# BAB II LANDASAN TEORI

## Kajian Penelitian Terkait

Sistem Informasi Manejemen (selanjutnya disebut SIM) merupakan penerapan sistem informasi di dalam organisasi untuk mendukung informasi-informasi yang dibutuhkan oleh semua tingkatan manajemen (Prasojo, 2013).

Pada penelitian Ginantra & Atmaja (2018) Aplikasi Manajemen Kegiatan Kemahasiswaan STIKI Indonesia Berbasis Web dapat memberikan kemudahan informasi kegiatan,nama kegiatan, kuota peserta dan jumlah SKP serta tempat pelaksanaan kegiatan kepada mahasiswa STIKI Indonesia. Selain itu, Aplikasi Manajemen Kegiatan Kemahasiswaan STIKI Indonesia Berbasis Web ini memberikan kemudahan bagian kemahasiswaan dalam monitoring kegiatan mahasiswa dan membantu bagian kerumahtanggan dalam mengatur ruangan. Tidak ada metode khusus dalam pengembangan aplikasi ini, hanya menggunakan Data Flow Diagram (DFD) sebagai alat bantu..

Menurut penelitian Ardiana, Suryawan, & Hartono (2018) STMIK STIKOM Indonesia memandang organisasi kemahasiswaan memiliki peran yang sangat penting bagi pengembangan diri mahasiswa. STMIK STIKOM Indonesia saat ini memiliki 32 ORMAWA yang wajib dibina dan diawasi. Pengelolalan administrasi kemahasiswaan saat ini dilakukan secara manual sehingga menyulitkan institusi untuk mengetahui kondisi dan memberikan penilaian terhadap kondisi ORMAWA. Sistem untuk mengelola administrasi organisasi pembinaan terhadap organisasi kemahasiswaan. Sistem ini akan memberikan informasi mengenai kondisi organisasi kemahasiswaan. Penilaian kondisi organisasi kemahasiswaan didasarkan pada keaktifan anggota organisasi kemahasiswaan, jumlah kegiatan, partisipasi mahasiswa dalam kegiatan kemahasiswaan, dan kedisiplinan organisasi kemahasiswaan dalam hal administrasi. Analisis dan perancangan yang digunakan dalam penelitian ini menggunakan metode Structured Analysis and Structured Design. Alat bantu yang digunakan adalah Statement of Purpose (SOP), Event List, Context Diagram, Data Flow Diagram (DFD). Perancangan database dilakukan dengan menggunakan Entity Relationship Diagram (ERD).

Menurut penelitian Pradnyana & Sugihartini (2016) tujuan dari pengembangan sistem informasi pendukung data kemahasiswaan adalah untuk memberikan kemudahan dalam pengumpulan dan mengelola data pendukung kemahasiswaan. Selain itu informasi yang didapat harus terpusat untuk mendukung proses pengisian borang. Arsip data ini nantinya sangat penting dalam proses akreditasi yang dilakukan oleh tim borang BAN PT dalam periode waktu tertentu terhadap kemahasiswaaan berdasarkan kebutuhan FTK UNDIKSHA dan standar evaluasi diri BAN-PT. Sistem yang diusulkan akan menghasilkan laporan-laporan terkait elemen penilaian akreditasi BAN PT tentang sistem kemahasiswaan khususnya mengenai prestasi akademik dan non akademik serta keterlibatan mahasiswa dalam organisasi. Metode yang digunakan dalam pengembangan sistem informasi pendukung data kemahasiswaan adalah metode pengembangan Software Development Life Cycle (SDLC) dengan model Waterfall yang terdiri dari tahap analisis, perancangan, coding/implementasi, pengujian dan perawatan.

Berdasarkan penelitian-penelitian diatas, kontribusi yang akan diusulkan melalui penelitian ini adalah sistem informasi manajemen yang akan dirancang dapat memudahkan bidang kemahasiswaan dalam pengorganisasian data kegiatan dan mempercepat proses pengukuran keaktifan mahasiswa guna memudahkan pelaporan kepada Direktorat Kemahasiswaan DIKTI dan pelaporan untuk syarat akreditasi. Pengembangan sistem ini mengunakan metode *Agile Requirement Engineering* dengan mengadopsi prinsip rekayasa kebutuhan aplikasi pada metodologi *SCRUM* guna mendapatkan *requirement* yang sesuai dengan kebutuhan *user*.

## Agile Software Development Methods

Berdasarkan penelitian yang dilakukan Mahendra & Tresno (2018) , Konsep *Agile Software Development* dicetuskan oleh Kent Beck dan 16 rekannya dengan menyatakan bahwa *agile software development* adalah cara membangun *software* dengan melakukannya dan membantu orang lain membangunnya sekaligus. *Agile software development methods* atau *agile methodology* merupakan sekumpulan metodologi pengembangan perangkat lunak yang berbasis pada pengembangan iteratif, di mana persyaratan dan solusi berkembang melalui kolaborasi antar tim yang terorganisir (Pressman, 2010).

Sementara Sommerville, n.d., (2011) mengemukakan metode *agile* merupakan metode pengembangan *incremental* yang fokus pada perkembangan yang cepat, perangkat lunak yang dirilis bertahap, mengurangi *overhead* proses, dan menghasilkan kode berkualitas tinggi dan pada proses perkembangannya melibatkan pelanggan secara langsung. Ada beberapa model pengembangan perangkat lunak yang termasuk *agile software development methods*, yaitu 1) *Extreme Programming*, 2) *Adaptive Software Development*, 3) *Dynamic Systems Development Method*, 4) *Model Scrum*, dan 5) *Agile Modeling*.

Tujuan dari metode *Agile* adalah untuk mengurangi *overhead cost* dalam proses pengembangan perangkat lunak (misalnya dengan membatasi dokumentasi) dan untuk dapat merespon dengan cepat *requirement* yang berubah tanpa pengerjaan ulang yang berlebihan.(Sommerville & Sommerville, 2016)

## Requirement Engineering

*Requirement Engineering* berfokus pada pengidentifikasian, pemodelan,

komunikasi dan dokumentasi *requirement* untuk sebuah sistem dan konteks dimana sistem akan dipakai. *Requirement* menggambarkan apa saja yang harus dilakukan pada pengembangan sebuah sistem, tetapi tidak dengan bagaimana mengimplementasikannya. Ada banyak teknik yang dapat digunakan selama proses *Requirement Engineering* untuk memastikan *requirement* tersebut lengkap, konsisten dan relevan (Eberlein, 2003).

Tujuan *Requirement Engineering* adalah untuk membantu mengetahui apa yang harus dibangun sebelum pengembangan sistem dimulai , hal ini bertujuan untuk mencegah pengerjaan ulang sistem yang membutuhkan biaya lebih mahal. Tujuan ini didasarkan pada dua asumsi utama :

* Kesalahan yang ditemukan setelah sistem selesai dibangun akan membutuhkan biaya yang lebih mahal untuk memperbaikinya.
* Memungkinkan untuk menentukan serangkaian requirement yang stabil sebelum tahap desain dan implementasi sistem dimulai.

Proses RE terdiri dari 5 tahapan utama, yaitu *Elicitation*, *Analysis* *and Negotiation, Documentation*, *Validation* dan *Management*.

 *Elicitation* adalah proses untuk menemukan *requirement* dan mengidentifikasi batasan sebuah sistem dengan berkonsultasi dengan pihak yang berkepentingan misalnya klien, *developer*, dan pengguna.

 *Analysis and Negotiation* adalah proses menganalisa *requirement* agar sesuai dengan kebutuhan, konsisten, lengkap, dan layak. Konflik dalam requirement diselesaikan melalui negosiasi para pihak yang berkepentingan.

 Tujuan dari *Requirement Documentation* adalah untuk mengkomunikasikan *requirement* antara *stakeholders* dan *developer*. Dokumen *requirement* adalah dasar untuk mengevaluasi produk dan proses selanjutnya, dan untuk mengontrol perubahan. Dokumen *requirement* yang baik tidak ambigu, lengkap, benar, dapat dimengerti, konsisten, ringkas, dan layak.

 Tujuan *Requirement Validation* adalah untuk menyatakan bahwa *requirement* adalah deskripsi sistem yang diterima dan dapat diimplementasikan. *Input* untuk proses validasi adalah dokumen *requirement*, standar organisasi, dan pengetahuan organisasi. *Output*nya adalah daftar yang berisi masalah yang dilaporkan dengan dokumen *requirement* dan tindakan yang perlu dilakukan untuk mengatasi masalah yang dilaporkan.

 Tujuan dari *Requirement Management* adalah untuk menangkap, menyimpan, menyebarluaskan, dan mengelola informasi. Manajemen *requirement* mencakup semua aktivitas yang berkaitan dengan perubahan dan *version control*, pelacakan *requirement*, dan pelacakan status *requirement* (Eberlein, 2003).

## SCRUM

Berdasarkan penelitian yang dilakukan Mahendra & Tresno (2018) , Menurut Pressman (2010) model *scrum* adalah metode pengembangan peranti lunak secara cepat (*agile*). Prinsip *scrum* sesuai dengan prinsip-prinsip yang terdapat pada metode pengembangan peranti secara cepat yang digunakan untuk menuntun kegiatan pengembangan peranti lunak, seperti pemenuhan kebutuhan, analisa, desain, dan penyampaian (*delivery*). Rangkaian kegiatan dalam model *scrum* terdiri dari 1) Aktivitas *Backlog*, 2) Aktivitas *Sprints*, 3) Aktivitas *Scrum Meeting*, dan 4) Demo.



Gambar 2.1 Alur kerja Scrum(Azdy & SN, 2012)

Aktivitas *Backlog* adalah aktivitas menyusun rincian prioritas pada fitur-fitur yang akan dibangun pada sistem informasi yang akan dibangun. Isi pada fitur-fitur dan pat ditambahkan setiap saat.

Aktivitas *Sprints* adalah aktivitas menyusun kegiatan yang akan dilakukan untuk memenuhi kebutuhan yang ditetapkan dalam *backlog*.

Aktivitas *Scrum Meeting* adalah aktivitas menyelenggarakan rapat dengan tim yang telah ditunjuk untuk membahas kemajuan kegiatan pengembangan sistem informasi.

Demo adalah menunjukkan fitur-fitur *software* yang telah dihasilkan untuk dievaluasi oleh pengguna sesuai dengan waktu yang telah ditentukan.

### Tim Scrum

Tim *Scrum* terdiri dari *Product Owner*, Tim Pengembang dan *Scrum Master*. Tim *Scrum* mengatur diri mereka sendiri dan berfungsi antar-lintas. Tim yang mengatur dirinya sendiri menentukan cara terbaik untuk menyelesaikan pekerjaannya, daripada diatur oleh pihak lain yang berada di luar anggota tim.(Schwaber & Sutherland, 2013)

Tim *Scrum* menghantarkan produk secara berkala dan bertahap untuk memperbesar kesempatan mendapatkan masukan. Penghantaran secara bertahap dari sebuah produk yang “Selesai”, memastikan produk yang berpotensi dapat digunakan, selalu siap tersedia.

* 1. *Product Owner*

*Product Owner* bertanggung-jawab untuk memaksimalkan nilai dari produk yang dihasilkan oleh Tim Pengembang. *Product Owner* merupakan satu-satunya orang yang bertanggung jawab untuk mengelola *Product Backlog*. Pengelolaan *Product Backlog* mencakup:

* Mendeskripsikan dengan jelas item *Product Backlog*
* Mengurutkan item di dalam *Product Backlog* untuk mencapai tujuan dan misi dengan cara terbaik
* Mengoptimalkan nilai dari hasil pekerjaan yang dilakukan Tim Pengembang
* Memastikan *Product Backlog* transparan, jelas, dan dapat dilihat semua pihak, dan menunjukkan apa yang akan dikerjakan oleh Tim *Scrum* selanjutnya
* Memastikan Tim Pengembang dapat memahami item dalam *Product Backlog* hingga batasan tertentu

*Product Owner* dapat melakukan semua pekerjaan di atas atau meminta Tim Pengembang untuk mengerjakannya. Namun, hanya *Product Owner* yang bertanggung jawab terhadap *Product Backlog*. *Product Owner* adalah satu orang dan bukan berupa komite. *Product Owner* dapat mewakili aspirasi dari komite ke dalam *Product Backlog*, namun mereka yang ingin merubah prioritas item *Product Backlog*, harus menyampaikannya melalui *Product Owner*.(Schwaber & Scrum, 2013)

* 1. Tim Pengembang

Tim Pengembang terdiri dari para professional yang bekerja untuk menghasilkan tambahan potongan produk (selanjutnya disebut inkremen) “Selesai”, yang berpotensi untuk dirilis di setiap akhir Sprint. Hanya anggota Tim Pengembang yang mengembangkan inkremen ini.

Tim pengembang memiliki karakteristik sebagai berikut:

* Mereka mengatur dirinya sendiri. Tidak ada satu orang pun (bahkan *Scrum Master*) yang memerintah Tim pengembang bagaimana cara merubah *Product Backlog* menjadi Inkremen yang berpotensi untuk dirilis.
* Tim pengembang berfungsi antar-lintas, sebagai sebuah tim, memiliki semua keahlian yang dibutuhkan untuk menghasilkan produk.
* *Scrum* tidak mengenal adanya jabatan tertentu untuk anggota Tim Pengembang selain Pengembang, apapun pekerjaan yang dikerjakan oleh masing-masing anggota tim; tidak ada pengecualian untuk aturan yang satu ini.
* Tim pengembang boleh memiliki spesialisasi keahlian dan fokus di satu area tertentu, namun akuntabilitas dari hasil pekerjaan secara keseluruhan adalah milik Tim Pengembang.
	1. *Scrum Master*

*Scrum Master* bertanggung-jawab untuk memastikan *Scrum* telah dipahami dan dilaksanakan. *Scrum Master* melakukan-nya dengan memastikan Tim *Scrum* mengikuti teori, praktik, dan aturan main *Scrum*.

*Scrum Master* adalah seorang pemimpin yang melayani Tim *Scrum*. *Scrum* Master membantu pihak di luar Tim *Scrum*, untuk memahami apakah interaksi mereka dengan Tim *Scrum* bermanfaat atau tidak. *Scrum Master* membantu setiap pihak untuk merubah interaksi-interaksi yang tidak bermanfaat sehingga bisa memaksimalkan nilai yang dihasilkan oleh Tim *Scrum*.

### Acara-acara Scrum

Acara-acara wajib dalam *Scrum* dihadiri untuk menciptakan sebuah kesinambungan dan mengurangi adanya acara-acara lain yang tidak tercantum di dalam *Scrum*. Setiap acara didalam *Scrum* memiliki batasan waktu, yang artinya selalu memiliki durasi maksimum. Pada saat *Sprint* dimulai, durasinya tetap dan tidak dapat diperpendek maupun diperpanjang. Acara-acara lainnya dapat diakhiri saat tujuan dari acara tersebut telah tercapai, memastikan waktu digunakan secukupnya tanpa ada yang terbuang sia-sia.(Schwaber & Sutherland, 2013)

1. *Sprint*

Jantung dari *Scrum* adalah *Sprint*, sebuah batasan waktu selama satu bulan atau kurang, di mana sebuah Inkremen yang “Selesai”, berfungsi, berpotensi untuk dirilis dikembangkan. *Sprint* biasanya memiliki durasi yang konsisten sepanjang proses pengembangan produk. *Sprint* yang baru, langsung dimulai setelah *Sprint* yang sebelumnya berakhir.

*Sprint* memuat dan terdiri dari *Sprint Planning*, *Daily Scrum*, pengembangan, *Sprint Review* dan *Sprint Retrospective*. Pada saat *Sprint* terdapat beberapa hal yang harus diperhatikan, yaitu:

* Tidak boleh ada perubahan yang dapat membahayakan tercapaianya *Sprint Goal*
* Kualitas dari *Sprint Goal* tidak boleh menurun
* *Scope* dapat diklarifikasikan dan dinegosiasikan ulang diantara *Product Owner* dan Tim Pengembang seiring dengan bertambahnya pengetahuan

Setiap *Sprint* dapat dikatakan sebagai sebuah proyek dengan batasan waktu tidak lebih dari satu bulan. Sama halnya dengan proyek, *Sprint* digunakan untuk menyelesaikan sesuatu. Setiap *Sprint* memiliki definisi mengenai apa yang akan dikembangkan, sebuah desain dan perencanaan yang fleksibel yang akan membimbing pengembangan, pekerjaan yang akan dilakukan dan hasil dari produk.(Schwaber & Sutherland, 2013)

### Artefak Scrum

Artefak *Scrum* merepresentasikan pekerjaan atau nilai, bertujuan untuk menyediakan transparansi, dan kesempatan-kesempatan untuk peninjauan dan adaptasi. Artefak yang didefinisikan oleh *Scrum* secara khusus dirancang untuk meningkatkan transparansi dari informasi kunci, dengan begitu semua pihak dapat memiliki pemahaman yang sama terhadap artefak (Schwaber & Scrum, 2013).

1. *Product Backlog*

*Product Backlog* adalah daftar terurut, dari setiap hal yang berkemungkinan dibutuhkan di dalam produk, dan juga merupakan sumber utama, dari daftar kebutuhan mengenai semua hal yang perlu dilakukan terhadap produk. *Product Owner* bertanggung-jawab terhadap *Product Backlog*, termasuk isinya, ketersediaannya, dan urutannya.

*Product Backlog* tidak pernah selesai. Pada awal pembuatannya hanya terjabar daftar kebutuhan yang paling diketahui dan dipahami pada saat itu. *Product Backlog* berkembang seiring dnegan berkembangnya produk dan lingkungan dimana produk tersebut digunakan. *Product Backlog* bersifat dinamis dimana senantiasa berubah agar produk dapat menjadi layak, kompetitif di pasar, dan bermanfaat bagi penggunanya. Selama produk masih eksis maka *Product Backlog* juga eksis.

*Product Backlog* menjabarkan semua fitur, fungsi, kebutuhan, penyempurnaan dan perbaikan terhadap produk di rilis mendatang. Item *Product Backlog* memiliki atribut deskripsi, urutan, estimasi dan nilai bisnis.

Seiring dengan digunakannya produk dan semakin bertambahnya nilai dari produk, dan bertambahnya masukan dari pasar, *Product Backlog* semakin berkembang menjadi lebih besar. Daftar kebutuhan tidak pernah berhenti berubah, sehingga *Product Backlog* dapat dikatakan sebagai artefak yang hidup. Perubahan dalam kebutuhan bisnis, keadaan pasar, ataupun teknologi dapat menyebabkan perubahan pada *Product Backlog*.

Item *Product Backlog* pada urutan yang lebih atas biasanya lebih jelas dan lebih detail dibandingkan item di bawahnya. Estimasi dengan presisi tinggi diberikan berdasarkan tingkat kejelasan dan detail yang tinggi dimana semakin bawah urutan dari item *Product Backlog*, maka semakin rendah pula tingkat kedetailannya. Item *Product Backlog* yang akan dikerjakan oleh Tim Pengembang untuk *Sprint* yang mendatang di-*refine* supaya setiap item yang dikerjakan dapat di-“Selesai”-kan dalam satu *Sprint*. Item *Product Backlog* yang dianggap dapat di-“Selesai”-kan oleh Tim Pengembang dalam satu *Sprint* dikatakan “Siap” untuk diseleksi pada saat *Sprint Planning*. Item *Product Backlog* biasanya memiliki tingkat transparansi yang tinggi karena adanya aktifitas *refinement* ini (Schwaber & Scrum, 2013).

### Definisi Selesai

Ketika sebuah item *Product Backlog* atau Inkremen dikatakan “Selesai”, setiap pihak harus mengerti dengan apa yang dimaksud dengan “Selesai”. Walupun definisi ini berbeda-beda antar tim *Scrum*, sesama anggota tim harus memiliki pemahaman yang sama mengenai pekerjaan yang harus mereka selesaikan guna memastikan adanya transparansi. Ini adalah definisi selesai untuk Tim *Scrum* dan ini digunakan untuk memeriksa apakah pekerjaan untuk mengembangkan Inkremen dianggap selesai.

Definisi yang sama akan membimbing Tim Pengembang dalam mengetahui berapa banyak item *Product Backlog* yang mereka bisa ambil pada saat *Sprint Planning*. Tujuan dari setiap *Sprint* adalah untuk menghantarkan Inkremen, yang berpotensi untuk dirilis, yang memenuhi definisi “Selesai” terkini yang dibuat oleh Tim *Scrum*.

Tim Pengembang menghantarkan Inkremen yang berfungsi setiap *Sprint*. Inkremen ini dapat digunakan, supaya *Product Owner* dapat merilis produk tersebut sesegera mungkin jika ia mau. Apabila definisi untuk sebuah Inkremen adalah bagian dari konvensi, standar atau panduan pengembangan dari organisasi, setiap Tim *Scrum* harus mengikuti seluruhnya sebagai minimum *requirement.* Apabila “Selesai” untuk sebuah Inkremen bukan merupakan bagian dari konvensi, standar atau panduan pengembangan dari organisasi, Tim Pengembang harus membuat definisi “Selesai” yang pantas untuk produk yang dikembangkan. Apabila ada beberapa Tim *Scrum* yang mengembangkan system atau produk yang sama, seluruh Tim Pengembang di setiap Tim *Scrum* harus menentukan definisi selesai yang sama bersama-sama.

Setiap Inkremen merupakan gabungan dari Inkremen *Sprint-sprint* sebelumnya dan diuji secara teliti, untuk memastikan setiap Inkremen dapat berfungsi secara penuh.

Seiring dengan bertambah dewasanya Tim *Scrum*, mereka diharapkan untuk membuat definisi selesai yang lebih baik dan ketat lagi demi peningkatan kualitas. Produk atau sistem manapun harus memiliki definisi “Selesai” yang merupakan sebuah standar untuk pekerjaan yang akan dilakukan. (Schwaber & Scrum, 2013)

## Basis Data

Basis data (*database*) adalah sebuah kumpulan dari file yang saling berkaitan. Teknologi *database* memiliki kelebihan dalam cara penyimpanan data dengan format yang fleksibel. Hal ini memungkinkan *database* terpisah dari sistem informasi dan aplikasi yang menggunakannya. *Database* dapat berkembang sesuai perubahan kebutuhan organisasi (Whitten, Jeffery, Bentley, Dittman, & Kevin, 2007).

Komponen-komponen penyusun sebuah database sebagai berikut:

a. *Field*

*Field* adalah unit terkecil dari data yang mempunyai arti untuk disimpan kedalam sebuah file atau *database*. Terdapat empat jenis *field* yang dapat tersimpan.

* Primary Key

Primary key adalah sebuah field atau kelompok dari field yang

bersifat unik sebagai identitas dari sebuah record.

- Secondary Key

Secondary key adalah sebuah field yang menjadi identitas dari sebuah record atau sebuah bagian dari record yang berkaitan.

- Foreign Key

Foreign key adalah sebuah field yang mengacu kepada record yang terdapat pada file berbeda dalam sebuah database.

- Descriptive key

Descriptive key adalah field lain yang tidak menjadi key

b. Record

Record adalah sebuah kumpulan dari field-field yang tersusun dari terikat berdasarkan format tertentu yang sudah ditentukan sebelumnya.

c. Files dan Tables

Files dalam konteks database adalah himpunan semua kejadian yang terjadi pada struktur record. Tables adalah database yang mempunyai relasi setara dengan sebuah file.

##  ERD (Entity Relationship Diagram)

 Menurut Bentley (2000) seringkali suatu entitas dapat memiliki lebih dari satu kunci. misalnya, karyawan entitas dapat diidentifikasi secara unik dengan nomor jaminan sosial, atau nomor perusahaan yang ditugaskan oleh perusahaan, atau alamat e-mail. Masing-masing atribut ini disebut kunci kandidat. Kunci kandidat adalah "kandidat untuk menjadi kunci utama" dari contoh entitas. Kadang-kadang disebut sebagai *identifier* kandidat (kunci kandidat dapat berupa atribut tunggal atau kunci gabungan).

 Kunci utama adalah kunci kandidat yang paling umum digunakan untuk mengidentifikasi satu entitas entitas secara unik. Default untuk kunci primer tidak selalu nol. Karena jika kunci tidak memiliki nilai, ia tidak dapat melayani tujuannya untuk mengidentifikasi turunan dari suatu entitas. Kunci kandidat apa pun yang tidak dipilih untuk menjadi kunci utama disebut sebagai tombol alternatif. sinonim umum adalah kunci sekunder. Margin mendemonstrasikan notasi untuk *primary key* dan *alternate key*. Semua kunci kandidat harus utama atau alternatif; oleh karena itu, tidak menggunakan notasi terpisah untuk kunci kandidat. Semua atribut yang bukan bagian dari *primary key* disebut atribut *nonkey*.

 Kadang-kadang, juga diperlukan untuk mengidentifikasi *subsetting* jika instansi entitas yang bertentangan, misalnya memerlukan cara sederhana untuk mengidentifikasi semua siswa laki-laki dan semua siswa perempuan. Kriteria *subsetting* adalah atribut (gabungan atribut) yang menahan nilai yang terbatas membagi semua contoh entitas menjadi subset yang berguna. Ini kadang-kadang disebut sebagai entri inversi dalam atau entitas siswa, atribut gender membagi contoh siswa menjadi dua subset siswa laki-laki dan siswa perempuan. Secara umum, kriteria *subsetting* hanya berguna ketika atribut memiliki batasan jumlah nilai yang sah (artinya terbatas) . Misalnya, nilai rata-rata tidak akan menjadi kriteria *subsetting* yang baik karena ada 999 kemungkinan nilai antara 0,00 dan 4,00 untuk atribut itu. Seni margin menunjukkan notasi untuk kriteria *subsetting.*

**Gambar 2.2** Simbol-simbol ERD





**Gambar 2.3** Contoh ERD