



MODUL PERKULIAHAN

Basis Data

Pengenalan Basis Data

Fakultas
Ilmu Komputer

Program Studi
Sistem Informasi

Tatap Muka
01

Kode MK
87010

Disusun Oleh
Tim Dosen

Abstract

Modul ini mempelajari pengenalan basis data.

Kompetensi

Mahasiswa mampu menjelaskan konsep basis data, mampu menjelaskan komponen dalam basis data, mampu menyebutkan dan menjelaskan keuntungan dan kelebihan penggunaan basis data.

A. Pendahuluan

Sebelum munculnya konsep basis data, untuk organisasi/perusahaan menggunakan pendekatan flat file untuk manajemen datanya. Pada pendekatan ini, user pada tiap departemen dalam suatu organisasi memiliki program aplikasi sendiri dan setiap program aplikasi menyimpan data mereka masing-masing

Pendekatan ini memiliki berbagai kelemahan, antara lain:

- **Tempat penyimpanan** – menghasilkan biaya penyimpanan tinggi untuk dokumen kertas/magnetik form yang digunakan.
- **Kerangkapan data** – data yang sama disimpan di berbagai tempat penyimpanan yang berbeda sehingga organisasi memiliki banyak data yang rangkap.
- **Data Updating** – perubahan terhadap data harus dilakukan berulang kali mengingat data yang sama terdapat di berbagai tempat penyimpanan.
- **Currency of Information** – memiliki potensi masalah jika gagal untuk memperbarui data pada semua file yang terpengaruh.
- **Task-Data Dependency** – pengguna tidak dapat memperoleh tambahan informasi saat membutuhkan informasi tambahan.

B. Konsep Basis Data

Pendekatan basis data mengubah model penyimpanan data yang ada pada pendekatan flat file dimana data disimpan di setiap departemen menjadi terkumpul dalam satu basis data yang dapat dipakai secara bersama oleh seluruh pengguna dalam organisasi / perusahaan.

C. Kelebihan Basis Data

Pendekatan basis data memberikan banyak keuntungan (kelebihan), antara lain:

1. Pengendalian terhadap kerangkapan data

Dilakukan dengan cara data hanya disimpan sekali. Hal ini mengurangi kerangkapan data dan mengurangi biaya untuk tempat penyimpanan.

2. Konsistensi Data

Dilakukan dengan cara data disimpan hanya sekali dalam basis data sehingga jika terjadi perubahan pada nilai data tersebut, perubahan hanya dilakukan satu kali dan nilai baru tersebut akan tersedia untuk semua pengguna.

3. Dapat memperoleh lebih banyak informasi dari data yang sama. Pengguna basis data dapat memperoleh informasi selain dari informasi rutin yang dikelolanya karena semua data lain berada dalam basis data yang sama. Dengan demikian kebutuhan akan informasi selain dari informasi rutin dapat terpenuhi.

4. Data dapat dipakai secara bersama (*shared*)

Data yang ada pada basis data menjadi milik seluruh organisasi dan dapat dipakai secara bersama oleh pengguna yang berwenang pada saat bersamaan.

5. Memperbaiki integritas data

Integritas data mengacu pada validitas dan konsistensi dari data yang disimpan. Integritas biasanya diekspresikan dalam batasan (*constraints*) yang merupakan aturan yang konsisten dan tidak dapat dilanggar. Jika kerangkapan data dikontrol dan kekonsistenan data dapat dijaga maka data menjadi akurat

6. Meningkatkan keamanan data

Keamanan basis data melindungi basis data dari pengguna yang tidak memiliki otorisasi. DBA dapat menentukan batasan-batasan pengaksesan data, misalnya dengan memberikan password dan pemberian hak akses bagi pemakai (misal : *modify*, *delete*, *insert*, *retrieve*).

7. *Economy of scale*

Dengan menggabungkan semua data operasional organisasi ke dalam satu basis data dengan aplikasi yang dibutuhkan dapat menghasilkan penghematan biaya. Anggaran yang biasanya dialokasikan ke setiap departemen untuk pengembangan dan pemeliharaan dari sistem file mereka dapat digabung sehingga menurunkan total biaya dan menciptakan *economy of scale*.

8. Meningkatkan aksesibilitas terhadap data dan respon yang lebih baik

Akibat dari integrasi data yang melewati batasan-batasan departemen dapat langsung diakses oleh pengguna. Ini berarti menyediakan sistem dengan fungsi yang lebih baik. Pengguna dapat memperoleh data yang dibutuhkan dengan cepat dengan menggunakan *query language*.

9. Dapat meningkatkan *data independence* (kemandirian data)

Dapat digunakan untuk bermacam-macam program aplikasi tanpa harus merubah format data yang sudah ada

D. Kelemahan Basis Data

Selain keuntungan, basis data juga memiliki beberapa kelemahan, antara lain :

1. Rumit

Harapan akan fungsi yang baik dari sebuah basis data yang baik membuat basis data menjadi software yang rumit. Perancang, pengembang, DBA, basis data administrator, dan pengguna akhir harus memahami fungsi basis data agar dapat mengambil manfaat dari basis data. Kegagalan dalam memahaminya akan menyebabkan keputusan yang buruk bagi organisasi dan membahayakan organisasi.

2. Biaya basis data

Biayanya sangat mahal karena menyangkut biaya-biaya untuk hardware and software.

3. Terdapat tambahan biaya hardware

Harus menggunakan biaya tambahan untuk hardware, storage, and network.

4. Terdapat biaya konversi

Diperlukan biaya yang besar untuk berpindah dari aplikasi/sistem yang lama ke dalam sistem dan hardware basis data yang baru. Diperlukan pula biaya untuk pelatihan staf untuk menggunakan sistem yang baru ini serta tambahan biaya untuk mempekerjakan staff khusus seperti DBA, dan lain-lain.

E. Karakteristik Basis Data

Untuk mempelajari basis data lebih lanjut, terlebih dahulu kita harus mengetahui karakteristik dari basis data itu sendiri. Karakteristik dari basis data antara lain:

1. Tujuan dari basis data adalah untuk membantu manusia agar dapat melacak segala sesuatu yang dianggap penting.

2. Data disimpan pada tabel. Tabel terdiri dari baris dan kolom seperti yang terlihat pada *spreadsheet*. Basis data memiliki beberapa tabel, dimana setiap tabel menyimpan data tentang suatu hal yang berbeda.

3. Setiap baris dalam suatu tabel menyimpan data dari *instance* yang berbeda. Misalnya pada tabel MAHASISWA:

- Baris 1 menyimpan data untuk Pratama
- Baris 2 menyimpan data untuk Parto

- Baris 3 menyimpan data untuk Pardede
4. Setiap kolom dalam suatu tabel menyimpan karakteristik dari suatu *instance* tersebut. Misalnya pada tabel MAHASISWA:
- Kolom 1 menyimpan data untuk NPM seorang mahasiswa
 - Kolom 2 menyimpan data untuk NamaMhsw seorang mahasiswa
 - Kolom 3 menyimpan data untuk Jurusan seorang mahasiswa
5. Basis data menyimpan data dan relasi. Basis data hanya terdiri dari data tentang *mahasiswa*, mata kuliah, dosen, dan *nilai*. Tetapi juga terdapat relasi(*relationship*) antara baris-baris pada tabel-tabel di dalam basis data tersebut.

NPM	NamaMhsw	Jurusan	NoTelpon
07001	Husni Pratama	Manajemen	08115577869
07002	Roni Parto	Akuntansi	08129992223
07003	Dono Pardede	Akuntansi	02178908764
07004	Dora Nakula	Sistem Informasi	02517778886
07005	Santi Arlisa	Sistem Informasi	08171414325

MAHASISWA

NPM	NamaMhsw	Jurusan	NoTelpon
07001	Husni Cook	Manajemen	08115577869
07002	Roni Parto	Akuntansi	08129992223
07003	Dono Pardede	Akuntansi	02178908764
07004	Dora Nakula	Sistem Informasi	02517778886
07005	Santi Arlisa	Sistem Informasi	08171414325

NILAI

NPM	KodeMatKul	Semester	Tahun	Nilai
07001	AKT100	Ganjil	2006	A
07001	TSI200	Ganjil	2006	B
07003	AKT100	Ganjil	2006	B
07003	TSI200	Ganjil	2006	C
07005	TSI240	Genap	2006	A
07005	AKT200	Genap	2006	C
07005	TSI300	Genap	2006	C

MATAKULIAH

KodeMatKul	NamaMatKul	KodeDosen
TSI200	Sistem Informasi Manajemen	D101
TSI240	SMBD	D220
TSI300	Jaringan Komputer	D224
AKT100	Dasar-dasar Akuntansi	D315
AKT200	Manajemen Keuangan	D315

DOSEN

KodeDosen	NamaDosen	NoTelpon
D101	Randy Martin	0217899666
D220	Dona Sanjaya	0812998877
D224	Paula Mahdi	0811772233
D315	Susi Abdul	0817223344
D421	Rully Rodolaly	0212345779

Konvensi Penamaan

- Nama tabel ditulis dengan huruf besar semua, misalnya **MAHASISWA**
- Nama kolom ditulis dengan huruf awal huruf besar, dan jika diperlukan lebih dari 1 kata maka huruf awal kata berikutnya dimulai dengan huruf besar. Misalnya **KodeDosen**, **NamaDosen**, **Nilai**.

F. Konsep Dasar Basis Data

Data adalah fakta atau angka yang disimpan atau dicatat. Informasi adalah data yang diolah dan disajikan sehingga memiliki suatu arti. Data pada tabel MAHASISWA, MATA KULIAH, dan NILAI dapat menghasilkan informasi tentang IPK mahasiswa.

Basis data selain untuk mencatat atau menyimpan data-data, juga dapat digunakan untuk membentuk suatu informasi. Basis data terdiri dari 2 tabel atau lebih dimana tabel-tabel tersebut saling berhubungan digunakan untuk memenuhi kebutuhan para pemakai dalam suatu organisasi. Setiap tabel pada basis data biasanya berisi tentang suatu hal.

Sistem manajemen basis data adalah perangkat lunak yang menciptakan, proses, dan mengelola basis data. Contoh dari sistem manajemen basis data adalah Microsoft Access, SQL Server, MySQL, ORACLE.

Saat ini basis data yang banyak digunakan adalah model basis data relasional. Model ini diperkenalkan oleh E.F. Codd pada tahun 1970, menggunakan matematika yang dikenal sebagai aljabar relasional. Saat ini model relasional digunakan sebagai model standar untuk basis data komersial.

Istilah Dasar dalam Model Basis data Relasional

Beberapa istilah yang digunakan dalam basis data relasional adalah:

- *Entitas*

Adalah suatu obyek yang terdapat di dunia nyata yang bisa dibedakan dengan obyek yang lainnya dimana obyek ini memiliki segala sesuatu yang dapat dilacak oleh penggunaanya seperti pelanggan, pegawai, penjualan.

- *Atribut*

Adalah karakteristik dari suatu entitas. Contoh: atribut dari pelanggan adalah nomor pelanggan, nama pelanggan, alamat, no telpon.

- *Relasi*

Relasi adalah tabel dua dimensi yang memiliki karakteristik sebagai berikut:

- Barisnya merupakan data tentang suatu entitas
- Kolom-kolomnya terdiri dari data tentang atribut sebuah entitas
- Semua data pada kolom yang sama memiliki jenis yang sama
- Setiap kolom mempunyai nama yang unik
- Sel dari suatu tabel menyimpan 1 nilai
- Urutan kolom tidak penting
- Urutan baris tidak penting
- Dua baris tidak mungkin identik (tidak ada duplikasi)

Tidak semua tabel dapat dikatakan sebagai relasi. Ada tabel-tabel yang bukan sebuah relasi.

MAHASISWA

NPM	NamaMhsw	Jurusan	NoTelpon
07001	Husni Cook	Manajemen	08115577869
07002	Roni Parto	Akuntansi	08129992223
07003	Dono Pardede	Akuntansi	02178908764
07004	Dora Nakula	Sistem Informasi	02517778886
07005	Santi Arlisa	Sistem Informasi	08171414325

MAHASISWA

NPM	NamaMhsw	Jurusan	NoTelpon
07001	Husni Cook	Manajemen	08115577869
			08124587589
			08569854712
07002	Roni Parto	Akuntansi	08129992223
07003	Dono Pardede	Akuntansi	02178908764
07004	Dora Nakula	Sistem Informasi	02517778886
			02184569852

07005	Santi Arlisa	Sistem Informasi	08171414325
-------	--------------	------------------	-------------

- Ketergantungan Fungsional (*Functional dependency*)
Ketergantungan fungsional terjadi ketika nilai satu atribut (satu set atribut) menentukan nilai atribut kedua (nilai satu set atribut kedua).
Contohnya:

NPM → NamaMahasiswa

(dibaca NPM menentukan NamaMahasiswa atau NamaMahasiswa bergantung secara fungsi pada NPM)

Ketergantungan fungsional bisa berdasarkan pada persamaan seperti pada:

TotalHarga= Jumlah x HargaSatuan

(Jumlah, HargaSatuan) → TotalHarga

Tetapi ketergantungan fungsional bukan persamaan. Contoh:

NPM → NamaMahasiswa

NPM → Alamat

NPM → (NamaMahasiswa, Alamat)

- *Determinant*
Pada ketergantungan fungsional **NPM → NamaMahasiswa**; atribut di sebelah kiri ketergantungan fungsional disebut sebagai determinant. Suatu determinant disebut unik jika dan hanya jika menentukan setiap kolom lain dalam suatu relasi.
Suatu determinan dapat disebut sebagai *composite determinant* jika determinant dari suatu ketergantungan fungsional terdiri dari lebih dari satu atribut. Contoh:

(NPM, KodeMatakuliah) → (Nilai)

- *Candidate key*
Key adalah kombinasi dari satu atau lebih kolom yang digunakan untuk mengidentifikasi baris-baris dalam suatu relasi. **Candidate Key** adalah key yang menentukan semua kolom lain dalam suatu relasi. Pada tabel MATAKULIAH, NamaMatkul adalah *Candidate Key*.
- *Composite key*
adalah key yang terdiri dari dua atau lebih kolom.
- *Primary key*

Adalah *candidate key* yang dipilih sebagai *primary key* sebagai sarana untuk mengidentifikasi baris-baris dalam sebuah relasi.

- Dalam satu tabel hanya ada satu *primary key*.
- *Primary key* dapat berupa *composite key*.
- *Primary key* yang ideal biasanya pendek, berupa angka, dan tidak pernah berubah.

- *Surrogate key*

adalah kolom yang ditambahkan pada suatu tabel untuk bertindak sebagai ***primary key***. DBMS akan membuat nilai kolom tersebut unik secara otomatis pada saat baris data pada tabel tersebut dibuat.

Contoh:

RENTAL_PROPERTY (Street, City, State/Province, Zip/PostalCode, Country, Rental_Rate)

RENTAL_PROPERTY(PropertyID, Street, City, State/Province, Zip/PostalCode, Country, Rental_Rate)

- *Foreign key*

adalah kolom atau komposite kolom dimana kolom tersebut merupakan ***primary key*** pada tabel lain (yang menjadi referensi).

Contoh:

DOSEN(KodeDosen, NamaDosen, NoTelpon)

MATAKULIAH(KodeMatKul, NamaMatKul, *KodeDosen*)

- *Referential integrity constraint*

Adalah pernyataan yang membatasi nilai-nilai foreign key ke nilai-nilai yang sudah ada sebagai *primary key* pada relasi yang berhubungan.

MATAKULIAH

KodeMatKul	NamaMatKul	KodeDosen
TSI200	Sistem Informasi Manajemen	D101
TSI240	SMBD	D220
TSI300	Jaringan Komputer	D224
AKT100	Dasar-dasar Akuntansi	D315
AKT200	Manajemen Keuangan	D315

DOSEN

KodeDosen	NamaDosen	NoTelpon
D101	Randy Martin	0217899666
D220	Dona Sanjaya	0812998877

D224	Paula Mahdi	0811772233
D315	Susi Abdul	0817223344
D421	Rully Rodolaly	0212345779

- *Normal form*

Tabel-tabel pada basis data harus memenuhi aturan normalisasi, yaitu bebas dari ketergantungan struktural atau anomali yang disebabkan oleh modifikasi data. Anomali ini disebut *modification anomalies*. Modification Anomalies terdiri dari:

- *Deletion anomaly*

Misalkan kita menghapus data dengan RepairNumber 2100. Data yang terhapus bukan hanya data tentang repair, tetapi data tentang mesin juga akan terhapus (informasi tentang mesin Lathe dengan AcquisitionPrice 4750 akan hilang).

Dari kasus di atas, ketika kita menghapus satu baris pada tabel tersebut, kita akan kehilangan 2 fakta sekaligus (machine dan repair) karena struktur tabel tersebut memaksa demikian. Kondisi tersebut dinamakan **deletion anomaly**.

- *Insertion anomaly*

Pada tabel EQUIPMENT_REPAIR, misalkan kita ingin memasukan data *repair*. Untuk memasukan data tersebut selain diperlukan:

- Data *Repair*, yaitu: RepairNumber, RepairDate, dan RepairAmount.
- Juga perlu data *Machine*, yaitu: ItemNumber, Type, dan AcquisitionCost.

Artinya ketika kita hanya perlu memasukan data tentang satu entity (*repair*), kita diharuskan memasukan data tentang entity lain (*machine*) karena kondisi struktur tabelnya mengharuskan demikian. Kondisi ini dinamakan **insertion anomaly**

- *Update anomaly*.

Pada tabel EQUIPMENT_REPAIR, misalkan kita akan mengubah data (update). Misalkan kita akan mengubah data baris terakhir menjadi:

Terlihat bahwa “Drill Press” memiliki dua nilai AcquisitionCost yang berbeda. Sehingga terjadi *data inconsistency*. Kondisi demikian dinamakan **update anomaly**

Bentuk normal terdiri dari: 1NF, 2NF, 3NF, BCNF, dan DK/NF (akan dipelajari lebih lanjut pada pertemuan berikutnya).

G. Komponen Basis Data

Komponen sistem basis data terdiri dari:

1. User (pengguna)
Pengguna sistem basis data
2. Aplikasi Basis Data
Program komputer yang digunakan oleh user untuk bekerja.
3. DBMS
Sistem manajemen basis data yang membuat, memproses, dan mengatur basis basis datas.
4. Basis data

Komponen Sistem Basis Data dengan SQL

1. User (pengguna)
Pengguna sistem basis data
2. Aplikasi basis data
Program komputer yang digunakan oleh user untuk bekerja.
3. *Structured Query Language (SQL)*
Bahasa standar (secara internasional) yang dapat digunakan untuk melakukan query terhadap data pada semua sistem basis data
4. DBMS
Sistem manajemen basis data yang membuat, memproses, dan mengatur basis data. basis datas.
5. Basis data

Komponen Sistem Basis Data kelas Perusahaan (*Enterprise-class Database System*)

Komponen dari *Enterprise-class Database System* terdiri dari:

1. User (pengguna)
Pengguna sistem basis data
2. *Database Application*
 - *Client/Server application*
 - ✓ Program aplikasi diinstal di client dan terkoneksi dengan database server
 - ✓ Dikembangkan dengan Visual Basic, C++, atau Java
 - *E-commerce*
 - ✓ Program berjalan pada web server
 - ✓ User menggunakannya melalui Web browser (Internet Explorer atau Firefox)
 - ✓ Web server yang sering digunakan: IIS atau Apache
 - ✓ Dikembangkan dengan PHP, Java, Microsoft .Net
 - *Reporting Application*

- ✓ Mempublikasikan hasil dari query pada basis data melalui portal atau web site suatu organisasi atau perusahaan
 - ✓ Dibuat menggunakan *third-party report generator* dan *digital dashboard product* seperti Cognos dan MicroStrategy
 - XML Web Services
 - ✓ Memanfaatkan teknologi XML sehingga memungkinkan komunikasi antar program aplikasi yang berbeda
 - ✓ Dikembangkan dengan Java atau Microsoft .Net (C## atau VB .Net)
3. Structured Query Language (SQL)
- Bahasa standar (secara internasional) yang dapat digunakan untuk melakukan query terhadap data pada semua sistem basis data
4. DBMS
- Memproses perintah SQL
 - Menyediakan fasilitas untuk membuat, memproses dan mengelola basis data
 - Contoh: Microsoft Access, Microsoft SQL Server, Oracle dan DB2
- Database application* mengambil dan menyimpan data dari basis data dengan mengirim perintah SQL ke DMBS
5. Basis data
- merupakan koleksi dari tabel-tabel yang terintegrasi

Daftar Pustaka

1. Connolly, Thomas M., Begg, Carolyn E., and Strachan, Anne D., Database System. A practical pproach to Design, Implementation, and Management, Addison Wesley Company, 1996.
2. David M. Kroenke, *Basis data Processing, Fundamentals, Design and Implementation*, 12nd Edition, Prentice-Hall Int'l Edition, 2012



MODUL PERKULIAHAN

Basis Data

Lingkup Basis Data

Fakultas
Ilmu Komputer

Program Studi
Sistem Informasi

Tatap Muka
02

Kode MK
87010

Disusun Oleh
Tim Dosen

Abstract

Modul ini mempelajari lingkup basis data.

Kompetensi

Mahasiswa mampu menjelaskan perbedaan arsitektur 3 level basis data, mampu mengidentifikasi komponen perangkat lunak dari DBMS dan menjelaskan fungsinya.

A. Pendahuluan

Seperti yang telah dijelaskan pada modul sebelumnya bahwa basis data merupakan jantung dari sebuah aplikasi, sehingga kedudukannya menjadi sangat penting dalam organisasi. Saat ini, pada suatu organisasi biasanya basis data yang digunakan adalah sumber informasi yang digunakan secara bersama oleh seluruh pengguna dalam organisasi tersebut. Setiap pengguna membutuhkan pandangan yang berbeda terhadap data yang disimpan pada basis data sesuai dengan tugas yang dikerjakannya. Untuk memenuhi hal tersebut diperlukan sebuah arsitektur DBMS. Arsitektur untuk DBMS komersial yang tersedia saat ini didasarkan pada arsitektur ANSI SPARC. Modul ini akan selain akan membahas tentang arsitektur basis data juga akan membahas tentang klasifikasi dalam model data, dan komponen perangkat lunak DBMS. Dengan demikian mahasiswa akan memiliki pengetahuan tentang ketiga hal tersebut yang merupakan elemen dalam lingkup basis data.

Kompetensi

Setelah mempelajari materi pokok bahasan disini, mahasiswa diharapkan:

- Mampu menjelaskan kembali perbedaan arsitektur 3 level basis data.
- Mampu menjelaskan perbedaan klasifikasi dalam model data.
- Mampu mengidentifikasi komponen perangkat lunak dari DBMS dan menjelaskan fungsinya.
- Mampu mengerjakan mengerjakan kasus-kasus yang terkait dengan materi lingkup basis data.

B. Arsitektur Tiga Level Basis Data

Tujuan utama dari sistem basis data adalah untuk menyediakan pandangan data abstrak untuk para penggunanya dengan menyembunyikan rincian bagaimana data disimpan dan dimanipulasi. Oleh sebab itu titik awal untuk merancang basis data adalah abstrak dan deskripsi umum tentang kebutuhan informasi organisasi yang akan disajikan dalam data basis data.

Basis data adalah sumber data yang digunakan secara bersama oleh seluruh pengguna dalam organisasi. Setiap pengguna yang berasal dari fungsi yang berbeda dalam suatu organisasi akan membutuhkan data yang berbeda untuk digunakan dalam mengerjakan tugasnya sehari-hari. Sehingga view (pandangan) terhadap data untuk setiap pengguna berbeda. Untuk memenuhi hal

ini terdapat arsitektur DBMS komersial yang tersedia saat ini dimana arsitektur ini didasarkan pada arsitektur ANSI SPARC yang terdiri dari 3 level arsitektur.

Adapun ke-tiga level arsitektur basis data ANSI PARC tersebut adalah:

1. Level Internal

Level internal merupakan representasi fisik dari basis data pada komputer. Level ini menggambarkan bagaimana data disimpan secara fisik pada basis data dan meliputi implementasi fisik dari database untuk mencapai kinerja *run time* dan penggunaan ruang penyimpanan yang optimal. Record disimpan dalam media penyimpanan dalam format byte.

Beberapa hal yang diperhatikan dalam level internal ini antara lain:

- Alokasi ruang penyimpanan data dan indeks
- Deskripsi record untuk penyimpanan (dengan ukuran penyimpanan untuk elemen data)
- Penempatan record
- Pemampatan data dan teknik enkripsi data

Di bawah level internal terdapat level fisik yang dikelola oleh sistem operasi di bawah pengarahan DBMS. Akan tetapi fungsi DBMS dan sistem operasi pada level fisik tidak jelas dan bervariasi dari sistem yang satu ke sistem yang lain. Beberapa manfaat yang diambil DBMS dari sistem operasi adalah metode akses.

2. Level Eksternal

Merupakan cara pandang pengguna terhadap basis data. Pengguna adalah *programmer*, *end user* atau DBA. Level eksternal menggambarkan bagian basis data yang relevan bagi seorang pengguna tertentu. Level eksternal terdiri dari sejumlah pandangan (*view*) yang berbeda dari sebuah basis data. Masing-masing pengguna merepresentasikan dalam bentuk yang sudah dikenalnya. Cara pandang secara eksternal hanya terbatas pada entitas, atribut dan hubungan antar entitas (*relationship*) yang diperlukan saja.

3. Level Konseptual

Level konseptual merupakan pandangan komunitas terhadap basis data. Pada tingkat ini digambarkan data apa yang disimpan dalam basis data dan hubungan antar datanya. Level konseptual ini menyajikan:

- Semua entitas dengan atribut-atribut dan relasinya.

- Batasan pada data.
- Informasi semantik tentang data.
- Keamanan dan integritas informasi.

Level ini mendukung setiap pandangan eksternal, yaitu setiap data apapun yang tersedia untuk pengguna harus terkandung dalam atau berasal dari level konseptual. Deskripsi data dari entitas pada tingkat ini hanya terdiri dari jenis data dan besarnya atribut tanpa memperhatikan besarnya penyimpanan dalam ukuran byte.

Schema, Mapping dan Instances

Schema

Gambaran keseluruhan dari database disebut dengan **database schema**. Ada 3 jenis schema yang berbeda dalam suatu database dan masing-masing didefinisikan menurut arsitektur 3 level abstraksi tersebut.

- Level tertinggi mempunyai beberapa **external schema** (juga subschema) yang berhubungan dengan view terhadap data yang berbeda-beda
- Pada conceptual level, kita mempunyai **conceptual schema**. Conceptual schema menggambarkan semua item data dan hubungan antara item data dengan batasan integritas. Hanya ada *satu conceptual schema per database*.
- Pada level terendah dari abstraksi, kita mempunyai **internal schema**. Internal schema merupakan deskripsi lengkap dari model internal, yang berisi definisi dari record yang disimpan, metoda representasi, data field, skema index dan skema hashing yang digunakan. Dan hanya ada satu internal schema.

Mapping

DBMS bertanggung jawab untuk memetakan ketiga jenis schema tersebut. Conceptual schema direlasikan ke internal schema melalui **conceptual /internal mapping**. Hal ini memungkinkan DBMS untuk menemukan record aktual atau kombinasi record dalam penyimpanan fisik yang menggantikan record logik pada conceptual schema bersama-sama dengan batasan-batasan yang dilakukan pada operasi untuk record logik tersebut. Setiap external schema direlasikan ke skema konseptual oleh **external / conceptual mapping**. Hal ini memungkinkan DBMS untuk memetakan nama dalam pandangan pemakai ke dalam bagian yang relevan dari conceptual schema.

Contoh dari level yang berbeda dapat dilihat pada gambar 2.2 di bawah ini. Terdapat 2 eksternal view, yaitu view yang pertama terdiri dari NPM, Nama, TglLahir, Alamat, dan view yang kedua terdiri dari NPM, Nama, Prodi. Kedua view ini digabung menjadi satu conceptual

view. Pada proses penggabungan perbedaan utamanya adalah pada field usia yang telah berubah menjadi TglLahir. DBMS memelihara external.conceptual mapping. Contohnya DBMS memetakan field NPM pada external view 1 ke field NPM pada conceptual record. Di level ini kita melihat definisi dari struktur bahasa tingkat tinggi yang berisi sebuah pointer, next yang memungkinkan daftar record MHS secara fisik dihubungkan ke form. Perhatikan bahwa urutan field pada level internal berbeda dengan level konseptual. DBMS memelihara pemetaan konseptual/internal.

External view 1

NPM	Nama	Usia	Alamat
-----	------	------	--------

External view 2

NPM	Nama	Prodi
-----	------	-------

Conceptual level

NPM	Nama	TglLahir	Alamat	Telpon	Prodi
-----	------	----------	--------	--------	-------

Internal level	<pre> struct MHS { int NPM; char Nama [25]; struct date Date_of_Birth; char Alamat [30] struct MHS *next; }; index NoBP; index ProgStudi </pre>	<pre> /* pointer ke record MHS berikutnya /* mendefinisikan indexuntuk MHS </pre>
-----------------------	--	---

Instance

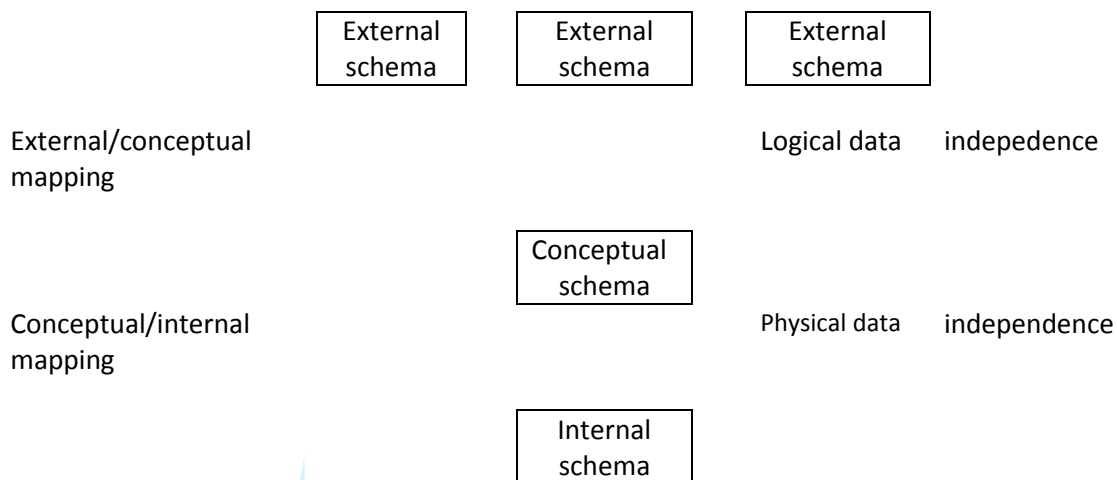
Database schema adalah deskripsi dari database. Ini ditentukan selama proses perancangan database diharapkan tidak sering berubah. Tetapi data aktual pada database bisa sering berubah karena proses insert atau update. Data pada database pada titik tertentu pada waktu tertentu disebut dengan **database instance**.

Data Independence

Tujuan utama dari tiga level arsitektur basis data adalah untuk menyediakan *data independence* dimana level diatasnya tidak berpengaruh oleh perubahan untuk level dibawahnya. Ada 2 jenis data independence , yaitu:

- *Logical data independence*,
Logical data independence menunjuk kepada kekebalan dari *external schema* untuk perubahan-perubahan dalam *conceptual schema*
- *Physical data independence*,
Physical data independence menunjuk kepada kekebalan dari *conceptual schema* untuk perubahan-perubahan dalam internal schema

Perubahan internal schema, seperti: penggunaan organisasi file atau struktur penyimpanan yang berbeda, penggunaan media penyimpanan yang berbeda, perubahan algoritma indeks atau hashing dimungkinkan tanpa harus mengganti/merubah conceptual atau external schema. Dari pandangan si pengguna efek yang dirasakan hanyalah perubahan dalam kinerja. Gambar 2.3 berikut ini menggambarkan setiap jenis data independence yang terjadi pada relasi dalam 3 level arsitektur.



C. Klasifikasi Dalam Model Data

Data model adalah sekumpulan konsep untuk menggambarkan data, hubungan antar data, dan batasan-batasan pada data dalam organisasi. Tujuan pemodelan data adalah untuk menyajikan data dan membuat data mudah dimengerti sehingga jika pemodelan ini dilakukan kita dapat dengan mudah merancang basis data.

Terdapat 3 kategori model data, yaitu: model data berbasis obyek, model data berbasis record, dan model data fisik. Disini, akan dibahas *model data berbasis obyek* dan *model data berbasis record*.

Model Data Berbasis Obyek

Model data ini menggunakan konsep seperti entitas, atribut, dan relasi.

- Entity adalah obyek yang dapat dibedakan dengan obyek lain dalam organisasi, contohnya orang, tempat, barang atau kejadian.
- Atribut adalah properti/karakteristik yang menggambarkan beberapa aspek dari obyek yang akan kita catat, contohnya untuk entitas mahasiswa, atributnya adalah NPM, nama, alamat, tempat lahir, tanggal lahir, dsb.
- Relasi adalah asosiasi antar entitas.

Beberapa jenis model data berbasis obyek yang umum, adalah:

- Hubungan entitas (Entity relationship)
- Semantik
- Fungsional
- Berbasis obyek

Model hubungan entitas muncul sebagai teknik utama untuk perancangan database konseptual. Model ini menjelaskan hubungan anatar data berdasarkan persepsi dunia nyata terdiri dari obyek-obyek yang mempunyai relasi antar obyek tersebut.

Model data semantik menyatakan hubungan antar obyek (dalam kata-kata). Contohnya seperti gambar berikut ini.

Model Data Berbasis Record

Pada model berbasis record, basis data terdiri dari sejumlah record dengan format tetap dari jenis-jenis yang berbeda. Setiap jenis record menentukan sejumlah field dimana masing-masing memiliki panjang tetap. Terdapat 3 jenis model data logik berbasis record, yaitu:

1. Model Database Jaringan (Network Database Model)
2. Model Database Hierarki (Hierarchical Database Model)
3. Model Database Relasi (Relational Database Model)

Berikut ini adalah penjelasan dari 3 jenis model data logik berbasis record tersebut.

1. Model Database Jaringan (Network Database Model)

Pada model database jaringan data disajikan sebagai kumpulan record dan relasi disajikan sebagai set-set. Dibandingkan dengan model relasional, relasinya secara jelas dimodelkan dengan set yang akan menjadi pointers (relasi) pada implementasinya. Record-record disusun sebagai struktur graf dengan record-record yang tampil sebagai *node* dan set sebagai *edge* pada graf. Model ini menyerupai model hirarki. Perbedaannya terdapat pada suatu simpul anak bisa memiliki lebih dari satu orang tua. Model ini bisa menyatakan hubungan 1:1 (satu orang tua punya satu anak), 1:M (satu orang tua punya banyak anak), maupun N:M (beberapa anak bisa mempunyai beberapa orangtua).

2. Model Hierarki (Hierarchical Database Model)

Model hierarki adalah suatu jenis terbatas dari model jaringan. Pada model ini data disajikan sebagai kumpulan record dan relasi disajikan sebagai set-set tetapi model ini memungkinkan sebuah node (anak) untuk hanya memiliki satu orang tua tetapi satu orang tua bisa memiliki beberapa anak. Model hierarki disajikan seperti model graf pohon (tree) dimana record-record ditampilkan sebagai node (anak) yang juga disebut sebagai segment dan set disebut edge.

3. Model Database Relasional (Relational Database Model)

Model database relasional berdasarkan konsep relasi matematika. Pada model relasional data dan relasi disajikan sebagai tabel dimana setiap tabel memiliki kolom dengan nama yang unik dan memiliki baris-baris dimana setiap baris menyimpan instance yang berbeda (lihat modul 01 tentang database relasional).

MAHASISWA

NPM	NamaMhsw	Jurusan	NoTelpon
07001	Husni Cook	Manajemen	08115577869
07002	Roni Parto	Akuntansi	08129992223
07003	Dono Pardede	Akuntansi	02178908764
07004	Dora Nakula	Sistem Informasi	02517778886
07005	Santi Arlisa	Sistem Informasi	08171414325

NILAI

NPM	KodeMatKul	Semester	Tahun	Nilai
07001	AKT100	Ganjil	2006	A
07001	TSI200	Ganjil	2006	B
07003	AKT100	Ganjil	2006	B
07003	TSI200	Ganjil	2006	C
07005	TSI240	Genap	2006	A
07005	AKT200	Genap	2006	C
07005	TSI300	Genap	2006	C

MATAKULIAH

KodeMatKul	NamaMatKul	KodeDosen
TSI200	Sistem Informasi Manajemen	D101
TSI240	SMBD	D220
TSI300	Jaringan Komputer	D224
AKT100	Dasar-dasar Akuntansi	D315
AKT200	Manajemen Keuangan	D315

D. Komponen Perangkat Lunak DBMS

Komponen perangkat lunak untuk DBMS antara lain adalah:

1. *Authorization control*. Modul ini memeriksa apakah pengguna memiliki otorisasi melakukan operasi yang dikehendaki.
2. *Command processor*. Begitu sistem sudah memeriksa bahwa pengguna memiliki otorisasi untuk melakukan operasi, kontrol akan dikirim ke prosesor perintah.
3. *Integrity checker*. Untuk operasi-operasi yang mengubah database, integrity checker memeriksa bahwa operasi yang diminta memenuhi semua batasan integritas (seperti key constraints).
4. *Query Optimizer*. Modul ini menentukan strategi optimal untuk eksekusi query.
5. *Transaction manager*. Modul ini melakukan pengolahan operasi yang diminta yang diterima dari transaksi.
6. *Scheduler*. Modul ini bertanggung jawab untuk memastikan konkurensi operasi pada database yang diminta tanpa bertentangan satu sama lain. Scheduler mengendalikan urutan relatif terhadap eksekusi dari operasi transaksi.
7. *Recovery manager*. Modul ini memastikan database tetap konsisten pada saat terjadi kegagalan. Recovery manager bertanggung jawab untuk menjalankan transaksi atau menggagalkan transaksi.
8. *Buffer manager*. Modul ini bertanggung jawab untuk mentransfer data antara memory utama dengan tempat penyimpanan sekunder, seperti disk dan tape. Recovery manager dan buffer manager disebut sebagai data manager.

Daftar Pustaka

3. Connolly, Thomas M., Begg, Carolyn E., and Strachan, Anne D., Database System. A practical pproach to Design, Implementation, and Management, Addison Wesley Company, 1996.
4. David M. Kroenke, *Basis data Processing, Fundamentals, Design and Implementation*, 12nd Edition, Prentice-Hall Int'l Edition, 2012



MODUL PERKULIAHAN

Basis Data

Model Relasional Basis Data

Fakultas

Ilmu Komputer

Program Studi

Sistem Informasi

Tatap Muka

03

Kode MK

87010

Disusun Oleh

Tim Dosen

Abstract

Modul ini mempelajari model relasional basis data.

Kompetensi

Mahasiswa mampu menjelaskan struktur data relasional, mampu menyebutkan dan menjelaskan kunci relasional, mampu mengidentifikasi komponen perangkat lunak dari DBMS dan menjelaskan fungsinya.

A. Pendahuluan

Ditemukan oleh E.F. Codd.

Model Data Relasional adalah suatu model basis data yang menggunakan tabel dua dimensi, yang terdiri atas baris dan kolom untuk menggambarkan sebuah berkas data.

Model ini menunjukkan cara mengelola/mengorganisasikan data secara fisik dalam memory sekunder, yang akan berdampak pula pada bagaimana kita mengelompokkan data dan membentuk keseluruhan data yang terkait dalam sistem yang kita buat.

Contoh Tabel dan keterhubungannya :

MHS

NPM	Nama	Alamat
10296832	Nurhayati	Jakarta
10296126	Astuti	Jakarta
31296500	Budi	Depok
41296525	Prananingrum	Bogor
50096487	Pipit	Bekasi
21196353	Quraish	Bogor

MKUL

KDMK	MTKULIAH	SKS
KK021	P. Basis Data	2
KD132	SIM	3
KU122	Pancasila	2

NILAI

NPM	KDMK	MID	FINAL
10296832	KK021	60	75
10296126	KD132	70	90
31296500	KK021	55	40
41296525	KU122	90	80
21196353	KU122	75	75
50095487	KD132	80	0
10296832	KD132	40	30

B. Keuntungan Model Data Relasional

1. Bentuknya sederhana
2. Mudah melakukan berbagai operasi data (query, update/edit, delete).

Contoh-contoh model basis data:

1. Model basis data hirarki
2. Model basis data network/jaringan
3. Model basis data relational (paling banyak digunakan)

Contoh DBMS yang mengelola basis data relational :

- dBase III+
- MS.Access
- Borland-Paradox
- Oracle
- DB2
- SYBASE
- Informix.

C. Contoh Pembuatan Tabel

MKUL

KDMK	MTKULIAH	SKS
KK021	P. Basis Data	2
KD132	SIM	3
KU122	Pancasila	2

Karakteristik dari tabel MKUL:

- data mata kuliah memiliki 3 buah kolom data
- kolom 1 berisi data string/alphanumeric dengan lebar tetap, yaitu 5 digit/char.
- Kolom 2 berisi data string dengan lebar maksimum 30 digit.
- Kolom 3 berisi data integer dengan lebar maksimum 1 digit.

Dari karakteristik di atas, kita bisa menetapkan struktur data tabel MKUL:

- nama kolom/field.
- Tipe data.

- Lebar (banyaknya dgiti maksimum yang bisa ditampung).

Jadi, struktur tabel MKUL :

Nama Kolom	Tipe	Lebar
KDMK	Char	5
MTKULIAH	Char	30
SKS	numerik	1

D. Istilah dalam Model Data Relasional :

Relasi:

Sebuah tabel yang terdiri dari beberapa kolom dan beberapa baris.

Atribut:

Kolom pada sebuah relasi (field).

Tupel

Baris pada sebuah relasi (record).

Domain

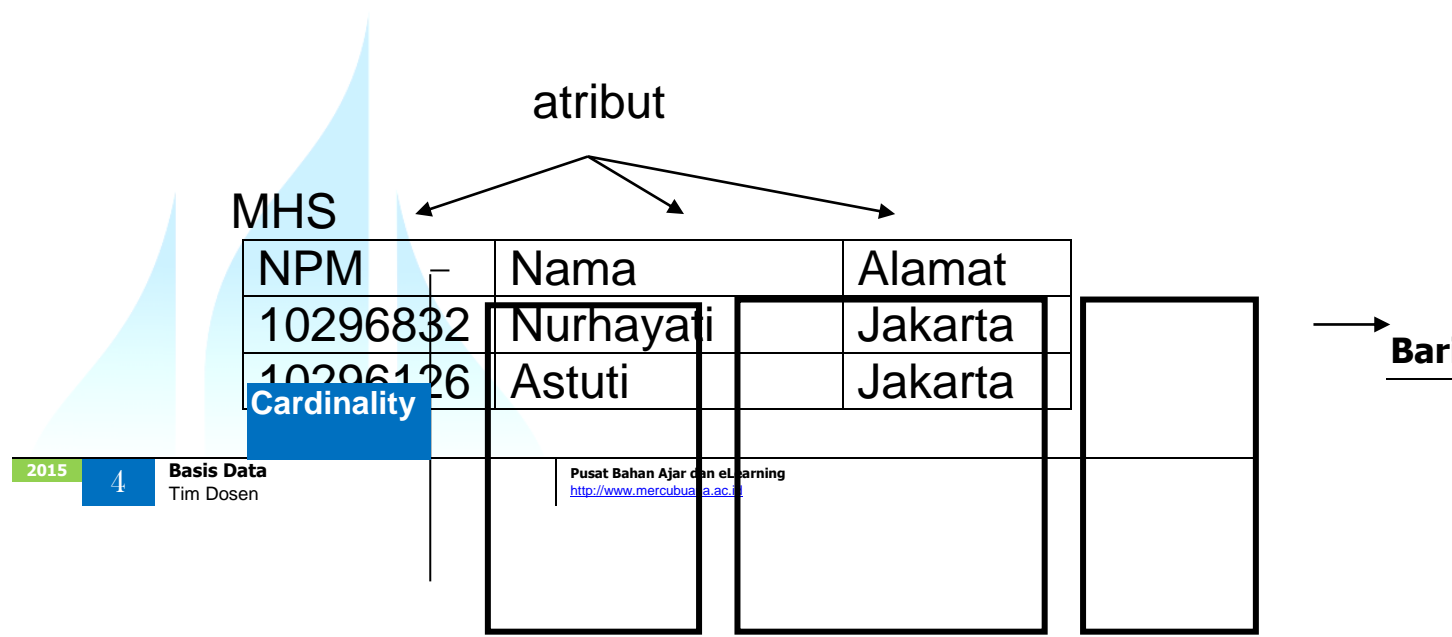
Kumpulan nilai yang valid untuk satu atau lebih atribut

Derajat (degree)

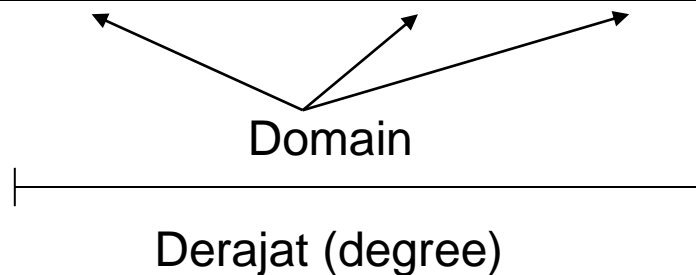
Jumlah atribut dalam sebuah relasi (jumlah field)

Cardinality

Jumlah tupel dalam sebuah relasi (jumlah record)



31296500	Budi	Depok
41296525	Prananingrum	Bogor
50096487	Pipit	Bekasi
21196353	Quraish	Bogor



E. Relational Key

Super key

Satu atribut/kumpulan atribut yang secara unik mengidentifikasi sebuah tupel di dalam relasi (*satu atau lebih field yang dapat dipilih untuk membedakan antara 1 record dengan record lainnya*).

Contoh: Untuk tabel MHS di atas, super key-nya:

- NPM
- NAMA (dengan syarat tidak ada nama yang sama)
- ALAMAT (dengan syarat tidak ada alamat yang sama)
- NPM + NAMA
- NPM + ALAMAT
- NAMA + ALAMAT
- NPM + NAMA + ALAMAT

Candidate key

Atribut di dalam relasi yang biasanya mempunyai nilai *unik* (*super key dengan jumlah field yang paling sedikit*)

Maka, candidate key-nya adalah NPM, NAMA dan ALAMAT (karena hanya terdiri dari 1 field saja)

Primary key

Candidate key yang dipilih untuk mengidentifikasi tupel secara unik dalam relasi

Maka, primary key yang dipilih adalah NPM (unik, tidak ada NPM yang sama).

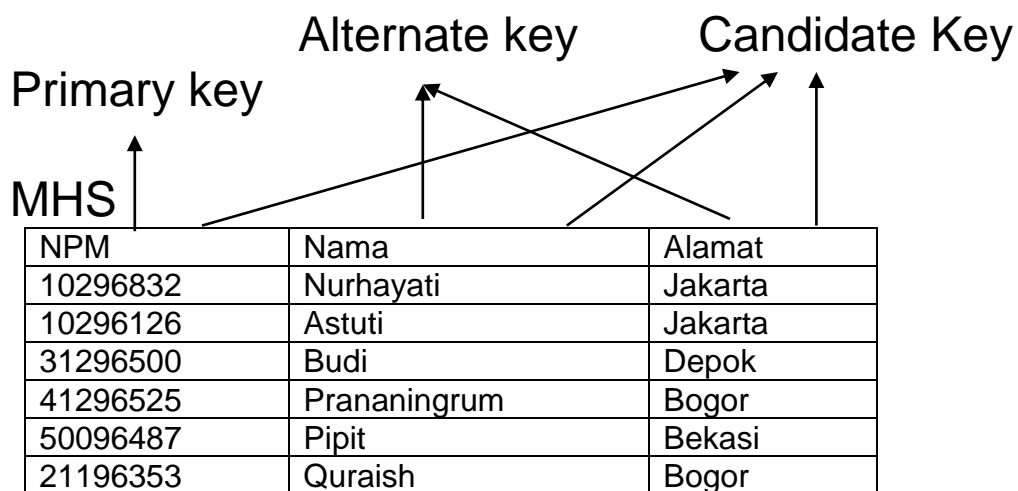
Alternate key

Candidate key yang tidak dipilih sebagai primary key

Maka, candidate key-nya NAMA dan ALAMAT

Foreign key

Atribut dengan domain yang sama yang menjadi kunci utama pada sebuah relasi tetapi pada relasi lain atribut tersebut hanya sebagai atribut biasa



F. Relational Integrity Rules

1. Null
Nilai suatu atribut yang tidak diketahui dan tidak cocok untuk baris (tuple) tersebut
2. Entity Integrity
Tidak ada satu komponen primary key yang bernilai null.
3. Referential Integrity

Suatu domain dapat dipakai sebagai kunci primer bila merupakan atribut tunggal pada domain yang bersangkutan.

G. Bahasa Pada Model Data Relasional

Menggunakan bahasa query, yaitu pernyataan yang diajukan untuk mengambil informasi

Terbagi 2 :

1. Bahasa Query Formal

Bahasa query yang diterjemahkan dengan menggunakan simbol-simbol matematis.

Terbagi 2, yaitu:

- a. Prosedural, yaitu pemakai memberi spesifikasi data apa yang dibutuhkan dan bagaimana cara mendapatkannya.

Contoh:

Aljabar Relasional, yaitu dimana query diekspresikan dengan cara menerapkan operator tertentu terhadap suatu tabel / relasi.

- b. Non Prosedural, yaitu pemakai menspesifikasikan data apa yang dibutuhkan tanpa menspesifikasikan bagaimana untuk mendapatkannya.

Contoh:

Kalkulus Relasional, dimana query menjelaskan set tuple yang diinginkan dengan cara menjelaskan predikat tuple yang diharapkan.

Terbagi 2 :

1. Kalkulus Relasional Tupel
2. Kalkulus Relasional Domain

2. Bahasa Query Komersial

Bahasa Query yang dirancang sendiri oleh programmer menjadi suatu program aplikasi agar pemakai lebih mudah menggunakannya (user friendly).

Contoh :

- QUEL

Berbasis pada bahasa kalkulus relasional

- QBE
Berbasis pada bahasa kalkulus relasional
- SQL
Berbasis pada bahasa kalkulus relasional dan aljabar relasional

Daftar Pustaka

5. Connolly, Thomas M., Begg, Carolyn E., and Strachan, Anne D., Database System. A practical pproach to Design, Implementation, and Management, Addison Wesley Company, 1996.
6. David M. Kroenke, *Basis data Processing, Fundamentals, Design and Implementation*, 12nd Edition, Prentice-Hall Int'l Edition, 2012



MODUL PERKULIAHAN

Basis Data

SQL – Data Manipulation Language Query Single Table

Fakultas
Ilmu Komputer

Program Studi
Sistem Informasi

Tatap Muka

04

Kode MK
87010

Disusun Oleh
Tim Dosen

Abstract

Modul ini mempelajari SQL Data manipulation language query single table.

Kompetensi

Mahasiswa mampu menulis perintah SQL select dengan menggunakan klausa where, order by untuk melakukan pengurutan, group by untuk mengelompokkan hasil query, menggunakan fungsi agregat.

A. Pendahuluan

STRUCTURE QUERY LANGUAGE (SQL)

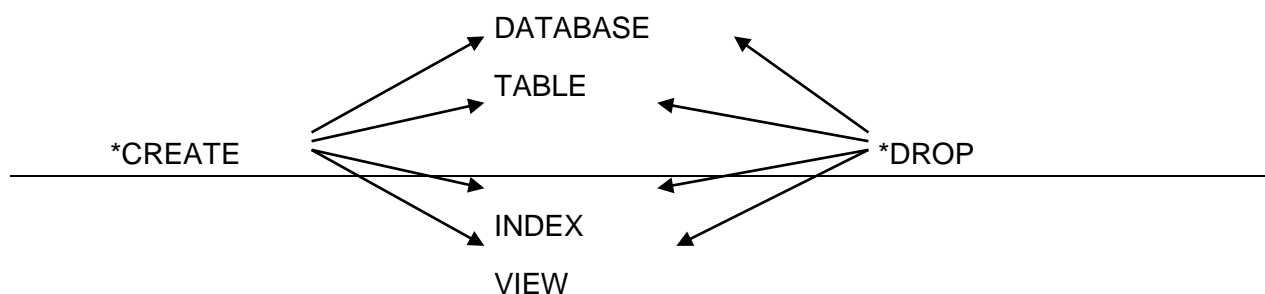
SQL dapat diterapkan pada beberapa software diantaranya adalah : Dbase IV, Informix, AS-400, Access .

Jenis SQL ada 2 macam :

1. Interactive : langsung dapat dioperasikan.
2. Embedded : disisipkan ke dalam sebuah program (Cobol, C, Fortran).

B. Pengelompokan Statement SQL

1. Data Definition Language (DDL)



*ALTER TABLE

2. Data Manipulation Language (DML)
📖 INSERT, SELECT, UPDATE, DELETE
3. Data Access
🔑 GRANT, REVOKE
4. Data Integrity

🔑 RECOVER TABLE

5. Auxiliary
🔑 UNLOAD, LOAD, RENAME COLUMN

C. Kasus DDL

📖 Pembuatan Database

Sintaks : CREATE DATABASE nama_db ;

Contoh: CREATE DATABASE latihan ;
(membuat database dengan nama latihan)

📖 Pembuatan Tabel

Sintaks : CREATE TABLE nama_tabel
(nama_kolom1 type_kolom1,
nama_kolom2 type_kolom2, ...)

Contoh:

➔ Struktur database

- MHS (NPM char(8),NAMA char(25),ALAMAT char(30))
- MKUL (KDMK char(5),MTKUL char(25),SKS smallint)
- NILAI (NPM char(8),KDMK char(5),MID smallint,FINAL smallint)

➔ Membuat table

- CREATE TABLE MHS
(NPM char(8) notnull, NAMA char(25) notnull, ALAMAT char(30) notnull);

- CREATE TABLE MKUL
(KDMK char(5) notnull, MTKULIAH char(25) notnull, SKS smallint notnull);
- CREATE TABLE NILAI
(NPM char(8) notnull, KDMK char(5) notnull, MID smallint, FINAL smallint);

Pembuatan Index

Sintaks :

```
CREATE [UNIQUE] INDEX nama_index
ON nama_tabel (nama_kolom);
```

Contoh:

- 1). Buat index dengan nama MHSIN berdasarkan NPM
dari tabel MHS !

```
CREATE UNIQUE INDEX MHSIN
ON MHS(NPM);
```

Hasil :

MHSIN

<i>NPM</i>	<i>NAMA</i>	<i>ALAMAT</i>
<i>10296126</i>	<i>Astuti</i>	<i>Jakarta</i>
<i>10296832</i>	<i>Nurhayati</i>	<i>Jakarta</i>
<i>21196353</i>	<i>Quraish</i>	<i>Bogor</i>
<i>31296500</i>	<i>Budi</i>	<i>Depok</i>
<i>41296525</i>	<i>Prananingrum</i>	<i>Bogor</i>

50096487	Pipit	Bekasi
-----------------	--------------	---------------

2). CREATE INDEX NILAIIN
 ON NILAI(KDMK);

Pembuatan View

Sintaks : **CREATE VIEW** nama_view
 [(nama_kolom1, ..., ...)] AS SELECT statement [WITH CHECK
 OPTION];

Contoh:

1). Buat view dengan nama MHSVIEW yang berisi
semua data mahasiswa !

CREATE VIEW MHSVIEW AS SELECT *
FROM MHS;

2). CREATE VIEW NILVIEW(NPM, KDMK, MID)
AS SELECT NPM, KDMK, MID
FROM NILAI;

Menghapus Database / Tabel / Index / View

Sintaks : **DROP DATABASE** nama_db
 DROP TABLE nama_tabel
 DROP INDEX nama_index

DROP VIEW nama_view

Contoh : - menghapus tabel MHS :

DROP TABLE MHS;

Merubah Struktur Tabel

Sintaks : **ALTER TABLE** nama_tabel
ADD (nama_kolom type_kolom
[BEFORE nama_kolom])
MODIFY (nama_kolom type_kolom)
DROP (nama_kolom type_kolom);

Contoh :

1). Tambahkan kolom JKEL pada tabel MHS

ALTER TABLE MHS ADD(JKEL char(1));

Hasil :

MHS

NPM	NAMA	ALAMAT	JKEL
10296832	Nurhayati	Jakarta	
10296126	Astuti	Jakarta	
31296500	Budi	Depok	
41296525	Prananingrum	Bogor	
50096487	Pipit	Bekasi	

21196353	Quraish	Bogor	
----------	---------	-------	--

2).Ubah panjang kolom MTKULIAH yang ada pada tabel MKUL !

```
ALTER TABLE MKUL
MODIFY(MTKULIAH char(30));
```

3). Hapus kolom JKEL dari tabel MHS !

```
ALTER TABLE MHS DROP(JKEL char(1));
```

→ Contoh Data

MHS

NPM	NAMA	ALAMAT
10296832	Nurhayati	Jakarta
10296126	Astuti	Jakarta
31296500	Budi	Depok
41296525	Prananingrum	Bogor
50096487	Pipit	Bekasi
21196353	Quraish	Bogor

MKUL

KDMK	MTKULIAH	SKS
KK021	P. Basis Data	2
KD132	SIM	3
KU122	Pancasila	2

NILAI

NPM	KDMK	MID	FINAL
10296832	KK021	60	75
10296126	KD132	70	90
31296500	KK021	55	40
41296525	KU122	90	80
21196353	KU122	75	75
50095487	KD132	80	0
10296832	KD132	40	30

Daftar Pustaka

7. Connolly, Thomas M., Begg, Carolyn E., and Strachan, Anne D., Database System. A practical approach to Design, Implementation, and Management, Addison Wesley Company, 1996.
8. David M. Kroenke, *Basis data Processing, Fundamentals, Design and Implementation*, 12nd Edition, Prentice-Hall Int'l Edition, 2012



MODUL PERKULIAHAN

Basis Data

SQL – Data Manipulation Language Multi Table

Fakultas
Ilmu Komputer

Program Studi
Sistem Informasi

Tatap Muka
05

Kode MK
87010

Disusun Oleh
Tim Dosen

Abstract

Modul ini mempelajari SQL Data manipulation language query multi table.

Kompetensi

Mahasiswa mampu menulis perintah SQL select dengan subquery dan join table dan menggunakan klausa union, intersect dan except dalam query, mampu melakukan pengubahan data dalam tabel basis data dengan menggunakan perintah insert, update dan delete.

A. Definisi

Data Manipulation Language (DML)

Digunakan untuk memanipulasi data dengan menggunakan perintah : *select, insert, update, delete*.

Data Manipulation Language merupakan bagian terpadu bahasa SQL. Perintah-perintahnya dapat dibuat secara interaktif atau ditempelkan pada sebuah program aplikasi. Pemakai hanya perlu menentukan 'APA' yang ia inginkan, DBMS menentukan 'BAGAIMANA' cara mendapatkannya.

Perintah – perintah yang digunakan dalam DML yaitu :

INSERT

Sintaks : INSERT INTO
 nama_tabel [(nama_kolom1, ...)]
 VALUES (data1, ...);

Contoh :

1). Masukkan data pada tabel MKUL !

```
INSERT INTO MKUL VALUES ("KK222",  
"Berkas Akses", 2);
```

Hasil :

MKUL

<i>KDMK</i>	MTKULIAH	SKS
-------------	----------	-----

KK021	P. Basis Data	2
KD132	SIM	3
KU122	Pancasila	2
KK222	Berkas Akses	2

2). INSERT INTO NILAI(NPM,KDMK,MID) VALUES
("32296222","KK222",30);

UPDATE

Sintaks : UPDATE nama_tabel
 SET nama_kolom = ekspresi
 WHERE kondisi;

Contoh :

1).Ubah alamat mahasiswa yang memiliki NPM =

"50096487" !

UPDATE MHS SET ALAMAT="Depok"
WHERE NPM="50096487";

Hasil :

MHS

NPM	NAMA	ALAMAT
50096487	Pipit	Depok

2). UPDATE NILAI SET MID=MID+10
WHERE KDMK="KK021";

DELETE

Sintaks: DELETE FROM nama_tabel
WHERE kondisi

Contoh :

- Hapus nilai mahasiswa yang mempunyai NPM="10296832" dan KDMK="KK021" !

DELETE FROM NILAI WHERE
NPM="10296832" AND KDMK="KK021";

Hasil :

NILAI

NPM	KDMK	MID	FINAL
10296126	KD132	70	90

31296500	KK021	55	40
41296525	KU122	90	80
21196353	KU122	75	75
50095487	KD132	80	0
10296832	KD132	40	30

SELECT

Sintaks: SELECT [DISTINCT] nama_kolom
 FROM nama_tabel
 [WHERE kondisi]
 [GROUP BY nama_kolom]
 [HAVING kondisi]
 [ORDER BY nama_kolom [ASC/DESC]]

Contoh satu tabel (Simple Query) :

1). Menampilkan data.

☉ Tampilkan semua data mahasiswa !

SELECT * FROM MHS;

Hasil :

MHS

NPM	NAMA	ALAMAT
10296832	Nurhayati	Jakarta
10296126	Astuti	Jakarta
31296500	Budi	Depok
41296525	Prananingrum	Bogor
50096487	Pipit	Bekasi
21196353	Quraish	Bogor

☛ SELECT NPM,NAMA,ALAMAT FROM MHS;

2). Mengambil data dari suatu tabel dengan satu/banyak kondisi :

☛ Tampilkan mata kuliah yang memiliki SKS = 2 !

```
SELECT MTKULIAH FROM MKUL
WHERE SKS = 2;
```

Hasil :

MKUL

MTKULIAH
P. Basis Data
Pancasila

- `SELECT * FROM NILAI WHERE MID >=60 OR FINAL > 75;`
- `SELECT NPM, KDMK, MID FROM NILAI WHERE MID BETWEEN 70 AND 100;`

3). Mengambil data dari suatu tabel

dengan menggunakan perintah LIKE:

- Tampilkan nama mahasiswa yang diawali dengan huruf "P" !

```
SELECT NAMA FROM MHS  
WHERE NAMA LIKE "P%";
```

Hasil :

MHS

NAMA
Prananingrum
Pipit

- ☯ SELECT NAMA FROM MHS WHERE NAMA NOT LIKE "%a%";
- ☯ SELECT NAMA FROM MHS WHERE NAMA LIKE "_u";

4). Mengambil data pada suatu tabel dengan hanya menampilkan satu kali saja data yang sama :

- ☯ Tampilkan alamat mahasiswa, dimana alamat yang sama hanya ditampilkan satu kali saja !

SELECT DISTINCT ALAMAT FROM MHS;

5). Memilih beberapa / semua data dari suatu tabel untuk diurutkan / dikelompokkan :

- ☯ Tampilkan semua data dari tabel MHS, dengan nama terurut dari "Z" ke "A" !

SELECT * FROM MHS ORDER BY NAMA DESC;

Hasil :

MHS

NPM	NAMA	ALAMAT
21196353	Quraish	Bogor

41296525	Prananingrum	Bogor
50096487	Pipit	Bekasi
10296832	Nurhayati	Jakarta
31296500	Budi	Depok
10296126	Astuti	Jakarta

- ☉ Tampilkan alamat mahasiswa dan jumlah masiswa yang bertempat tinggal di alamat tersebut !

```
SELECT ALAMAT, COUNT(*) FROM MHS
GROUP BY ALAMAT;
```

- ☉ Tampilkan alamat dan jumlah masiswa yang bertempat tinggal pada alamat yang jumlahnya lebih dari satu !

```
SELECT ALAMAT, COUNT(*) FROM MHS
GROUP BY ALAMAT
HAVING COUNT(*) > 1;
```

6). Penggunaan Agregate Function.

- ☉ Tampilkan data tertinggi dan terendah dari nilai Midtest pada KDMK = "KD132" !

```
SELECT MAX(MID), MIN(MID) FROM NILAI  
WHERE KDMK="KD132";
```

Hasil :

NILAI

MID
80
40

- ☉ Tampilkan rata-rata nilai final test dan jumlah nilai final test dengan KDMK = "KD132" !

```
SELECT AVG(FINAL), SUM(FINAL) FROM NILAI  
WHERE KDMK="KD132";
```

Hasil : 40 120

NILAI

KDMK	FINAL
KD132	90
KD132	0

Daftar Pustaka

9. Connolly, Thomas M., Begg, Carolyn E., and Strachan, Anne D., Database System. A practical pproach to Design, Implementation, and Management, Addison Wesley Company, 1996.
10. David M. Kroenke, *Basis data Processing, Fundamentals, Design and Implementation*, 12nd Edition, Prentice-Hall Int'l Edition, 2012



MODUL PERKULIAHAN

Basis Data

SQL – Data Definition Language (Create & Alter)

Fakultas
Ilmu Komputer

Program Studi
Sistem Informasi

Tatap Muka
06

Kode MK
87010

Disusun Oleh
Tim Dosen

Abstract

Modul ini mempelajari SQL Data definition language pernyataan create & alter.

Kompetensi

Mahasiswa mampu mengidentifikasi tipe data dalam SQL dengan menggunakannya dalam mendefinisikan struktur tabel.

A. Pendahuluan

Contoh lebih dari satu tabel (Sub Query dan Join) :

- 1). Tampilkan nama mahasiswa yang mempunyai nilai midtest lebih kecil dari 60 !

a. **Sub Query**

☉ SELECT NAMA FROM MHS WHERE NPM IN
(SELECT NPM FROM NILAI WHERE MID <= 60);

b. **JOIN** :

☉ SELECT NAMA FROM MHS, NILAI WHERE
MHS.NPM = NILAI.NPM AND NILAI.MID <= 60;

Hasil :

MHS

NAMA
Nurhayati
Budi
Quraish

2). **a. Sub Query :**

```
☉ SELECT NAMA FROM MHS WHERE NPM
    IN
    (SELECT NPM FROM NILAI WHERE
    KDMK IN
    (SELECT KDMK FROM MKUL WHERE
    MTKULIAH = "SIM");
```

b. JOIN :

```
☉ SELECT NAMA FROM MHS, NILAI, MKUL
    WHERE MKUL.MTKULIAH="SIM" AND
    NILAI.KDMK = MKUL.KDMK AND
    MHS.NPM = NILAI.NPM;
```

3). Penggunaan EXISTS / NOT EXISTS

☉ Tampilkan nama mahasiswa yang tidak mengambil KDMK = "KK021" !

```
SELECT NAMA FROM MHS
    WHERE NOT EXISTS
    (SELECT * FROM NILAI
    WHERE NILAI.NPM = MHS.NPM AND
    KDMK = "KK021" );
```

Hasil :

MHS

NAMA
Astuti
Prananingrum
Pipit
Quraish

4). Penggunaan **UNION**

- ☛ Tampilkan NPM mahasiswa yang bernama Budi dan yang memiliki nilai final > 75 !

```
SELECT NPM FROM MHS
WHERE NAMA = "Budi" UNION
SELECT NPM FROM NILAI
WHERE FINAL > 75;
```

B. Kasus Data Access

▼ **GRANT**

Sintaks: - GRANT hak_akses ON nama_db

TO nama_pemakai
[WITH GRANT OPTION]
[AS GRANTOR];

- GRANT hak_akses ON nama_tabel
TO nama_pemakai
[WITH GRANT OPTION]
[AS GRANTOR]

Contoh : - Berikan hak akses kepada Avi untuk
menampilkan nilai final test !

GRANT SELECT(FINAL)
ON NILAI TO AVI;

▼ **REVOKE**

Sintaks: - REVOKE hak_akses ON nama_db
FROM nama_pemakai;

- REVOKE hak_akses ON nama_tabel
FROM nama_pemakai;

Contoh : - Tarik kembali hak akses untuk
 menampilkan nilai final test dari Avi !

```
REVOKE SELECT(FINAL)
ON NILAI FROM AVI;
```

C. Kasus Statement Auxiliary

UNLOAD

Sintaks: UNLOAD TO “nama_path”
 [DELIMITER “char_pemisah”]
 SELECT statement;

Contoh : - Merubah semua data mahasiswa ke
 bentuk ASCII dan disimpan ke file teks di
 directory /home/avi :

```
UNLOAD TO “/home/avi/teks”
DELIMITER “|” SELECT * FROM MHS;
```

LOAD

Sintaks: LOAD FROM “nama_path”
 DELIMITER “char_pemisah”
 INSERT INTO
 nama_tabel [nama_kolom];

Contoh : - Merubah file teks ke tabel MHS_2 di
 directory /home/avi :

```
LOAD FROM "/home/avi/teks"  
DELIMITER "|" INSERT INTO MHS_2;
```

RENAME

Sintaks: **RENAME COLUMN** nama_kolom_lama
 TO Nama_kolom_baru;

Contoh : - Mengganti kolom ALAMAT yang ada
 pada tabel MHS menjadi KOTA :
 RENAME COLUMN MHS.ALAMAT TO KOTA;

Daftar Pustaka

11. Connolly, Thomas M., Begg, Carolyn E., and Strachan, Anne D., Database System. A practical pproach to Design, Implementation, and Management, Addison Wesley Company, 1996.
12. David M. Kroenke, *Basis data Processing, Fundamentals, Design and Implementation*, 12nd Edition, Prentice-Hall Int'l Edition, 2012



MODUL PERKULIAHAN

Basis Data

SQL – Data Definition Language (View)

Fakultas
Ilmu Komputer

Program Studi
Sistem Informasi

Tatap Muka

7

Kode MK
87010

Disusun Oleh
Tim Dosen

Abstract

Modul ini mempelajari SQL Data definition language pernyataan view.

Kompetensi

Mahasiswa mampu menjelaskan penggunaan dari view, mampu mengidentifikasi dan menyebutkan keuntungan dan kerugian dari penggunaan view serta mampu membuat dan menghapus view dengan perintah sql

A. Membuat View (CREATE VIEW)

1. Membuat view untuk suplier yang statusnya lebih besar dari 15

```
CREATE VIEW GOOD_SUPPLIERS  
AS SELECT Sn, Status, City FROM S  
WHERE Status > 15;
```

2. Membuat view yang berisi supplier yang tinggal di Paris

```
CREATE VIEW Paris_Suppliers  
AS SELECT * FROM Supliers  
WHERE City = ' Paris '
```

3. Membuat view dengan mengganti nama_atributnya

```
CREATE VIEW Parts (PNum, Part_Name, WT)  
AS SELECT P#, Pname, Weight FROM Part  
WHERE COLOR = 'Red'
```

B. Fungsi Perhitungan

COUNT : jumlah baris dan kolom

SUM : jumlah nilai dalam kolom

AVG : rata - rata nilai dalam kolom

MAX : nilai terbesar dalam kolom

MIN : nilai terkecil dalam kolom

Untuk SUM dan AVG nilainya harus numerik (INT, SMALLINT, FLOAT). Fungsi-fungsi tsb jika dikenakan pada nilai yang NULL maka nilainya akan diabaikan kecuali untuk COUNT(*)

1. Menghitung jumlah supplier

```
SELECT COUNT(*) FROM S
```

atau

```
SELECT COUNT (Sn) FROM S
```

2. Menampilkan nomor part dan total kuantitas pengiriman dari setiap part

```
SELECT Pn, SUM(QTY) FROM SP
```

```
GROUP BY Pn
```

3. Menghitung jumlah kuantitas dari P2 yang telah disupply

```
SELECT SUM (QTY) FROM SP WHERE Pn = 'P2'
```

4. Menampilkan jumlah pengiriman barang dengan nomor P4 dan dipasok oleh nomor supplier S1

```
SELECT COUNT(*) FROM SP  
  
WHERE Pn = 'P4' AND Sn = 'S1'
```

5. Menampilkan nomor part dan total kuantitas dari masing-masing part

```
SELECT Pn, SUM(QTY) FROM SP  
  
GROUP BY P3
```

C. DATA CONTROL LANGUAGE

1. GRANT

Fungsi : digunakan untuk memberikan izin akses kepada user

Sintaks : GRANT privileges ON tname TO user

Contoh :

```
GRANT SELECT ON CLUB TO PUBLIC
```

```
GRANT SELECT, INSERT, UPDATE, DELETE ON CLUB TO USER01
```

2. REVOKE

Fungsi : digunakan untuk mencabut izin akses kepada user

Sintaks : REVOKE privileges ON tbname FROM user

Contoh :

REVOKE INSERT, UPDATE, DELETE ON CLUB FROM USER01

REVOKE ALL ON CLUB FROM PUBLIC

Daftar Pustaka

13. Connolly, Thomas M., Begg, Carolyn E., and Strachan, Anne D., Database System. A practical pproach to Design, Implementation, and Management, Addison Wesley Company, 1996.
14. David M. Kroenke, *Basis data Processing, Fundamentals, Design and Implementation*, 12nd Edition, Prentice-Hall Int'l Edition, 2012



MODUL PERKULIAHAN

Basis Data

Pemodelan Entitas Relasi

Fakultas
Ilmu Komputer

Program Studi
Sistem Informasi

Tatap Muka

09

Kode MK
87010

Disusun Oleh
Tim Dosen

Abstract

Modul ini mempelajari pemodelan entitas relasi.

Kompetensi

Mahasiswa mampu menjelaskan konsep dasar tentang model entitas dan hubungannya beserta atribut yang terdapat dalam entitas

A. Model Data

Model Data sendiri dapat didefinisikan sebagai kumpulan perangkat konseptual untuk menggambarkan data, hubungan data, semantik (makna) data dan batasan data. Oleh karena yang ingin ditunjukkan adalah makna dari data dan keterhubungannya dengan data lain, maka Model Data ini lebih tepat jika disebut Model Data Logik.

Model Keterhubungan Entitas (*Entity-Relationship Models*), yang merupakan Model Data yang paling populer digunakan dalam perancangan basis data.

B. Model Entity-Relationship (Model Keterhubungan-Entitas)

Pemakaian istilah 'Model Keterhubungan-Entitas' dalam bahasa Indonesia dapat digunakan sebagai padanan dari istilah asing : *Entity-Relationship Model (E-R Model)*.

Istilah Model Entity-Relationship telah demikian populer/umum digunakan dalam berbagai pembahasan tentang analisis/perancangan Basis Data.

Pada Model Entity-Relationship, semesta data yang ada diterjemahkan atau ditransformasikan dengan memanfaatkan sejumlah perangkat konseptual menjadi sebuah diagram data, yang umum disebut sebagai Diagram Entity-Relationship (Diagram E-R). Sebelum kita membahas lebih jauh tentang bagaimana Diagram E-R tersebut dapat kita gambarkan, maka yang harus lebih dulu diketahui adalah komponen-komponen pembentuk Model Entity-Relationship. Sesuai namanya, ada 2 (dua) komponen utama pembentuk Model Entity-Relationship, yaitu:

- ✓ Entitas (*Entity*)
- ✓ Relasi (*Relation*)

C. Entitas (*Entity*) dan Himpunan Entitas (*Entitas Set*)

Entitas merupakan individu yang mewakili sesuatu yang nyata (eksistensinya) dan dapat dibedakan dari sesuatu yang lain. Sebuah kursi yang kita duduki, seseorang yang menjadi pegawai di sebuah perusahaan dan sebuah mobil yang melintas di depan kita adalah Entitas. Sekelompok Entitas yang sejenis dan berada dalam lingkup yang sama membentuk sebuah Himpunan Entitas (*Entity Set*).

Sederhananya, Entitas menunjuk pada individu suatu objek, sedang Himpunan Entitas menunjuk pada rumpun (*famili*) dari individu tersebut. Seseorang memang dapat menjadi sebuah entitas, tapi dapat berada pada Himpunan Entitas yang berbeda dengan seseorang yang lain. Seorang pasien, misalnya, akan dimasukkan

dalam Himpunan Entitas Pasien, sedang seorang dokter akan ditempatkan dalam Himpunan Entitas Dokter.

Dalam berbagai pembahasan/literatur, penyebutan Himpunan Entitas (yang kurang praktis) ini seringkali digantikan dengan sebutan Entitas saja. Karena itu sering kita temui, penggunaan istilah Entitas (*Entity*) di sebuah literatur sebenarnya menunjuk pada Himpunan Entitas.

Contoh-contoh Himpunan Entitas:

- ✓ Semua Pelanggan, atau Pelanggan saja.
dengan entitas Budiman, Suherman, Aminah, dan seterusnya.
- ✓ Semua Mobil, atau Mobil saja.
dengan entitas mobil Suzuki, mobil Toyota, mobil Honda, dan lain-lain.
- ✓ Semua Mahasiswa, atau Mahasiswa saja.
dengan entitas Ali, Budi, Iman, dan seterusnya.

NIM	Nama_Mhs	Alamat_Mhs	Tgl_Lahir
980001	Ali Akbar	Jl.Merdeka No.10, Jakarta	02-01-1979
980002	Budi Haryanto	Jl.Gajah Mada No.2c, Jakarta	06-10-1978
980003	Iman Faisal	Komp. Taman Anggrek Jakarta	13-05-1979
980004	Indah Susanti	Jl. Tengku Rian, Jakarta	21-06-1979

Gambar Himpunan Entitas mahasiswa

D. Relasi (*Relationship*) dan Himpunan Relasi (*Relationship Sets*)

Relasi menunjukkan adanya hubungan di antara sejumlah entitas yang berasal dari himpunan entitas yang berbeda. Misalnya, entitas seorang mahasiswa dengan *NIM*='980001' dan *Nama_Mhs*='Ali Akbar' (yang ada di himpunan entitas Mahasiswa) mempunyai relasi dengan entitas sebuah mata kuliah dengan *Kode_Kul*=IF-110' dan *Nama_Kul*='Struktur Data'. Relasi di antara kedua entitas tadi mengandung arti bahwa mahasiswa tersebut sedang mengambil/mempelajari mata kuliah tersebut di sebuah perguruan tinggi yang kita tinjau.

Kumpulan semua relasi di antara entitas-entitas yang terdapat pada himpunan entitas-himpunan entitas tersebut membentuk Himpunan Relasi (*Relationship Sets*). Sebagaimana istilah Himpunan Entitas yang banyak sekali disingkat menjadi Entitas (walaupun sebenarnya memiliki perbedaan makna), istilah Himpunan Relasi jarang sekali digunakan dan lebih sering disingkat dengan istilah Relasi saja.

Himpunan entitas Mata Kuliah ini memiliki relasi dengan himpunan entitas Mahasiswa yang dapat digambarkan sebagai berikut:

NIM	Nama_Mhs	Alamat_Mhs	Tgl_Lahir
980001	Ali Akbar	Jl.Merdeka No.10, Jakarta	02-01-1979
980002	Budi Haryanto	Jl.Gajah Mada No.2c, Jakarta	06-10-1978
980003	Iman Faisal	Komp. Taman Anggrek Jakarta	13-05-1979
980004	Indah Susanti	Jl. Tengku Rian, Jakarta	21-06-1979

Kode_Kul	Nama_Kul
IF-110	Struktur Data
IF-310	Basis Data
KU-234	Bahasa Indonesia
MA-115	Matematika 1

Untuk memudahkan, kita sebut saja masing-masing entitas sesuai dengan nilai dari atribut Nama-nya (jadi pada himpunan entitas Mahasiswa, ada entitas 'Ali Akbar', entitas 'Budi Haryanto', entitas 'Iman Faisal' dan entitas 'Indah Susanti' dan pada himpunan entitas Kuliah, ada entitas 'Struktur Data', entitas 'Basis Data', entitas 'Bahasa Indonesia' dan entitas 'Matematika 1'). :

Untuk menjelaskan hubungan apa yang terjadi di antara kedua himpunan entitas tersebut, kita dapat memberi nama Himpunan Relasi '*Mempelajari*', atau Himpunan Relasi '*Belajar*'.

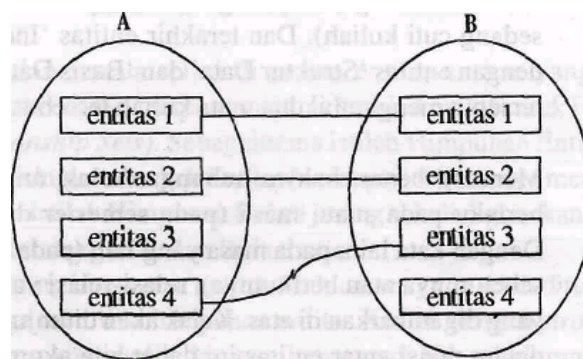
E. Kardinalitas/Derajat Relasi

Kardinalitas Relasi menunjukkan jumlah maksimum entitas yang dapat berelasi dengan entitas pada himpunan entitas yang lain. Pada contoh di atas, maka hubungan maksimum dari himpunan entitas Mahasiswa ke himpunan entitas Kuliah adalah Banyak (lebih dari satu) dan demikian pula hubungan maksimum dari himpunan entitas Kuliah ke himpunan entitas Mahasiswa adalah Banyak (lebih dari satu). Dengan demikian Kardinalitas Relasi antara kedua himpunan entitas adalah Banyak ke Banyak.

Kardinalitas Relasi yang terjadi di antara dua himpunan entitas (misalnya A dan B) dapat berupa:

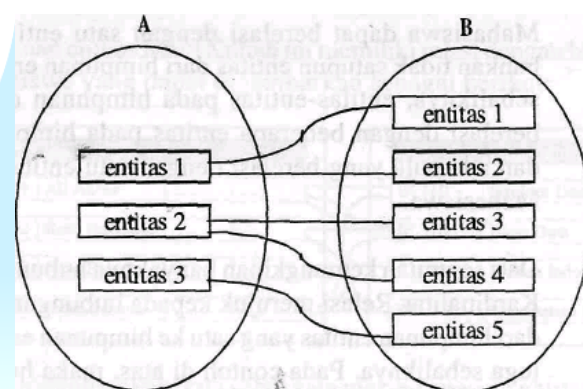
- **Satu ke Satu (*One to One*)**

Berarti setiap entitas pada himpunan entitas A berhubungan dengan paling banyak dengan satu entitas pada himpunan entitas B, dan begitu juga sebaliknya setiap entitas pada himpunan entitas B berhubungan dengan paling banyak dengan satu entitas pada himpunan entitas A.



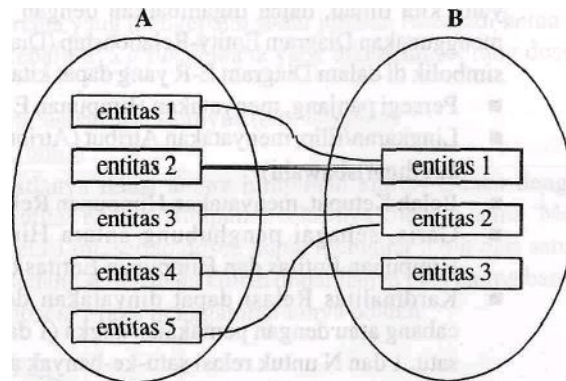
- **Satu ke Banyak (*One to Many*),**

Berarti setiap entitas pada himpunan entitas A dapat berhubungan dengan banyak entitas pada himpunan entitas B, tetapi tidak sebaliknya, di mana setiap entitas pada himpunan entitas B berhubungan dengan paling banyak satu entitas pada himpunan entitas A.



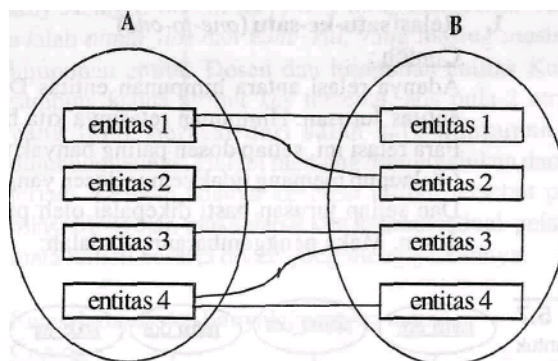
- **Banyak ke Satu (*Many to One*)**

Berarti setiap entitas pada himpunan entitas B berhubungan dengan banyak entitas pada himpunan entitas A, tetapi tidak sebaliknya, di mana setiap entitas pada himpunan entitas A berhubungan dengan paling banyak satu entitas pada himpunan entitas B.



- **Banyak ke Banyak (*Many to Many*)**


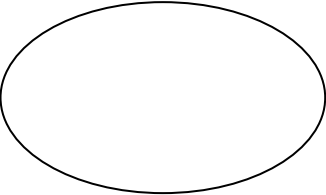
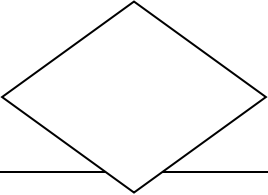

yang berarti setiap entitas pada himpunan entitas A dapat berhubungan dengan banyak entitas pada himpunan entitas B, dan demikian juga sebaliknya, di mana setiap entitas pada himpunan entitas B dapat berhubungan dengan banyak entitas pada himpunan entitas A.



Kardinalitas Relasi Satu ke Banyak dan Banyak ke Satu dapat dianggap sama, karena tinjauan Kardinalitas Relasi selalu dilihat dari dua sisi (dari himpunan entitas A ke himpunan entitas B dan dari himpunan entitas B ke himpunan entitas A). Jadi kalau penggambaran pada contoh Kardinalitas Relasi Banyak ke Satu, di mana himpunan entitas A kita tempatkan di sebelah kanan dan himpunan entitas B kita tempatkan di sebelah kiri (dan hal ini boleh-boleh saja dilakukan), maka Kardinalitas Relasinya menjadi Satu ke Banyak.

F. Diagram Entity-Relationship (Diagram E-R)

Model Entity-Relationship yang berisi komponen-komponen Himpunan Entitas dan Himpunan Relasi yang masing-masing dilengkapi dengan atribut-atribut yang merepresentasikan seluruh fakta dari '*dunia nyata*' yang kita tinjau, dapat digambarkan dengan lebih sistematis dengan menggunakan Diagram Entity-Relationship (Diagram E-R). Notasi-notasi simbolik di dalam Diagram E-R yang dapat kita gunakan adalah:

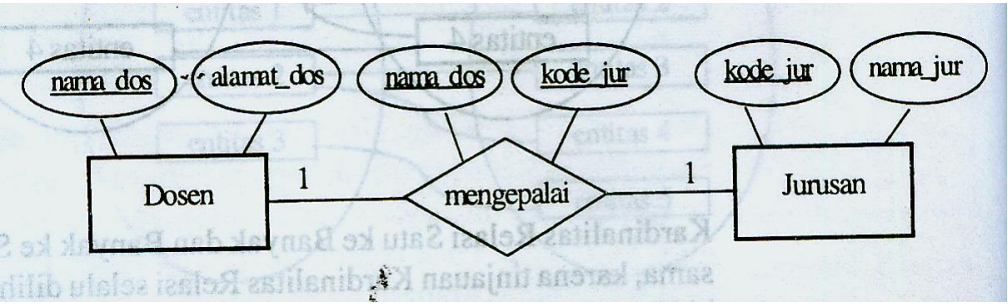
Diagram	Keterangan
	Persegi panjang, menyatakan Himpunan Entitas.
	Lingkaran/Elip, menyatakan Atribut (Atribut yang berfungsi sebagai key digarisbawahi).
	<i>Belah Ketupat, menyatakan Himpunan Relasi</i>
	Garis, sebagai penghubung antara Himpunan Relasi dengan Himpunan Entitas dan Himpunan Entitas dengan Atributnya.

Berikut adalah contoh penggambaran relasi antar himpunan entitas lengkap dengan kardinalitas relasi dan atribut-atributnya:

1. Relasi satu-ke-satu (*one-to-one*)

Misal :

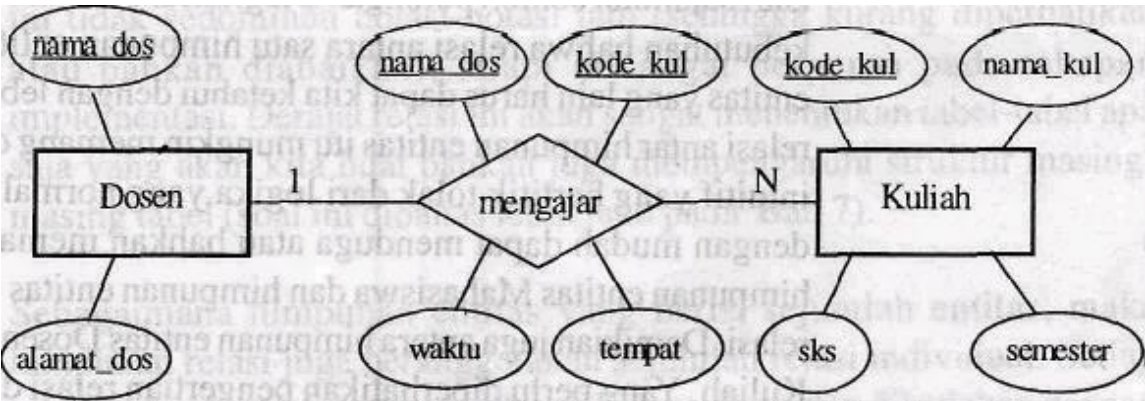
Adanya relasi antara himpunan entitas Dosen dengan himpunan entitas Jurusan. Himpunan relasinya kita beri nama '**Mengepalai**'. Pada relasi ini, setiap dosen paling banyak mengepalai satu jurusan (walaupun memang tidak semua dosen yang menjadi ketua jurusan). Dan setiap jurusan pasti dikepalai oleh paling banyak satu orang dosen. Maka penggambarannya adalah:



2. Relasi satu-ke-banyak (one-to-many)

Misal :

Adanya relasi antara himpunan entitas Dosen dengan himpunan entitas Kuliah. Himpunan relasinya kita beri nama 'Mengajar'. Pada relasi ini, setiap dosen dapat mengajar lebih dari satu mata kuliah, sedang setiap mata kuliah diajar hanya oleh paling banyak satu orang 'dosen'. Maka penggambarannya adalah:



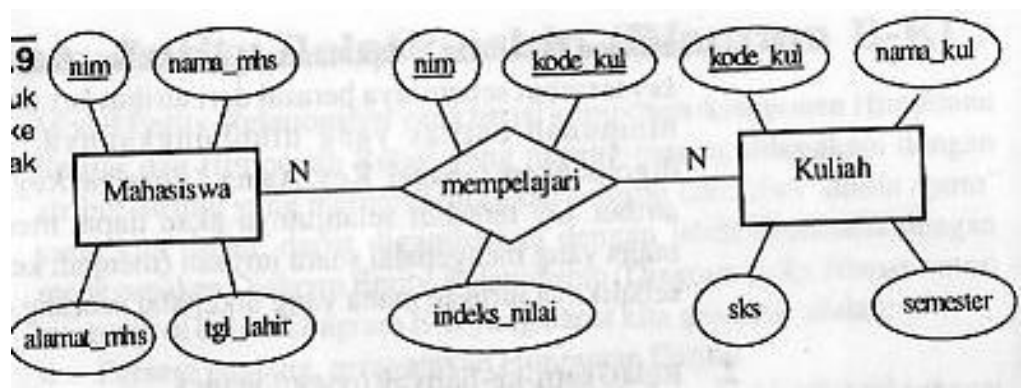
Key Asing (Foreign-Key) dari himpunan relasi Mengajar di atas adalah *nama_dos* dan *kode_kul*, yang masing-masing berasal dari himpunan entitas Dosen dan himpunan entitas Kuliah. Tetapi di samping kedua atribut *key* tersebut, ada pula 2 atribut tambahan yang tidak berasal dari salah satu himpunan entitas yang dihubungkannya. Hal ini memang dimungkinkan dan bahkan umum terjadi. Dengan adanya keempat atribut tersebut pada

himpunan relasi Mengajar, maka dapat kita ketahui jadwal pelaksanaan setiap mata kuliah beserta dosen yang mengajarkannya.

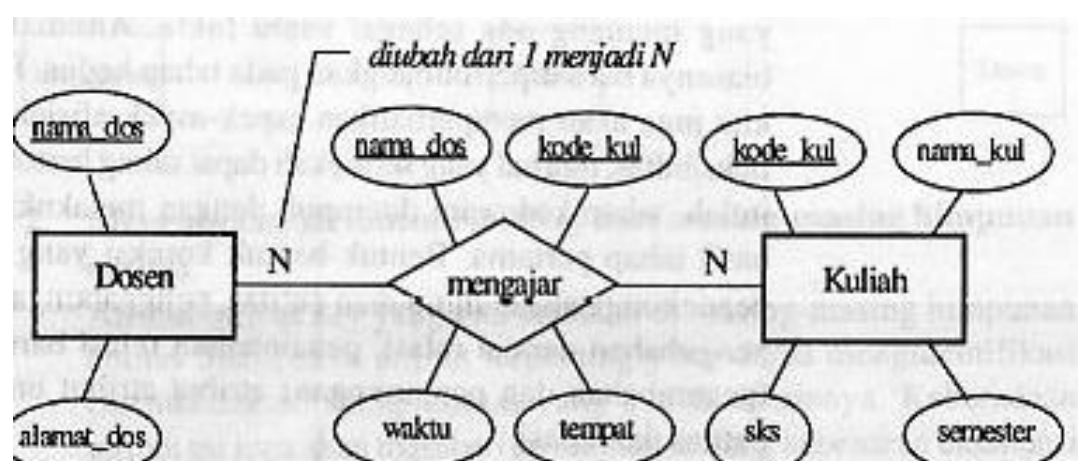
3. Relasi banyak-banyak(*many-to-many*)

Misal:

: Adanya relasi antara himpunan entitas Mahasiswa dengan himpunan entitas Kuliah. Himpunan relasinya kita beri nama 'Mempelajari'. Pada relasi ini, setiap mahasiswa dapat mempelajari lebih dari satu mata kuliah. Demikian juga sebaliknya, setiap mata kuliah dapat dipelajari oleh lebih dari satu orang mahasiswa. Maka penggambarannya adalah



Keberadaan himpunan relasi 'Mempelajari' di atas akan memiliki dua fungsi, yaitu untuk menunjukkan mata kuliah mana saja yang diambil oleh seorang mahasiswa (atau mahasiswa mana saja yang mengambil mata kuliah tertentu) dan indeks nilai yang diperoleh seorang mahasiswa untuk mata kuliah tertentu (tentu saja setelah data indeks nilai tersebut disimpan). Akan tetapi, pada setiap relasi yang ada kita juga harus menentukan kardinalitas/ derajat relasi dan atribut-atribul relasi tersebut.



G. Tahapan Pembuatan Diagram E-R

Diagram E-R selalu dibuat secara bertahap. Paling tidak ada dua kelompok tahapan yang biasa ditempuh di dalam pembuatan Diagram E-R, yaitu:

1. Tahap pembuatan Diagram E-R awal (*preliminary design*).
 2. Tahap optimasi Diagram E-R (*final design*).
- Objektif dari tahap yang pertama adalah:
 1. Mendapatkan sebuah rancangan basis data minimal yang dapat mengakomodasi kebutuhan penyimpanan data terhadap sistem yang sedang kita tinjau.
 2. Umumnya juga mengabaikan anomali-anomali (sejumlah pengecualian) yang memang ada sebagai suatu fakta. Anomali-anomali tersebut biasanya baru dipertimbangkan pada tahap kedua.
 - Objektif pada tahap kedua ini perlu memperhatikan aspek-aspek efisiensi, performansi dan fleksibilitas, tiga hal yang seringkali dapat saling bertolak belakang. Karena itulah, tahap kedua ini ditempuh dengan:
 - ☞ melakukan koreksi terhadap hasil tahap pertama.Bentuk-bentuk koreksi yang terjadi bisa berupa pendekomposisian himpunan entitas, penggabungan himpunan entitas, pengubahan derajat relasi, penambahan relasi baru hingga perubahan (penambahan dan pengurangan) atribut-atribut untuk masing-masing entitas dan relasi.

Langkah-langkah teknis pada tahap pertama tersebut untuk mewujudkan perancangan basis data pada lingkup sistem perkuliahan yang telah kita bahas, maka urutan penggambarannya adalah sebagai berikut:

1. Mengidentifikasi dan menetapkan seluruh himpunan entitas yang akan terlibat

Himpunan entitas mana saja yang akan kita pilih/libatkan tidak hanya tergantung pada jenis topik/sistem yang kita tinjau, tetapi juga ditentukan oleh seberapa jauh ruang lingkup yang ingin kita akomodasi dalam rancangan basis data kita. Dalam lingkup sistem perkuliahan sesungguhnya ada banyak sekali himpunan entitas yang bisa kita libatkan seperti Mahasiswa, Kuliah, Praktikum, Dosen, Asisten, Ruang, Jurusan, Literatur dan lain-lain. Namun, dalam lingkup sistem perkuliahan yang sederhana

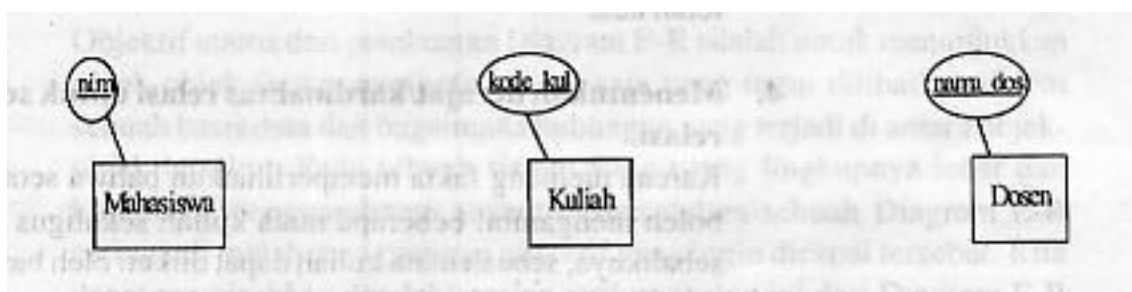
sesuai dengan pembahasan sebelumnya, dapat kita identifikasi adanya tiga buah himpunan entitas, yaitu Mahasiswa, Kuliah dan Dosen:



2. Menentukan atribut-atribut key dari masing-masing himpunan entitas

Salah satu ciri dari himpunan entitas adalah kemandiannya. Kemandirian itu terlihat dari kejelasan atribut yang menjadi *key* dan perbedaannya dengan *key* yang ada di himpunan entitas yang lain.

Dan pengidentifikasi setiap entitas secara unik di himpunan entitas Mahasiswa adalah atribut *nim*, lalu di himpunan entitas Kuliah adalah atribut *kode_kul* dan di himpunan entitas Dosen adalah atribut *nama_dos*:



Atribut *nim*, *kode_kul* dan *nama_dos* merupakan atribut-atribut yang tidak saling tergantung, karena itulah dapat kita yakini bahwa Mahasiswa, Kuliah dan Dosen memang merupakan himpunan entitas yang tepat.

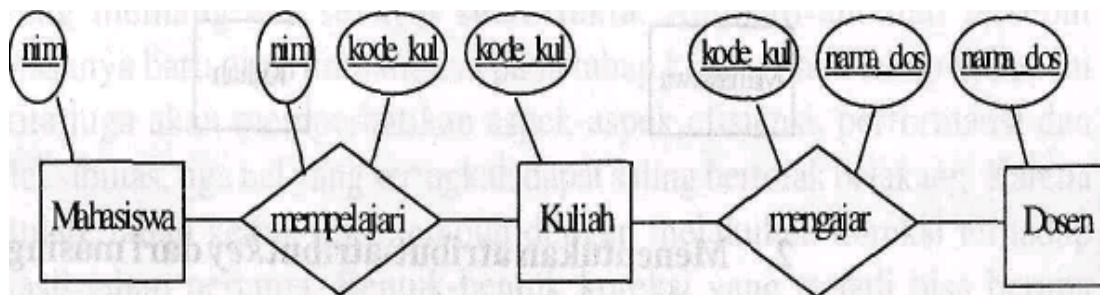
3. Mengidentifikasi dan menetapkan seluruh himpunan relasi di antara himpunan entitas-himpunan entitas yang ada beserta *foreign key*-nya.

Langkah ketiga ini merupakan langkah terpenting dalam pembentukan Diagram E-R dimana:

- ❑ Ketepatan kita dalam menentukan relasi-relasi yang terjadi di antara himpunan entitas akan sangat menentukan kualitas rancangan basis data yang kita bangun.
- ❑ Relasi-relasi yang kita tetapkan harus dapat mengakomodasi semua fakta yang ada dan menjamin semua kebutuhan penyajian data, tetapi di sisi lain juga harus dibuat seoptimal mungkin agar tidak memakan ruang penyimpanan yang lebih besar dan tidak menyulitkan operasi pengelolaan data.

- ❑ Relasi-relasi yang sifatnya tidak langsung harus ditiadakan.

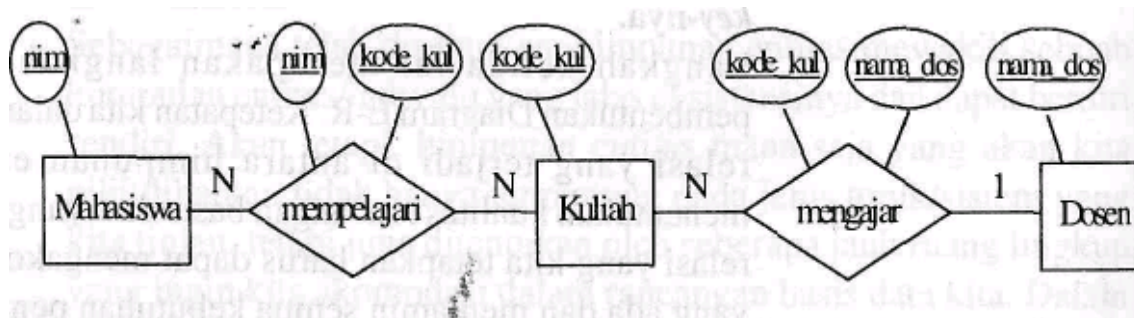
Himpunan relasi Mempelajari dan himpunan relasi Mengajar merupakan relasi langsung yang terjadi antara himpunan entitas Mahasiswa dan Kuliah serta antara himpunan entitas Dosen dan Kuliah:



Himpunan relasi Mempelajari akan dapat mengakomodasi adanya fakta tentang sejumlah mahasiswa yang mengambil mata kuliah tertentu dan sebaliknya sejumlah mata kuliah yang diambil/dipelajari oleh mahasiswa tertentu. Demikian juga dengan himpunan relasi Mengajar yang dapat mengakomodasi fakta tentang dosen yang mengajar mata kuliah tertentu. Kendati, katakanlah, ada kebutuhan untuk menyajikan informasi tentang mahasiswa mana saja yang diajar oleh seorang dosen, kita tidak perlu membuat relasi antara himpunan entitas Mahasiswa dan Dosen, karena kebutuhan penyajian informasi semacam itu telah dapat dipenuhi dengan melakukan *query* yang melibatkan himpunan entitas Kuliah dan kedua himpunan relasi yang telah ada.

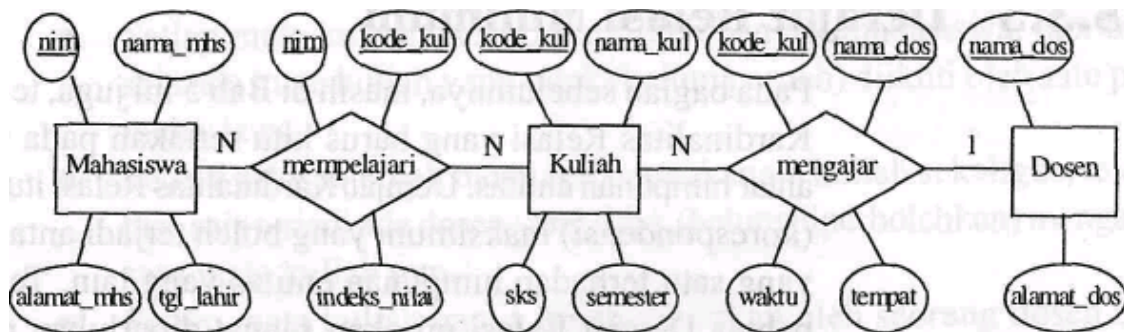
4. Menentukan derajat/kardinalitas relasi untuk setiap himpunan relasi.

Karena memang fakta memperlihatkan bahwa seorang mahasiswa boleh mengambil beberapa mata kuliah sekaligus dan begitu juga sebaliknya, sebuah mata kuliah dapat diikuti oleh banyak mahasiswa sekaligus, maka derajat relasi antara himpunan entitas Mahasiswa dan Kuliah adalah banyak-ke-banyak. Sementara itu, fakta yang ada juga menunjukkan bahwa seorang dosen dapat mengajar beberapa mata kuliah (pada semester yang sedang berjalan), tetapi setiap mata kuliah hanya dipegang oleh seorang dosen, maka derajat relasi antara himpunan entitas Dosen dan Kuliah adalah satu-ke-banyak. Berangkat dari fakta tersebut, maka Diagram E-R kita sekarang menjadi:



5. Melengkapi himpunan entitas dan himpunan relasi dengan atribut-atribut deskriptif (non key).

Berangkat dari fakta yang ada, atribut-atribut deskriptif yang dapat kita sertakan pada masing-masing himpunan entitas dan himpunan relasi adalah (di dalam elip dan tidak bergaris bawah):



Langkah terakhir ini merupakan langkah pelengkap sehingga tidak terpenting langkah-langkah sebelumnya. Keberadaan atribut-atribut deskriptif ini merupakan refleksi pengakomodasian terhadap fakta yang memang ada dan kebutuhan penyajian data di saat yang lain. Atribut deskriptif ini juga tidak banyak berperan dalam membentuk pemahaman kita jika 'membaca' sebuah Diagram E-R, bahkan cenderung mengganggu karena biasanya jumlah atribut demikian cukup banyak. Karena itu, khususnya pada sebuah sistem yang besar dan kompleks, langkah terakhir ini acapkali tidak dilakukan, sehingga Diagram E-R yang dibangun hanya sampai pada langkah keempat.

H. Diagram E-R dengan Kamus Data

Objektif utama dari pembuatan Diagram E-R adalah untuk menunjukkan objek-objek (himpunan entitas) apa saja yang ingin dilibatkan dalam sebuah basis data dan bagaimana hubungan yang terjadi di antara objek-objek tersebut. Pada

sebuah sistem yang ruang lingkupnya lebar dan kompleks, penggambaran atribut-atribut dalam sebuah Diagram E-R seringkali malah mengganggu objektif yang ingin dicapai tersebut. Kita dapat memisahkan pendeklarasian atribut-atribut ini dari Diagram E-R dan menyatakannya dalam sebuah kamus data. Kamus data berisi daftar atribut yang diapit kurung kurawal ('{' dan '}'). Atribut yang berfungsi sebagai *key* juga dibedakan dengan yang bukan *key* dengan menggarisbawahi atribut tersebut.

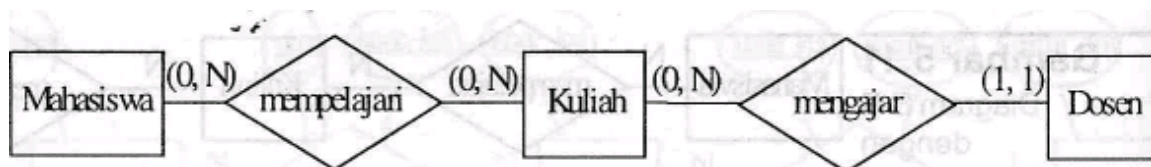
Kamus Data:

- ✓ Mahasiswa = {nim. nama_mhs, alamat_mhs, tgl_lahir}
- ✓ Kuliah = {kode kul. nama_kul, sks, semester}
- ✓ Dosen = {nama dos. alamat dos}
- ✓ mempelajari = {mm, kode kul. indeks_nilai}
- ✓ mengajar = {kode kul. nama dos. waktu, tempat}

I. Derajat Relasi Minimum

Derajat/Kardinalitas Relasi minimum merupakan hubungan (korespondensi) minimum yang boleh terjadi antara himpunan entitas yang satu terhadap himpunan entitas yang lain, contoh relasi antara himpunan entitas Mahasiswa dan Kuliah. Kita telah mengetahui bahwa seorang mahasiswa boleh mengambil banyak mata kuliah sekaligus dan demikian juga sebaliknya (sehingga Kardinalitas Relasinya adalah banyak-ke-banyak atau N-N). Sementara derajat minimum dalam relasi itu dapat kita ketahui dari fakta bahwa seorang mahasiswa boleh tidak mengambil mata kuliah satupun (karena sedang cuti, misalnya) dan bisa terjadi sebuah mata kuliah tidak diikuti oleh seorang mahasiswa pun (karena merupakan mata kuliah pilihan, misalnya). Dengan begitu Derajat Relasi Minimum-nya sama-sama 0). Nilai 0 (nol) memang merupakan Derajat Relasi Minimum yang sering terjadi. Tetapi tidak selalu demikian. Misalnya, relasi antara Kuliah dan Dosen, ada fakta bahwa setiap mata kuliah harus sudah ditentukan dosen yang akan mengajarkannya (sehingga Derajat Relasi Minimum-nya adalah 1).

Dapat digambarkan Diagram E-R untuk sistem perkuliahan sebelumnya sebagai berikut (atribut-atributnya sengaja tidak diperlihatkan):



Dengan Diagram E-R di atas, pemahaman hubungan antara himpunan entitas-himpunan entitas tersebut adalah:

- ❑ Seorang mahasiswa dapat mempelajari banyak mata kuliah sekaligus, tapi

- boleh juga tidak (belum) mempelajari mata kuliah satu pun.
- ❑ Setiap mata kuliah dapat diikuti oleh banyak mahasiswa, tapi bisa saja ada mata kuliah yang tidak (belum pernah) diikuti oleh satu pun mahasiswa.
 - ❑ Seorang dosen boleh mengajar banyak mata kuliah sekaligus, tetapi bisa saja terjadi ada dosen yang tidak (belum diperbolehkan) mengajar satu mata kuliah pun.
 - ❑ Setiap mata kuliah hanya boleh diajarkan oleh seorang dosen dan tidak boleh ada mata kuliah yang belum ditentukan siapa dosennya.

Daftar Pustaka

15. Connolly, Thomas M., Begg, Carolyn E., and Strachan, Anne D., Database System. A practical pproach to Design, Implementation, and Management, Addison Wesley Company, 1996.
16. David M. Kroenke, *Basis data Processing, Fundamentals, Design and Implementation*, 12nd Edition, Prentice-Hall Int'l Edition, 2012



MODUL PERKULIAHAN

Basis Data

Entity Relationship Diagram

Fakultas
Ilmu Komputer

Program Studi
Sistem Informasi

Tatap Muka
10

Kode MK
87010

Disusun Oleh
Tim Dosen

Abstract

Modul ini mempelajari Entity Relationship Diagram.

Kompetensi

Mahasiswa mampu menggambarkan model hubungan entitas dalam sebuah diagram ERD.

A. Pendahuluan

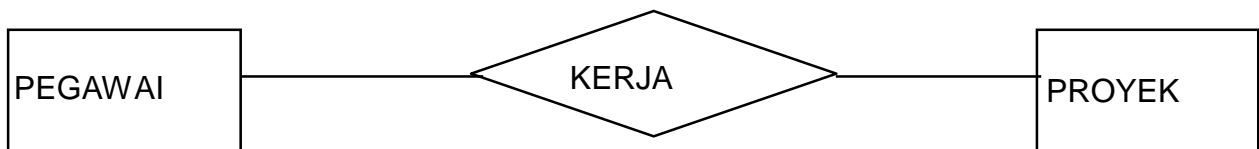
Model Entity Relationship merupakan suatu penyajian data dengan menggunakan Entity dan Relationship

Entity

- Entity adalah obyek yang dapat dibedakan dalam dunia nyata
- Entity set adalah kumpulan dari entity yang sejenis
- Entity set dapat berupa :
 - Obyek secara fisik : Rumah, Kendaraan, Peralatan
 - Obyek secara konsep : Pekerjaan , Perusahaan, Rencana

Relationship

- Relationship adalah hubungan yang terjadi antara satu atau lebih entity.
- Relationship set adalah kumpulan relationship yang sejenis.



Gambar 1. Entity dan Relationship

Atribut

- Atribut adalah karakteristik dari entity atau relationship, yang menyediakan penjelasan detail tentang entity atau relationship tersebut.
- Nilai Atribut merupakan suatu data aktual atau informasi yang disimpan pada suatu atribut di dalam suatu entity atau relationship.

Jenis-jenis atribut :

- Key

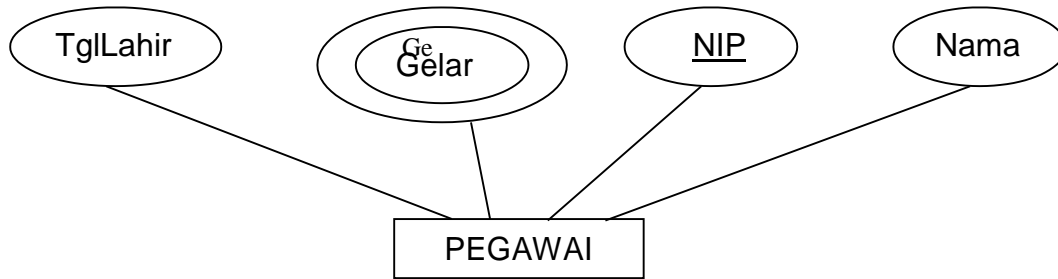
Atribut yang digunakan untuk menentukan suatu entity secara unik.

- ❑ Atribut Simple

Atribut yang bernilai tunggal.

- ❑ Atribut Multivalued

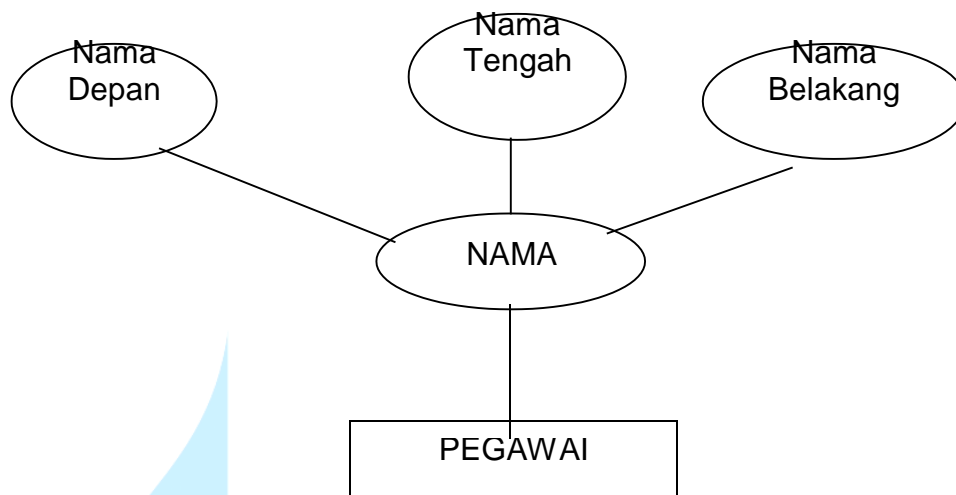
Atribut yang memiliki sekelompok nilai untuk setiap instan entity.



Gambar 2. Atribut Multivalued

- ❑ Atribut Composite

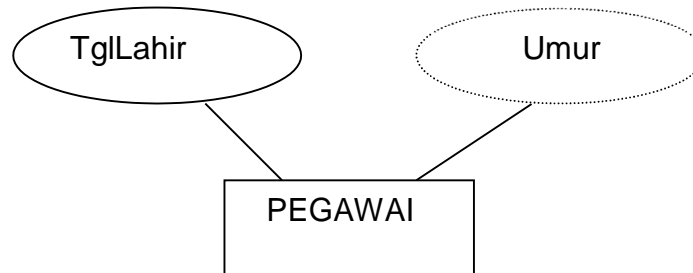
Suatu atribut yang terdiri dari beberapa atribut yang lebih kecil yang mempunyai arti tertentu.



Gambar 3. Atribut Composite

- ❑ Atribut Derivatif

Suatu atribut yang dihasilkan dari atribut yang lain.

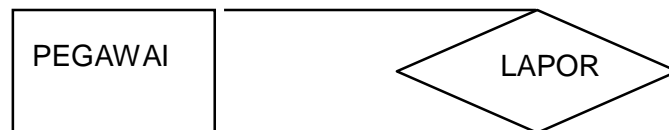


Gambar 4. Atribut Derivatif

B. Derajat dari relationship

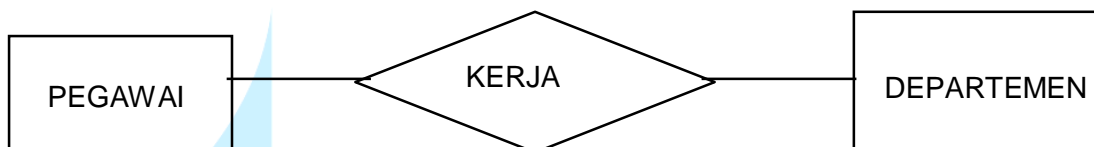
- ❑ Menjelaskan jumlah entity yang berpartisipasi dalam suatu relationship

Unary Degree (Derajat Satu)



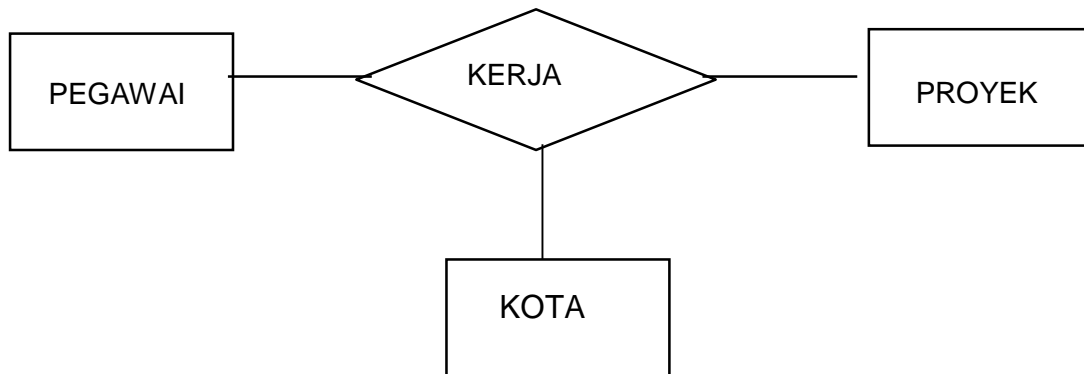
Gambar 5. Unary Degree

Binary Degree (Derajat Dua)



Gambar 6. Binary Degree

Ternary Degree (Derajat Tiga)



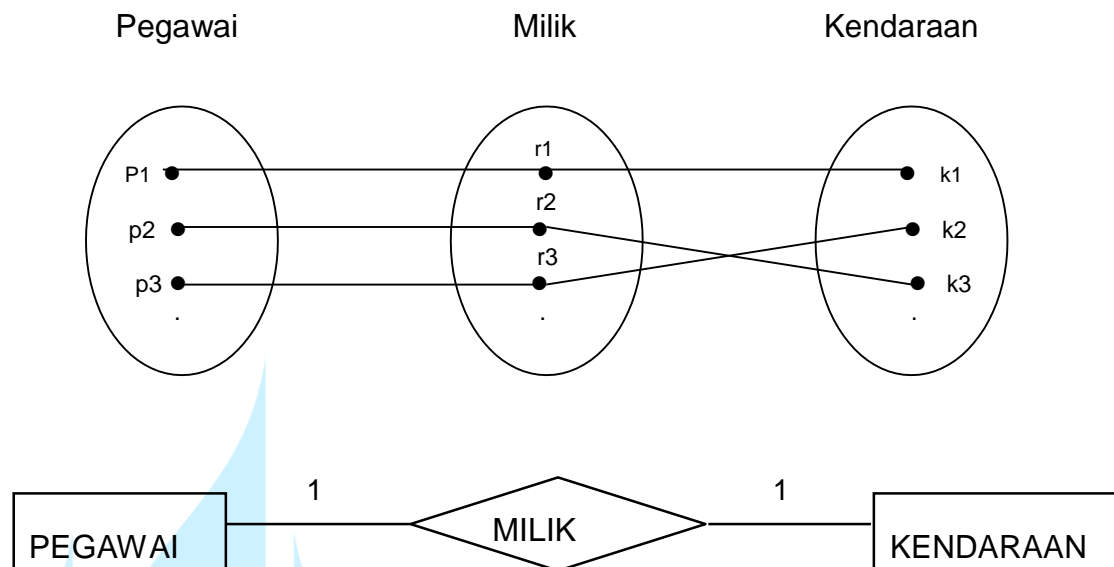
Gambar 7. Ternary Degree

C. Cardinality Ratio Constraint

Menjelaskan batasan jumlah keterhubungan satu entity dengan entity lainnya.

□ Jenis Cardinality Ratio

1 : 1



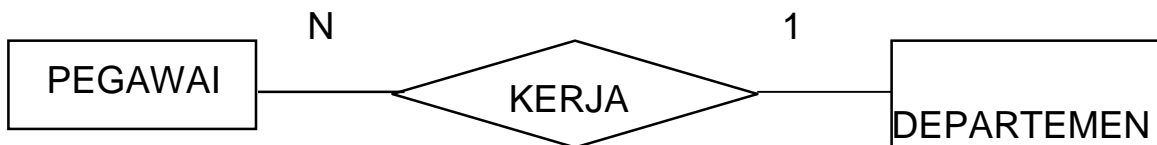
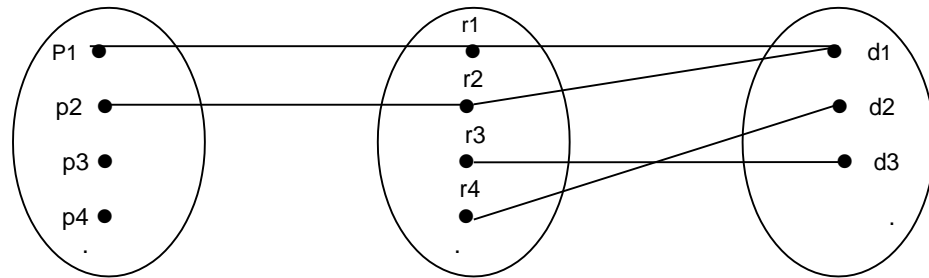
Gambar 8. Cardinatlity Ratio 1:1

1 : N / N : 1

Pegawai

Kerja

Departemen

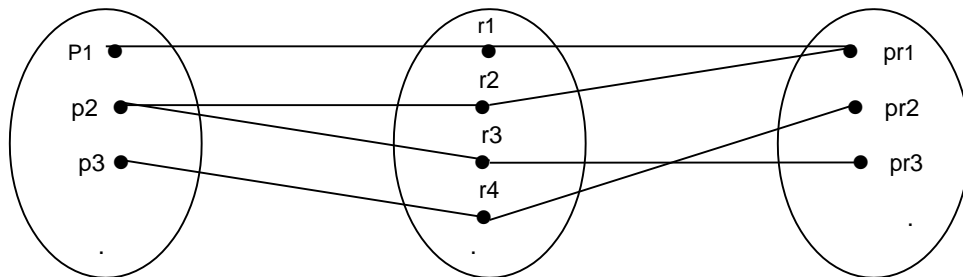


M : N

Pegawai

Kerja

Proyek



D. Participation Constraint

- Menjelaskan apakah keberadaan suatu entity tergantung pada hubungannya dengan entity lain .

Terdapat 2 macam Participation Constraint :

Total Participation

Keberadaan suatu entity tergantung pada hubungannya dengan entity lain.



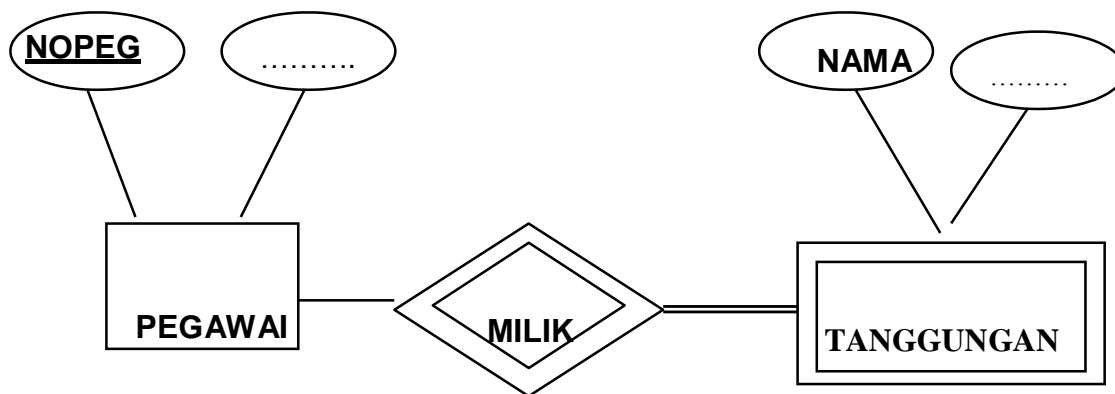
Partial Participation

Keberadaan suatu entity tidak tergantung pada hubungannya dengan entity lain.


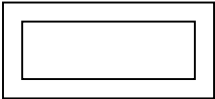
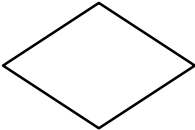
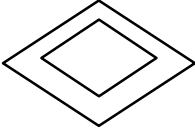



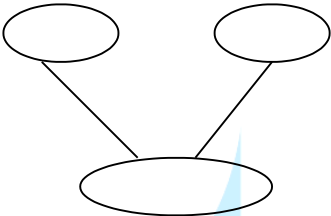



E. Weak entity

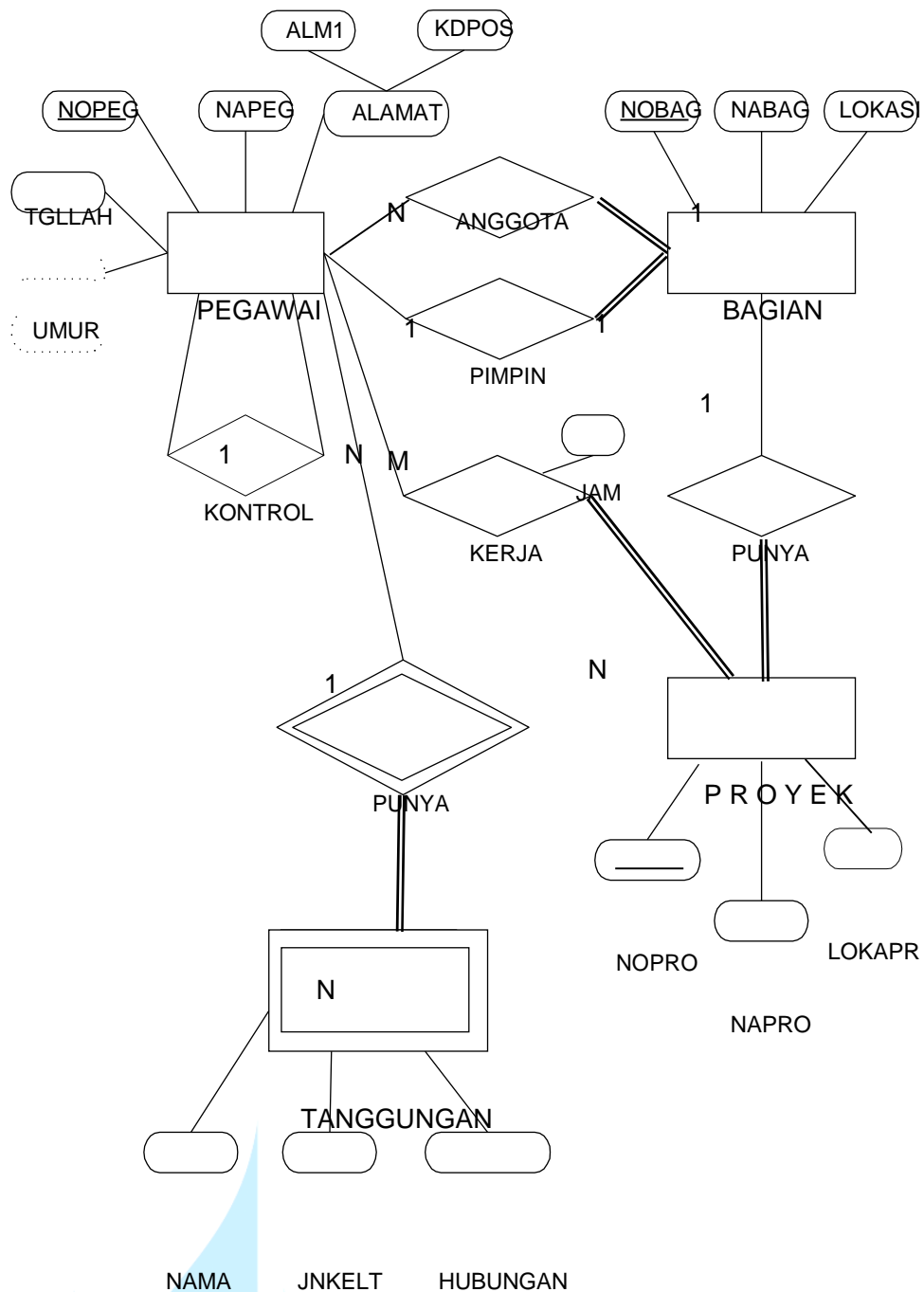
- ❑ Weak Entity adalah suatu Entity dimana keberadaan dari entity tersebut tergantung dari keberadaan entity lain.
- ❑ Entity yang merupakan induknya disebut *Identifying Owner* dan relationshipnya disebut *Identifying Relationship*.
- ❑ Weak Entity selalu mempunyai Total Participation constraint dengan Identifying Owner.



F. Simbol-simbol ER-Diagram

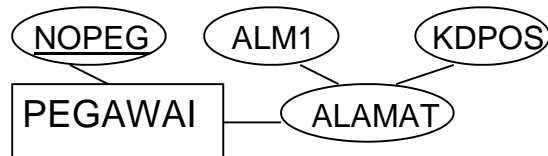
<u>Notasi</u>	<u>Arti</u>
1. 	1. Entity
2. 	2. Weak Entity
3. 	3. Relationship
4. 	4. Identifying Relationship
5. 	5. Atribut
6. 	6. Atribut Primary Key
7. 	7. Atribut Multivalue
8. 	8. Atribut Composite
9. 	9. Atribut Derivatif

Contoh Penggambaran Diagram ER



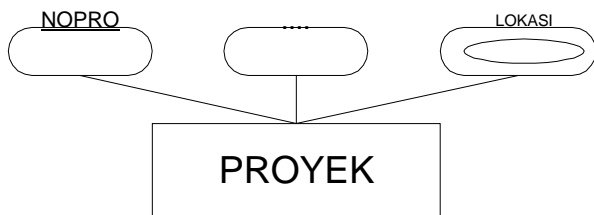
G. Transformasi dari ERD ke Database Relasional

1. Setiap tipe Entity dibuat suatu relasi yang memuat semua atribut simple, sedangkan untuk atribut composite hanya dimuat komponen-komponennya saja.



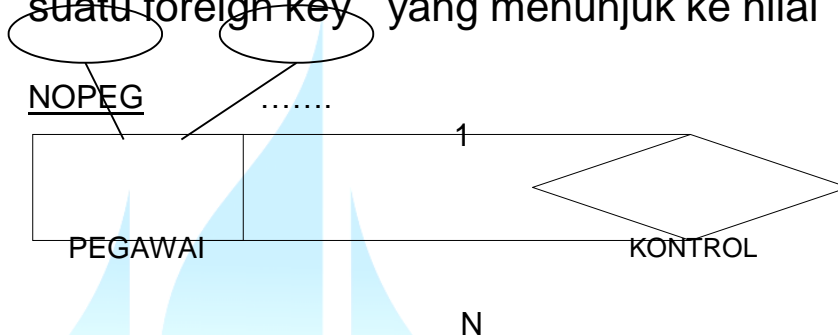
PEGAWAI (NOPEG, ALM1, KDPOS,)

2. Setiap relasi yang mempunyai atribut multivalued, buatlah relasi baru dimana Primary Keynya merupakan gabungan dari Primary Key dari relasi tersebut dengan atribut multivalued.



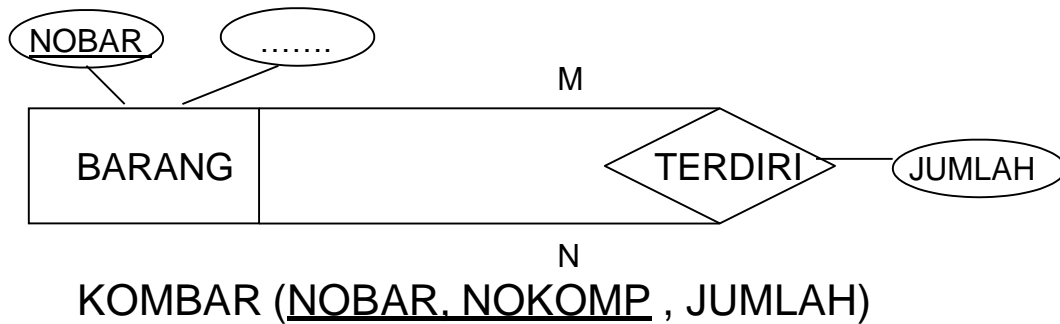
LOKPR(NOPRO, LOKASI)

3. Setiap Unary Relationship 1:N, pada relasi perlu ditambahkan suatu foreign key yang menunjuk ke nilai primary keynya.

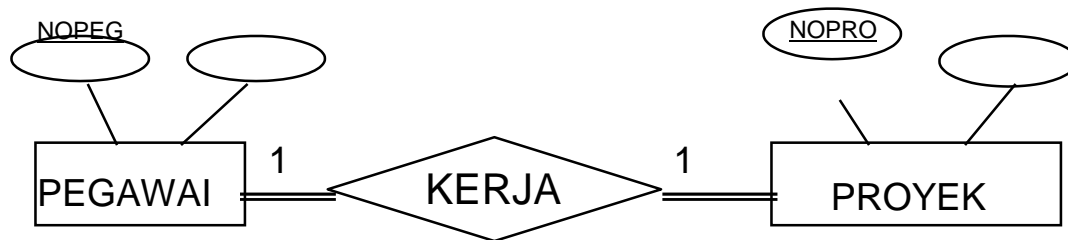


PEGAWAI (NOPEG, , SUPERVISOR-ID)

4. Setiap Unary Relationship M:N, buatlah relasi baru dimana primary keynya merupakan gabungan dari dua atribut dimana keduanya menunjuk ke primary key relasi awal dengan penamaan yang berbeda.

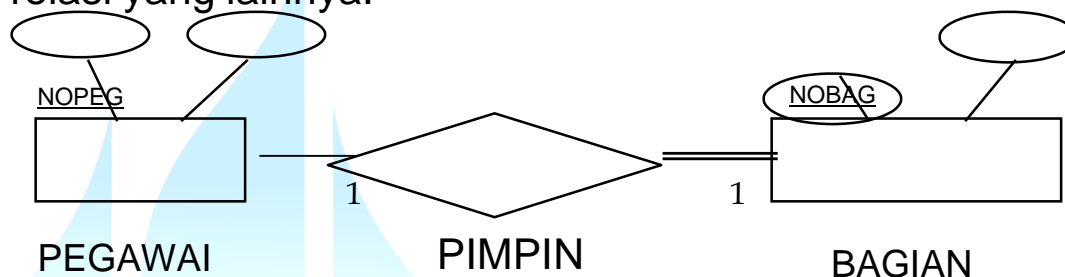


5. Setiap Binary Relationship 1:1, dimana Participation Constraint keduanya total, buatlah suatu relasi gabungan dimana Primary Keynya dapat dipilih salah satu.



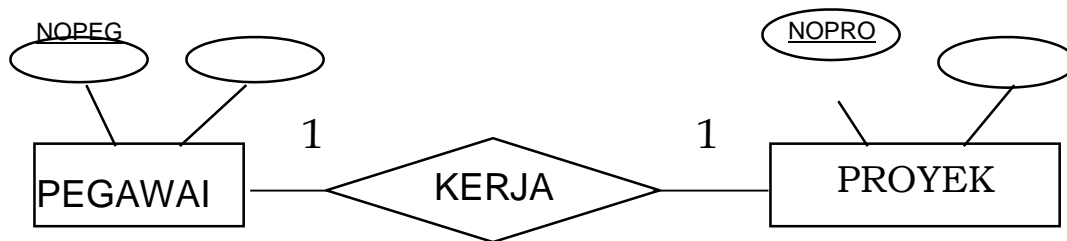
PEGAWAI (NOPEG, ... , NOPRO, ...).

6. Setiap Binary Relationship 1:1 dan salah satu Participation Constraintnya Total, maka Primary Key pada relasi yang Participation Constraintnya Partial menjadi Foreign Key pada relasi yang lainnya.



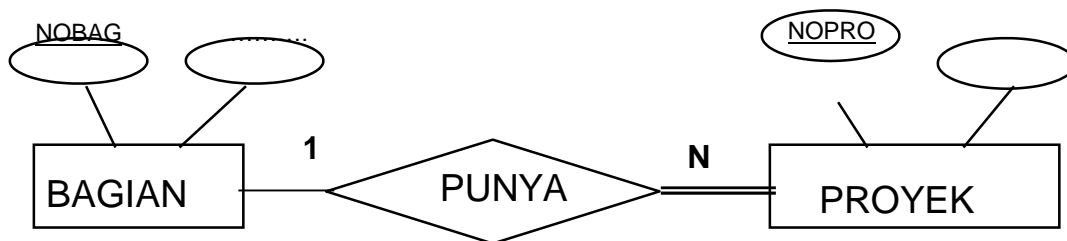
BAGIAN (NOBAG, ... , MANAGER)

7. Setiap Binary Relationship 1:1, dimana kedua Participation Constraintnya partial, maka selain kedua relasi perlu dibuat relasi baru yang berisi Primary Key gabungan dari Primary Key kedua tipe Entity yang berelasi.



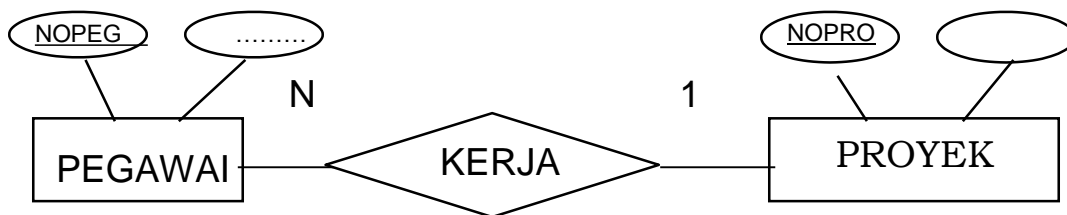
PEKERJAAN (NOPEG, NOPRO, ...)

8. Setiap Binary Relationship 1 : N, dimana tipe Entity yang bersisi N mempunyai Participation Constraint Total, maka Primary Key pada relasi yang bersisi 1 dijadikan Foreign Key pada relasi yang bersisi N.



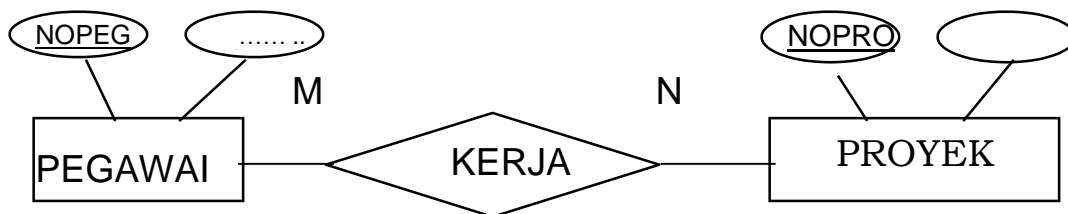
PROYEK (NOPRO, ... , NOBAG)

9. Setiap Binary Relationship 1 : N, dimana tipe Entity yang bersisi N mempunyai Participation Constraint partial, buatlah relasi baru dimana Primary Keynya merupakan gabungan dari Primary Key kedua tipe Entity yang berelasi.



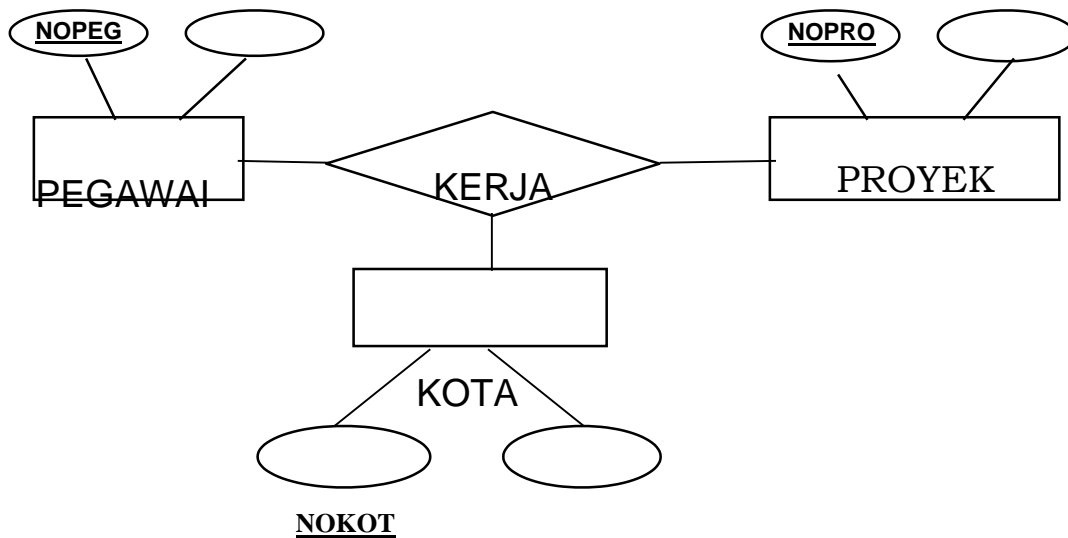
PEKERJAAN (NOPEG, NOPRO,)

10. Setiap Binary Relationship M:N, buatlah relasi baru dimana Primary Keynya merupakan gabungan dari Primary Key kedua tipe Entity yang berelasi.



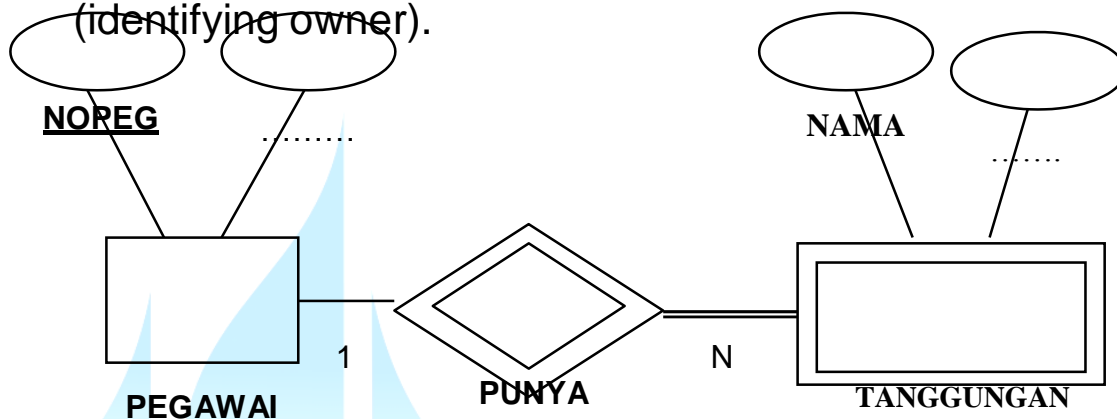
PEKERJAAN (NOPEG, NOPRO,)

11. Setiap Ternary Relationship, buatlah relasi baru dimana Primary Keynya merupakan gabungan dari Primary Key ketiga tipe Entity yang berelasi.



PEKERJAAN (NOPEG, NOPRO, NOKOT)

12. Setiap tipe Weak Entity, dibuat suatu relasi yang memuat semua atributnya dimana Primary Keynya adalah gabungan dari Partial Key dan Primary Key dari relasi induknya (identifying owner).



TANGGUNGAN (NOPEG.NAMA,)

H. Hasil Transformasi dari Diagram ER ke database relasional :

Skema Database

PEGAWAI (NOPEG, NAPEG, ALM1, KDPOS, TGLLAH, UMUR, SUPERVISOR-ID, NOBAG)

BAGIAN (NOBAG, NABAG, LOKASI, MANAGER)

PROYEK (NOPRO, NAPRO, NOBAG)

LOKPR (NOPRO, LOKAPR) PEKERJAAN (NOPEG, NOPRO, JAM)

TANGGUNGAN (NOPEG, NAMA, JNKELT, HUBUNGAN)

Daftar Pustaka

17. Connolly, Thomas M., Begg, Carolyn E., and Strachan, Anne D., Database System. A practical pproach to Design, Implementation, and Management, Addison Wesley Company, 1996.
18. David M. Kroenke, *Basis data Processing, Fundamentals, Design and Implementation*, 12nd Edition, Prentice-Hall Int'l Edition, 2012



MODUL PERKULIAHAN

Basis Data

Konsep Normalisasi & Anomali Tabel

Fakultas
Ilmu Komputer

Program Studi
Sistem Informasi

Tatap Muka
11

Kode MK
87010

Disusun Oleh
Tim Dosen

Abstract

Modul ini mempelajari konsep normalisasi & anomali tabel

Kompetensi

Mahasiswa mampu menjelaskan konsep normalisasi, konsep anomali serta mengidentifikasi anomali dalam sebuah relasi.

A. Pendahuluan

Normalisasi adalah suatu teknik untuk mengorganisasi data ke dalam tabel-tabel untuk memenuhi kebutuhan pemakai di dalam suatu organisasi.

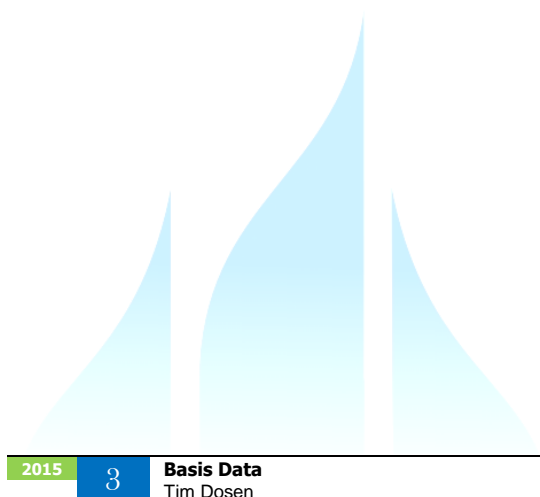
B. Tujuan dari normalisasi

- Untuk menghilangkan kerangkapan data
- Untuk mengurangi kompleksitas
- Untuk mempermudah pemodifikasian data

C. Proses Normalisasi

- Data diuraikan dalam bentuk tabel, selanjutnya dianalisis berdasarkan persyaratan tertentu ke beberapa tingkat.
- Apabila tabel yang diuji belum memenuhi persyaratan tertentu, maka tabel tersebut perlu dipecah menjadi beberapa

tabel yang lebih sederhana sampai memenuhi bentuk yang optimal.



D. Tahapan Normalisasi

Bentuk Tidak Normal



Menghilangkan perulangan group

Bentuk Normal Pertama (1NF)



Menghilangkan ketergantungan sebagian

Bentuk Normal Kedua (2NF)



Menghilangkan ketergantungan transitif

Bentuk Normal Ketiga (3NF)



Menghilangkan anomali-anomali hasil dari
ketergantungan fungsional

Bentuk Normal Boyce-Codd (BCNF)



Menghilangkan Ketergantungan Multivalue

Bentuk Normal Keempat (4NF)



Menghilangkan anomali-anomali yang tersisa

Bentuk Normal Kelima

E. Ketergantungan Fungsional

Definisi :

Atribut Y pada relasi R dikatakan tergantung fungsional pada atribut X ($R.X \rightarrow R.Y$), jika dan hanya jika setiap nilai X pada relasi R mempunyai tepat satu nilai Y pada R.

Misal, terdapat skema database Pemasok-barang :

Pemasok (No-pem, Na-pem)

Tabel PEMASOK-BARANG

<u>No-pem</u>	Na-pem
P01	Baharu
P02	Sinar
P03	Harapan

Ketergantungan fungsional dari tabel PEMASOK-BARANG adalah :

No-pem \rightarrow Na-pem

F. Ketergantungan Fungsional Penuh

Definisi :

Atribut Y pada relasi R dikatakan tergantung fungsional penuh pada atribut X pada relasi R, jika Y tidak tergantung pada subset dari X (bila X adalah key gabungan)

Contoh :

KIRIM-BARANG(No-pem, Na-pem, No-bar, Jumlah)

<u>No-pem</u>	Na-pem	<u>No-bar</u>	Jumlah
P01	Baharu	B01	1000
P01	Baharu	B02	1500
P01	Baharu	B03	2000
P02	Sinar	B03	1000
P03	Harapan	B02	2000

Ketergantungan fungsional :

No-pem --> Na-pem

No-bar, No-pem --> Jumlah (Tergantung penuh thd keynya)

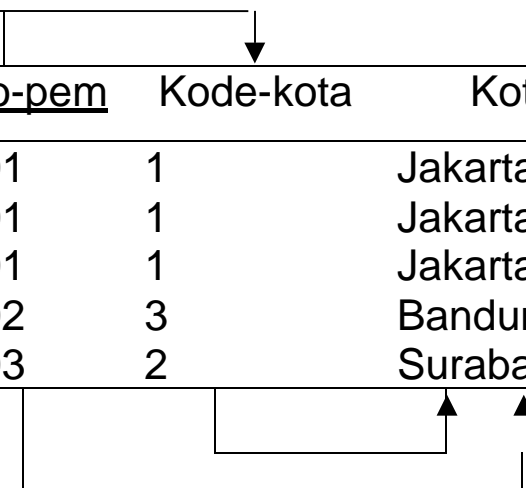
G. Ketergantungan Transitif

Definisi :

Atribut Z pada relasi R dikatakan tergantung transitif pada atribut X , jika atribut Y tergantung pada atribut X pada relasi R dan atribut Z tergantung pada atribut Y pada relasi R.

($X \twoheadrightarrow Y$, $Y \rightarrow Z$ maka $X \twoheadrightarrow Z$)

Contoh :



<u>No-pem</u>	Kode-kota	Kota	<u>No-bar</u>	Jumlah
P01	1	Jakarta	B01	1000
P01	1	Jakarta	B02	1500
P01	1	Jakarta	B03	2000
P02	3	Bandung	B03	1000
P03	2	Surabaya	B02	2000

Ketergantungan transitif :

No-pem \rightarrow Kode-kota

Kode-kota \rightarrow Kota , maka

No-pem \rightarrow Kota

Bentuk Normal Kesatu (1NF)

Suatu relasi dikatakan sudah memenuhi Bentuk Normal Kesatu bila setiap data bersifat atomik yaitu setiap irisan baris dan kolom hanya mempunyai satu nilai data

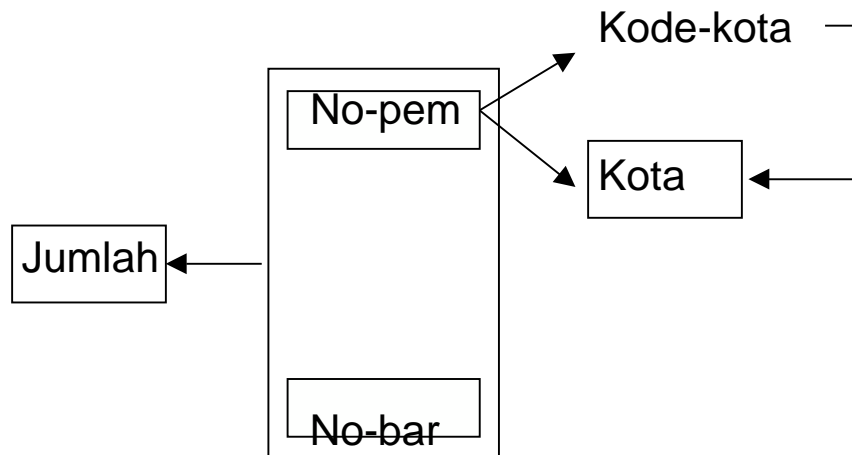
Tabel KIRIM-1 (Unnormal)

No-pem	Kode-kota	Kota	No-bar	Jumlah
P01	1	Jakarta	B01	1000
			B02	1500
			B03	2000
P02	3	Bandung	B03	1000
P03	2	Surabaya	B02	2000

Tabel KIRIM-2 (1NF)

<u>No-pem</u>	Kode-kota	Kota	<u>No-bar</u>	Jumlah
P01	1	Jakarta	B01	1000
P01	1	Jakarta	B02	1500
P01	1	Jakarta	B03	2000
P02	3	Bandung	B03	1000
P03	2	Surabaya	B02	2000

Diagram Ketergantungan Fungsional



Bentuk Normal Kedua (2NF)

Suatu relasi dikatakan sudah memenuhi Bentuk Normal Kedua bila relasi tersebut sudah memenuhi bentuk Normal kesatu, dan atribut yang bukan key sudah tergantung penuh terhadap keynya.

Tabel PEMASOK-1 (2NF)

<u>No-pem</u>	Kode-kota	Kota
P01	1	Jakarta
P02	3	Bandung
P03	2	Surabaya

Bentuk Normal Ketiga (3NF)

Suatu relasi dikatakan sudah memenuhi Bentuk Normal ketiga bila relasi tersebut sudah memenuhi bentuk Normal kedua dan atribut yang bukan key tidak tergantung transitif terhadap keynya.

Tabel KIRIM-3 (3NF)

<u>No-pem</u>	<u>No-bar</u>	Jumlah
P01	B01	1000
P01	B02	1500
P01	B03	2000
P02	B03	1000
P03	B02	2000

Tabel PEMASOK-2 (3NF)

<u>No-pem</u>	Kode-kota
P01	1
P02	3
P03	2

Tabel PEMASOK-3 (3NF)

<u>Kode-kota</u>	Kota
1	Jakarta
2	Surabaya
3	Bandung

Normalisasi pada database perkuliahan

Asumsi :

- %% Seorang mahasiswa dapat mengambil beberapa mata kuliah
- %% Satu mata kuliah dapat diambil oleh lebih dari satu mahasiswa
- %% Satu mata kuliah hanya diajarkan oleh satu dosen
- %% Satu dosen dapat mengajar beberapa mata kuliah
- %% Seorang mahasiswa pada mata kuliah tertentu hanya mempunyai satu nilai

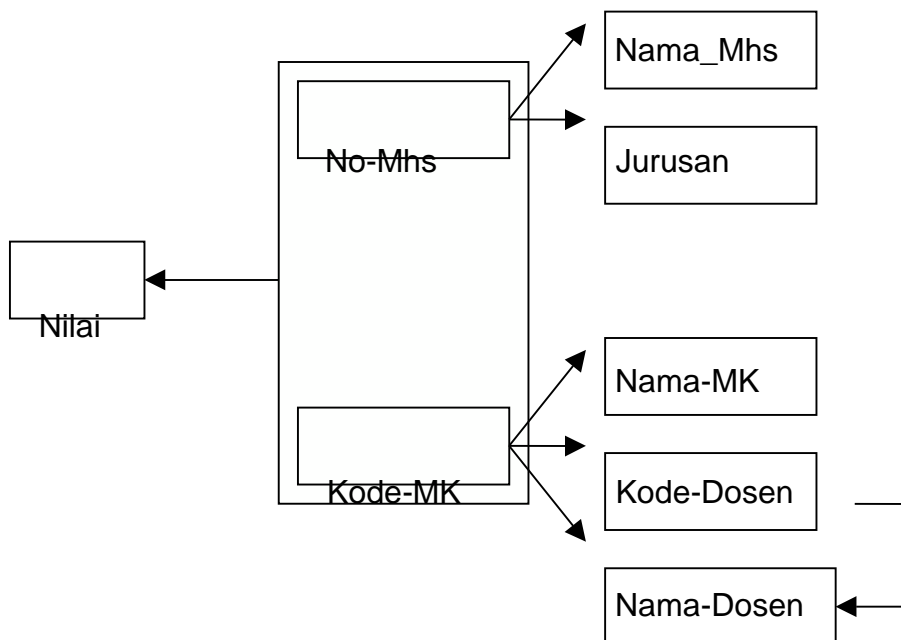
Tabel MAHASISWA-1 (Unnormal)

No-Mhs	Nama - Mhs	Jurusan	Kode- MK	Nama-MK	Kode-Dosen	Nama- Dosen	Nilai
2683	Welli	MI	MI350	Manajamen DB	B104	Ati	A
			MI465	Analsis Prc. Sistem	B317	Dita	B
5432	Bakri	Ak.	MI350	Manajemen DB	B104	Ati	C
			AKN201	Akuntansi Keuangan	D310	Lia	B
			MKT300	Dasar Pemasaran	B212	Lola	A

Tabel MAHASISWA-2 (1NF)

<u>No-Mhs</u>	Nama- Mhs	Jurusan	<u>Kode-MK</u>	Nama-MK	Kode-Dosen	Nama- Dosen	Nilai
2683	Welli	MI	MI350	Manajamen DB	B104	Ati	A
2683	Welli	MI	MI465	Analsis Prc. Sistem	B317	Dita	B
5432	Bakri	Ak.	MI350	Manajemen DB	B104	Ati	C
5432	Bakri	Ak.	AKN201	Akuntansi Keuangan	D310	Lia	B
5432	Bakri	Ak.	MKT300	Dasar Pemasaran	B212	Lola	A

H. Diagram Ketergantungan Fungsional



Tabel KULIAH (2NF)

<u>Kode-MK</u>	Nama-MK	Kode-Dosen	Nama-Dosen
MI350	Manajamen DB	B104	Ati
MI465	Analisis Prc. Sistem	B317	Dita
AKN201	Akuntansi Keuangan	D310	Lia
MKT300	Dasar Pemasaran	B212	Lola

Tabel MAHASISWA-3 (3NF)

<u>No-Mhs</u>	Nama-Mhs	Jurusan
2683	Welli	MI
5432	Bakri	Ak.

Tabel NILAI (3NF)

<u>No-Mhs</u>	<u>Kode MK</u>	Nilai
2683	MI350	A
2683	MI465	B
5432	MI350	C
5432	AKN201	B
5432	MKT300	A

Tabel MATAKULIAH (3NF)

<u>Kode-MK</u>	Nama-MK	Kode-Dosen
MI350	Manajamen DB	B104
MI465	Analisis Prc. Sistem	B317
AKN201	Akuntansi Keuangan	D310
MKT300	DasarPemasaran	B212

Tabel DOSEN (3NF)

<u>Kode- Dosen</u>	Nama-Dosen
B104	Ati
B317	Dita
B310	Lia
B212	Lola

Daftar Pustaka

19. Connolly, Thomas M., Begg, Carolyn E., and Strachan, Anne D., Database System. A practical pproach to Design, Implementation, and Management, Addison Wesley Company, 1996.

20. David M. Kroenke, *Basis data Processing, Fundamentals, Design and Implementation*,
12nd Edition, Prentice-Hall Int'l Edition, 2012



MODUL PERKULIAHAN

Basis Data

Ketergantungan Fungsional Dalam Normalisasi

Fakultas
Ilmu Komputer

Program Studi
Sistem Informasi

Tatap Muka
12

Kode MK
87010

Disusun Oleh
Tim Dosen

Abstract

Modul ini mempelajari ketergantungan fungsional dalam normalisasi & bentuk normal 1, 2 dan 3.

Kompetensi

Mahasiswa mampu memahami konsep dari ketergantungan fungsional dengan mengidentifikasi jenis normalisasi serta mampu memahami aturan pembentukan bentuk normal 1, 2 dan 3 dengan melakukan proses normalisasi.

A. Tujuan

Mengupayakan untuk memperoleh skema relasi yang "baik" yaitu untuk mengukur secara formal mengapa satu set pengelompokan attribute menjadi sejumlah skema relasi adalah lebih baik dari yang lain.

B. Petunjuk Informal Dalam Desain Skema Relasi

Empat "ukuran informal" mengenai kualitas desain skema relasi :

- Semantik dari attributes
- Reduksi nilai-nilai yang redundan (rangkap) dalam tuples
- Reduksi nilai-nilai null dalam tuples (record)
- Tidak mempunyai tuples yang aneh (spurious tuples)

C. Semantik dari Attribute

- Berkaitan dengan asumsi mengenai arti (semantic) tertentu yang diasosiasikan terhadap sejumlah attribute yang membentuk suatu skema relasional.
- Semantik (arti) menjelaskan bagaimana menginterpretasikan nilai-nilai attribute yang disimpan dalam suatu tuples dari suatu relasi (bagaimana nilai-nilai attribute dalam suatu tuples berkaitan dengan yang lain).

❖ PETUNJUK 1:

- Desain suatu skema relasional sedemikian rupa sehingga semantik yang dikandungnya mudah untuk untuk dijelaskan.

- Jangan mengkombinasikan attribute-attribute dari sejumlah entity type dan relationship types menjadi satu relasi tunggal.
- Secara intuitif, jika suatu skema berkorespondensi dengan satu entity type atau satu relationship type saja, maka arti yang dikandungnya akan cenderung menjadi lebih jelas.
 → Skema-skema relasi dalam basis data COMPANY yang diberikan dalam contoh-contoh kuliah sebelumnya, merupakan skema relasi yang memenuhi petunjuk-1 di atas.

Dibawah ini diberikan contoh penurunan skema relasional yang menyalahi petunjuk-1 di atas :

- a) EMP_DEPT (ENAME, SSN, BDATE, DNUMBER, DNAME, DMGRSSN)
 ---- mengkombinasikan relasi-2 EMPLOYEE dan DEPARTMENT
- b) EMP_PROJ (SSN, PNUMBER, HOURS, ENAME, PNAME, PLOCATION)
 ---- mengkombinasikan relasi-2 WORKS_ON dan PROJECT

D. Informasi yang Redundan dan Update Anomalies

- Salah satu tujuan dari desain skema adalah untuk meminimumkan pemakaian storage yang dipakai oleh base relations (files).
- Pengelompokan sejumlah attribute menjadi skema-2 relasi yang baik mempunyai dampak yang berarti dalam mengurangi pemakaian storage.
 EMP_DEPT (ENAME, SSN, BDATE, DNUMBER, DNAME, DMGRSSN)

Yang merupakan hasil NATURAL JOIN dari sebagian attribute EMPLOYEE dan DEPARTMENT, dan relasi :

EMP_PROJ (SSN, PNUMBER, HOURS, ENAME, PNAME, PLOCATION)

Yang merupakan modifikasi dari relasi WORKS_ON dengan tambahan attribute dari PROJECT dan EMPLOYEE, akan membutuhkan pemakaian storage yang lebih besar, karena adanya pengulangan (repeating group) dari :

(DName, DNumber, DMgrSSN)

→ dalam relasi EMP_DEPT (untuk setiap employee dalam satu department

Yang sama)

(PName, PLocation)

→ dalam relasi EMP_PROJ (untuk setiap employee yang bekerja dalam satu project yang sama)

Persoalan lain yang lebih serius dari kedua relasi di atas bilamana dijadikan sebagai base relations adalah timbulnya “Update Anomalies”, yang meliputi : (3)

- Insertion anomalies
- Deletion anomalies
- Modification anomalies

** Insertion Anomalies.

Terdiri dari dua :

- Persoalan kecenderungan terjadinya inkonsistensi data.

Sebagai contoh, dalam relasi EMP_DEPT, setiap kali suatu tuple baru ditambahkan, maka attribute-2 dari Department dimana seorang employee bekerja HARUS dituliskan secara tepat. Jika tidak maka akan terjadi nilai-2 yang inkonsisten untuk sejumlah nomor Department yang sama.

- Persoalan kesulitan penyisipan tuple yang baru

Sebagai contoh, masih untuk relasi EMP_DEPT, jika suatu Department telah ada (didefinisikan) tapi belum ada employee di dalamnya, maka satu-satunya cara adalah dengan mengisi nilai-nilai

NULL pada sejumlah attribute untuk employee. Tetapi cara ini menyalahi entity integrity constraint, dimana key dari suatu relasi tidak boleh bernilai NULL.

** Deletion Anomalies.

Anomali ini berkaitan erat dengan persoalan kedua dalam insertion anomalies; dimana untuk kasus relasi EMP_DEPT, jika suatu tuple employee yang merupakan satu-satunya employee untuk suatu Department dihapus, maka informasi mengenai Department akan terhapus dari basis data.

** Modification Anomalies.

Dalam relasi EMP_DEPT, jika nilai dari salah satu attribute employee untuk suatu Department tertentu diubah (misalnya nama department diubah), maka semua tuple employee yang bekerja pada Department tersebut juga harus diubah. Jika ada tuple yang tertinggal tidak diubah, maka akan terdapat dua nama yang berbeda untuk satu department yang sama (yang seharusnya tidak boleh terjadi)

❖ PETUNJUK-2:

- Desain suatu skema relasi dasar (base relation schema) sedemikian rupa sehingga ketiga jenis anomali (insertion, deletion dan modification) tidak akan terjadi.

E. Nilai-nilai NULL dalam tuples

Dalam hasil desain suatu skema relasi, mungkin saja terdapat pengelompokan sejumlah attribute menjadi suatu relasi dengan jumlah attribute yang “besar”. Jika terdapat sejumlah sub-set attributes yang tidak berlaku untuk semua tuple dalam relasi, maka akan terdapat sejumlah nilai-2 NULL dalam sejumlah tuples tersebut yang mengakibatkan:

- Pemborosan storage

- Timbulnya persoalan semantik dari attribute
- Kesulitan dalam merealisasikan operasi Join (kosong dengan kosong)
- Kesulitan dalam merealisasikan fungsi – fungsi agregate (seperti SUM, COUNT, dan AVERAGE)

Selain kesulitan-2 di atas, nilai – nilai NULL dapat memberikan interpretasi jamak (multiple, interpretations) terhadap tuple yang dalamnya terdapat attribute – attribute dengan nilai null :

- Attribute – 2 tersebut tidak terpakai untuk tuple
- Nilai – nilai attribute untuk tuple tidak diketahui
- Nilai – nilainya diketahui, tapi belum tercatat atau tersedia

❖ PETUNJUK 3:

- Sedapat mungkin, hindari penempatan attribute – attribute dalam suatu base relation yang memungkinkan timbulnya nilainya null. Jika nilai – nilai null tidak dapat dihindari, yakinkan bahwa hal tersebut hanya berlaku untuk kasus – kasus khusus dan jangan diberlakukan terhadap sebagian besar dari tuple dalam suatu relasi

→ Sebagai contoh, jika hanya terdapat 10% dari keseluruhan employee yang mempunyai kantor pribadi, maka merupakan suatu cara perancangan yang beralasan apabila satu attribute OFFICE_NUMBER dimasukkan dalam relasi EMPLOYEE; tetapi akan lebih baik apabila dibuatkan satu relasi baru yang terdiri dari dua attribute (ESSN, OFFICE_NUMBER) yang dipakai untuk menyimpan data employee yang mempunyai kantor pribadi.

F. Tuples yang Tidak Dikehendaki (Spurious Tuples)

Untuk ini, seandainya relasi:

EMP_PROJ (SSN, PNumber, Hours, EName, PLocation)

Didekomposisi menjadi dua relasi:

EMP_LOCS (EName, PLocation)

EMP_PROJ1 (SSN, PNUMBER, Hours, PName, PLocation)

Maka, jika kedua relasi hasil dekomposisi di atas diupayakan untuk dilakukan NATURAL JOIN (lewat attribute "PLocation"), akan diperoleh sejumlah tuple yang melebihi (tidak ada) dalam EMP_PROJ.

Keadaan ini dapat terjadi karena terdapat sejumlah tuple hasil JOIN untuk "PLOCATION" yang sama, pasangan tuple "SSn" dan "EName" yang dihasilkan bukan merupakan pasangan yang valid.

Sejumlah tuple yang ada dalam hasil JOIN tetapi tidak ada dalam relasi EMP_PROJ disebut Spurious tuples, yaitu tuples yang tidak dikehendaki dan tidak valid.

❖ PETUNJUK 4:

- Dalam mendesain skema-skema relasi harus diupayakan sehingga skema-skema yang dihasilkan dapat dilakukan JOIN dengan kondisi keamanan (EQUI JOIN atau NATURAL JOIN) pada attribute-attribute yang berupa primary key atau foreign key, dengan cara yang menjamin bahwa spurious tuples tidak akan dihasilkan.

G. KETERGANTUNGAN FUNGSIONAL (FUNCTIONAL DEPENDENCIES)

Konsep Ketergantungan Fungsional merupakan salah satu konsep yang sangat penting dalam desain skema relasional, karena ini dapat secara formal mendefinisikan bentuk-bentuk relasi yang normal (normalisasi data).

H. Definisi Ketergantungan Fungsional

Ketergantungan fungsional merupakan satu constraint antara dua set attribute suatu basis data.

Jika suatu skema basis data relasional dengan n buah attribute dinyatakan dalam bentuk universal:

$$R = \{A_1, A_2, \dots, A_{n-1}, A_n\}$$

Maka ketergantungan fungsional (disingkat FD) antara dua set attribute X dan Y (keduanya subset dari R), dinotasikan $X \rightarrow Y$, menyatakan suatu constraint pada sejumlah tuples yang memungkinkan dapat membentuk "relation instance" r dari R; yaitu:

Untuk sembarang pasangan tuples t_1 dan t_2 dalam r sedemikian rupa sehingga berlaku $t_1[X] = t_2[X]$, maka juga harus berlaku $t_1[Y] = t_2[Y]$. (menyatakan konsep key (FK, PK))

Dari constraint di atas, dapat dikatakan bahwa nilai-nilai komponen tuple dari X dapat secara unik (atau secara fungsional) menentukan nilai-nilai dari komponen Y. Sebaliknya, dapat juga dikatakan bahwa Y secara fungsional tergantung pada X.

Jadi, X secara fungsional menentukan Y dalam suatu skema relasi R dan hanya jika, bilamana dua tuples dari r(R) mempunyai nilai X yang sama, maka kedua tuples ini juga harus mempunyai nilai Y sama

→ Perlu diingat bahwa:

- Jika suatu constraint pada R berlaku bahwa tidak boleh ada lebih dari satu tuple untuk suatu nilai X dalam sembarang relation instance r(R) (yaitu X merupakan candidate key dari R) mengisyaratkan bahwa $X \rightarrow Y$ untuk sembarang subset attribute Y dari R.

- Jika berlaku $X \rightarrow Y$ dalam R, hal ini tidak menyatakan bahwa apakah berlaku atau tidak $Y \rightarrow X$ dalam R.
 - Penggunaan utama dari konsep ketergantungan fungsional adalah untuk memberikan penjelasan lebih jauh suatu skema relasi R dengan menyatakan constraint pada sejumlah atributnya yang harus berlaku pada setiap saat.

Sebagai contoh, perhatikan skema relasi EMP_PROJ:

EMP_PROJ (SSN, Pnumber, Hours, EName, Pname, Plocation)

Yang dari semantik atributnya berlaku ketergantungan fungsional berikut:

- (a) $SSN \rightarrow EName$
- (b) $Pnumber \rightarrow \{Pname, Plocation\}$
- (c) $\{SSN, Pnumber\} \rightarrow Hours$

→ Ketergantungan fungsional merupakan sifat dari skema (intension) relasi R, bukan merupakan keadaan relasi tertentu(extension). Dengan demikian, suatu FD tidak dapat diturunkan secara otomatis dari suatu relasi, tetapi harus didefinisikan secara eksplisit oleh mereka yang mengerti semantik attribute dari relasi R.

I. Aturan-Aturan Penurunan (Inference Rules)

Utk. Fd.

Suatu FD $X \rightarrow Y$ diturunkan dari satu set dependencies F dalam R jika $X \rightarrow Y$ berlaku dalam setiap keadaan relasi r ; yaitu bilamana r memenuhi semua dependencies dalam F, maka $X \rightarrow Y$ juga berlaku dalam r.

Satu set dari semua functional dependencies yang dapat diturunkan dari F disebut: "Closure F+ dari F".

Untuk memperoleh cara yang sistematis dalam menurunkan dependencies, diperlakukan satu set "INFERENCE RULES" yang dapat digunakan untuk menurunkan dependencies yang baru dari satu set dependencies yang diberikan.

- Notasi $F \models X \rightarrow Y$ digunakan untuk menyatakan bahwa functional dependency $X \rightarrow Y$ diturunkan dari satu set FDF.
- Untuk tujuan mempersingkatkan penulisan variable-variabel attribute, digunakan notasi:
 $FD\{X, Y\} \rightarrow Z$ disingkat $XY \rightarrow Z$
 $FD\{X, Y, Z\} \rightarrow \{U, V\}$ disingkat $XYZ \rightarrow UV$

Terdapat 6 aturan(rumus) untuk functional dependencies:

1. Rumus Reflexive:
Jika $X \geq Y$, maka $X \rightarrow Y$
2. Rumus Augmentation:
 $\{X \rightarrow Y\} \models XZ \rightarrow YZ$
3. Rumus Transitif;
 $\{X \rightarrow Y, Y \rightarrow Z\} \models X \rightarrow Z$
4. Rumus Decomposition(Projection):
 $\{X \rightarrow YZ\} \models X \rightarrow Y$
5. Rumus Union(Additive):
 $\{X \rightarrow Y, X \rightarrow Z\} \models X \rightarrow YZ$
6. Rumus Pseudotransitive:
 $\{X \rightarrow Y, WX \rightarrow Z\} \models WX \rightarrow Z$

Rumus 1 s/d 3 dikenal sebagai 'Armstrong's Inference Rules', dimana set dependence F dapat diturunkan hanya dengan menggunakan rumus-rumus.

1s/d3 di atas (telah dibutuhkan oleh Armstrong pada 1974)

→ Pembuktian keenam rumus diatas dibahas dikelas!

Algoritma mencari X^+

(X^+ : closure of X under F)

- Biasanya, perancang basis data pertama mendefinisikan functional dependencies F yang dapat ditentukan dari semantik attribute dalam R
- Functional dependencies tambahan yang juga berlaku dalam R dapat diturunkan dengan menggunakan Armstrong's rule pada F

└─ secara sistematis dapat diperoleh dengan cara, pertama menentukan setiap set attribute X yang muncul disisi sebelah kiri dari FD dalam F yang kemudian dengan menggunakan Armstrong's rule cari semua attribute yang tergantung pada X.

Algoritmanya:

```
X+ := X;

REPEAT

    Oldx+ := x+ ;

    FOR each FD  $Y \rightarrow Z$  dalam F DO

        IF  $Y \subseteq X^+$  THEN  $X^+ := X^+ \cup Z$ ;

UNTIL (oldx+ = x+);
```

Attribute-attribute yang bisa ditentukan disuatu attribute

CONTOH:

Perhatikan skema EMP_PROJ yang mempunyai satu set FD F berikut :

$F = \{ \text{SSN} \rightarrow \text{EName},$

$$\begin{aligned}
 & \text{Pnumber} \rightarrow \{\text{Pname, Plocation}\}, \\
 & \{\text{SSN, PNumber}\} \rightarrow \text{Hours} \\
 & \}
 \end{aligned}$$

Dengan menggunakan algoritma untuk menghitung X^+ dengan berdasarkan pada F , maka diperoleh:

- $\{\text{SSN}\}^+ = \{\text{SSN, EName}\}$
- $\{\text{Pnumber}\}^+ = \{\text{Pnumber, Pname, Plocation}\}$
- $\{\text{SSN, PNumber}\}^+ = \{\text{SSN, PNumber, EName, PName, PLocation, Hours}\}$

$\text{EMP_PROJ} = \{\text{SSN, EName, PNumber, PName, PLocation, Hours}\}$

1. $\text{SSN}^+ = \{\text{SSN}\}$
 $\text{Ssn} \rightarrow \text{Ename} \rightarrow \text{SSN} \leq \text{SSN}^+ \rightarrow \text{SSN}^+ = \{\text{SSN}\} \cup \{\text{Ename}\} = \{\text{SSN, Ename}\}$
2. $\text{Pnumber}^+ = \{\text{Pnumber}\}$
 $\text{Pnumber} \rightarrow \{\text{Pname, Plocation}\} \rightarrow \text{Pnumber} \leq \text{Pnumber}^+ \rightarrow \text{Pnumber}^+ = \{\text{Pnumber, Pname, Plocation}\}$
3. $\{\text{SSN, Pnumber}\}^+ = \rightarrow \{\text{SSN, PNumber}\}$
 - a. $\text{SSN} \rightarrow \text{Ename} \rightarrow x^+ = \{\text{SSN, Ename, Pnumber}\}$
 - b. $\text{Pnumber} \rightarrow \{\text{Pname, Plocation}\} \rightarrow x^+ = \{\text{SSN, Ename, Pnumber, Pname, Plocation}\}$
 - c. $\{\text{SSN, Pnumber}\} \rightarrow \text{Hours} \rightarrow x^+ = \{\text{SSN, Ename, Pnumber, Pname, Plocation}\}$

J. Set Functional Dependencies Yang Ekvivalen

DEFINISI:

Satu set FD E dilingkupi (covered) oleh satu set FD F (F melingkupi E), jika setiap FD dalam E juga ada dalam F ; yaitu E dilingkupi oleh F jika setiap dependency dalam E dapat diturunkan dari F .

→ Dua set functional dependencies E dan F dikatakan ekivalen ($E=F$) jika $E^+ = F^+$. ekivalen berarti bahwa setiap FD dalam E dapat diturunkan dari F, dan setiap FD dalam F dapat diturunkan dari E.

Jadi $E=F$ jika kedua kondisi, yaitu E melingkupi F dan F melingkupi E terpenuhi.

→ Untuk menentukan apakah F melingkupi E dapat dilakukan dengan ;

1. Menghitung x^+ dengan berdasarkan pada F untuk setiap FD $X \rightarrow Y$ dalam E, maka dikatakan F melingkupi E.
2. Periksa apakah attribute – attribute dalam Y ada dalam X^+ . Jika "Ya" untuk setiap FD dalam D maka, dikatakan F melingkupi E.

K. Set Functional Dependencies Yang Minimal

Satu set functional dependencies F dikatakan minimal apabila memenuhi kondisi-kondisi:

- a) Setiap dependency dalam F mempunyai satu atribut tunggal pada sisi kanannya (canonical form).
- b) Sembarang dependency dalam F tidak dapat dihapus dan tetap mempertahankan bahwa satu set FD yang dihasilkan adalah ekivalen dengan F.
- c) Sembarang dependency $X \rightarrow A$ tidak dapat diganti dengan satu dependency $Y \rightarrow A$ (di mana $Y \subset X$), dan tetap menghasilkan FD yang ekivalen dengan F.

Set FD yang minimal di atas dapat dipandang sebagai satu set dependencies dalam bentuk standar (atau canonical) tanpa redundansi.

→ kondisi b) dan c) menjamin bahwa tidak ada redundansi dalam dependencies.

→ kondisi a) menjamin bahwa setiap dependency ada dalam bentuk canonical dengan satu atribut tunggal pada sisi kanannya.

Daftar Pustaka

21. Connolly, Thomas M., Begg, Carolyn E., and Strachan, Anne D., Database System. A practical pproach to Design, Implementation, and Management, Addison Wesley Company, 1996.
22. David M. Kroenke, *Basis data Processing, Fundamentals, Design and Implementation*, 12nd Edition, Prentice-Hall Int'l Edition, 2012



MODUL PERKULIAHAN

Basis Data

Pengenalan Aljabar Relasional & Kalkulus Relasional

Fakultas
Ilmu Komputer

Program Studi
Sistem Informasi

Tatap Muka
13

Kode MK
87010

Disusun Oleh
Tim Dosen

Abstract

Modul ini mempelajari pengenalan aljabar relasional & kalkulus relasional pada basis data.

Kompetensi

Mahasiswa mampu memahami konsep penggunaan aljabar relasional & kalkulus relasional dengan menjelaskan fungsinya serta mampu membuat queri sederhana dengan operasi aljabar relasional dan operasi kalkulus relasional.

A. Pendahuluan Aljabar Relasional

➤ Operasi – Operasi Dasar

- Select
- Project
- Cartesian Product
- Union
- Set Defference

➤ Operasi – Operasi Tambahan

- Natural Join
- Theta Join
- Intersection
- Division

B. SELECT

Memperoleh tupel – tupel dari suatu relasi yang memenuhi predikat tertentu

Simbol : σ (sigma)

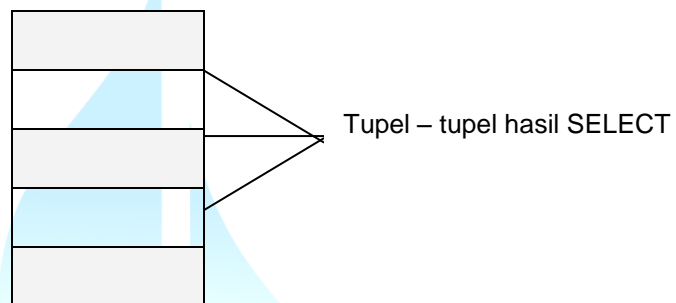
Operasi SELECT akan melibatkan :

Operand : konstanta / bilangan

Operator aritmatika : $<, =, >, \geq, \neq, \leq$

Operator logika : \wedge (and), \vee (or), \neg (not)

Ilustrasi : R





Contoh Query :

Skema relasi mahasiswa (npm, nama, alamat, kota, jkel)

Dicari informasi mengenai mahasiswa yang mempunyai NPM = '50100333'

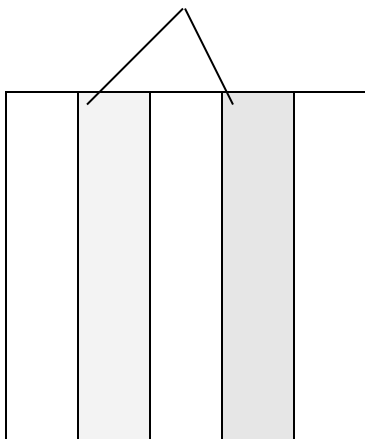
$\sigma_{npm='50100333'}$ (MAHASISWA)

C. PROJECT

Memperoleh atribut – atribut tertentu dari suatu relasi

Simbol : π (pi)

Ilustrasi : Atribut-atribut
 hasil project



Contoh Query :

Skema relasi MAHASISWA (npm, nama, alamat, kota, jkel)

Dicari informasi mengenai nama dan kota mahasiswa

D. CARTESIAN PRODUCT

Membentuk suatu relasi dari dua relasi yang terdiri dari kombinasi tupelo-tupel yang mungkin

Simbol : X (cros)

Ilustrasi :

R	S	R X S	
a	1	a	1
b	2	a	2
	3	a	3
		b	1
		b	2
		b	3

Contoh Queri :

Skema relasi MHS (npm, nama, alamat, tgl_lahir)

Skema relasi MTKULIAH (kd_mk, nama_mk, sks)

MHS X MTKULIAH

E. UNION

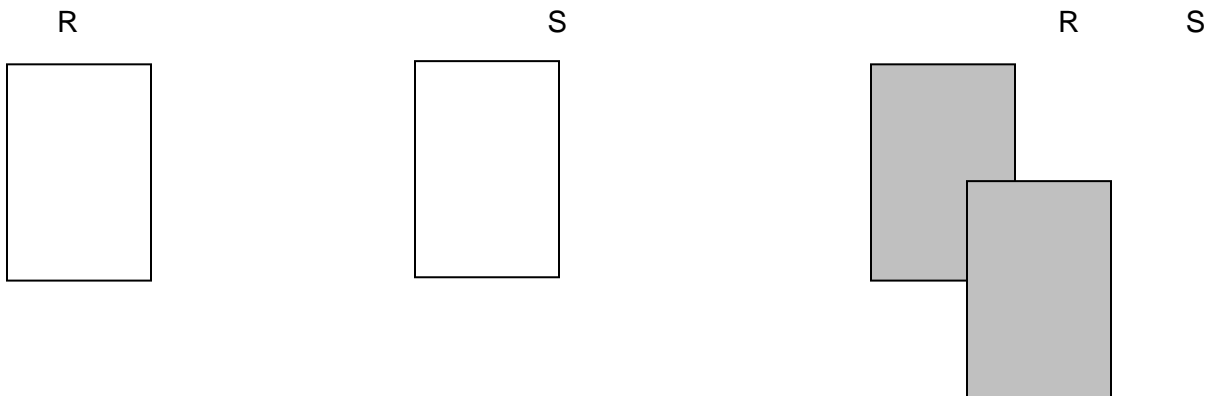
Membentuk suatu relasi yang terdiri dari tupel – tupelo yang berada pada salah satu relasi atau pada kedua relasi, dengan syarat :

Misalnya ada relasi R dan S, maka jumlah atribut relasi R dan S harus sama

Domain dari atribut ke i dari R harus sama dengan domain dari atribut ke i dari S

Simbol : \cup (union)

Ilustrasi :



Contoh Query :

Skema relasi MHS (npm, nama, alamat, tgl_lahir)

Skema relasi MTKULIAH (kd_mk, nama_mk, sks)

Gabungkan data dari relasi MHS dengan data dari relasi MTKULIAH

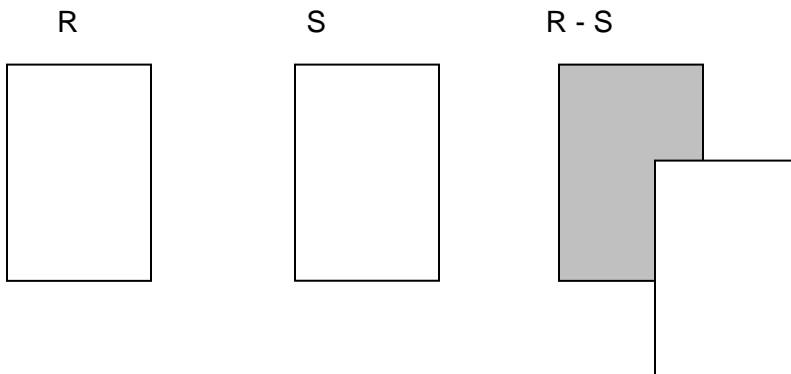
MHS MTKULIAH

F. SET DIFFERENCE

Membentuk suatu relasi yang terdiri dari tupel – tupel yang berada pada relasi pertama dan tidak berada pada relasi kedua atau kedua-duanya

Simbol : - (minus)

Ilustrasi :



Contoh Queri :

Skema relasi MTKULIAH (kd_mk, nama_mk, sks)

Skema relasi NILAI (npm, kd_mk, nil_mid, nil_uas)

Dicari kode mata kuliah yang bersks 2 dan nilai mid untuk kode mata kuliah tersebut dibawah 50

$$\Pi_{kd_mk} (\sigma_{sks = 2}^{(MTKULIAH)}) - \Pi_{kd_mk} (\sigma_{nil_mid > 50}^{(NILAI)})$$

G. NATURAL JOIN

Membentuk suatu relasi dari dua relasi yang terdiri dari kombinasi yang mungkin dari relasi – relasi.

Simbol : \bowtie

Ilustrasi

R		S		R \bowtie S		
A	1	1	X	a	1	X
B	2	1	Y	a	1	Y
		3	Z			

Contoh Query :

Skema relasi MHS (npm, nama, alamat, tgl_lahir)

Skema relasi Nilai (npm, kd_mk, nil_mid, nil_uas)

Dicari nama mahasiswa yang mengambil matakuliah dengan kode matakuliah “KK021”

Π nama ($\sigma_{kd_mk = 'KK021'}$ (MHS \bowtie NILAI))

H. THETA JOIN

Membentuk suatu relasi dari dua relasi yang terdiri dari kombinasi yang mungkin dari relasi – relasi dengan kondisi tertentu

Simbol : \bowtie

Ilustrasi :

R			S		R \bowtie S B < D				
A	B	C	D	E	A	B	C	D	E
1	2	3	4	6	1	2	3	4	6
4	5	6	7	8	1	2	3	7	8
7	8	9			4	5	6	7	8

Contoh Query :

Skema relasi MTKULIAH (kd_mk, nama_mk, sks)

Skema relasi Nilai (npm, kd_mk, nil_mid, nil_uas)

Dicari nama matakuliah yang diambil oleh mahasiswa dengan NPM “50100333” dengan kode matakuliah pada relasi nilai harus sama dengan relasi mtkuliah

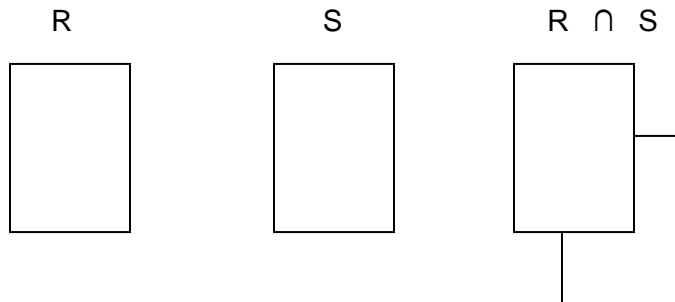
Π nama ($\sigma_{kd_mk = 'KK021' \wedge (mtkuliah.kd_mk = nilai.kd_mk)}$ (MTKULIAH \bowtie NILAI))

I. INTERSECTION

Membentuk suatu relasi yang terdiri atas tupel –tupel yang sama dari dua relasi

Simbol : \cap

Ilustrasi :



Contoh Queri :

Skema relasi MTKULIAH (kd_mk, nama_mk, sks)

Skema relasi Nilai (npm, kd_mk, nil_mid, nil_uas)

Dicari kode matakuliah yang mempunyai sks = 4 yang ambil oleh mahasiswa dengan NPM
"50100333"

$$\Pi_{kd_mk} (\sigma_{sks = 4} (MTKULIAH)) \cap \Pi_{kd_mk} (\sigma_{npm = '50100333'} (NILAI))$$

J. DIVISION

Untuk memndapatkan nilai yang ada pada salah satu atribut dari relasi ' pembilang ' yang nilai atributnya sama dengan nilai atribut relasi ' penyebut '

Simbol : \div

Ilustrasi :

R				S		R ÷ S		R	
a	b	c	d	c	d	a	b	R	
a	b	e	f	e	f	e	d	÷	S
b	c	e	f					S	
e	d	c	d					Sisa bagi	
e	d	e	f						
a	b	d	e						

Contoh Query :

Skema relasi MHS (npm, nama, alamat, tgl_lahir)

Skema relasi NILAi (npm, kd_mk, nil_mid, nil_uas)

Dicari matakuliah yang diambil oleh mahasiswa yang bernama "SADIKIN"

$\Pi_{kd_mk, npm} (NILAi) \cap \Pi_{npm} (\sigma_{nama = 'SADIKIN'} (MHS))$

Daftar Pustaka

23. Connolly, Thomas M., Begg, Carolyn E., and Strachan, Anne D., Database System. A practical pproach to Design, Implementation, and Management, Addison Wesley Company, 1996.
24. David M. Kroenke, *Basis data Processing, Fundamentals, Design and Implementation*, 12nd Edition, Prentice-Hall Int'l Edition, 2012



MODUL PERKULIAHAN

Basis Data

Keamanan Sistem Basis Data

Fakultas
Ilmu Komputer

Program Studi
Sistem Informasi

Tatap Muka
14

Kode MK
87010

Disusun Oleh
Tim Dosen

Abstract

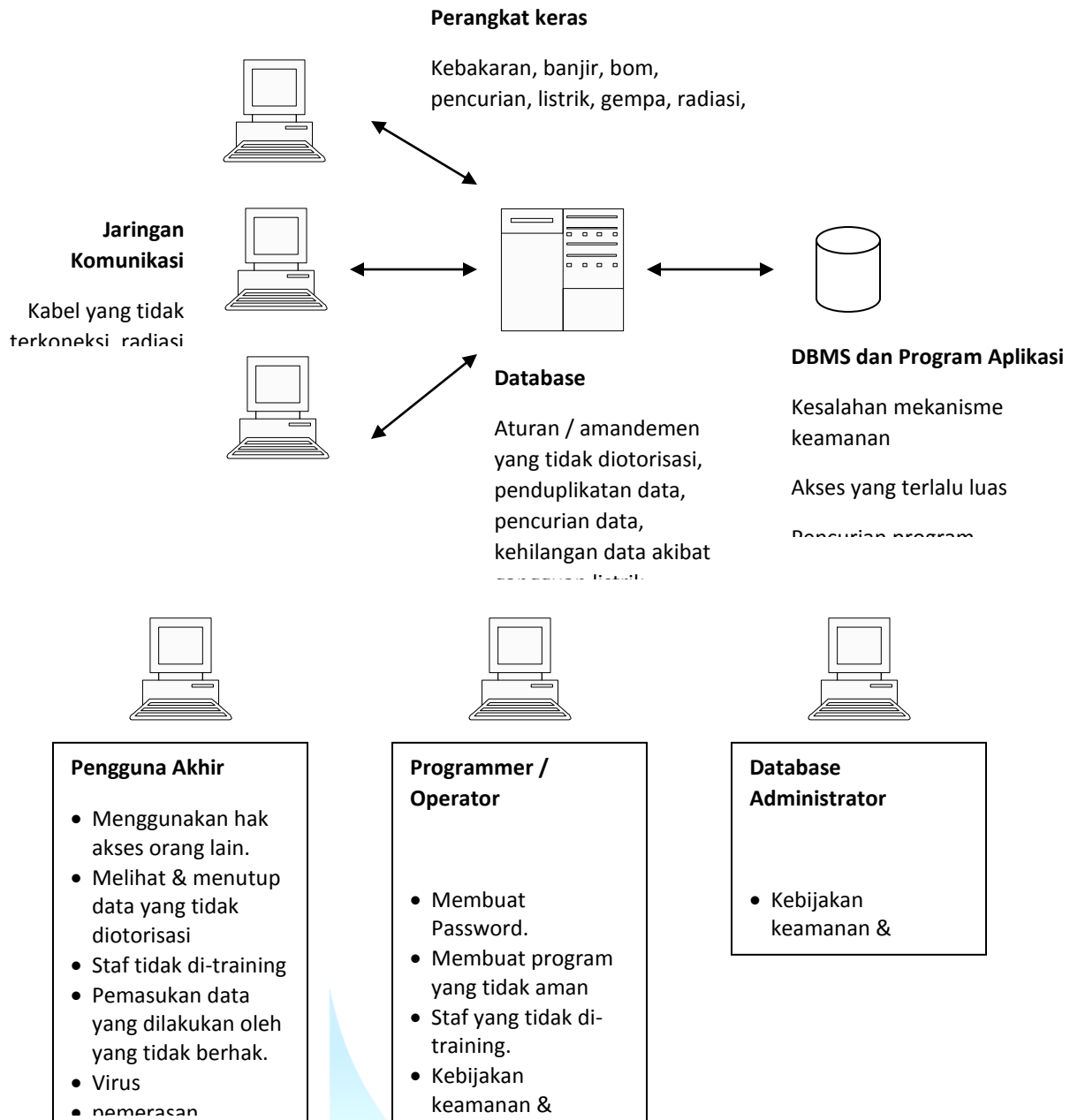
Modul ini mempelajari keamanan sistem basis data.

Kompetensi

Mahasiswa mampu menjelaskan lingkup keamanan basis data serta mampu mengidentifikasi tipe ancaman yang dapat mempengaruhi keamanan sistem basis data.

A. Definisi Keamanan Sistem Basis Data

Keamanan merupakan suatu proteksi terhadap pengrusakan data dan pemakaian data oleh pemakai yang tidak punya kewenangan.



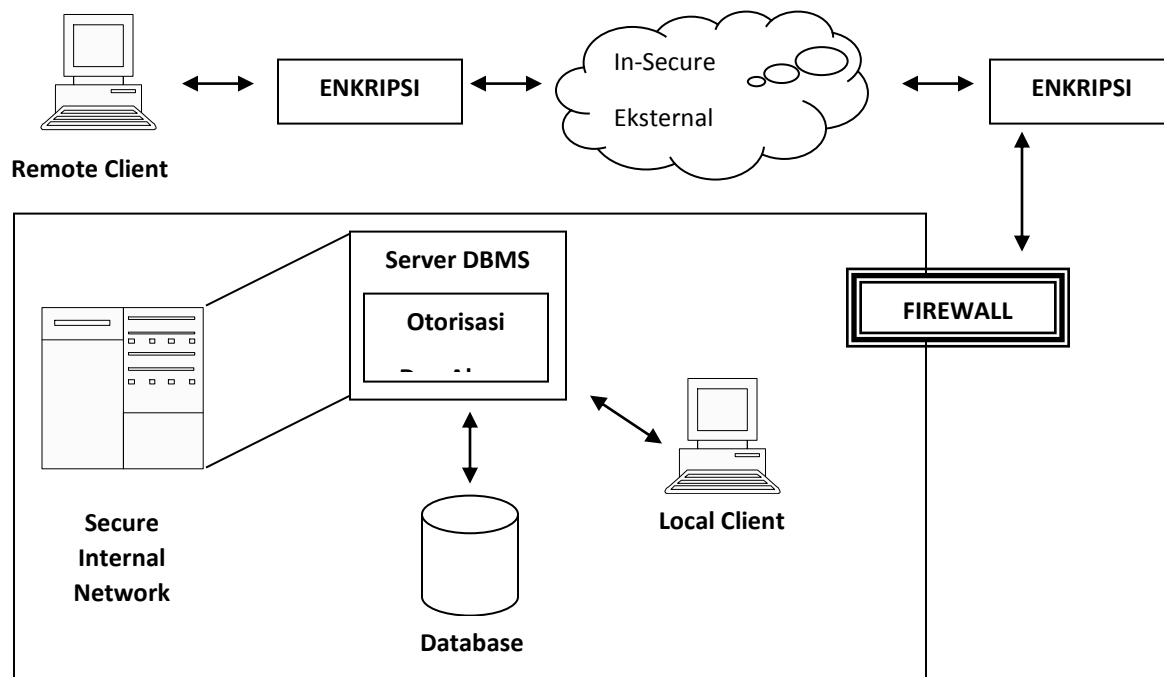


B. Penyalahgunaan Database

1. Tidak disengaja, jenisnya :
 - a. kerusakan selama proses transaksi
 - b. anomali yang disebabkan oleh akses database yang konkuren
 - c. anomali yang disebabkan oleh pendistribusian data pada beberapa komputer
 - d. logika error yang mengancam kemampuan transaksi untuk mempertahankan konsistensi database.
2. Disengaja, jenisnya :
 - a. Pengambilan data / pembacaan data oleh pihak yang tidak berwenang.
 - b. Pengubahan data oleh pihak yang tidak berwenang.
 - c. Penghapusan data oleh pihak yang tidak berwenang.

C. Tingkatan Pada Keamanan Database

1. Fisikal → lokasi-lokasi dimana terdapat sistem komputer haruslah aman secara fisik terhadap serangan perusak.
2. Manusia → wewenang pemakai harus dilakukan dengan berhati-hati untuk mengurangi kemungkinan adanya manipulasi oleh pemakai yang berwenang
3. Sistem Operasi → Kelemahan pada SO ini memungkinkan pengaksesan data oleh pihak tak berwenang, karena hampir seluruh jaringan sistem database menggunakan akses jarak jauh.
4. Sistem Database → Pengaturan hak pemakai yang baik.



D. Keamanan Data

1. Otorisasi :

- Pemberian Wewenang atau hak istimewa (priviledge) untuk mengakses sistem atau obyek database
- Kendali otorisasi (=kontrol akses) dapat dibangun pada perangkat lunak dengan 2 fungsi :
- Mengendalikan sistem atau obyek yang dapat diakses
- Mengendalikan bagaimana pengguna menggunakannya
- Sistem administrasi yang bertanggungjawab untuk memberikan hak akses dengan membuat account pengguna.

2. Tabel View :

- Merupakan metode pembatasan bagi pengguna untuk mendapatkan model database yang sesuai dengan kebutuhan perorangan. Metode ini dapat menyembunyikan data yang tidak digunakan atau tidak perlu dilihat oleh pengguna.
- Contoh pada Database relasional, untuk pengamanan dilakukan beberapa level :
 1. Relasi → pengguna diperbolehkan atau tidak diperbolehkan mengakses langsung suatu relasi
 2. View → pengguna diperbolehkan atau tidak diperbolehkan mengakses data yang terapat pada view
 3. Read Authorization → pengguna diperbolehkan membaca data, tetapi tidak dapat memodifikasi.
 4. Insert Authorization → pengguna diperbolehkan menambah data baru, tetapi tidak dapat memodifikasi data yang sudah ada.
 5. Update Authorization → pengguna diperbolehkan memodifikasi data, tetapi tidak dapat menghapus data.
 6. Delete Authorization → pengguna diperbolehkan menghapus data.
- Untuk Modifikasi data terdapat otorisasi tambahan :
 1. Index Authorization → pengguna diperbolehkan membuat dan menghapus index data.
 2. Resource Authorization → pengguna diperbolehkan membuat relasi-relasi baru.
 3. Alteration Authorization → pengguna diperbolehkan menambah/menghapus atribut suatu relasi.
 4. Drop Authorization → pengguna diperbolehkan menghapus relasi yang sudah ada.
- Contoh perintah menggunakan SQL :

GRANT : memberikan wewenang kepada pemakai

Syntax : GRANT <priviledge list> ON <nama relasi/view> TO <pemakai>

Contoh :

```
GRANT SELECT ON S TO BUDI
```

```
GRANT SELECT,UPDATE (STATUS,KOTA) ON S TO ALI,BUDI
```

REVOKE : mencabut wewenang yang dimiliki oleh pemakai

Syntax : REVOKE <priviledge list> ON <nama relasi/view> FROM <pemakai>

Contoh :

```
REVOKE SELECT ON S TO BUDI
```

```
REVOKE SELECT,UPDATE (STATUS,KOTA) ON S TO ALI,BUDI
```

Priviledge list : READ, INSERT, DROP, DELETE, INEX, ALTERATION, RESOURCE

3. Backup data dan recovery :

Backup : proses secara periodik untuk mebuat duplikat ari database dan melakukan logging file (atau program) ke media penyimpanan eksternal.

Jurnaling : proses menyimpan dan mengatur log file dari semua perubahan yang dibuat di database untuk proses recovery yang efektif jika terjadi kesalahan.

Isi Jurnal :

- Record transaksi
 1. Identifikasi dari record
 2. Tipe record jurnal (transaksi start, insert, update, delete, abort, commit)
 3. Item data sebelum perubahan (operasi update dan delete)
 4. Item data setelah perubahan (operasi insert dan update)
 5. Informasi manajemen jurnal (misal : pointer sebelum dan record jurnal selanjutnya untuk semua transaksi)
- Record checkpoint : suatu informasi pada jurnal untuk memulihkan database dari kegagalan, kalau sekedar redo, akan sulit penyimpanan sejauh mana jurnal untuk mencarinya kembali, maka untuk membatasi pencarian menggunakan teknik ini.

Recovery : merupakan upaya uantuk mengembalikan basis data ke keadaaan yang dianggap benar setelah terjadinya suatu kegagalan.

4. Jenis Pemulihan :

1. Pemulihan terhadap kegagalan transaksi : Kesatuan prosedur alam program yang dapat mengubah / memperbarui data pada sejumlah tabel.

2. Pemulihan terhadap kegagalan media : Pemulihan karena kegagalan media dengan cara mengambil atau memuat kembali salinan basis data (backup)
3. Pemulihan terhadap kegagalan sistem : Karena gangguan sistem, hang, listrik terputus alirannya.

Fasilitas pemulihan pada DBMS :

1. Mekanisme backup secara periodik
2. fasilitas logging dengan membuat track pada tempatnya saat transaksi berlangsung dan pada saat database berubah.
3. fasilitas checkpoint, melakukan update database yang terbaru.
4. manager pemulihan, memperbolehkan sistem untuk menyimpan ulang database menjadi lebih konsisten setelah terjadinya kesalahan.

Teknik Pemulihan :

1. deferred upate / perubahan yang ditunda : perubahan pada DB tidak akan berlangsung sampai transaksi ada pada poin disetujui (COMMIT). Jika terjadi kegagalan maka tidak akan terjadi perubahan, tetapi diperlukan operasi redo untuk mencegah akibat dari kegagalan tersebut.
2. Immediate Upadate / perubahan langsung : perubahan pada DB akan segera tanpa harus menunggu sebuah transaksi tersebut disetujui. Jika terjadi kegagalan diperlukan operasi UNDO untuk melihat apakah ada transaksi yang telah disetujui sebelum terjadi kegagalan.
3. Shadow Paging : menggunakan page bayangan imana paa prosesnya terdiri dari 2 tabel yang sama, yang satu menjadi tabel transaksi dan yang lain digunakan sebagai cadangan. Ketika transaksi mulai berlangsung kedua tabel ini sama dan selama berlangsung tabel transaksi yang menyimpan semua perubahan ke database, tabel bayangan akan digunakan jika terjadi kesalahan. Keuntungannya adalah tidak membutuhkan REDO atau UNDO, kelemahannya membuat terjadinya fragmentasi.

5. Kesatuan data dan Enkripsi :

- Enkripsi : keamanan data
- Integritas :metode pemeriksaan dan validasi data (metode integrity constrain), yaitu berisi aturan-aturan atau batasan-batasan untuk tujuan terlaksananya integritas data.
- Konkuren : mekanisme untuk menjamin bahwa transaksi yang konkuren pada database multi user tidak saling mengganggu operasinya masing-masing. Adanya penjadwalan proses yang akurat (time stamping).

Daftar Pustaka

25. Connolly, Thomas M., Begg, Carolyn E., and Strachan, Anne D., Database System. A practical pproach to Design, Implementation, and Management, Addison Wesley Company, 1996.
26. David M. Kroenke, *Basis data Processing, Fundamentals, Design and Implementation*, 12nd Edition, Prentice-Hall Int'l Edition, 2012



MODUL PERKULIAHAN

Basis Data

Pengenalan Objek Sistem Manajemen Basis Data

Fakultas
Ilmu Komputer

Program Studi
Sistem Informasi

Tatap Muka
15

Kode MK
87010

Disusun Oleh
Tim Dosen

Abstract

Modul ini mempelajari pengenalan objek sistem manajemen basis data.

Kompetensi

Mahasiswa mampu menjelaskan alasan dibutuhkannya OODB serta mampu menjelaskan konsep yang terkait dengan OODB .

A. Object Oriented Data Base

Objek database mulai populer pada pertengahan tahun 1990 an. Bermula dari Objek Oriented Programming (OOP) yang kemudian dikembangkan menjadi Objek Oriented Database (OOD) dan pada akhirnya menjadi Objek Oriented Analysis (OOA). Didalam konsep objek oriented database kita dapat melakukan pemodelan data dari semua phenomena dan dapat dinyatakan dalam bahasa umum (natural). Objek Oriented Database pada dasarnya merupakan kosep dari pemrograman berorientasi objek secara umum ditambah dengan database sebagai media penyimpanan datanya yang berbentuk class-class, sehingga dalam hal ini masih berhubungan erat dengan E-R Model. Objek Oriented Database muncul karena kekomplekan dari penyimpanan objek-objek yang akan disimpan didalam database sehingga konsep dari Relational Database Manejemen Sistem (RDBMS) masih tetap digunakan. Mekanisme penyimpanan objek-objek didalam Relational Database Menejemen Sistem ini sering dikenal dengan istilah ORDBMS (Objek Relational Database Managemen System).

Object oriented database adalah sebuah model basisdata dimana informasi disimpan daam bentuk object. Object yang dimaksud tersebut digunakan dalam OOP (object oriented programming). Ketika kemampuan basisdata bergabung dengan kemampuan OOP, hasilnya berupa object database management sistem (ODBMS). ODBMS ideal untuk pada pemrogaman object oriented karena mereka ketika malakukan proses developer, controller ddan model mamiliki persamaan yaitu sama – sama menggunakan object. ODBMS mendukung data yang kompleks seperti vidio, suara, gambar, dll secara native. Berbeda dengan Relational Databse Mangement Sistem (RDBMS) yang tidak native mendukung data kompleks karena harus membagi menjasi dua bagian yaitu : model basis data dan model aplikasi. Bagi yang telah mengenal RDBMS, object pada ODBMS bertindak sama dengan tabel RDBMS. ODBMS dibuat untuk menggantikan RDBMS jika bahasa pemrograman yang digunakan adalah OOP.

Kelebihan OODB antara

lain:

1. Proses penyimpanan dan pengambilan data lebih sederhana;
2. Program mengakses data dengan objeknya secara langsung sehingga kinerja program akan lebih tinggi.

Kekurangan OODB antara lain :

1. Kekurangan dukungan platform, kebanyakan OODB hanya mendukung bahasa pemrograman C++, C# dan Java saja;
2. Kebutuhan keterampilan, dikarenakan OODB masih tergolong baru dan masih relatif jarang penggunaanya;
3. Sulit bermigrasi, dibutuhkan komitmen yang kuat dalam memilih DBMS yang akan digunakan, sekali bermigrasi ke OODB, akan sulit untuk kembali ke RDBMS

B. PERKEMBANGAN APLIKASI BASIS DATA

1. Computer-Aided Design (CAD)

Database CAD menyimpan data yang berhubungan dengan rancangan mekanik dan elektrik, sebagai contoh : gedung, pesawat, dan chips IC.

2. Computer-Aided Manufacturing (CAM)

Database CAM menyimpan data yang jenisnya sama dengan sistem CAD, ditambah data yang berhubungan dengan produksi yang mempunyai ciri-ciri tersendiri (seperti mobil pada saat perakitan) dan produksi yang continue (seperti sintesa kimia).

3. Computer-Aided Software Engineering (CASE)

Database CASE menyimpan data yang berhubungan dengan langkah-langkah dari siklus pengembangan software yaitu : planning, requirements collection analysis, design, implementation, test, maintenance and documentation.

4. Computer-Aided Publishing (CAP)

Database CAP menyimpan dokumen yang kompleks. Sama seperti otomatisasi kantor, aplikasi CAP telah diperluas untuk menangani dokumen-dokumen multimedia yang berisikan teks, audio, gambar, video data, dan animasi.

5. Office Automation (OA)

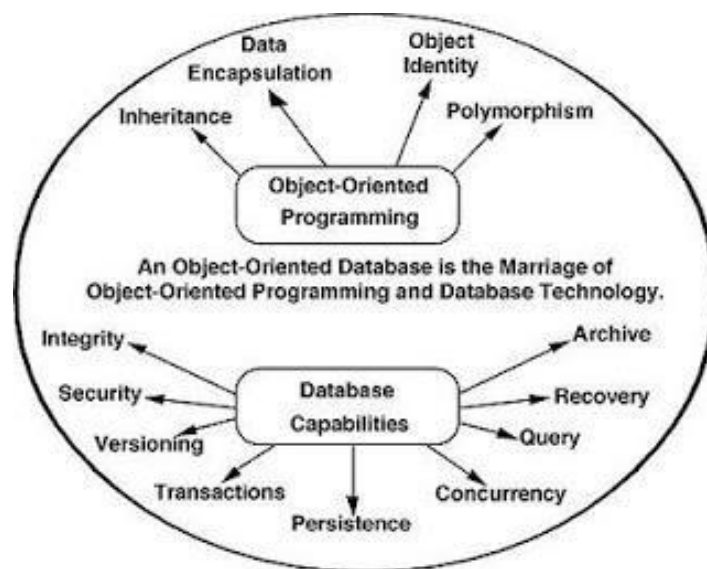
Database OA menyimpan data yang berhubungan dengan pengontrolan informasi computer dalam bidang bisnis, termasuk e-mail, dokumen-dokumen, invoice, dsb.

Agar menyediakan dukungan yang lebih baik untuk area ini, dibutuhkan penanganan yang lebih luas terhadap jenis data daripada nama, alamat, tanggal dan uang.

Sekarang ini sistem yang modern dapat menangani text yang berjenis bebas, foto, diagram, audio dan video. Sebagai contoh: dokumen multimedia yang mengangani teks, foto, spreadsheets dan suara.

C. KONSEP OBJECT ORIENTED

Dalam Object Oriented baik Programming maupun database terdapat beberapa konsep mengenai object oriented yang perlu diketahui. Berikut adalah penjelasannya.



Beberapa konsep Object Oriented Database :

- Kenyataan dalam dunia ini direpresentasikan sebagai object
- Setiap object memiliki state dan behavior
- State merupakan nilai dari properties dan atribut dari object
- Behavior merupakan method yang dijalankan oleh state
- Object yang memiliki kesamaan state dan behavior dikelompokkan dalam satu class yang sama
- Setiap object hanya dapat diturunkan (instace) kedalam satu class saja
- Class-class dikeompokkan dalam sebuah hierarki. Subclass memiliki turunan (inherits) dari superclass-nya.
- Dimungkinkan juga terjadi overriding dimana sebuah class men-substitute domain dari propertiesnya dengan method lain dalam implementasi yang berbeda.

D. Abstraksi dan Enkapsulasi

• **Abstraksi**

adalah proses identifikasi aspek-aspek yang perlu dari entitas dan mengabaikan property yang tidak penting.

• **Enkapsulasi (information hiding)**

adalah memisahkan aspek-aspek eksternal sebuah objek dari rincian internalnya (internal details), yang tidak terlihat dari dunia luar. Dengan cara ini, internal detail sebuah objek dapat dirubah tanpa mempengaruhi aplikasi yang menggunakan objek tersebut, begitu juga dengan external detail. Dengan kata lain, encapsulation menyediakan data independence.

Contoh : Objek Roti, objek ini mempunyai method Pembuatan Roti. Jika kita ingin memakan roti, tentu kita tidak perlu tahu bagaimana cara membuatnya. Dengan demikian pembuatan Roti menjadi sesuatu yang menjadi dasar bagi konsep information hiding.



E. Objek dan Atribut

- **Objek**

adalah sebuah entitas yang dapat diidentifikasi secara unik, berisikan atribut- atribut yang menerangkan keadaan atau kondisi (state) objek dunia nyata (real world object) dan aksi-aksi yang berhubungan dengan sebuah objek dunia nyata. Definisi objek serupa dengan definisi entitas. Perbedaannya : objek menunjukkan keadaan (state) dan tingkah laku (behaviour), sedangkan entitas menunjukkan models state.

- **Atribut**

adalah nama-nama property dari sebuah kelas yang menjelaskan batasan nilainya dari property yang dimiliki oleh sebuah kelas tersebut. Atribut dari sebuah kelas mempresentasikan property-property yang dimiliki oleh kelas tersebut. Atribut mempunyai tipe yang menjelaskan tipe instanisasinya. Hanya sebuah instanisasi dari kelas (objek) yang dapat mengubah nilai dari atributnya.

Keadaan (state) dari sebuah objek dijelaskan dengan nilai dari atribut-atribut yang dimilikinya (selain keberadaan hubungan dengan objek lainnya). Dalam sebuah kelas atribut hanya dinyatakan keberadaan dan batasan nilainya saja, sedangkan dalam sebuah objek atributnya sudah dinyatakan nilai dan menjelaskan kedudukan / keadaan dari objek tersebut.

F. Identitas Objek (Object Identity)

Pada saat objek dibuat, object identifier (OID) langsung ditentukan. OID tersebut unik dan berbeda. OID membedakan objek yang satu dengan objek lainnya di dalam sistem. Sekali objek dibuat, OID tersebut tidak dapat digunakan kembali untuk objek-objek lainnya, walaupun objek tersebut telah dihapus.



G. Metode dan Pesan

- **Metode (Methods)**

Dalam teknologi objek, function biasanya disebut methods. Methods mendefinisikan tingkah laku dari sebuah objek. Methods dapat digunakan untuk merubah kondisi objek dengan memodifikasi nilai atribut-atributnya, atau meng-query nilai atribut yang diseleksi.

- **Pesan (Message)**

Message mempunyai arti komunikasi antara objek. Sebuah message merupakan permintaan sederhana dari suatu objek (pengirim) ke objek lain (penerima) dan menanyakan objek tsb untuk mengeksekusi salah satu method-nya. Pengirim dan penerima bisa pada objek yang sama. Notasi 'dot' biasanya digunakan untuk mengakses sebuah method.

E. CLASS

Class merupakan pendefinisian himpunan objek yang sejenis. Objek yang mempunyai atribut yang sama dan meresponse message yang sama dapat dikelompokkan bersama membentuk sebuah class. Atribut dan method yang berhubungan cukup sekali saja didefinisikan untuk class, daripada didefinisikan terpisah untuk setiap objek.

Contoh :

seluruh objek cabang dideskripsikan oleh sebuah class cabang (branch). Objek-objek pada sebuah class disebut instance dari class. Setiap instance mempunyai nilainya sendiri untuk setiap atribut, tetapi nama atribut dan method-nya sama seperti instance lainnya dari sebuah class.



F. SUBCLASS, SUPERCLASS dan INHERITANCE

Inheritance memungkinkan satu class objek didefinisikan sebagai kasus spesial (special case) dari sebuah class pada umumnya. Special case ini dikenal dengan subclass, dan kasus umum lainnya dikenal sebagai superclass. Proses pembentukan superclass sama seperti generalization, sedangkan subclass seperti specialization. Konsep dari superclass, subclass, dan inheritance sama seperti EER, kecuali dalam paradigm object-oriented, inheritance meliputi state dan behaviour.

Ada beberapa bentuk inheritance :

1. Single inheritance

Subclass merupakan turunan dari satu superclass.

Contoh : subclass Manager dan Sales_Staff merupakan turunan property dari superclass Staff.

2. Multiple inheritance

Subclass Sales_Manager merupakan turunan dari superclass Manager dan Sales_Staff.



3. Repeated inheritance

Kasus spesial dari multiple inheritance, dimana sebuah superclass merupakan turunan dari sebuah superclass biasa. Melanjutkan contoh multiple inheritance, class Manager dan Sales_staff bisa saja merupakan turunan dari superclass biasa yaitu superclass Staff. Dalam kasus ini, mekanisme inheritance harus meyakinkan bahwa class Sales_manager tidak diturunkan sebanyak dua kali dari superclass Staff.

4. Selective inheritance

Mengizinkan subclass menurunkan sejumlah property dari superclass. Keistimewaan ini secara fungsional sama seperti mekanisme view, dengan membatasi akses ke beberapa detail tapi tidak seluruhnya.

Object Oriented Database (OOD) merupakan salah satu jenis database dimana data direpresentasikan dalam bentuk object. Pendekatan ini sangat dipengaruhi oleh bahasa pemrograman object-oriented dan dapat dipahami sebagai usaha untuk menambah fungsionalitas DBMS pada lingkup bahasa pemrograman.

Daftar Pustaka

27. Connolly, Thomas M., Begg, Carolyn E., and Strachan, Anne D., Database System. A practical pproach to Design, Implementation, and Management, Addison Wesley Company, 1996.
28. David M. Kroenke, *Basis data Processing, Fundamentals, Design and Implementation*, 12nd Edition, Prentice-Hall Int'l Edition, 2012