2026 LPPM Politeknik Enjinering Indorama. Hak Cipta Dilindungi

Artikel ini merupakan artikel akses terbuka yang didistribusikan berdasarkan ketentuan lisensi CC BY.

(<https://creativecommons.org/licenses/by/4.0/>)

DOI: 10.61713/jrt.v6i2.377.

angka ke huruf berdasarkan skala IPB dan memvalidasi bahwa input nilai hanya diizinkan pada KRS yang sudah disetujui. Mitigasi serangan SQL Injection berhasil dioptimalkan di lapisan aplikasi melalui kueri terparameterisasi (prepared statements) berbasis PHP PDO [11]. Selain itu, mekanisme audit logging bawaan MySQL (General Query Log dengan output tabel) diaktifkan untuk memantau seluruh aktivitas kueri pada basis data SIAKAD secara otomatis, sehingga membentuk arsitektur pertahanan berlapis yang komprehensif.

DAFTAR PUSTAKA

- [1] Al Maliki M, Faisol A, Pranoto YA. Pengembangan Sistem Informasi Akademik Mahasiswa (SIAM) Berbasis Web di Lingkungan ITN Malang. JATI (Jurnal Mahasiswa Teknik Informatika). 2025; 9(6): 10819-10826. doi: <https://doi.org/10.36040/jati.v9i6.15415>
- [2] Fachri F, Fadlil A, Riadi I. Analisis Keamanan Webserver menggunakan Penetration Test. Jurnal Informatika. 2021; 8(2): 183-190. doi: <https://doi.org/10.31294/ji.v8i2.10854>
- [3] Nafisah N, Ujianti MH, Riyanto. Sistem Informasi Akademik Pengolahan Data Nilai Siswa Berbasis Website pada SD Negeri Mangunsaren 02. JATI (Jurnal Mahasiswa Teknik Informatika). 2025; 9(4): 6648-6654. doi: <https://doi.org/10.36040/jati.v9i4.14169>
- [4] Pratiwi SD, Nugroho IM, Jaelani I. Rancang Bangun SIAS Menggunakan Metode RAD pada SMK S Mutiara Bangsa Purwakarta Berbasis Mobile. JATI (Jurnal Mahasiswa Teknik Informatika). 2024; 8(6): 11080-11085. doi: <https://doi.org/10.36040/jati.v8i6.11214>
- [5] Di Juni P, Hasan MA, Sitompul YSP, Alexander J, Hidayat I, Bilbo RM. Audit Keamanan Basis Data Menggunakan SQL Server Audit untuk Deteksi Aktivitas Tidak Sah. JATI (Jurnal Mahasiswa Teknik Informatika). 2025; 9(2): 2896-2903. doi: <https://doi.org/10.36040/jati.v9i2.13198>
- [6] Dina DFM. Normalisasi Database Rancangan Sistem Penyewaan Buku Berbayar. Computing Insight: Journal of Computer Science. 2024; 4(1): 56-61. doi: https://doi.org/10.30651/comp_insight.v4i1.15814
- [7] Andarini RY, Hendradi P, Nugroho S. Meningkatkan Keamanan Terhadap SQL Injection Studi Kasus Sistem Kepegawaian BNN. Indonesian Journal of Business Intelligence (IJUBI). 2023; 6(1). doi: <https://doi.org/10.21927/ijubi.v6i1.3161>
- [8] Akbar FB, Sipayung AFP, Lisara AD, Syakira N, Raisa D. Analisis Keamanan Sistem Akademik Berbasis Web terhadap Serangan SQL Injection. JIKUM: Jurnal Ilmu Komputer. 2026; 2(1): 60-64. doi: <https://doi.org/10.62671/jikum.v2i1.161>
- [9] Kusuma G. Implementasi OWASP ZAP untuk Pengujian Keamanan Sistem Informasi Akademik. Jurnal Teknologi Informasi. 2022; 16(2): 178-186. doi: <https://doi.org/10.47111/jti.v16i2.3995>
- [10] Rizky AMN, Tahir M, Sapitri TA, Islam MF. Implementasi SQL Injection dalam Penetration Testing untuk Deteksi Celah Keamanan Web. JATI (Jurnal Mahasiswa Teknik Informatika). 2025; 9(4): 6964-6968. doi: <https://doi.org/10.36040/jati.v9i4.14350>
- [11] Choi J, Jung YA, Ko H. Comparative Analysis of SQL Injection Defense Mechanisms Based on Three Approaches: PDO, PVT, and ART. Applied Sciences. 2025; 15(23): 12351. doi: <https://doi.org/10.3390/app152312351>
- [12] Mardiyono A, Hermawan I, Wirawan C, Iswara RW, Ramadhan MB, Azzahirah S. Pengembangan Database Sistem Informasi Akademik Sekolah Alam Indonesia Cibinong. Jurnal Abdi Insani. 2025; 12(3): 1148-1160. doi: <https://doi.org/10.29303/abdiinsani.v12i3.2418>
- [13] Andrijević N, Lovreković Z, Salkić H, Šarčević Đ, Perišić J. Benchmarking PHP–MySQL Communication: A Comparative Study of MySQLi and PDO Under Varying Query Complexity. Electronics. 2025; 15(1): 21. doi: <https://doi.org/10.3390/electronics15010021>
- [14] Hamidi A, Hamraz AR, Rahmani K. Database Security Mechanisms in MySQL. Afghanistan Research Journal - Natural Science. 2024; 4(1). doi: <https://doi.org/10.70648/naturalscience.v4i1.66>
- [15] Hamraz AR, Hamidi A. Securing MySQL Databases: A Comprehensive Examination of Advanced Security Measures. Kabul Journal of Engineering and Technology (KJET). 2024; 3(1). doi: <https://doi.org/10.31841/kjet.2024.39>

Implementasi Normalisasi Dan Kontrol Keamanan Pada Basis Data Sistem Informasi Akademik Berbasis Mysql,
Reza Asriano Maulana, Halimil Fathi, Sukrina Herman

©2026 LPPM Politeknik Enjinereng Indorama. Hak Cipta Dilindungi

Artikel ini merupakan artikel akses terbuka yang didistribusikan berdasarkan ketentuan lisensi CC BY.

(<https://creativecommons.org/licenses/by/4.0/>)

DOI: 10.61713/jrt.v6i2.377.