# BAB II
# KAJIAN LITERATUR

## 2.1. Konsep Manajemen Jadwal

Manajemen jadwal, dalam konteks organisasi atau bisnis, melibatkan proses perencanaan, pengorganisasian, dan pelaksanaan waktu dan sumber daya untuk mencapai tujuan yang ditetapkan. Konsep manajemen jadwal ditekankan sebagai bagian integral dari manajemen operasional yang efektif [3].

Sebuah penelitian yang diselenggarakan dalam jurnal [4] mengembangkan sebuah aplikasi manajemen jadwal mengajar guru sekolah menengah atas menyimpulkan bahwa dengan mengubah pengelolaan jadwal guru sekolah dari pencatatan dan penyimpanan data yang dilakukan secara manual berbentuk arsip-arsip berkas menjadi dalam bentuk aplikasi yang terkomputerisasi mendapati bahwa aplikasi tersebut membuat proses menjadi lebih cepat dan efektif. Proses pembaruan dan penginputan jadwal menjadi cepat dan minim akan kesalahan, serta pembuatan laporan yang jauh lebih efektif. Hal ini membuktikan bahwa proses penjadwalan yang terkomputerisasi adalah langkah yang tepat dan efektif dalam studi kasus kami. Dengan menerapkan konsep manajemen jadwal yang baik, sebuah perguruan tinggi dapat mengoptimalkan penggunaan sumber daya dan meningkatkan kepuasan mahasiswa serta efektivitas operasional secara keseluruhan.

## 2.2. Website

Website atau situs dapat dianggap sebagai kumpulan halaman yang menampilkan beragam informasi seperti teks, gambar, animasi, suara, dan video, serta kombinasi dari semua unsur tersebut. Halaman-halaman ini bisa memiliki sifat statis atau dinamis, membentuk struktur terhubung di mana setiap halaman tersambung dengan halaman lainnya melalui *hyperlink* atau jaringan halaman [5].

### 2.2.1. Unsur-unsur dalam Website

Terdapat beberapa unsur yang mendukung keberadaan sebuah website [6], yaitu:

1. Nama Domain

   Nama domain merupakan alamat khusus yang digunakan untuk mengidentifikasi suatu situs web di internet. Dalam bahasa Indonesia, istilah domain sering disebut sebagai "nama domain" atau "alamat domain". Nama domain terdiri dari dua komponen utama, yaitu nama dan ekstensi domain. Bagian nama adalah pengenal unik yang dipilih oleh pemilik situs web, sementara bagian ekstensi domain menunjukkan jenis atau kategori

situs tersebut. Sebagai contoh, "google.com" memiliki "google" sebagai nama dan ".com" sebagai ekstensi domain. Fungsi utama nama domain adalah memudahkan pengguna internet dalam mengakses situs web tanpa harus mengingat alamat IP yang rumit. Selain itu, nama domain juga dapat mencerminkan identitas, tujuan, atau konten dari situs web tersebut.

2. *Web Hosting*

   *Web hosting* adalah layanan yang menyediakan infrastruktur dan teknologi yang diperlukan untuk membuat situs web dapat diakses melalui internet. Secara sederhana, *web hosting* adalah menyewa ruang di server yang terhubung ke internet agar situs web Anda dapat diakses oleh pengguna di seluruh dunia. Ada berbagai jenis *web hosting*, mulai dari *shared hosting* (berbagi server dengan banyak situs web lain), *virtual private server* (VPS), *dedicated server* (server khusus untuk satu situs web), hingga *cloud hosting* (menggunakan sumber daya dari beberapa server yang terhubung dalam satu jaringan).

3. Desain Website

   Kualitas dan daya tarik sebuah website sangat dipengaruhi oleh desainnya. Desain yang baik atau buruk akan memengaruhi pandangan pengunjung terhadap website tersebut. Kualitas sebuah situs sangat bergantung pada keahlian desainernya. Semakin mahir seorang desainer web dalam menggunakan berbagai program atau perangkat lunak yang mendukung pembuatan situs, maka kualitas situs yang dihasilkan juga akan semakin baik. Sebaliknya, jika desainer kurang terampil, situs yang dibuatnya kemungkinan akan kurang berkualitas.

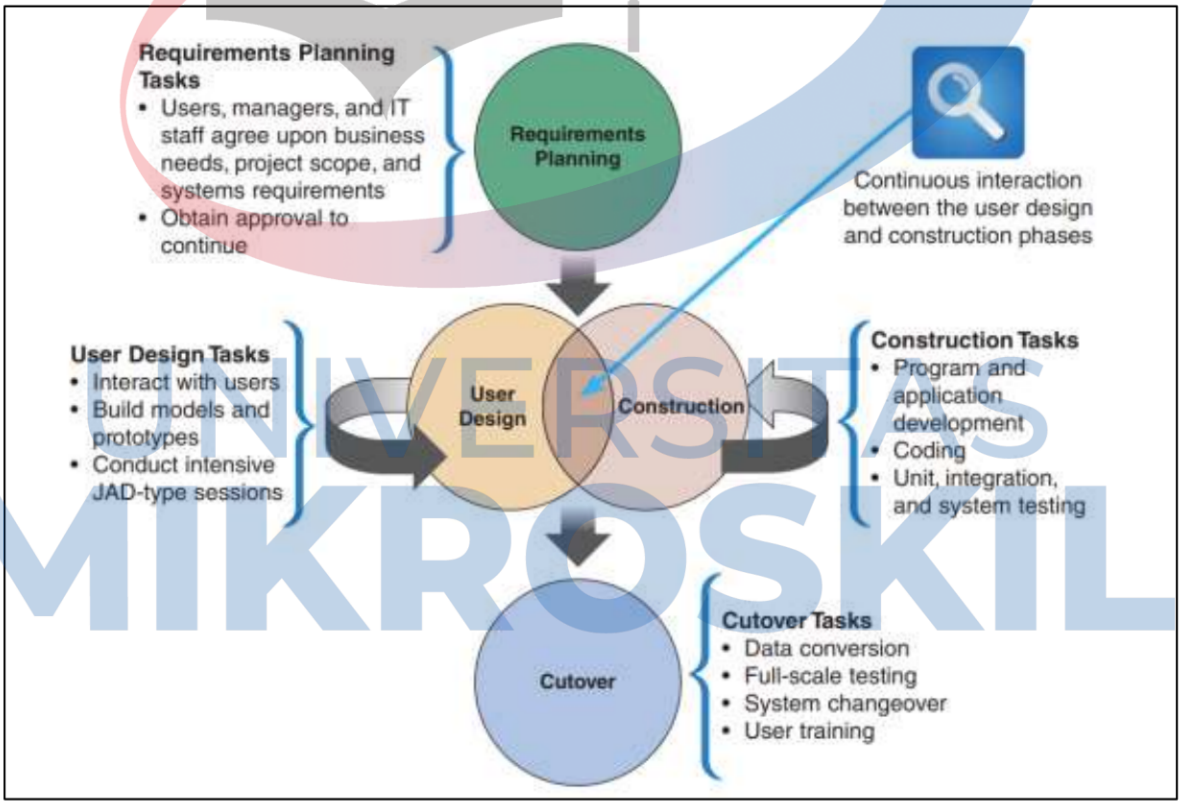
4. Publikasi Website

   Membangun sebuah website tidak akan memberikan manfaat jika tidak memiliki kunjungan atau pengakuan dari masyarakat atau pengguna internet. Kualitas dan keefektifan sebuah situs sangat ditentukan oleh jumlah pengunjung yang signifikan dan umpan balik yang diterima. Untuk memperkenalkan situs kepada masyarakat, diperlukan tindakan publikasi atau promosi. Promosi situs dapat dilakukan melalui berbagai cara, seperti penggunaan pamflet, brosur, spanduk, kartu nama, dan lain sebagainya, namun metode ini sering kali terbatas dan kurang efektif. Salah satu cara yang umum dilakukan dan paling efektif adalah dengan mempublikasikan situs secara langsung di internet melalui mesin pencari.

## 2.3. Rapid Application Development

*Rapid Application Development* (RAD) adalah teknik berbasis tim yang mempercepat pengembangan sistem informasi dan menghasilkan sistem informasi yang berfungsi. RAD adalah metodologi lengkap, dengan siklus hidup empat tahap yang sejalan dengan fase SDLC tradisional. Perusahaan menggunakan RAD untuk mengurangi biaya dan waktu pengembangan serta meningkatkan probabilitas kesuksesan. RAD sangat bergantung pada *prototyping* dan keterlibatan pengguna. Proses RAD memungkinkan pengguna untuk memeriksa model kerja sesegera mungkin, menentukan apakah memenuhi kebutuhan mereka, dan menyarankan perubahan yang diperlukan. Berdasarkan masukan pengguna, prototipe dimodifikasi, dan proses interaktif berlanjut sampai sistem sepenuhnya dikembangkan dan pengguna puas. Tim proyek menggunakan alat CASE untuk membangun prototipe dan membuat aliran dokumentasi yang berkelanjutan [7].

### 2.3.1. Fase dan Aktivitas RAD




Gambar 2.1 Fase dan Aktivitas RAD

Model RAD terdiri dari empat fase meliputi [7]:

1. Perencanaan kebutuhan. Fase perencanaan kebutuhan menggabungkan elemen-elemen perencanaan sistem dan analisis sistem dari SDLC. Pengguna, manajer, dan anggota staf

TI mendiskusikan dan menyetujui kebutuhan bisnis, lingkup proyek, kendala, dan persyaratan sistem. Fase perencanaan kebutuhan berakhir ketika tim setuju pada isu-isu kunci dan mendapatkan otorisasi manajemen untuk melanjutkan.

2. Desain pengguna. Selama fase desain pengguna, pengguna berinteraksi dengan analis sistem dan mengembangkan model dan prototipe yang mewakili semua proses sistem, output, dan input. Desain pengguna adalah proses interaktif yang berkelanjutan yang memungkinkan pengguna memahami, memodifikasi, dan akhirnya menyetujui model kerja sistem yang memenuhi kebutuhan mereka.
3. Konstruksi. Fase konstruksi berfokus pada tugas pengembangan program dan aplikasi yang mirip dengan SDLC. Namun, dalam RAD, pengguna terus berpartisipasi dan masih dapat menyarankan perubahan atau perbaikan saat layar atau laporan aktual dikembangkan.
4. Pergantian. Fase pergantian menyerupai tugas terakhir dalam fase implementasi SDLC, termasuk konversi data, pengujian, pergantian ke sistem baru, dan pelatihan pengguna. Dibandingkan dengan metode tradisional, seluruh proses dijalankan lebih cepat. Sebagai hasilnya, sistem baru dibangun, disampaikan, dan dioperasikan jauh lebih cepat.

### 2.3.2. Tujuan RAD

Tujuan utama dari semua pendekatan RAD adalah untuk memotong waktu dan biaya pengembangan dengan melibatkan pengguna dalam setiap fase pengembangan sistem. Karena itu merupakan proses yang berkelanjutan, RAD memungkinkan tim pengembangan untuk membuat modifikasi yang diperlukan dengan cepat, seiring dengan evolusi desain. Selain keterlibatan pengguna, sebuah tim RAD yang sukses harus memiliki sumber daya TI, keterampilan, dan dukungan manajemen. Karena itu adalah proses dinamis yang didorong oleh pengguna, RAD terutama berharga ketika sebuah perusahaan membutuhkan sistem informasi untuk mendukung fungsi bisnis baru. Dengan mendapatkan masukan pengguna dari awal, RAD juga membantu tim pengembangan merancang sebuah sistem yang membutuhkan antarmuka pengguna yang sangat interaktif atau kompleks [7].

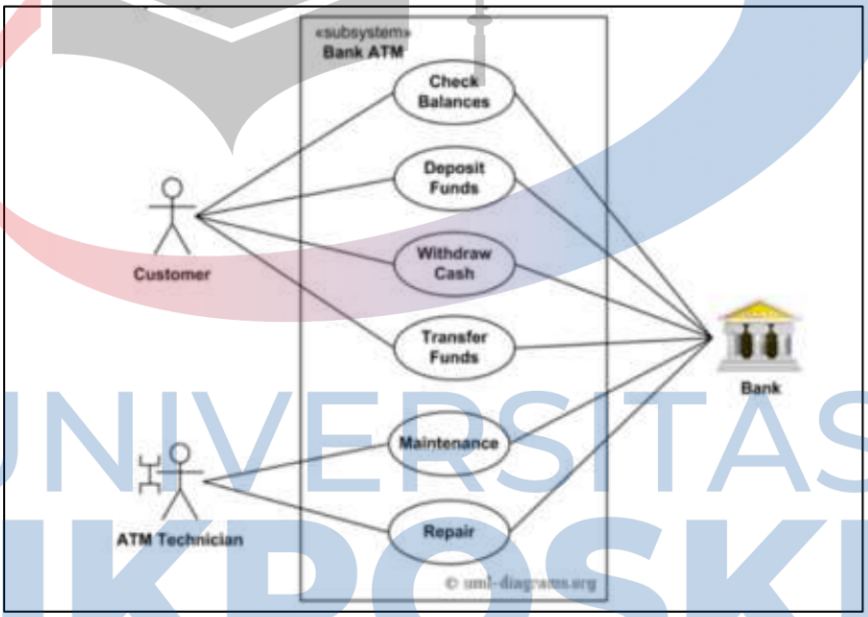
### 2.3.3. Keuntungan dan Kekurangan RAD

RAD memiliki kelebihan dan kekurangan dibandingkan dengan metode analisis terstruktur tradisional. Keunggulan utamanya adalah bahwa sistem dapat dikembangkan lebih cepat dengan penghematan biaya yang signifikan. Kekurangannya adalah bahwa RAD menekankan mekanika sistem itu sendiri dan tidak menekankan kebutuhan bisnis strategis perusahaan. Risikonya adalah bahwa sebuah sistem mungkin berfungsi dengan baik dalam

jangka pendek, tetapi tujuan perusahaan dan jangka panjang untuk sistem tersebut mungkin tidak terpenuhi. Kekurangan potensial lainnya adalah bahwa siklus waktu yang dipercepat mungkin memungkinkan lebih sedikit waktu untuk mengembangkan standar kualitas, konsistensi, dan desain. Namun, RAD dapat menjadi alternatif yang menarik jika sebuah organisasi memahami risiko-risiko yang mungkin terjadi [7].

## 2.4. Use Case Diagram

Diagram *use case* berperan penting dalam menangkap fungsionalitas sistem dari perspektif penggunanya. Representasi visual ini menggambarkan interaksi (aliran informasi) yang terjadi antara aktor (pengguna atau sistem eksternal) dan sistem itu sendiri. Dengan menggunakan diagram *use case*, para pemangku kepentingan, termasuk developer, pengguna, dan analis bisnis, dapat memperoleh pemahaman bersama tentang fungsionalitas yang diharapkan dari sistem tersebut [8].




Gambar 2.2 Contoh Diagram *Use Case*

Terdapat beberapa komponen utama dalam Diagram *Use Case* [8], yaitu:

1. Aktor: Ini mewakili entitas yang berinteraksi dengan sistem untuk mencapai tujuan tertentu. Aktor bisa berupa pengguna manusia, sistem perangkat lunak lain, atau perangkat eksternal. Dalam diagram *use case*, aktor biasanya digambarkan sebagai figur tongkat.
2. *Use Case:* Ini mewakili fungsionalitas yang ditawarkan oleh sistem. Setiap *use case* menangkap urutan interaksi tertentu antara aktor dan sistem, yang mengarah pada hasil

yang terdefinisi dengan baik. *Use case* biasanya ditampilkan sebagai elips dalam diagram.

3. Hubungan: Hubungan antara aktor dan *use case*, serta antara *use case* itu sendiri, digambarkan menggunakan panah dan notasi. Hubungan ini dapat mencakup:
4. Asosiasi: Interaksi dasar antara aktor dan *use case*.
5. *Include:* Aliran fungsionalitas umum yang digunakan oleh beberapa *use case*.
6. *Extend:* Fungsionalitas opsional yang dapat ditambahkan ke *use case* inti dalam kondisi tertentu.
7. Generalisasi: Hubungan hierarkis di mana satu *use case* mewarisi perilaku *use case* lain yang lebih umum.

Terdapat beberapa manfaat dari menggunakan Diagram *Use Case* [8], yaitu:

1. Komunikasi yang Lebih Baik: Diagram *use case* menyediakan representasi visual yang jelas dan ringkas dari fungsionalitas sistem, mendorong komunikasi yang lebih baik antara pemangku kepentingan dengan latar belakang teknis yang beragam.
2. Penggalian Kebutuhan: Diagram ini memfasilitasi identifikasi dan dokumentasi kebutuhan sistem. Aktor dan *use case* mewakili apa yang perlu dilakukan sistem dan untuk siapa.
3. Desain yang Ditingkatkan: Dengan memvisualisasikan interaksi, diagram *use case* dapat memandu proses desain dan pengembangan, memastikan sistem memenuhi tujuan yang diinginkan.
4. Deteksi Kesalahan Dini: Non-konsistensi atau fungsionalitas yang hilang dapat dengan mudah diidentifikasi selama pembuatan diagram *use-case*, yang mengarah pada koreksi dini dan peningkatan kualitas perangkat lunak.

## 2.5. Activity Diagram

*Activity diagram* adalah diagram yang digunakan dalam *Unified Modeling Language* (UML) untuk memodelkan perilaku sistem melalui serangkaian aktivitas. Dalam konteks pengembangan website, diagram aktivitas dapat menggambarkan alur kerja dan interaksi pengguna dengan sistem secara jelas dan terstruktur. Misalnya, pada website *e-commerce*, diagram aktivitas dapat menunjukkan langkah-langkah yang dilakukan pengguna dari memilih produk, menambahkannya ke keranjang, hingga menyelesaikan transaksi pembelian. Diagram ini membantu dalam memahami dan menganalisis proses bisnis serta alur kerja yang kompleks, sehingga memudahkan pengembang dalam merancang dan mengimplementasikan fitur-fitur website dengan lebih efisien [9].

Terdapat simbol-simbol yang digunakan dalam perancangan *Activity Diagram*, yaitu:

1. *Initial Node (Start Point)*: Simbol ini menandai awal dari aktivitas dan digambarkan sebagai lingkaran hitam kecil.
2. *Activity (Action State)*: Simbol ini mewakili tugas atau operasi yang dilakukan dalam aliran kerja dan digambarkan sebagai persegi panjang dengan sudut melengkung.
3. *Control Flow (Edge)*: Panah yang menunjukkan arah dan urutan eksekusi aktivitas dari satu elemen ke elemen lainnya.
4. *Decision Node*: Simbol ini digunakan untuk menunjukkan titik percabangan dalam aliran kontrol, di mana kondisi tertentu menentukan jalur mana yang diambil. Digambarkan sebagai belah ketupat.
5. *Merge Node*: Simbol ini menggabungkan kembali beberapa jalur kontrol ke dalam satu jalur dan juga digambarkan sebagai belah ketupat.
6. *Fork Node*: Simbol ini menunjukkan pemisahan aktivitas menjadi beberapa jalur paralel dan digambarkan sebagai garis tebal horizontal atau vertikal.
7. *Join Node*: Simbol ini menggabungkan beberapa jalur paralel menjadi satu jalur dan juga digambarkan sebagai garis tebal horizontal atau vertikal.

## 2.6. UI/UX

*User Interface*, yang sering disebut sebagai UI, mengacu pada tempat di mana pengguna berinteraksi dengan sistem komputer, aplikasi, atau perangkat lunak. Sebagai komponen penting dalam desain, UI memungkinkan pengguna untuk berkomunikasi, mengontrol, dan memanfaatkan fitur yang tersedia dalam sistem tersebut. Berbagai elemen seperti tombol, ikon, menu, dan kotak dialog tercakup dalam UI dan membantu pengguna dalam melakukan tindakan atau membuat pilihan. Perangkat masukan seperti keyboard, mouse, dan layar sentuh juga terlibat dalam UI, digunakan oleh pengguna untuk memberikan input kepada sistem. Selain itu, respons dan umpan balik dari sistem seperti pesan kesalahan atau animasi juga memiliki peran penting dalam UI. Desain visual yang efektif, termasuk tata letak yang baik dan penggunaan warna yang tepat, membantu menciptakan pengalaman yang menarik dan mudah dimengerti bagi pengguna. Aksesibilitas juga menjadi perhatian dalam UI yang baik, memastikan bahwa semua pengguna, termasuk yang memiliki keterbatasan, dapat dengan mudah mengakses dan menggunakan sistem tersebut. Dengan menyatukan semua elemen ini, sebuah UI yang baik dapat meningkatkan kegunaan dan kenyamanan pengguna serta memberikan pengalaman yang positif saat berinteraksi dengan teknologi.

*User Experience* (UX) mencerminkan interaksi pengguna dengan suatu produk, sistem, atau layanan serta persepsinya terhadapnya. Ini mencakup segala aspek mulai dari interaksi langsung dengan antarmuka pengguna (UI) hingga elemen yang lebih luas seperti citra merek, kenyamanan pengguna, dan kepuasan. Fokus utama UX adalah menciptakan produk atau layanan yang tidak hanya mudah digunakan, tetapi juga memberikan nilai tambah kepada pengguna. Proses desain UX dimulai dengan studi mendalam tentang kebutuhan, preferensi, dan harapan pengguna, di mana tim desain memusatkan perhatiannya pada pengguna dari awal hingga akhir proses, mulai dari perencanaan hingga pengujian. Tujuan utamanya adalah memastikan bahwa produk atau layanan tidak hanya memenuhi kebutuhan fungsional pengguna, tetapi juga memberikan pengalaman yang memuaskan secara emosional. Ini melibatkan desain yang sederhana, konsisten, dan responsif, serta upaya berkelanjutan untuk mendengarkan dan merespons umpan balik pengguna. Dengan memperhatikan semua aspek ini, produk atau layanan dapat menciptakan pengalaman pengguna yang positif, membangun loyalitas pengguna, dan meningkatkan reputasi merek [10].

## 2.7. Basis Data

Basis data (*database*) merupakan komponen penting dalam pengembangan website yang digunakan untuk menyimpan, mengelola, dan mengakses data. Basis data memungkinkan pengembang untuk menyimpan informasi yang diperlukan oleh aplikasi web, seperti informasi pengguna, konten situs, dan data transaksi [11]. Berbagai jenis basis data yang umum digunakan dalam pengembangan web termasuk basis data relasional, seperti *MySQL, PostgreSQL*, dan *SQLite*, serta basis data *NoSQL*, seperti *MongoDB* dan *Redis*. Penggunaan basis data yang tepat sangat penting untuk memastikan kehandalan, kinerja, dan skalabilitas aplikasi web. Selain itu, konsep seperti normalisasi data, *indexing*, dan transaksi juga menjadi bagian integral dalam desain dan pengelolaan basis data, yang memastikan integritas data dan efisiensi operasional. Dengan menggunakan basis data dengan bijak, pengembang dapat membangun aplikasi web yang kuat, efisien, dan dapat diandalkan untuk memenuhi kebutuhan pengguna dengan baik.

### 2.7.1. Class Diagram

*Class diagram* adalah salah satu jenis diagram dalam *Unified Modeling Language* (UML) yang digunakan untuk memodelkan struktur statis dari suatu sistem. Diagram ini menggambarkan kelas-kelas yang ada dalam sistem beserta atribut, metode, dan hubungan antar kelas tersebut. *Class diagram* memainkan peran penting dalam tahap analisis dan

desain perangkat lunak dengan menyediakan gambaran yang jelas tentang bagaimana entitas-entitas dalam sistem saling berinteraksi.

Elemen-elemen utama dalam *class diagram* terdiri dari kelas, atribut, metode, dan hubungan antar kelas. Kelas (*class*) adalah entitas utama yang mengandung atribut dan metode. Atribut (*attributes*) merepresentasikan karakteristik atau properti yang dimiliki oleh kelas, sementara metode (*methods/operations*) adalah fungsi atau operasi yang bisa dilakukan oleh kelas tersebut. Sebagai contoh, kelas Person mungkin memiliki atribut name dan age, serta metode *getName()* dan *getAge()*.

Hubungan (r*elationships*) antara kelas dalam *class diagram* dapat berupa asosiasi, agregasi, komposisi, dan generalisasi. Asosiasi (*association*) menunjukkan hubungan antara dua kelas yang dapat bersifat satu arah (*unidirectional*) atau dua arah (*bidirectional*). Agregasi (*aggregation*) menggambarkan hubungan bagian-keseluruhan di mana bagian dapat berdiri sendiri tanpa keseluruhan. Sebaliknya, komposisi (*composition*) adalah hubungan bagian-keseluruhan yang kuat, di mana bagian tidak dapat berdiri sendiri tanpa keseluruhan. Generalisasi (*generalization/inheritance*) menggambarkan hirarki kelas di mana subclass mewarisi atribut dan metode dari superclass [12].

### 2.7.2. Entity Relationship Diagram (ERD)

*Entity Relationship Diagram* (ERD) adalah alat visual yang digunakan dalam desain basis data untuk menggambarkan struktur data dan hubungan antara entitas (*entities*). ERD membantu pengembang dalam memodelkan struktur basis data dengan jelas dan terorganisir, sehingga memudahkan dalam pemahaman dan komunikasi tentang bagaimana data disimpan dan berinteraksi dalam suatu sistem. Dengan menggunakan ERD, pengembang dapat mengidentifikasi entitas-entitas utama dalam basis data, menentukan atribut-atribut yang terkait dengan setiap entitas, serta menetapkan hubungan antara entitas-entitas tersebut, seperti hubungan *one-to-one, one-to-many*, atau *many-to-many* [11].

Terdapat simbol-simbol yang umum digunakan dalam *Entity Relationship Diagram* (ERD) meliputi [11]:

1. Persegi Panjang (*Rectangle*): Merepresentasikan entitas atau tabel dalam basis data. Setiap entitas memiliki atribut yang terkait dengannya.
2. Garis (*Line*): Merepresentasikan hubungan antara entitas. Garis ini dapat berupa garis tunggal atau panah, tergantung pada jenis hubungan yang dijelaskan (misalnya, hubungan *one-to-one, one-to-many*, atau *many-to-many*).

3. Panah (*Arrow*): Digunakan untuk menunjukkan arah hubungan antara entitas. Panah ini menunjukkan arah yang ditunjukkan oleh hubungan, misalnya, dari entitas induk ke entitas anak.

Dengan demikian, ERD adalah alat yang sangat penting dalam fase perancangan basis data, yang membantu memastikan bahwa struktur basis data yang dirancang sesuai dengan kebutuhan aplikasi dan dapat mengakomodasi ketergantungan dan hubungan antar data dengan baik.