SISTEM MANAJEMEN EVENT KAMPUS MENGGUNAKAN MODEL DATABASE TERSTRUKTUR

Rizky Tito Prasetyo, S.Si, M.T.I, Al-Ghifari Rahbani Ramadhan, Dzulhas Syahara Muthahari, Muhammad Bagas Ramadhan, Bagas Malik Ibrahim.

Program Studi S1 Sistem Informasi, Universitas Pembangunan Nasional "VETERAN" Jakarta

---

ABSTRAK

Perancangan sistem basis data merupakan tahap penting dalam pengembangan sistem informasi, khususnya untuk mendukung pengelolaan data yang terstruktur dan efisien. Penelitian ini bertujuan untuk merancang sistem basis data pada Sistem Informasi Manajemen Event Kampus, yang digunakan untuk mendukung kegiatan kemahasiswaan seperti seminar, workshop, dan pelatihan. Pendekatan yang digunakan dalam penelitian ini adalah **Database Life Cycle (DBLC)**, yang terdiri dari enam tahapan utama: studi awal, perancangan, implementasi, pengujian, pengoperasian, serta pemeliharaan.

Data kebutuhan sistem diperoleh melalui observasi terhadap dokumen pelaksanaan event kampus, seperti formulir pendaftaran dan laporan kegiatan, serta disimulasikan berdasarkan praktik umum penyelenggaraan event. Hasil dari penelitian ini adalah rancangan basis data dalam bentuk Entity Relationship Diagram (ERD), tabel relasional yang telah dinormalisasi, dan skema awal implementasi menggunakan MySQL. Rancangan ini diharapkan dapat menjadi dasar bagi pengembangan sistem informasi event kampus yang lebih efisien dan terintegrasi.

Kata kunci: Basis data, perancangan sistem, event kampus, DBLC, ERD.

ABSTRACT

Database design is a crucial phase in developing information systems, especially for supporting structured and efficient data management. This study aims to design a database system for a Campus Event Management Information System, intended to support student activities such as seminars, workshops, and trainings. The research adopts the ***Database Life Cycle (DBLC)*** *approach, which consists of six main stages: initial study, design, implementation, testing, operation, and maintenance.*

System requirements were identified through observation of event-related documents, such as registration forms and activity reports, and simulated based on common practices in campus event organization. The output of this study includes a database design in the form of an Entity Relationship Diagram (ERD), normalized relational tables, and an initial implementation schema using MySQL. This design is expected to serve as a foundation for developing a more efficient and integrated campus event information system.

Keywords: *Database, system design, campus event, DBLC, ERD.*

BAB I

PENDAHULUAN

Perguruan tinggi merupakan salah satu lingkungan yang aktif dalam menyelenggarakan berbagai kegiatan kemahasiswaan, seperti seminar, workshop, pelatihan, lomba, dan kegiatan sosial. Kegiatan-kegiatan ini tidak hanya bertujuan untuk pengembangan soft skill mahasiswa, tetapi juga menjadi bagian penting dari pencapaian indikator kinerja institusi. Namun, dalam pelaksanaannya, manajemen data event kampus sering kali dilakukan secara manual atau menggunakan sistem yang terpisah-pisah, sehingga menimbulkan berbagai kendala seperti duplikasi data, kesalahan pencatatan, dan kesulitan dalam pelaporan.

Kelemahan dalam pengelolaan informasi event, terutama terkait data peserta, panitia, jadwal, dan dokumentasi, dapat mengganggu efisiensi operasional serta akurasi informasi. Sebagai contoh, pengumpulan data peserta secara manual melalui formulir kertas atau Google Form tanpa sistem terintegrasi dapat menyebabkan kehilangan data atau kesalahan input. Oleh karena itu, diperlukan suatu sistem informasi manajemen event kampus yang terstruktur dan terintegrasi, yang diawali dengan perancangan basis data yang baik.

Basis data merupakan komponen inti dalam sistem informasi, karena berperan dalam menyimpan, mengelola, dan menyediakan akses terhadap data secara sistematis. Perancangan sistem basis data yang tepat dapat meningkatkan efisiensi dalam pengolahan data serta memudahkan proses pelaporan dan evaluasi kegiatan kampus. Penelitian ini bertujuan untuk merancang sistem basis data untuk sistem informasi manajemen event kampus, dengan pendekatan analisis kebutuhan data, perancangan Entity Relationship Diagram (ERD), dan normalisasi hingga bentuk normal ke-3.

Ruang lingkup penelitian ini difokuskan pada perancangan basis data yang mencakup pengelolaan data event, data peserta, data panitia, registrasi, serta kehadiran peserta. Penelitian tidak mencakup pengembangan antarmuka pengguna (UI) maupun pengkodean sistem secara menyeluruh. Hasil akhir dari penelitian ini diharapkan dapat menjadi fondasi yang solid untuk implementasi sistem informasi manajemen event kampus berbasis web maupun aplikasi mobile, serta dapat dikembangkan lebih lanjut sesuai kebutuhan institusi.

BAB II

TINJAUAN PUSTAKA

2.1 Sistem Informasi

Sistem informasi merupakan kombinasi dari teknologi informasi dan aktivitas manusia yang menggunakan teknologi tersebut untuk mendukung operasi dan manajemen. Menurut Jogiyanto (2005), sistem informasi adalah suatu sistem di dalam suatu organisasi yang mempertemukan kebutuhan pengolahan transaksi harian, mendukung operasi, bersifat manajerial, dan kegiatan strategis dari suatu organisasi serta menyediakan pihak luar dengan laporan-laporan yang diperlukan. Dalam konteks event kampus, sistem informasi digunakan untuk membantu proses pendaftaran peserta, pengelolaan panitia, dokumentasi kegiatan, dan pembuatan laporan.

2.2 Basis Data

Basis data (database) adalah kumpulan data yang saling berhubungan dan disimpan secara sistematis agar dapat diakses, dikelola, dan diperbarui dengan mudah. Menurut Kadir (2019), basis data bertujuan untuk menghindari redudansi data, menjaga integritas data, serta memberikan akses yang cepat dan aman terhadap data. Basis data menjadi fondasi utama dalam pembangunan sistem informasi, termasuk dalam sistem informasi manajemen event kampus

2.3 Database Management System (DBMS)

DBMS adalah perangkat lunak yang digunakan untuk mengelola basis data, termasuk menyimpan, mengubah, dan mengambil data. Beberapa contoh DBMS populer adalah MySQL, PostgreSQL, dan Microsoft SQL Server. DBMS memberikan antarmuka yang memungkinkan pengguna untuk berinteraksi dengan basis data menggunakan bahasa kueri seperti SQL (Structured Query Language). 2.4 Entity Relationship Diagram (ERD) ERD merupakan model perancangan basis data yang menggambarkan hubungan antar entitas dalam sistem. Menurut Connolly dan Begg (2015), ERD digunakan untuk membantu proses perancangan logis dari basis data, dengan mendefinisikan entitas, atribut, dan relasi antar entitas tersebut. ERD sangat berguna untuk memahami struktur data dan menjadi acuan dalam pembuatan skema relasional.

2.4 Entity Relationship Diagram (ERD)

ERD merupakan model perancangan basis data yang menggambarkan hubungan antar entitas dalam sistem. Menurut Connolly dan Begg (2015), ERD digunakan untuk membantu proses perancangan logis dari basis data, dengan mendefinisikan entitas, atribut, dan relasi antar entitas tersebut. ERD sangat berguna untuk memahami struktur data dan menjadi acuan dalam pembuatan skema relasional.

2.5 Normalisasi Data

Normalisasi proses pengorganisasian data dalam basis data untuk mengurangi duplikasi data dan meningkatkan integritas. Proses normalisasi terdiri dari beberapa bentuk normal (normal forms) seperti 1NF, 2NF, dan 3NF. Setiap bentuk normal memiliki kriteria tertentu yang harus dipenuhi. Menurut Sudarsono (2016), dengan menerapkan normalisasi, desain basis data

akan menjadi lebih efisien dan terstruktur, serta memudahkan proses pemeliharaan data. 2.6 Sistem Informasi Event Kampus Beberapa penelitian sebelumnya telah membahas tentang sistem informasi event kampus. Misalnya, pada penelitian oleh Pratama & Rahmawati (2020), dijelaskan bahwa digitalisasi sistem event kampus dapat meningkatkan efektivitas koordinasi antara panitia, peserta, dan pihak kampus. Namun, banyak dari sistem tersebut belum mengadopsi perancangan basis data yang optimal, sehingga menimbulkan kendala dalam pengelolaan data jangka panjang.

2.6 Sistem Informasi Event Kampus

Beberapa penelitian sebelumnya telah membahas tentang sistem informasi event kampus. Misalnya, pada penelitian oleh Pratama & Rahmawati (2020), dijelaskan bahwa digitalisasi sistem event kampus dapat meningkatkan efektivitas koordinasi antara panitia, peserta, dan pihak kampus. Namun, banyak dari sistem tersebut belum mengadopsi perancangan basis data yang optimal, sehingga menimbulkan kendala dalam pengelolaan data jangka panjang.

BAB III

METODOLOGI PENELITIAN

Penelitian ini menggunakan pendekatan Database Life Cycle (DBLC) untuk merancang sistem basis data pada Sistem Informasi Manajemen Event Kampus. Pendekatan ini mencakup seluruh siklus hidup basis data, mulai dari studi awal hingga tahap pemeliharaan dan pengembangan.

3.1 Database Initial Study (Studi Awal Database)

Pada tahap ini dilakukan analisis kebutuhan informasi untuk sistem berdasarkan dua metode utama:

- **A. Observasi Dokumen**

Peneliti melakukan pengamatan terhadap dokumen-dokumen yang umum digunakan dalam pelaksanaan event kampus, seperti:

  - Formulir pendaftaran peserta (online/offline)
  - Daftar kehadiran
  - Struktur organisasi panitia
  - Laporan kegiatan dan dokumentasi d
  - ari dokumen ini diidentifikasi entitas penting seperti: Event, Peserta, Panitia, Absensi, dan Sertifikat.

- **B. Simulasi Kebutuhan Fungsional**

Karena tidak dilakukan wawancara langsung dengan pihak penyelenggara event, maka peneliti menyusun simulasi alur sistem berdasarkan praktik umum manajemen event kampus. Asumsi kebutuhan disusun berdasarkan logika kegiatan kampus dan referensi dari sistem informasi sejenis yang telah tersedia secara daring atau dalam literatur terdahulu.

Asumsi-asumsi tersebut mencakup:

  - Panitia mengelola data event dan peserta
  - Peserta mendaftar secara online
  - Absensi dilakukan selama kegiatan
  - Sertifikat diberikan setelah kegiatan selesai

Metode ini dipilih untuk menyesuaikan dengan keterbatasan akses terhadap pengguna, namun tetap menjaga validitas kebutuhan sistem melalui dokumentasi yang relevan dan praktik umum kegiatan kampus.

3.2 Database Design (Perancangan Database)

Perancangan basis data dimulai dengan menganalisis kebutuhan informasi dari sistem informasi manajemen event kampus. Dari hasil analisis, diidentifikasi tujuh entitas utama: Event, Peserta, Panitia, Registrasi, Absensi, Sertifikat, dan Feedback. Setiap entitas menggambarkan komponen nyata yang terlibat dalam proses penyelenggaraan event kampus, seperti seminar, workshop, dan lomba. Perancangan dilakukan dalam tiga tahap, yaitu desain konseptual, desain logikal, dan desain fisik.

Pada tahap desain konseptual, dibangun model Entity Relationship Diagram (ERD) yang menggambarkan relasi antar entitas. Misalnya, entitas Peserta memiliki relasi

many-to-many dengan Event yang dimediasi oleh entitas Registrasi. Selanjutnya, Peserta juga terhubung ke Absensi, Sertifikat, dan Feedback, masing-masing untuk mencatat kehadiran, bukti partisipasi, dan umpan balik terhadap event.

Dalam desain logikal, struktur ERD dikonversi ke dalam bentuk skema relasional. Proses ini dilakukan dengan mempertimbangkan normalisasi hingga bentuk normal ketiga (3NF) untuk memastikan efisiensi penyimpanan data dan menghindari duplikasi informasi. Setiap entitas menjadi tabel dalam basis data, dilengkapi dengan atribut dan relasi yang sesuai. Misalnya, tabel Registrasi memiliki atribut `id_registrasi`, `id_event`, `id_peserta`, `waktu_registrasi`, dan `status_pendaftaran`.

Selanjutnya, pada desain fisik, tipe data dan panjang karakter didefinisikan secara eksplisit sesuai kebutuhan masing-masing atribut. Relasi antar tabel diimplementasikan dengan primary key dan foreign key. Contohnya, `id_event` dan `id_peserta` pada tabel Registrasi merujuk ke tabel Event dan Peserta. Desain ini memastikan integritas referensial data antar entitas di dalam sistem.

3.3 Implementation and Loading (Implementasi dan Pengisian Data)

Implementasi basis data dilakukan menggunakan sistem manajemen basis data relasional (RDBMS) MySQL. Setiap tabel dibuat berdasarkan skema yang telah ditentukan dalam tahap perancangan. Proses implementasi dilakukan menggunakan perintah SQL `CREATE TABLE`, dengan mendefinisikan atribut, tipe data, kunci primer, dan kunci asing untuk masing-masing tabel. Misalnya, tabel Event dibuat dengan atribut `id_event` sebagai kunci primer dan `status_event` sebagai enumerasi dengan nilai seperti 'aktif', 'selesai', dan 'dibatalkan'.

Setelah struktur tabel selesai diimplementasikan, dilakukan proses data loading menggunakan data simulasi. Data ini dimasukkan secara manual atau melalui skrip `INSERT` untuk mengisi tabel seperti Peserta, Event, dan Panitia. Data simulasi ini digunakan untuk melakukan pengujian awal sistem, seperti simulasi registrasi peserta, pencatatan kehadiran, pemberian sertifikat, dan pengisian feedback. Dengan pengisian data awal ini, sistem dapat diuji dari sisi aliran data serta keterhubungan antar tabel.

3.4 Testing and Evaluation (Pengujian dan Evaluasi)

Tahap pengujian dilakukan untuk memastikan bahwa struktur basis data telah berjalan sesuai fungsionalitas yang diharapkan. Pengujian dilakukan dengan menggunakan berbagai query SQL, baik untuk manipulasi data (`INSERT`, `UPDATE`, `DELETE`) maupun untuk pengambilan data (`SELECT`, `JOIN`). Sebagai contoh, dilakukan pengujian untuk mengambil daftar peserta yang sudah registrasi pada suatu event dan mencetak laporan kehadiran mereka. Relasi antar tabel diuji menggunakan perintah join untuk memastikan integritas referensial terjaga.

Evaluasi dilakukan terhadap kemampuan sistem dalam menangani skenario nyata, seperti mencegah pendaftaran ganda pada event yang sama, atau menolak absensi

dari peserta yang belum mendaftar. Selain itu, dilakukan validasi terhadap kunci asing untuk memastikan bahwa data yang tidak sesuai referensi tidak dapat dimasukkan. Hasil pengujian menunjukkan bahwa database bekerja stabil, integritas data terjaga, dan proses yang dirancang dapat dijalankan sesuai dengan logika bisnis yang ditentukan.

3.5 Operation (Pengoperasian Sistem)

Setelah database berhasil diimplementasikan dan diuji, tahap operasional dilakukan dalam bentuk simulasi alur kerja sistem informasi event kampus. Simulasi ini mencakup proses penambahan event oleh panitia, pendaftaran peserta melalui entitas registrasi, pencatatan kehadiran selama kegiatan berlangsung, serta pemberian sertifikat bagi peserta yang memenuhi syarat kehadiran. Selain itu, peserta juga dapat memberikan feedback melalui pengisian rating dan komentar terhadap event yang telah diikuti.

Pengoperasian ini dilakukan dengan bantuan antarmuka sederhana yang memanfaatkan perintah SQL dan alat bantu GUI seperti phpMyAdmin. Selama proses ini, sistem mampu memproses transaksi data secara konsisten, seperti menolak peserta yang mencoba absen pada event yang belum didaftari. Operasional sistem memperlihatkan bahwa database telah mampu memenuhi fungsionalitas dasar sistem manajemen event kampus secara menyeluruh.

3.6 Maintenance and Evolution (Pemeliharaan dan Pengembangan)

Tahap terakhir dalam siklus pengembangan basis data adalah pemeliharaan dan pengembangan lebih lanjut. Sistem dirancang secara modular, sehingga memudahkan penambahan fitur baru atau perubahan struktur data tanpa mengganggu sistem yang sudah berjalan. Sebagai contoh, jika di masa depan dibutuhkan fitur manajemen sponsor atau integrasi dengan sistem akademik kampus, pengembang cukup menambahkan entitas baru dan melakukan penyesuaian relasi tanpa harus merombak struktur yang ada.

Dokumentasi skema database, ERD, dan relasi antar tabel juga disiapkan untuk mempermudah proses pemeliharaan jangka panjang. Selain itu, sistem dapat dikembangkan lebih lanjut dengan menambahkan lapisan API untuk integrasi ke antarmuka web atau aplikasi mobile. Dengan pendekatan ini, sistem informasi manajemen event kampus menjadi fleksibel, dapat beradaptasi dengan kebutuhan pengguna, dan siap untuk dikembangkan ke versi yang lebih kompleks secara bertahap.

BAB IV

HASIL DAN PEMBAHASAN

4.1 Analisis Kebutuhan Tabel

4.1.1 Tabel Event

Kebutuhan:

- Sistem membutuhkan informasi mengenai kegiatan atau acara yang diselenggarakan oleh kampus, seperti seminar, workshop, lomba, dan sebagainya.

Fungsi Data:

- Menyimpan detail event agar dapat ditampilkan kepada peserta.
- Mengelola status event (aktif/selesai/dibatalkan) untuk pengendalian operasional.

Atribut:

- id_event (PK): identifikasi unik setiap event.
- nama_event, deskripsi, tanggal_mulai/selesai, lokasi: sebagai informasi utama event.
- jenis_event, status_event: mengklasifikasi dan menentukan status operasional event.

4.1.2 Tabel Peserta

Kebutuhan:

- Sistem perlu menyimpan informasi mahasiswa atau pihak eksternal yang ingin mendaftar dan mengikuti event.

Fungsi Data:

- Digunakan saat registrasi, absensi, pemberian sertifikat, dan evaluasi event.

Atribut:

- id_peserta (PK): identifikasi unik peserta.

- nama, nim, email, no_hp, institusi: data pribadi peserta untuk keperluan administratif dan komunikasi.

4.1.3 Tabel Panitia

Kebutuhan:

- Sistem membutuhkan data siapa saja yang terlibat sebagai penyelenggara dalam setiap event.

Fungsi Data:

- Digunakan untuk otorisasi pengelolaan data event, serta dokumentasi struktur kepanitiaan.

Atribut:

- id_panitia (PK): identifikasi unik panitia.
- nama, nim, email, jabatan: identitas dan peran dalam event.
- id_event (FK): relasi ke event yang dikelola.

4.1.4 Tabel Registrasi

Kebutuhan:

- Sistem perlu mencatat proses pendaftaran peserta ke event tertentu.

Fungsi Data:

- Digunakan untuk validasi peserta, seleksi kepesertaan, dan pemantauan status pendaftaran.

Atribut:

- id_registrasi (PK): identifikasi unik tiap pendaftaran.
- id_event (FK), id_peserta (FK): relasi ke event dan peserta terkait.
- waktu_registrasi: waktu pendaftaran dilakukan.
- status_pendaftaran: status validasi pendaftaran oleh panitia.

4.1.5 Tabel Absensi

Kebutuhan:

- Sistem perlu mencatat kehadiran peserta saat pelaksanaan event.

Fungsi Data:

- Digunakan untuk evaluasi kehadiran dan sebagai syarat penerbitan sertifikat.

Atribut:

- id_absen (PK): identifikasi unik absensi.
- id_event (FK), id_peserta (FK): relasi untuk mencatat siapa hadir di event apa.
- waktu_masuk/keluar, status: mencatat waktu kehadiran dan status (hadir/tidak).

4.1.6 Tabel Sertifikat

Kebutuhan:

- Sistem perlu menyimpan data sertifikat yang diberikan kepada peserta sebagai bukti keikutsertaan.

Fungsi Data:

- Digunakan untuk pencetakan atau pengunduhan sertifikat oleh peserta.

Atribut:

- id_sertifikat (PK): identifikasi unik sertifikat.
- id_event (FK), id_peserta (FK): relasi siapa mendapat sertifikat dari event mana.
- nomor_sertifikat, tanggal_terbit, file_sertifikat: detail sertifikat.

4.1.7 Tabel Feedback

Kebutuhan:

- Sistem ingin memperoleh umpan balik dari peserta terkait event yang diikuti.

Fungsi Data:

- Digunakan untuk evaluasi dan peningkatan kualitas event selanjutnya.

Atribut:

- id_feedback (PK): identifikasi unik setiap masukan.
- id_event (FK), id_peserta (FK): relasi peserta memberikan masukan terhadap event.

- rating, komentar, waktu_input: penilaian dan isi ulasan.

4.2 Desain Konseptual

Pada fase ini, model data dirancang secara konseptual berdasarkan hasil analisis kebutuhan sistem informasi manajemen event kampus. Perancangan dimulai dengan mengidentifikasi entitas utama yang diperlukan dalam sistem, seperti event, peserta, panitia, registrasi, absensi, sertifikat, dan feedback. Setiap entitas mewakili objek nyata yang terlibat dalam proses manajemen event di lingkungan kampus. Relasi antar entitas juga ditentukan, misalnya peserta dapat melakukan registrasi ke banyak event, dan satu event dapat memiliki banyak peserta. Desain konseptual ini divisualisasikan dalam bentuk Entity Relationship Diagram (ERD) untuk menggambarkan struktur data secara logis sebelum diterapkan ke dalam model fisik. Pendekatan ini memastikan bahwa kebutuhan data pengguna telah terakomodasi dengan baik dalam perancangan sistem.

4.2.1 Identifikasi Tipe Entitas

Pada tahap ini dilakukan identifikasi tipe entitas yang merepresentasikan objek-objek nyata dalam sistem informasi manajemen event kampus. Setiap entitas berperan penting dalam mendukung proses bisnis sistem, mulai dari perencanaan, pelaksanaan, hingga evaluasi sebuah event. Berikut ini adalah tipe-tipe entitas yang teridentifikasi:

1. **Event**
   Event merupakan entitas utama yang menggambarkan kegiatan atau acara yang diselenggarakan di lingkungan kampus, seperti seminar, pelatihan, lomba, atau talkshow. Event memiliki atribut seperti nama event, deskripsi, tanggal pelaksanaan, lokasi, jenis, dan status. Entitas ini menjadi pusat dari relasi dengan entitas lainnya seperti peserta, panitia, dan sertifikat.

2. **Peserta**
   Peserta adalah entitas yang mewakili individu (mahasiswa, dosen, atau umum) yang mendaftar dan mengikuti event. Entitas ini menyimpan informasi seperti nama, NIM, email, nomor HP, dan institusi asal. Peserta dapat terhubung dengan banyak event melalui entitas relasi seperti registrasi dan absensi.

3. **Panitia**
   Panitia mewakili pengguna yang memiliki hak akses untuk mengelola event tertentu. Entitas ini menyimpan data seperti nama panitia, email, jabatan, dan referensi ke event yang dikelola. Panitia dapat berasal dari organisasi mahasiswa, dosen pembimbing, atau pihak ketiga penyelenggara event.

4. **Registrasi**
   Registrasi merupakan entitas hubungan yang mencatat aktivitas pendaftaran peserta terhadap event tertentu. Tipe ini termasuk weak entity karena keberadaannya bergantung pada entitas Event dan Peserta. Atribut pentingnya

meliputi waktu registrasi dan status pendaftaran.

5. **Absensi**
   Absensi merupakan entitas yang mencatat kehadiran peserta pada saat event berlangsung. Sama seperti registrasi, absensi juga merupakan entitas relasional yang menghubungkan peserta dengan event. Absensi memiliki atribut seperti waktu masuk, waktu keluar, dan status kehadiran.

6. **Sertifikat**
   Sertifikat adalah entitas yang mewakili bukti partisipasi peserta dalam sebuah event. Sertifikat diberikan kepada peserta yang memenuhi syarat kehadiran dan validasi panitia. Atributnya mencakup nomor sertifikat, tanggal terbit, dan file sertifikat.

7. **Feedback**
   Feedback merupakan entitas opsional yang digunakan untuk mencatat ulasan atau evaluasi peserta terhadap event yang diikuti. Entitas ini berguna dalam peningkatan mutu pelaksanaan event di masa depan. Atributnya antara lain rating, komentar, dan waktu input.

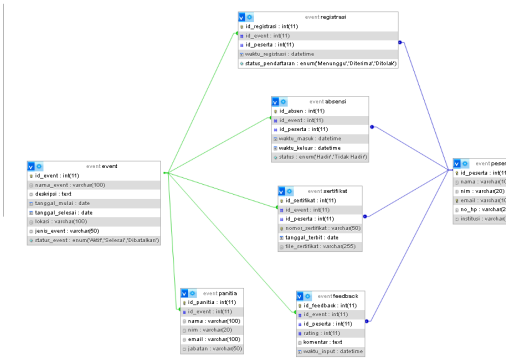
4.3 Design Logikal

Desain logikal merupakan tahapan perancangan struktur basis data yang bertujuan untuk memodelkan hubungan antar data berdasarkan kebutuhan informasi organisasi secara sistematis dan terstruktur. Pada tahap ini, seluruh entitas, atribut, dan relasi yang telah diidentifikasi pada desain konseptual digambarkan secara lebih rinci ke dalam bentuk Entity Relationship Diagram (ERD).

ERD ini menjadi dasar untuk implementasi basis data ke dalam Database Management System (DBMS), karena menggambarkan bagaimana data saling terhubung dan dapat diakses secara efisien. Desain logikal juga memperhatikan normalisasi agar terhindar dari redundansi data dan menjaga integritas data antar tabel.

Dalam konteks sistem informasi manajemen event kampus, desain logikal mencakup entitas utama seperti event, peserta, panitia, registrasi, absensi, sertifikat, dan feedback. Relasi antar entitas dirancang untuk mencerminkan aktivitas nyata seperti pendaftaran peserta ke event, kehadiran peserta, pemberian sertifikat, serta pengumpulan evaluasi dari peserta.

Dengan pendekatan ini, sistem dapat mengelola data event kampus secara terpadu dan mendukung fungsi operasional seperti monitoring, pencetakan sertifikat, hingga evaluasi kegiatan. Desain logikal yang telah dibangun ini kemudian menjadi acuan utama untuk tahapan desain fisik dan implementasi basis data yang sesungguhnya di DBMS seperti MySQL.



Gambar ERD(Entity Relasionship Diagram)

4.4 Tabel Relasional

Tabel relasional merupakan representasi data dalam bentuk dua dimensi yang terdiri dari baris dan kolom, di mana setiap tabel merepresentasikan satu entitas atau hubungan antar entitas dalam sistem. Pada tahap ini, data yang telah dianalisis dan dimodelkan secara logis kemudian diorganisasikan ke dalam struktur tabel yang jelas dan saling terhubung melalui kunci primer dan kunci asing. Dalam sistem informasi manajemen event kampus, tabel-tabel relasional digunakan untuk menyimpan data seperti informasi event, peserta, panitia, registrasi, absensi, sertifikat, dan umpan balik. Setiap tabel memiliki peran spesifik serta keterkaitan satu sama lain guna menjamin integritas dan keterpaduan data di dalam sistem. Desain tabel relasional ini menjadi dasar dalam membangun basis data yang efisien, terstruktur, dan mudah diakses sesuai dengan kebutuhan sistem yang telah ditentukan.

4.4.1 Tabel Event

Nama Kolom	Tipe Data	Kete nga
id_event	INT (PK)	ID unik event
nama_event	VARCHAR(100)	Nama acara
deskripsi	TEXT	Penjelasan acara
tanggal_mulai	DATE	Tanggal mulai
tanggal_selesai	DATE	Tanggal selesai
lokasi	VARCHAR(100)	Lokasi pelaksanaan
jenis_event	VARCHAR(50)	Seminar, Workshop, dll
status_event	ENUM('aktif','selesai','dibatalkan')	Status kegiatan

4.4.2 Tabel Peserta

Nama Kolom	Tipe Data	Keterangan
id_peserta	INT (PK)	ID peserta
nama	VARCHAR(100)	Nama lengkap
nim	VARCHAR(20)	Nomor Induk Mahasiswa
email	VARCHAR(100)	Email

no_hp	VARCHAR(20)	Nomor telepon
institusi	VARCHAR(100)	Asal universitas/instansi

4.4.3 Tabel Panitia

Nama Kolom	Tipe Data	Keterangan
id_panitia	INT (PK)	ID panitia
nama	VARCHAR(100)	Nama panitia
nim	VARCHAR(20)	NIM
email	VARCHAR(100)	Email
jabatan	VARCHAR(50)	Ketua, Sekretaris, dll
id_event	INT (FK)	Relasi ke Event

4.4.4 Tabel Registrasi

Nama Kolom	Tipe Data	Keterangan
id_registrasi	INT (PK)	ID registrasi
id_event	INT (FK)	Relasi ke Event
id_peserta	INT (FK)	Relasi ke Peserta
waktu_registrasi	DATETIME	Tanggal dan waktu daftar
status_pendaftaran	ENUM('terdaftar','lolos','ditolak')	Status daftar

4.4.5 Tabel Absensi

Nama Kolom	Tipe Data	Keterangan
id_absen	INT (PK)	ID absensi
id_event	INT (FK)	Relasi ke Event
id_peserta	INT (FK)	Relasi ke Peserta
waktu_masuk	DATETIME	Waktu kehadiran masuk
waktu_keluar	DATETIME	Waktu keluar (opsional)
status	ENUM('hadir','tidak hadir')	Status hadir

4.4.6 Tabel Sertifikat

Nama Kolom	Tipe Data	Keterangan
id_sertifikat	INT (PK)	ID unik sertifikat
id_event	INT (FK)	Relasi ke Event
id_peserta	INT (FK)	Relasi ke Peserta

nomor_sertifikat	VARCHAR(50)	Nomor unik sertifikat
tanggal_terbit	DATE	Tanggal diterbitkan
file_sertifikat	VARCHAR(255)	Nama file PDF (opsional)

4.4.7 Tabel Feedback

Nama Kolom	Tipe Data	Keterangan
id_feedback	INT (PK)	ID feedback
id_event	INT (FK)	Relasi ke Event
id_peserta	INT (FK)	Relasi ke Peserta
rating	INT	Skor 1–5
komentar	TEXT	Ulasan dari peserta
waktu_input	DATETIME	Waktu penilaian dilakukan

4.5 DDL (Data Definition Language)

Data Definition Language (DDL) digunakan untuk membuat dan mengubah struktur objek dalam basis data menggunakan perintah yang telah ditetapkan dan sintaksis tertentu . Objek basis data ini meliputi tabel , urutan, lokasi, alias , skema , dan indeks. DDL adalah bahasa standar dengan perintah untuk menentukan grup penyimpanan (stogroup), berbagai struktur dan objek dalam basis data. Pernyataan DDL membuat, mengubah, dan menghapus objek basis data, seperti tabel, indeks, dan stogroup. DDL juga digunakan dalam pengertian umum untuk merujuk ke bahasa apa pun yang menjelaskan data. DDL mencakup pernyataan-pernyataan Structured Query Language (SQL) untuk membuat dan menghapus basis data, alias, lokasi, indeks, tabel, dan urutan.

```
MariaDB [(none)]> CREATE DATABASE IF NOT EXISTS event_kampus;
Query OK, 1 row affected (0.003 sec)

MariaDB [(none)]> USE event_kampus;
Database changed
```

4.5.1 Trigger

Dalam DBMS (*Database Management System*), trigger merupakan kumpulan script yang berhubungan dengan table, view ataupun skema yang dijalankan secara otomatis ketika terdapat event yang dijalankan. Event tersebut meliputi operasi yang biasa dilakukan dalam mengolah database, seperti :

- DML (Data Manipulation Language) yang meliputi DELETE, INSERT atau UPDATE
- DDL (Data Definition Language) yang meliputi CREATE, ALTER atau DROP
- Operasi Database lainnya, seperti SERVERERROR, LOGON, LOGOFF, STARTUP atau SHUTDOWN)

```
MariaDB [event_kampus]> DELIMITER $$
MariaDB [event_kampus]>
MariaDB [event_kampus]> CREATE TRIGGER after_insert_registrasi
    -> AFTER INSERT ON registrasi
    -> FOR EACH ROW
    -> BEGIN
    -> IF NEW.status_pendaftaran = 'Diterima' THEN
    -> INSERT INTO absensi (id_event, id_peserta, waktu_masuk, waktu_keluar,
 status)
    -> VALUES (NEW.id_event, NEW.id_peserta, NULL, NULL, 'Hadir');
    -> END IF;
    -> END $$
Query OK, 0 rows affected (0.014 sec)

MariaDB [event_kampus]>
MariaDB [event_kampus]> DELIMITER ;
```

4.5.2 Function

Fungsi (function) adalah prosedur yang mengembalikan nilai dan dapat digunakan untuk melakukan perhitungan atau logika tertentu dalam database. Fungsi membantu menyederhanakan query dan menjaga konsistensi pengolahan data. Berikut adalah contoh implementasi fungsi yang diterapkan dalam sistem.

```
MariaDB [event_kampus]> DELIMITER $$
MariaDB [event_kampus]>
MariaDB [event_kampus]> CREATE FUNCTION jumlah_peserta_hadir(id_event_param
INT)
    -> RETURNS INT
    -> DETERMINISTIC
    -> BEGIN
    -> DECLARE total INT;
    -> SELECT COUNT(*) INTO total
    -> FROM absensi
    -> WHERE id_event = id_event_param AND status = 'Hadir';
    -> RETURN total;
    -> END $$
Query OK, 0 rows affected (0.029 sec)

MariaDB [event_kampus]>
MariaDB [event_kampus]> DELIMITER ;
```

4.5.3 View

VIEW atau disebut juga tabel virtual adalah sebuah tabel atau *logical* yang tidak berbentuk fisik dan terbuat dari himpunan hasil *query*. Jadi, VIEW ini tidak menyimpan data hasil *query*, melainkan hanya menyimpan *query* SELECT *statement* dan dibentuk menjadi tabel virtual.

Untuk hasil yang ditampilkan di VIEW merupakan data yang diperoleh dari tabel sumber dengan bentuk yang sama dengan tabel aslinya. Menambahkan SQL *statement* dan fungsi-fungsi SQL ke dalam VIEW.

Jadi, dengan membuat VIEW selain mendapatkan isi yang sama dengan tabel asli dan juga dapat menghemat ruang penyimpanan lebih banyak. Tujuan dari VIEW ini antara lain meningkatkan keamanan data, meningkatkan efisiensi *query*, dan menghemat penyimpanan memori.

```
MariaDB [event_kampus]> CREATE VIEW view_rekap_peserta AS
    -> SELECT
    -> p.nama AS nama_peserta,
    -> e.nama_event,
    -> r.status_pendaftaran,
    -> a.status AS status_kehadiran
    -> FROM peserta p
    -> JOIN registrasi r ON p.id_peserta = r.id_peserta
    -> JOIN event e ON r.id_event = e.id_event
    -> LEFT JOIN absensi a ON a.id_event = e.id_event AND a.id_peserta = p.i
d_peserta;
Query OK, 0 rows affected (0.004 sec)
```

4.5.4 Store Prosedure

Stored procedure adalah blok kode yang terdiri dari satu atau lebih pernyataan SQL yang disimpan dalam *database*. Pengguna dapat memanggil stored procedure ini untuk melakukan operasi tertentu tanpa harus menulis ulang kode SQL setiap kali diperlukan. Ini meningkatkan efisiensi dan konsistensi saat berinteraksi dengan *database*.

Stored procedure dapat menerima parameter, sehingga fungsinya dapat disesuaikan dengan kebutuhan. Misalnya, jika ingin mengambil data berdasarkan kriteria tertentu,

```
MariaDB [event_kampus]> DELIMITER $$
MariaDB [event_kampus]>
MariaDB [event_kampus]> CREATE PROCEDURE sp_detail_peserta_event(IN eventID
INT)
    -> BEGIN
    -> SELECT
    -> p.nama,
    -> r.status_pendaftaran,
    -> a.status AS status_kehadiran
    -> FROM peserta p
    -> JOIN registrasi r ON p.id_peserta = r.id_peserta
    -> LEFT JOIN absensi a ON a.id_peserta = p.id_peserta AND a.id_event = r
.id_event
    -> WHERE r.id_event = eventID;
    -> END $$
Query OK, 0 rows affected (0.012 sec)

MariaDB [event_kampus]>
MariaDB [event_kampus]> DELIMITER ;
```

4.6 Studi Implementasi

4.6.1 Insert data table

```
MariaDB [event]> select * from peserta;
+-------------+---------------+------------+-----------------+-------------+----------+
| id_peserta | nama           | nim        | email           | no_hp       | institusi |
+-------------+---------------+------------+-----------------+-------------+----------+
|          1 | Bagas Ramadhan | 2022101001 | bagas@upnvj.ac.id | 081234567801 | UPNVJ    |
|          2 | Nadia Aulia    | 2022101002 | nadia@upnvj.ac.id | 081234567802 | UPNVJ    |
|          3 | Rizky Maulana  | 2022101003 | rizky@upnvj.ac.id | 081234567803 | UPNVJ    |
|          4 | Salsabila Putri | 2022101004 | salsa@upnvj.ac.id | 081234567804 | UPNVJ    |
|          5 | Fajar Nugroho  | 2022101005 | fajar@upnvj.ac.id | 081234567805 | UPNVJ    |
|          6 | Dewi Anggraini | 2022101006 | dewi@upnvj.ac.id  | 081234567806 | UPNVJ    |
|          7 | Andi Pratama   | 2022101007 | andi@upnvj.ac.id  | 081234567807 | UPNVJ    |
|          8 | Laras Ayu      | 2022101008 | laras@upnvj.ac.id | 081234567808 | UPNVJ    |
|          9 | Ilham Ramadhan | 2022101009 | ilham@upnvj.ac.id | 081234567809 | UPNVJ    |
|         10 | Tiara Safira   | 2022101010 | tiara@upnvj.ac.id | 081234567810 | UPNVJ    |
+-------------+---------------+------------+-----------------+-------------+----------+
10 rows in set (0.023 sec)
```

Setelah struktur tabel berhasil dirancang dan diimplementasikan, langkah berikutnya adalah mengisi tabel dengan data yang diperlukan. Proses penyisipan data ini dilakukan menggunakan perintah SQL INSERT INTO, yang memungkinkan penambahan data secara manual maupun otomatis ke dalam tabel sesuai dengan format dan tipe data yang telah ditentukan pada desain fisik.

Penyisipan data bertujuan untuk menyediakan data awal atau contoh uji coba yang nantinya dapat digunakan dalam proses pengujian sistem maupun pembuatan laporan. Data yang dimasukkan mencakup informasi dasar seperti peserta, panitia, event, serta relasi antar entitas yang terlibat dalam sistem.

Contoh implementasi perintah penyisipan data ke dalam tabel dapat dilihat pada gambar berikut, yang menunjukkan data dari tabel peserta hasil eksekusi perintah SELECT setelah proses INSERT berhasil dilakukan.

4.6.2 Uji coba triggers

```
MariaDB [event_kampus]> DELIMITER $$
MariaDB [event_kampus]>
MariaDB [event_kampus]> CREATE TRIGGER after_insert_registrasi
    -> AFTER INSERT ON registrasi
    -> FOR EACH ROW
    -> BEGIN
    -> IF NEW.status_pendaftaran = 'Diterima' THEN
    -> INSERT INTO absensi (id_event, id_peserta, waktu_masuk, waktu_keluar, status)
    -> VALUES (NEW.id_event, NEW.id_peserta, NULL, NULL, 'Hadir');
    -> END IF;
    -> END $$
Query OK, 0 rows affected (0.014 sec)

MariaDB [event_kampus]>
MariaDB [event_kampus]> DELIMITER ;
MariaDB [event_kampus]> INSERT INTO registrasi (id_event, id_peserta, status_pendaftaran)
    -> VALUES (2, 6, 'Diterima');
Query OK, 1 row affected (0.005 sec)

MariaDB [event_kampus]> SELECT * FROM absensi WHERE id_event = 2 AND id_peserta = 6;
+-------------+----------+------------+-------------+--------------+--------+
| id_absensi | id_event | id_peserta | waktu_masuk | waktu_keluar | status |
+-------------+----------+------------+-------------+--------------+--------+
|          6 |        2 |          6 | NULL        | NULL         | Hadir  |
+-------------+----------+------------+-------------+--------------+--------+
1 row in set (0.001 sec)
```

Dalam sistem manajemen event kampus, efisiensi proses operasional dapat ditingkatkan melalui otomatisasi pada level basis data. Salah satu metode yang digunakan adalah penerapan trigger (pemicu), yaitu prosedur yang secara otomatis dijalankan sebagai respons terhadap perubahan pada tabel tertentu. Pada studi kasus ini, trigger digunakan untuk secara otomatis menambahkan data ke dalam tabel absensi apabila seorang peserta berhasil terdaftar (status pendaftaran = 'Diterima') pada suatu event.

Trigger bernama after_insert_registrasi dibuat dengan perintah:

```
CREATE TRIGGER after_insert_registrasi
 AFTER INSERT ON registrasi
 FOR EACH ROW
 BEGIN
   IF NEW.status_pendaftaran = 'Diterima' THEN
 INSERT INTO absensi (id_event, id_peserta, waktu_masuk, waktu_keluar, status)
  VALUES (NEW.id_event, NEW.id_peserta, NULL, NULL, 'Hadir');
 END IF;
 END
```

Langkah pertama adalah mengubah delimiter menjadi $$ untuk membedakan batas akhir dari trigger. Trigger tersebut diatur untuk dijalankan setelah (AFTER) sebuah data dimasukkan ke dalam tabel registrasi. Logika trigger memastikan bahwa jika status_pendaftaran bernilai 'Diterima', maka data baru akan disisipkan ke dalam tabel absensi secara otomatis, dengan nilai id_event dan id_peserta yang sama, serta status kehadiran langsung tercatat sebagai 'Hadir'. Nilai waktu_masuk dan waktu_keluar dibiarkan NULL dan dapat diperbarui nanti sesuai kedatangan peserta secara aktual.

Sebagai contoh implementasi, dilakukan penyisipan data ke tabel registrasi dengan query berikut:

INSERT INTO registrasi (id_event, id_peserta, status_pendaftaran)
 VALUES (2, 6, 'Diterima');

Secara otomatis, data baru pada tabel absensi juga akan ditambahkan oleh trigger tersebut. Hasil verifikasi menggunakan perintah:

SELECT * FROM absensi WHERE id_event = 2 AND id_peserta = 6;

menunjukkan bahwa peserta dengan ID 6 telah dicatat sebagai hadir pada event ID 2 tanpa perlu penyisipan manual ke dalam tabel absensi. Hal ini membuktikan bahwa trigger bekerja sesuai dengan logika yang dirancang. Penerapan trigger ini sangat berguna dalam menjaga konsistensi dan mempercepat proses bisnis, sekaligus mengurangi risiko kesalahan manusia (human error) dalam entri data yang bersifat rutin dan berulang.

4.6.3 Uji coba View

```
MariaDB [event_kampus]> CREATE VIEW view_rekap_peserta AS
    -> SELECT
    -> p.nama AS nama_peserta,
    -> e.nama_event,
    -> r.status_pendaftaran,
    -> a.status AS status_kehadiran
    -> FROM peserta p
    -> JOIN registrasi r ON p.id_peserta = r.id_peserta
    -> JOIN event e ON r.id_event = e.id_event
    -> LEFT JOIN absensi a ON a.id_event = e.id_event AND a.id_peserta = p.i
d_peserta;
Query OK, 0 rows affected (0.004 sec)
```

Perintah SQL pada gambar digunakan untuk membuat sebuah view bernama view_rekap_peserta di database event_kampus. View ini berfungsi untuk menampilkan data gabungan dari beberapa tabel, yaitu peserta, registrasi, event, dan absensi. Data yang ditampilkan mencakup nama peserta, nama event, status pendaftaran, dan status kehadiran.

Tabel peserta digabung dengan registrasi berdasarkan id_peserta, lalu digabung lagi dengan event berdasarkan id_event. Selanjutnya, digunakan LEFT JOIN pada tabel absensi untuk mengambil data kehadiran jika tersedia, namun tetap menampilkan peserta meskipun belum tercatat hadir (hasilnya akan bernilai NULL).

View ini mempermudah pengguna untuk melihat rekapitulasi peserta dalam suatu event tanpa harus menulis query yang panjang setiap kali.

4.6.4 Uji coba Function

```
MariaDB [event_kampus]> DELIMITER $$
MariaDB [event_kampus]>
MariaDB [event_kampus]> CREATE FUNCTION jumlah_peserta_hadir(id_event_param INT)
    -> RETURNS INT
    -> DETERMINISTIC
    -> BEGIN
    -> DECLARE total INT;
    -> SELECT COUNT(*) INTO total
    -> FROM absensi
    -> WHERE id_event = id_event_param AND status = 'Hadir';
    -> RETURN total;
    -> END $$
Query OK, 0 rows affected (0.029 sec)

MariaDB [event_kampus]>
MariaDB [event_kampus]> DELIMITER ;
MariaDB [event_kampus]> SELECT jumlah_peserta_hadir(1) AS total_hadir_event_1;
+---------------------+
| total_hadir_event_1 |
+---------------------+
|                   5 |
+---------------------+
1 row in set (0.001 sec)
```

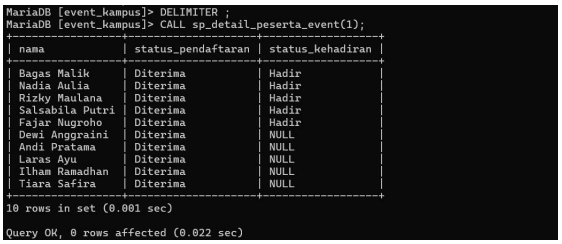
Dalam pengelolaan sistem informasi manajemen event kampus, terkadang dibutuhkan perhitungan otomatis terhadap jumlah peserta yang hadir pada suatu event tertentu. Untuk mendukung kebutuhan tersebut, sistem dapat dilengkapi dengan fungsi buatan (user-defined function) di dalam basis data. Fungsi ini dibuat menggunakan perintah CREATE FUNCTION, yang memungkinkan pengguna mendefinisikan fungsi tersendiri berdasarkan logika tertentu. Pada studi kasus ini, fungsi jumlah_peserta_hadir dibuat untuk menghitung total peserta yang berstatus 'Hadir' pada event tertentu berdasarkan ID event sebagai parameter masukannya. Langkah pertama dimulai dengan mengganti delimiter standar menjadi simbol khusus (misalnya $$) agar MariaDB dapat membedakan akhir dari pernyataan SQL kompleks. Kemudian, fungsi dideklarasikan dengan parameter id_event_param bertipe INT dan mengembalikan nilai berupa jumlah (INT). Di dalam blok fungsi, digunakan SELECT COUNT(*) untuk menghitung data dari tabel absensi yang memenuhi kondisi tertentu, yaitu id_event sesuai dengan parameter masukan

dan status kehadiran peserta adalah 'Hadir'. Setelah fungsi berhasil dibuat, sistem dapat langsung menggunakannya dengan perintah SELECT, seperti pada contoh:

SELECT jumlah_peserta_hadir(1) AS total_hadir_event_1;

Perintah ini akan menghasilkan jumlah peserta yang hadir pada event dengan ID 1, dan hasilnya dapat ditampilkan atau dipakai dalam laporan lainnya. Implementasi fungsi ini mendukung efisiensi sistem karena perhitungan dilakukan langsung di level database tanpa perlu proses tambahan dari sisi aplikasi.

4.6.4 Uji coba Prosedure



```
MariaDB [event_kampus]> DELIMITER ;
MariaDB [event_kampus]> CALL sp_detail_peserta_event(1);
+----------------+--------------------+------------------+
| nama           | status_pendaftaran | status_kehadiran |
+----------------+--------------------+------------------+
| Bagas Malik    | Diterima           | Hadir            |
| Nadia Aulia    | Diterima           | Hadir            |
| Rizky Maulana  | Diterima           | Hadir            |
| Salsabila Putri| Diterima           | Hadir            |
| Fajar Nugroho  | Diterima           | Hadir            |
| Dewi Anggraini | Diterima           | NULL             |
| Andi Pratama   | Diterima           | NULL             |
| Laras Ayu      | Diterima           | NULL             |
| Ilham Ramadhan | Diterima           | NULL             |
| Tiara Safira   | Diterima           | NULL             |
+----------------+--------------------+------------------+
10 rows in set (0.001 sec)

Query OK, 0 rows affected (0.022 sec)
```

Pada gambar tersebut ditunjukkan sebuah prosedur tersimpan (stored procedure) dalam database MariaDB dengan nama sp_detail_peserta_event. Prosedur ini dibuat untuk menampilkan data peserta berdasarkan ID suatu event tertentu. Prosedur ini menerima satu parameter masukan berupa eventID yang bertipe INT. Di dalam prosedur, dilakukan proses pengambilan data (SELECT) yang mencakup nama peserta, status pendaftaran, dan status kehadiran. Data ini diambil dari tiga tabel, yaitu peserta, registrasi, dan absensi. Tabel peserta dihubungkan dengan tabel registrasi melalui kolom id_peserta, sedangkan tabel absensi dihubungkan secara LEFT JOIN dengan peserta berdasarkan id_peserta dan id_event. Penggunaan LEFT JOIN ini bertujuan agar peserta tetap ditampilkan meskipun belum memiliki data absensi, yang ditandai dengan nilai NULL pada kolom status_kehadiran. Data yang ditampilkan hanya yang memiliki id_event sesuai dengan parameter yang diberikan. Setelah prosedur dibuat, dilakukan pemanggilan dengan CALL sp_detail_peserta_event(1);, yang berarti data peserta ditampilkan untuk event dengan ID 1. Hasilnya menampilkan daftar 10 peserta dengan status pendaftaran “Diterima”. Dari 10 peserta tersebut, 5 di antaranya telah hadir dan memiliki status kehadiran “Hadir”, sementara 5 lainnya belum tercatat hadir sehingga status kehadirannya bernilai NULL.

BAB V

KESIMPULAN

Penelitian ini menghasilkan sebuah rancangan sistem basis data yang terstruktur dan efisien untuk mendukung pengelolaan data dalam Sistem Informasi Manajemen Event Kampus. Berdasarkan pendekatan Database Life Cycle (DBLC), telah dilakukan tahapan studi awal, perancangan, implementasi, pengujian, pengoperasian, dan pemeliharaan database yang menyeluruh.

Rancangan ini berhasil mengidentifikasi tujuh entitas utama, yaitu: Event, Peserta, Panitia, Registrasi, Absensi, Sertifikat, dan Feedback. Masing-masing entitas telah dirancang dalam bentuk Entity Relationship Diagram (ERD), kemudian dikonversi ke dalam skema relasional dan diimplementasikan pada MySQL. Sistem mampu menangani berbagai kebutuhan operasional event kampus seperti pendaftaran peserta, kehadiran, pemberian sertifikat, dan evaluasi kegiatan.

Dari pengujian yang dilakukan, sistem terbukti dapat menjaga integritas data, mencegah redundansi, serta mengotomatisasi beberapa proses melalui fitur trigger, function, view, dan stored procedure. Sistem ini efektif mengatasi permasalahan lama seperti pencatatan manual, kehilangan data, dan duplikasi informasi.

Saran:

Untuk pengembangan lebih lanjut, sistem ini disarankan:

- Diintegrasikan dengan sistem akademik kampus guna menyelaraskan data mahasiswa secara otomatis.
- Dikembangkan antarmuka pengguna (user interface) berbasis web atau mobile untuk meningkatkan kemudahan akses.
- Dilengkapi dengan fitur notifikasi (email atau WhatsApp API) agar peserta dan panitia mendapatkan informasi event secara real-time.
- Mendukung manajemen sponsor dan pendanaan acara sebagai tambahan fitur pengembangan sistem di masa dep

DAFTAR PUSTAKA

Kadir, A. (2019). *Dasar Pemrograman dan Basis Data*. Yogyakarta: Andi Offset.

Sutabri, T. (2012). *Analisis Sistem Informasi*. Yogyakarta: Andi.

Connolly, T. & Begg, C. (2015). *Database Systems: A Practical Approach to Design, Implementation, and Management*. Pearson Education.

Connolly, T., & Begg, C. (2015). *Database Systems: A Practical Approach to Design, Implementation, and Management* (6th ed.). Pearson Education.

Jogiyanto, H. M. (2005). *Analisis dan Desain Sistem Informasi: Pendekatan Terstruktur Teori dan Praktik Aplikasi Bisnis*. Andi.

Kadir, A. (2019). *Dasar Pemrograman dan Basis Data*. Yogyakarta: Andi Offset.

Pratama, D. A., & Rahmawati, S. (2020). Rancang Bangun Sistem Informasi Manajemen Event Kampus Berbasis Web (Studi Kasus: Universitas XYZ). *Jurnal Teknologi dan Sistem Informasi*, 8(1), 45–52. https://doi.org/10.1234/jtsi.v8i1.1234

Sudarsono, H. (2016). *Konsep Dasar Basis Data*. Jakarta: Penerbit Informatika.

Sutabri, T. (2012). *Analisis Sistem Informasi*. Yogyakarta: Andi.

LAMPIRAN

1. DDL

```
MariaDB [(none)]> CREATE DATABASE IF NOT EXISTS event_kampus;
Query OK, 1 row affected (0.003 sec)

MariaDB [(none)]> USE event_kampus;
Database changed
```

Gambar 4.5

2. Trigger

```
MariaDB [event_kampus]> DELIMITER $$
MariaDB [event_kampus]>
MariaDB [event_kampus]> CREATE TRIGGER after_insert_registrasi
    -> AFTER INSERT ON registrasi
    -> FOR EACH ROW
    -> BEGIN
    -> IF NEW.status_pendaftaran = 'Diterima' THEN
    -> INSERT INTO absensi (id_event, id_peserta, waktu_masuk, waktu_keluar,
 status)
    -> VALUES (NEW.id_event, NEW.id_peserta, NULL, NULL, 'Hadir');
    -> END IF;
    -> END $$
Query OK, 0 rows affected (0.014 sec)

MariaDB [event_kampus]>
MariaDB [event_kampus]> DELIMITER ;
```

Gambar 4.5.1

3. Function

```
MariaDB [event_kampus]> DELIMITER $$
MariaDB [event_kampus]>
MariaDB [event_kampus]> CREATE FUNCTION jumlah_peserta_hadir(id_event_param
INT)
    -> RETURNS INT
    -> DETERMINISTIC
    -> BEGIN
    -> DECLARE total INT;
    -> SELECT COUNT(*) INTO total
    -> FROM absensi
    -> WHERE id_event = id_event_param AND status = 'Hadir';
    -> RETURN total;
    -> END $$
Query OK, 0 rows affected (0.029 sec)

MariaDB [event_kampus]>
MariaDB [event_kampus]> DELIMITER ;
```

Gambar 4.5.2

4. View

```
MariaDB [event_kampus]> CREATE VIEW view_rekap_peserta AS
    -> SELECT
    -> p.nama AS nama_peserta,
    -> e.nama_event,
    -> r.status_pendaftaran,
    -> a.status AS status_kehadiran
    -> FROM peserta p
    -> JOIN registrasi r ON p.id_peserta = r.id_peserta
    -> JOIN event e ON r.id_event = e.id_event
    -> LEFT JOIN absensi a ON a.id_event = e.id_event AND a.id_peserta = p.i
d_peserta;
Query OK, 0 rows affected (0.004 sec)
```

Gambar 4.5.3

5. Store Procedure

```
MariaDB [event_kampus]> DELIMITER $$
MariaDB [event_kampus]>
MariaDB [event_kampus]> CREATE PROCEDURE sp_detail_peserta_event(IN eventID
INT)
    -> BEGIN
    -> SELECT
    -> p.nama,
    -> r.status_pendaftaran,
    -> a.status AS status_kehadiran
    -> FROM peserta p
    -> JOIN registrasi r ON p.id_peserta = r.id_peserta
    -> LEFT JOIN absensi a ON a.id_peserta = p.id_peserta AND a.id_event = r
.id_event
    -> WHERE r.id_event = eventID;
    -> END $$
Query OK, 0 rows affected (0.012 sec)

MariaDB [event_kampus]>
MariaDB [event_kampus]> DELIMITER ;
```

Gambar 4.5.4

6. Insert Data Tabel

```
MariaDB [event]> select * from peserta;
+-----------+---------------+------------+-----------------+-------------+-----------+
| id_peserta | nama          | nim        | email           | no_hp       | institusi |
+-----------+---------------+------------+-----------------+-------------+-----------+
|          1 | Bagas Ramadhan  | 2022101001 | bagas@upnvj.ac.id | 081234567801 | UPNVJ     |
|          2 | Nadia Aulia     | 2022101002 | nadia@upnvj.ac.id | 081234567802 | UPNVJ     |
|          3 | Rizky Maulana   | 2022101003 | rizky@upnvj.ac.id | 081234567803 | UPNVJ     |
|          4 | Salsabila Putri | 2022101004 | salsa@upnvj.ac.id | 081234567804 | UPNVJ     |
|          5 | Fajar Nugroho   | 2022101005 | fajar@upnvj.ac.id | 081234567805 | UPNVJ     |
|          6 | Dewi Anggraini  | 2022101006 | dewi@upnvj.ac.id  | 081234567806 | UPNVJ     |
|          7 | Andi Pratama    | 2022101007 | andi@upnvj.ac.id  | 081234567807 | UPNVJ     |
|          8 | Laras Ayu       | 2022101008 | laras@upnvj.ac.id | 081234567808 | UPNVJ     |
|          9 | Ilham Ramadhan  | 2022101009 | ilham@upnvj.ac.id | 081234567809 | UPNVJ     |
|         10 | Tiara Safira    | 2022101010 | tiara@upnvj.ac.id | 081234567810 | UPNVJ     |
+-----------+---------------+------------+-----------------+-------------+-----------+
10 rows in set (0.023 sec)
```

Gambar 4.6.1

7. Uji Coba Trigger

```
MariaDB [event_kampus]> DELIMITER $$
MariaDB [event_kampus]>
MariaDB [event_kampus]> CREATE TRIGGER after_insert_registrasi
    -> AFTER INSERT ON registrasi
    -> FOR EACH ROW
    -> BEGIN
    -> IF NEW.status_pendaftaran = 'Diterima' THEN
    -> INSERT INTO absensi (id_event, id_peserta, waktu_masuk, waktu_keluar, status)
    -> VALUES (NEW.id_event, NEW.id_peserta, NULL, NULL, 'Hadir');
    -> END IF;
    -> END $$
Query OK, 0 rows affected (0.014 sec)

MariaDB [event_kampus]>
MariaDB [event_kampus]> DELIMITER ;
MariaDB [event_kampus]> INSERT INTO registrasi (id_event, id_peserta, status_pendaftaran)
    -> VALUES (2, 6, 'Diterima');
Query OK, 1 row affected (0.005 sec)

MariaDB [event_kampus]> SELECT * FROM absensi WHERE id_event = 2 AND id_peserta = 6;
+------------+----------+------------+-------------+--------------+--------+
| id_absensi | id_event | id_peserta | waktu_masuk | waktu_keluar | status |
+------------+----------+------------+-------------+--------------+--------+
|          6 |        2 |          6 | NULL        | NULL         | Hadir  |
+------------+----------+------------+-------------+--------------+--------+
1 row in set (0.001 sec)
```

Gambar 4.6.2

8. Uji Coba View

```
MariaDB [event_kampus]> CREATE VIEW view_rekap_peserta AS
    -> SELECT
    -> p.nama AS nama_peserta,
    -> e.nama_event,
    -> r.status_pendaftaran,
    -> a.status AS status_kehadiran
    -> FROM peserta p
    -> JOIN registrasi r ON p.id_peserta = r.id_peserta
    -> JOIN event e ON r.id_event = e.id_event
    -> LEFT JOIN absensi a ON a.id_event = e.id_event AND a.id_peserta = p.i
d_peserta;
Query OK, 0 rows affected (0.004 sec)
```

Gambar 4.6.3

9. Uji Coba Function

```
MariaDB [event_kampus]> DELIMITER $$
MariaDB [event_kampus]>
MariaDB [event_kampus]> CREATE FUNCTION jumlah_peserta_hadir(id_event_param INT)
    -> RETURNS INT
    -> DETERMINISTIC
    -> BEGIN
    -> DECLARE total INT;
    -> SELECT COUNT(*) INTO total
    -> FROM absensi
    -> WHERE id_event = id_event_param AND status = 'Hadir';
    -> RETURN total;
    -> END $$
Query OK, 0 rows affected (0.029 sec)

MariaDB [event_kampus]>
MariaDB [event_kampus]> DELIMITER ;
MariaDB [event_kampus]> SELECT jumlah_peserta_hadir(1) AS total_hadir_event_1;
+----------------------+
| total_hadir_event_1 |
+----------------------+
|                    5 |
+----------------------+
1 row in set (0.001 sec)
```

Gambar 4.6.4

10. Uji Coba Procedure

```
MariaDB [event_kampus]> DELIMITER ;
MariaDB [event_kampus]> CALL sp_detail_peserta_event(1);
+----------------+--------------------+------------------+
| nama           | status_pendaftaran | status_kehadiran |
+----------------+--------------------+------------------+
| Bagas Malik    | Diterima           | Hadir            |
| Nadia Aulia    | Diterima           | Hadir            |
| Rizky Maulana  | Diterima           | Hadir            |
| Salsabila Putri | Diterima           | Hadir            |
| Fajar Nugroho  | Diterima           | Hadir            |
| Dewi Anggraini | Diterima           | NULL             |
| Andi Pratama   | Diterima           | NULL             |
| Laras Ayu      | Diterima           | NULL             |
| Ilham Ramadhan | Diterima           | NULL             |
| Tiara Safira   | Diterima           | NULL             |
+----------------+--------------------+------------------+
10 rows in set (0.001 sec)

Query OK, 0 rows affected (0.022 sec)
```

Gambar 4.6.5