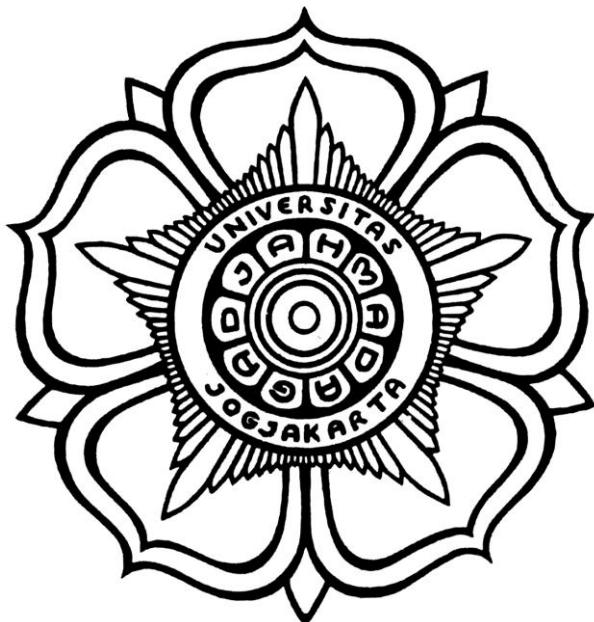


BUKU PETUNJUK (MODUL) PRAKTIKUM

BASIS DATA (MII2502)



**LABORATORIUM KOMPUTER DASAR
JURUSAN ILMU KOMPUTER DAN ELEKTRONIKA
Fakultas Matematika dan Ilmu Pengetahuan Alam
Universitas Gadjah Mada
Yogyakarta
2017**

TATA TERTIB PRAKTIKAN

1. Masuk sesuai jadwal/waktu yang telah ditentukan/disepakati bersama, terlambat lebih dari 15 menit dipersilahkan tidak memasuki ruang LABORATORIUM KOMPUTER DASAR.
2. Praktikan yang tidak mengikuti praktikum lebih dari 3 kali tanpa keterangan resmi, tidak berhak mengikuti responsi/ujian.
3. Tidak diadakan ujian susulan baik ujian tengah semester maupun akhir semester, kecuali atas persetujuan dosen/pengampu mata kuliah bersangkutan.
4. Memakai Pakaian rapi dan sopan:
 - **Pria** : - Kemeja Lengan panjang atau pendek
- Celana panjang Rapi
- Bersepatu
- Tidak boleh memakai T-shirt tanpa krah atau tanpa lengan
 - **Wanita** : - Kemeja Lengan panjang/pendek (tidak ketat dan atau transparan)
- Rok ATAU Celana panjang (tidak ketat dan atau transparan)
- Bersepatu
- Tidak boleh memakai T-shirt tanpa krah atau tanpa lengan
5. Mahasiswa tidak diperbolehkan merokok, makan dan minum pada saat kuliah praktikum.
6. Tas dan perlengkapan lain harus diletakkan pada tempat yang telah disediakan (hanya diperbolehkan membawa barang berharga, Ex: Handphone, dompet dan alat yang diperlukan untuk praktikum)
7. Barang berharga milik peserta kuliah praktikum menjadi tanggung jawab sendiri (laboran tidak bertanggungjawab atas kehilangan barang tersebut).
8. Dering HP harus dimatikan (silent) pada saat kuliah praktikum.
9. Selesai Praktikum, Komputer dimatikan dan kursi dirapikan kembali.
10. Mahasiswa diwajibkan menjaga kebersihan dan ketertiban serta ketenangan belajar.
11. Mahasiswa tidak diperbolehkan menggunakan komputer untuk bermain games.

12. Mahasiswa tidak diperkenankan men-install program/software tanpa petugas lab.
13. Mahasiswa tidak diperkenankan memindah posisi hardware (mouse, keyboard, monitor, CPU)
14. Mahasiswa tidak diperbolehkan membawa atau mengambil (secara sengaja atau tidak sengaja) perlengkapan praktikum yang ada di Laboratorium komputer).
15. Mahasiswa wajib menjaga keutuhan semua peralatan yang ada di Laboratorium Komputer serta tidak diperbolehkan memakai komputer Pengajar/Instruktur.
16. Mengisi daftar hadir dan mencatat nomor kursi/komputer yang digunakan selama praktikum berlangsung ke dalam form daftar hadir yang telah disediakan.
17. Melaporkan keadaan komputer dan atau peralatan yang digunakan (yang rusak/tidak berfungsi) sebelum, sesaat dan atau sesudah penggunaan ke Pengajar/Instruktur atau kepada laboran.
18. Praktikan wajib mematikan Komputer yang telah selesai digunakan dan merapikan kembali kursi, meja dan perlengkapan pendukung praktikum lainnya setelah praktikum selesai dilaksanakan/berakhir.
19. Laboran berhak mencatat, memberikan sanksi atau melakukan tindakan seperlunya terhadap praktikan yang melanggar tata tertib.

LABORATORIUM KOMPUTER DASAR

DIKE F.MIPA UGM

KATA PENGANTAR

Assalamu'alaikum Wr. Wb.

Alhamdullilah, atas berkat rahmat Allah Yang Maha Kuasa dengan didorongkan oleh keinginan luhur memperluas wawasan dalam pengembangan pengetahuan tentang Basis Data, maka buku petunjuk (modul) praktikum ini disusun/dibuat.

Buku petunjuk (modul) Praktikum Basis Data 1 ini dibuat untuk membantu jalannya Praktikum Basis Data 1 prodi S1 Ilmu Komputer. Buku petunjuk (modul) ini dibuat sedemikian rupa sehingga dapat dengan mudah dipahami dan dipelajari oleh mereka yang belum pernah mengenal basis data sekalipun, dan bagi mereka yang pernah mengenal membaca buku ini akan menyegarkan ingatan.

Terima kasih kami ucapkan kepada semua pihak yang telah membantu dan mendukung pembuatan buku ini.

Wassalamu'alaikum Wr. Wb.

LABORATORIUM KOMPUTER DASAR
DIKE F.MIPA UGM

DAFTAR ISI

BAB I ENTITY RELATIONSHIP.....	1
1.1 Tujuan Praktikum.....	1
1.2 Materi, Contoh Dan Ilustrasi.....	1
1.3 Aktivitas Mahasiswa.....	6
1.4 Latihan (Dikerjakan Di Kelas).....	6
BAB II NORMALISASI	7
2.1 Tujuan Praktikum.....	7
2.2 Materi, Contoh Dan Ilustrasi.....	7
2.3 Aktivitas Mahasiswa.....	12
2.4 Latihan/Tugas	13
BAB III DATA DEFINTION LANGUAGE (DDL)	14
3.1 Tujuan Praktikum.....	14
3.2 Materi, Contoh Dan Ilustrasi.....	14
3.3 Aktivitas Mahasiswa	23
3.4 Latihan/Tugas	24
BAB IV DATA MANIPULATION LANGUAGE (DML)	25
4.1. Tujuan Praktikum.....	25
4.2. Materi, Contoh Dan Ilustrasi.....	25
4.3 Aktivitas Mahasiswa	48
4.4 Latihan Tugas.....	48
BAB V STORED ROUTINE : PROCEDURE.....	49
5.1 Tujuan Praktikum.....	49
5.2 Materi, Contoh Dan Ilustrasi.....	49
5.3 Aktivitas Mahasiswa	50
5.4 Latihan/Tugas	53

BAB VI STORED ROUTINE : FUNCTION.....	54
6.1 Tujuan Praktikum.....	54
6.2 Materi, Contoh Dan Ilustrasi.....	54
6.3 Aktivitas Mahasiswa.....	54
6.4 Latihan/Tugas	58
BAB VII TRIGGER	59
7.1 Tujuan Praktikum.....	59
7.2 Materi, Contoh Dan Ilustrasi.....	59
7.3 Tugas/Latihan.....	60
BAB VIII TRANSACTION	61
8.1 Tujuan Praktikum.....	61
8.2 Materi, Contoh Dan Ilustrasi.....	61
8.3 Aktivitas Mahasiswa.....	63
8.4 Latihan/Tugas	66
LAMPIRAN PROSES INSTALASI MySQL.....	67
9.1. Tujuan Praktikum.....	67
9.2. Materi, Contoh Dan Ilustrasi.....	67
9.3. Aktivitas Mahasiswa.....	77

BAB I

ENTITY RELATIONSHIP

1.1 TUJUAN PRAKTIKUM

1. Mahasiswa dapat memahami penggunaan simbol ERD.
2. Mahasiswa dapat menggambar ERD berdasarkan kasus yang diberikan.

1.2 MATERI, CONTOH DAN ILUSTRASI

Dalam ERD, terdapat 3 hal penting, yaitu: *Entity set*, Atribut dan *Relational*.

1.2.1 Entity Set

Entity set adalah sesuatu atau objek yang ada di dalam dunia nyata yang berbeda dengan objek lainnya, memiliki atribut penyusun, dan merupakan pembangun suatu sistem. Contoh, manusia yang bekerja di suatu perusahaan adalah sebuah *entity*. *Entity* mempunyai atribut bernilai (*values*), misal: 000-11-3452 merupakan sebuah nomer induk seorang pekerja (atribut nomer induk pekerja). Selain itu, seorang pekerja juga mempunyai tanggal lahir. Di sini tanggal lahir merupakan atribut dari *entity* pekerja yang berkedudukan sejajar dengan atribut nomer induk pekerja.

Entity mempunyai beberapa tipe atribut sebagai berikut:

1. *Simple attribute*: *entity* yang atributnya tidak dapat dibagi menjadi bagian yang lebih kecil.
2. *Composite attribute*: *entity* yang atributnya dapat dibagi menjadi atribut yang lebih kecil. Misalnya: atribut nama bisa dibagi menjadi nama awal, nama tengah, dan nama akhir.
3. *Single-valued attribute*: *entity* yang atributnya hanya dapat berisi satu nilai. Misal nomer induk pekerja.
4. *Multivalued attribute*: *entity* yang atributnya dapat berisi nol, satu atau lebih dari satu nilai. Misalnya: atribut telepon, bisa jadi seorang pekerja mempunyai satu atau lebih no. telepon.
5. Atribut turunan: *entity* yang atributnya dapat diturunkan dari atribut lainnya. Misalnya: atribut umur dapat diketahui dari atribut tanggal lahir dan tanggal pada saat itu.



1.2.2. Atribut

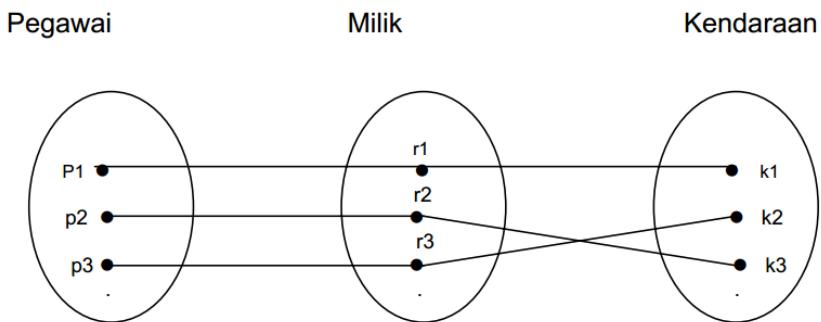
Atribut adalah keterangan-keterangan yang menjelaskan dari suatu entitas, seperti NIM, nama, fakultas, jurusan untuk entitas mahasiswa.

1.2.3 Relational

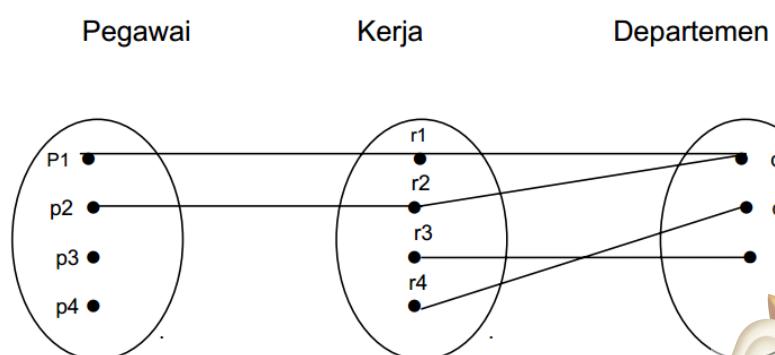
Relational adalah hubungan antara *entity*. Semisal pada contoh di atas *entity* manusia mempunyai hubungan dengan *entity* alamat yaitu "tinggal di". Di dalam merancang *database* hendaknya seluruh *entity* yang ada mempunyai hubungan dengan *entity* yang lain, minimal satu. Jika ada *entity* dalam *database* yang tidak mempunyai hubungan dengan satupun *entity* yang lain, maka akan timbul kesalahan dalam desain. Biasanya *entity* yang tidak berhubungan akan dihilangkan.

Terdapat beberapa jenis relasi, antara lain :

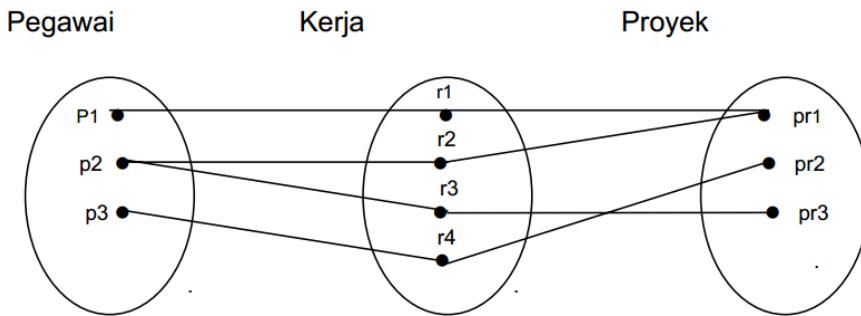
1. One to one



2. One to many/Many to one



3. Many to many



1.2.4. Key

Key adalah sejumlah atribut yang mengidentifikasi record/baris dalam sebuah *relation* secara unik. *Key* terdiri dari:

1. *Super Key* : satu atribut atau gabungan atribut (kolom) pada tabel yang dapat membedakan semua baris secara unik.
2. *Candidate key* : atribut-atribut yang menjadi determinan yang dapat dijadikan identitas record pada sebuah relation, bisa satu atau lebih *candidate key*. *Candidate key* disebut juga dengan *minimal super key*, yaitu *super key* yang tidak mengandung *super key* yang lain. Setiap *candidate key* pasti merupakan *super key*, namun tidak semua *super key* akan menjadi *candidate key*.
3. *Primary key* : *candidate key* yang menjadi identitas record karena dapat mengidentifikasi record secara unik.
4. *Alternate key* : *candidate key* yang tidak dijadikan *primary key*.
5. *Composite key* : *key* yang terdiri dari 2 atribut atau lebih. Atribut-atribut tersebut bila berdiri sendiri tidak menjadi identitas record, tetapi bila dirangkaikan menjadi satu kesatuan dapat mengidentifikasi secara unik.
6. *Foreign key* : *non key* atribut pada sebuah relation yang juga menjadi *key (primary)* atribut di relation lainnya.

Ada beberapa landasan pemuatan *primary key* suatu entitas ke entitas lain yang berhubungan. Landasan ini memakai ciri relasi yang digunakan.

1. *One to one* : entitas A berhubungan dengan entitas B secara *one to one*, maka *primary key* entitas A dimuat ke dalam entitas B atau sebaliknya.
2. *Many to one* : entitas A berhubungan dengan entitas B secara *many to one*, maka *primary key* entitas B dimuat ke dalam entitas A.



3. *Many to many* : entitas A berhubungan dengan entitas B secara *many to many* maka pemuatan *primary key* dari masing-masing entitas akan melibatkan suatu entitas baru.

1.2.5. Entity-Relationship Diagram

E-R diagram digunakan untuk membuat suatu model *database*. Kemudian dari model tersebut dibuatlah sistem *database*. Adapun macam-macam komponen dalam E-R diagram adalah :

Tabel 1.1 Simbol ERD

No	Notasi	Nama	Arti
1		<i>Entity</i>	Objek yang dapat dibedakan dalam dunia nyata
2		<i>Weak Entity</i>	Suatu <i>entity</i> dimana keberadaan dari <i>entity</i> tersebut tergantung dari keberadaan <i>entity</i> yang lain
3		<i>Relationship</i>	Hubungan yang terjadi antara satu atau lebih <i>entity</i>
4		<i>Identifying Relationship</i>	Hubungan yang terjadi antara satu atau lebih <i>weak entity</i>
5		<i>Atribut Simple</i>	<i>Atribut</i> yang bernilai tunggal atau <i>atribut atomic</i> yang tidak dapat dipisah-pisah lagi
6		<i>Atribut Primary Key</i>	Satu atau gabungan dari beberapa <i>atribut</i> yang membedakan semua baris data (<i>row</i>) dalam <i>table</i> secara unik
7		<i>Atribut Composite</i>	<i>Atribut</i> yang masih dapat diuraikan lagi menjadi sub-sub <i>atribut</i> yang masing-masing memiliki makna
8		<i>Atribut Multivalue</i>	Suatu <i>atribut</i> yang memiliki sekelompok nilai untuk setiap <i>instant entity</i>

Ada dua jenis entitas, yang pertama adalah entitas kuat yaitu entitas yang memiliki *primary key*. Kedua adalah entitas lemah yaitu entitas yang tidak memiliki *primary key*. Entitas kuat dan lemah ini akan dibahas lebih dalam pada materi normalisasi.

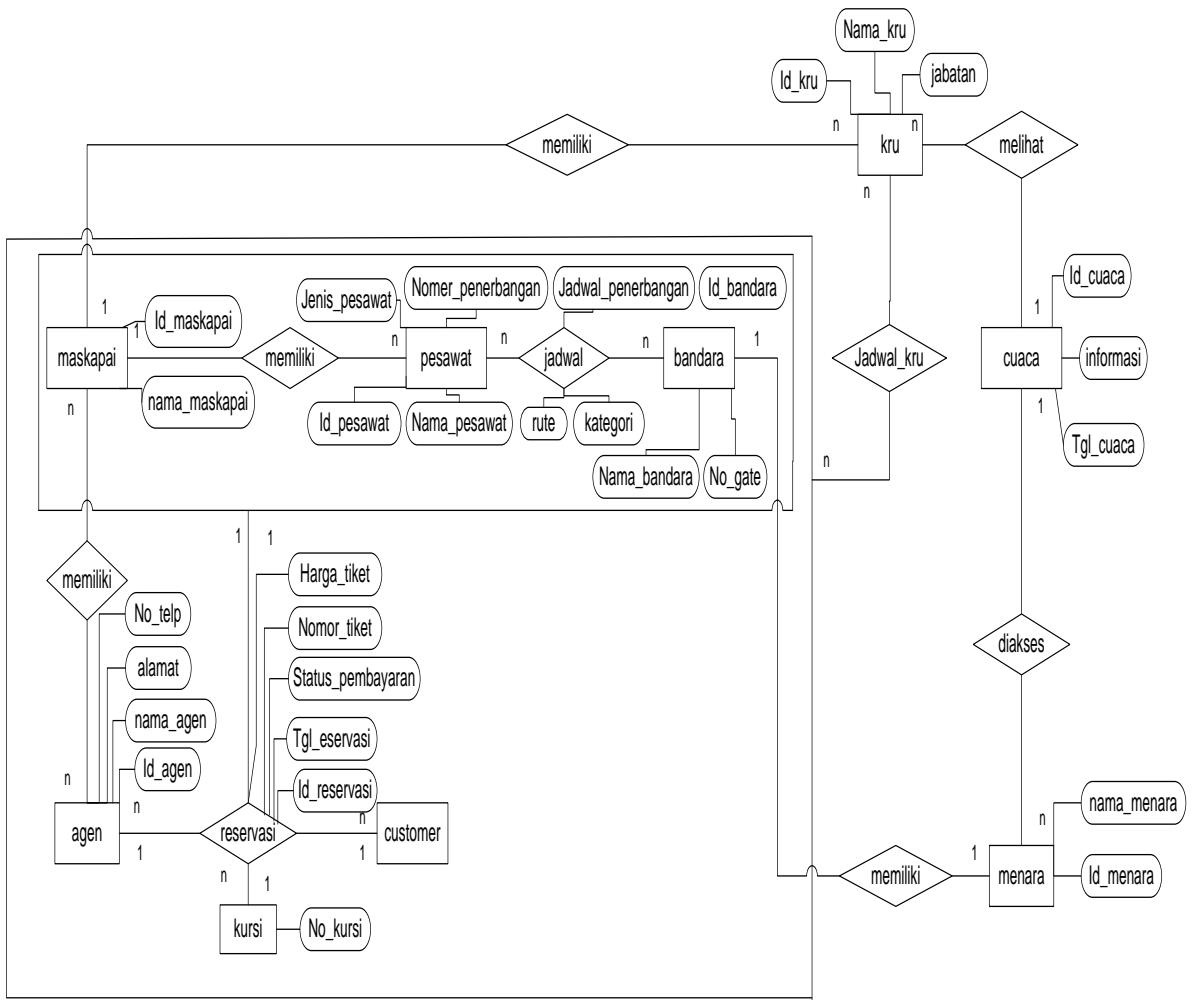


1.2.5.1 Contoh E-R Diagram

Pada setiap bandara sudah mempunyai jadwal penerbangan untuk semua maskapai baik penerbangan domestik maupun internasional. Dimana pihak bandara telah melakukan penjadwalan terhadap pesawat dan rute penerbangan baik keberangkatan serta kedatangan. Petugas yang mengatur lalu lintas udara yang biasa disebut air traffic controller (ATC) memiliki peranan yang penting terutama pada pesawat udara untuk mencegah antar pesawat tidak terlalu dekat satu sama lain, mencegah tabrakan antar pesawat udara dan pesawat udara dengan rintangan yang ada di sekitarnya selama beroperasi. Selain itu ATC mempunyai peran dalam pengaturan kelancaran arus lalu lintas, membantu pilot dalam mengendalikan keadaan darurat, memberikan informasi yang dibutuhkan pilot (seperti informasi cuaca, informasi navigasi penerbangan, dan informasi lalu lintas udara). ATC adalah rekan terdekat pilot selama di udara, peran ATC sangat besar dalam tercapainya tujuan penerbangan. Dalam sistem pelayanan penerbangan mempunyai beberapa pelayanan yang ditawarkan melalui web diantaranya sebagai berikut:

1. Pihak bandara dapat melihat keadaan cuaca saat ini, baik atau buruk untuk melakukan perbangsan, status penerbangan (*landing, takeoff, delay*), jadwal kedatangan, penerbangan dan tujuan pesawat.
2. Maskapai dapat melihat jadwal tugas penerbangan dari dan ke mana, siapa pilot, pramugari dan kru-krunya.
3. Agen bisa menjual, boking, memantau harga-harga pesawat dan jadwal penerbangan.
4. *Customer* bisa melihat jadwal penerbangan, rute penerbangan, melihat harga tiket pesawat, boking dan pesan tiket.





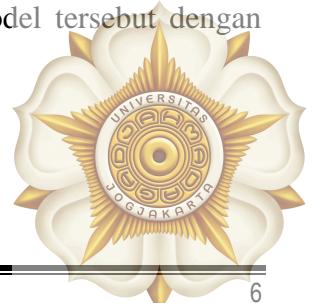
Gambar. 1.1 ERD Sistem Informasi Penerbangan Pesawat

1.3 AKTIVITAS MAHASISWA

1. Mahasiswa mempelajari dan mencoba ERD Diagram pada gambar 1.1.
2. Mahasiswa mengerjakan latihan 1.4.

1.4 LATIHAN (dikerjakan di kelas)

Buatlah sebuah model *database* dari sistem perpustakaan. Tentukan entitas-entitas yang ada di dalamnya dan jelaskan atribut-atribut penyusun entitas. Tentukan *primary key* entitas yang kuat serta relasi antar entitas. Setelah itu gambarkan model tersebut dengan ERD!



BAB II

NORMALISASI

2.1 TUJUAN PRAKTIKUM

1. Mahasiswa dapat memahami urgensi normalisasi pada tabel basis data.
2. Mahasiswa dapat memahami bentuk tidak normal.
3. Mahasiswa dapat memahami bentuk 1NF, 2NF dan 3NF.

2.2 MATERI, CONTOH DAN ILUSTRASI

2.2.1 Normalisasi

Proses normalisasi adalah sebuah teknik dalam logical desain sebuah basis data yang mengelompokkan atribut dari suatu relasi sehingga membentuk struktur relasi yang baik (tanpa redundansi), sehingga sebagian besar *ambiguity* bisa dihilangkan. Tujuan dari normalisasi adalah menghilangkan kerakangan data (*redundancy*), mengurangi kompleksitas dan untuk mempermudah pemodifikasi data.

Selain itu normalisasi diperlukan untuk menghindari anomali *insertion*, anomali *deletion*, dan anomali *update*. Pada database yang sudah normal, anomali *insertion* dapat dihindari karena pengisian data yang sama pada beberapa tabel hanya dilakukan cukup dengan mengisikan pada satu tabel dan yang lain akan mengikuti, begitu pula dengan perintah *update* dan *delete*. Namun pada tabel yang belum normal, pengisian data yang sama pada beberapa tabel dilakukan satu per satu. Hal ini memungkinkan terjadinya kesalahan input oleh manusia (human error) sehingga beberapa *record* yang seharusnya memiliki data yang sama, akan memiliki data yang berbeda.

Bentuk normalisasi ada beberapa macam, mulai dari normalisasi 1, normalisasi 2, dst. Beberapa kasus yang mencapai tahap normalisasi hingga lebih dari 5. Namun ada juga kasus yang hanya mencapai tahap normalisasi 2 saja. Normalisasi tiap kasus berbeda-beda. Hal ini disesuaikan dengan kebutuhan. Apabila dengan normalisasi ke 2 database sudah mencukupi syarat di atas, maka tidak diperlukan tahap normalisasi berikutnya.

2.2.2 Non-Normalisasi

Beberapa hal yang harus diperhatikan sebelum memulai dilakukan proses normalisasi, yaitu:

1. Setiap tabel harus memiliki *primary key*.
2. Sebuah tabel tidak boleh memiliki kumpulan data yang berulang.



Gambar 2.1 **orders** menunjukkan contoh tabel yang berbentuk non-normalisasi, namun sudah memenuhi syarat untuk dilakukan proses normalisasi.

Orders

OrderId	CustomerId	OrderDate	Items
1	4	5/1/2001	5 32-hammer, 3 2-screwdriver, 6 76-monkey wrench
2	23	5/9/2001	1 32-hammer
3	15	7/4/2001	2 113-deluxe garden hose, 2 121-economy nozzle
4	2	8/1/2001	15 1024-hack saw
5	23	8/2/2001	1 2-screwdriver
6	2	8/2/2001	5 52-key

Gambar 2.1 Tabel orders non normalisasi

Dari Gambar 2.1 dapat dilihat bahwa tabel **orders** tidak memiliki kumpulan data yang berulang, namun memiliki kumpulan nilai yang berulang, yaitu pada field **items**. Pada field items terdapat 2 permasalahan, yaitu kolom ini bersifat *multi-part field*, yaitu field ini dapat dipecah ke dalam bagian yang unik, kemudian bersifat *multi-value*, artinya 1 **OrderId** dapat diasosiasikan dengan 1 atau lebih field items. Oleh karena itu, proses selanjutnya adalah mengubah ke dalam bentuk normal pertama.

2.2.3 Normalisasi I

Normal pertama memiliki aturan sebagai berikut:

- Tidak adanya atribut *multi-value*, atribut komposit atau kombinasinya.
- Mendefinisikan atribut kunci.
- Setiap atribut dalam tabel harus bernilai *atomic* (tidak dapat dibagi lagi).

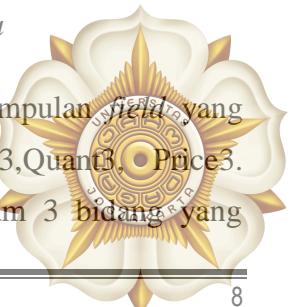
Dari Gambar 2.1 untuk melakukan normalisasi pada tabel **Orders** dilakukan dengan menghilangkan *multi value* dari field items.

Orders

OrderId	OrderDate	CustomerId	Item1	Qty1	Price1	Total1	Item2	Qty2	Price2	Total2	Item3	Qty3	Price3	Total3
1	5/1/2001	4	32-hammer	5	\$12.99	\$64.95	2-screwdriver	8	\$7.99	\$63.92	76-monkey wrench	6	\$8.99	\$53.94
2	5/9/2001	23	32-hammer	1	\$12.99	\$12.99								
3	7/4/2001	15	113-deluxe garden hose	2	\$4.50	\$9.00	121-economy nozzle	2	\$3.85	\$7.70				
4	8/1/2001	2	1024-hack saw	15	\$17.00	\$255.00								
5	8/2/2001	23	2-screwdriver	1	\$7.99	\$7.99								
6	8/2/2001	2	52-key	5	\$1.75	\$8.75								

Gambar 2.2 Langkah awal pembuatan normal pertama

Dari Gambar 2.2 dapat dilihat bahwa sekarang kita memiliki kumpulan field yang berulang: Item1, Quant1, Price1, Item2, Quant2, Price2, dan Item3, Quant3, Price3. Selanjutnya, untuk mengatasinya dilakukan dengan membagi ke dalam 3 bidang yang



berbeda, menjadi: Item, Quant dan Price. Selain itu, kita harus menghapus karakteristik multipart dari item field dengan membaginya menjadi 2 bidang yang berbeda : ProductID dan Product. Gambar 2.3 menunjukan hasil dari modifikasi.

Orders

OrderId	OrderDate	CustomerId	ProductId	Product	Quantity	Price	Total
1	5/1/2001	4	32	hammer	5	\$12.99	\$64.95
1	5/1/2001	4	2	screwdriver	8	\$7.99	\$63.92
1	5/1/2001	4	76	monkey wrench	6	\$8.99	\$53.94
2	5/9/2001	23	32	hammer	1	\$12.99	\$12.99
3	7/4/2001	15	113	deluxe garden hose	2	\$4.50	\$9.00
3	7/4/2001	15	121	economy nozzle	2	\$3.85	\$7.70
4	8/1/2001	2	1024	hack saw	15	\$17.00	\$255.00
5	8/2/2001	23	2	screwdriver	1	\$7.99	\$7.99
6	8/2/2001	2	52	key	5	\$1.75	\$8.75

Gambar 2.3 Bentuk normal pertama Tabel orders

Walaupun bentuk tabel orders masih belum sempurna, namun tabel sudah berada pada kondisi normal pertama dan dapat diujikan pada bentuk normal kedua.

2.2.4 Normalisasi II

Pada normalisasi bentuk kedua memiliki aturan sebagai berikut:

- Bentuk normal 2NF terpenuhi dalam sebuah tabel jika telah memenuhi bentuk 1NF, dan semua atribut selain primary key, secara utuh memiliki Functional Dependency pada primary key
- Sebuah tabel tidak memenuhi 2NF, jika ada atribut yang ketergantungannya (Functional Dependency) hanya bersifat parsial saja (hanya tergantung pada sebagian dari primary key)
- Jika terdapat atribut yang tidak memiliki ketergantungan terhadap primary key, maka atribut tersebut harus dipindah atau dihilangkan

Tabel orders pada Gambar 2.3 belum memenuhi syarat dari bentuk normalisasi kedua, karena terdapat 3 permasalahan berbeda,yaitu:

1. Tabel Orders memiliki 2 subjek, yaitu Orders dan OrderDetails.
2. Tabel Orders mengandung *field* dengan proses perhitungan (Total), dan
3. Tabel Orders memiliki ketergantungan parsial antara OrderID dan Product.

Langkah pertama adalah membagi tabel ka dalam 2 tabel yang lebih kecil, yaitu Tabel Orders dan OrderDetails. Pada tahap ini mamstikan bahwa Tabel Orders hanya menjelaskan mengenai satu dan hanya satu subjek.



Orders

OrderId	CustomerId	OrderDate
1	1	5/1/2001
2	3	5/9/2001
3	1	7/4/2001
4	2	8/1/2001
5	1	8/2/2001
6	2	8/2/2001

OrderDetails

OrderId	ProductId	Product	Quantity	Price	Total
1	32	hammer	5	\$12.99	\$64.95
1	2	screwdriver	8	\$7.99	\$63.92
1	76	monkey wrench	6	\$8.99	\$53.94
2	32	hammer	1	\$12.99	\$12.99
3	113	deluxe garden hose	2	\$4.50	\$9.00
3	121	economy nozzle	2	\$3.85	\$7.70
4	1024	hack saw	15	\$17.00	\$255.00
5	2	screwdriver	1	\$7.99	\$7.99
6	52	key	5	\$1.75	\$8.75

Gambar 2.4 Menerapkan bentuk normal kedua pada Tabel Orders

Sekarang Tabel Orders telah berada pada bentuk normal kedua, yang perlu diperhatikan sekarang adalah Tabel OrderDetails, karena pada tabel ini terdapat *field* perhitungan dan ketergantungan parsial. Untuk *field* yang mengandung proses perhitungan dapat diatasi dengan menghapusnya dari tabel, sedangkan untuk mengatasi ketergantungan parsial yang dapat dilakukan adalah dengan membaginya ke dalam 2 subjek, yaitu OrderDetails dan Product.

Untuk mengatasi ketergantungan parsial dilakukan dengan menghapus *field* ProductID dari OrderDetails, dan membuat tabel baru yaitu tabel Product dengan ProductID dan Product merupakan isi dari *field*nya. ProductID sangat penting pada tabel Product, karena ProductID akan merelasikan tabel Product dengan OrderDetails. Setelah proses ini dilakukan, Tabel OrderDetails sudah berada pada bentuk normal kedua.



OrderDetails

OrderId	ProductId	Quantity	Price
1	32	5	\$12.99
1	2	8	\$7.99
1	76	6	\$8.99
2	32	1	\$12.99
3	113	2	\$4.50
3	121	2	\$3.85
4	1024	15	\$17.00
5	2	1	\$7.99
6	52	5	\$1.75

Products

ProductId	Product	Price
2	screwdriver	\$7.99
32	hammer	\$12.99
52	key	\$1.75
113	deluxe garden hose	\$4.50
121	economy nozzle	\$3.85
1024	hack saw	\$17.00
76	monkey wrench	\$8.99

Gambar 2.5 Bentuk normal kedua dari tabel OrderDetails dan Product

2.2.5 Normalisasi III

Bentuk normalisasi ketiga mempunyai ciri, yaitu:

- Setiap nilai dalam *field* diperbarui secara independen. Jika mengubah nilai dari suatu *field* dari data yang diberikan, maka perubahan tersebut tidak memberikan pengaruh buruk pada nilai dari *field* yang lain,
- Setiap *field* mengidentifikasi karakteristik tertentu dari subjek tabel
- Setiap *field* non- *key* dalam tabel bergantung secara fungsional (*functional dependency*) pada *primary key*
- Tabel hanya menjelaskan satu dan hanya satu subjek.

Tabel Orders dan OrderDetail telah memiliki bentuk normal kedua, maka proses normal ketiga diterapkan pada tabel tersebut. Pada tabel Orders, syarat dari bentuk normal ketiga telah terpenuhi, maka tabel Orders sudah berada pada bentuk normal ketiga.

Orders

OrderId	CustomerId	OrderDate
1	1	5/1/2001
2	3	5/9/2001
3	1	7/4/2001
4	2	8/1/2001
5	1	8/2/2001
6	2	8/2/2001

Gambar 2.6 Bentuk normal ketiga dari tabel Orders



Jika kita melihat bentuk tabel OrderDetails pada Gambar 2.5, dapat dilihat bahwa pada tabel tersebut terdapat *field* yang tidak menjelaskan mengenai subjek tabel. *Field* ini adalah *field* Price. Price tidak merepresentasikan karakteristik yang spesifik dari tabel OrderDetails, selanjutnya nilai Price hanya ditentukan oleh ProductID. Pada tabel OrderDetails memiliki gabungan *primary key* yang terdiri dari OrderID dan ProductID, namun *field* Price tidak bergantung secara penuh pada seluruh *primary key*, maka *field* Price dapat dihilangkan dari tabel OrderDetails.

OrderDetails

OrderId	ProductId	Quantity
1	32	5
1	2	8
1	76	6
2	32	1
3	113	2
3	121	2
4	1024	15
5	2	1
6	52	5

Gambar 2.7 Bentuk normal ketiga dari tabel OrderDetails

2.3 AKTIVITAS MAHASISWA

1. Mahasiswa memperhatikan penjelasan instruktur mengenai normalisasi
2. Mahasiswa mencoba dan mempelajari bentuk tabel non-normalisasi pada Gambar 2.1.
3. Mahasiswa mencoba membuat normalisasi bentuk pertama seperti pada Gambar 2.2 dan Gambar 2.3
4. Mahasiswa mencoba membuat normalisasi bentuk kedua seperti pada Gambar 2.4 dan Gambar 2.5
5. Mahasiswa mencoba membuat normalisasi bentuk kedua seperti pada Gambar 2.6 dan Gambar 2.7
6. Mahasiswa mengerjakan latihan/tugas untuk melatih pemahaman mengenai normalisasi



2.4 LATIHAN/TUGAS

HEALTH HISTORY REPORT

<u>PET ID</u>	<u>PET NAME</u>	<u>PET TYPE</u>	<u>PET AGE</u>	<u>OWNER</u>	<u>VISIT DATE</u>	<u>PROCEDURE</u>
246	ROVER	DOG	12	SAM COOK	JAN 13/2002	01 - RABIES VACCINATION
					MAR 27/2002	10 - EXAMINE and TREAT WOUND
					APR 02/2002	05 - HEART WORM TEST
298	SPOT	DOG	2	TERRY KIM	JAN 21/2002	08 - TETANUS VACCINATION
					MAR 10/2002	05 - HEART WORM TEST
341	MORRIS	CAT	4	SAM COOK	JAN 23/2001	01 - RABIES VACCINATION
					JAN 13/2002	01 - RABIES VACCINATION
519	TWEEDY	BIRD	2	TERRY KIM	APR 30/2002	20 - ANNUAL CHECK UP
					APR 30/2002	12 - EYE WASH

- 1 Dari kasus tersebut buatlah proses normalisasinya, dengan menjelaskan per langkah dan satu per satu. Berikan alasan detail, seperti misalnya suatu atribut dihapus karena redundant, dll.



BAB III

DATA DEFINITION LANGUAGE (DDL)

3.1 TUJUAN PRAKTIKUM

1. Mahasiswa dapat menjelaskan konsep *Data Definition Language*.
2. Mahasiswa dapat menggunakan perintah-perintah perintah – perintah dasar untuk mendefinisikan objek dari basis data.

3.2 MATERI, CONTOH DAN ILUSTRASI

Data Definition Language (DDL) merupakan kelompok perintah yang digunakan untuk melakukan pendefinisian database maupun tabel. Perintah dasar dalam DDL ini adalah:

1. CREATE : perintah ini digunakan untuk membuat database, tabel, view dan index.
2. ALTER : perintah ini digunakan untuk mengubah struktur tabel.
3. DROP : perintah ini digunakan untuk menghapus database, tabel, view dan index.

3.2.1 Database

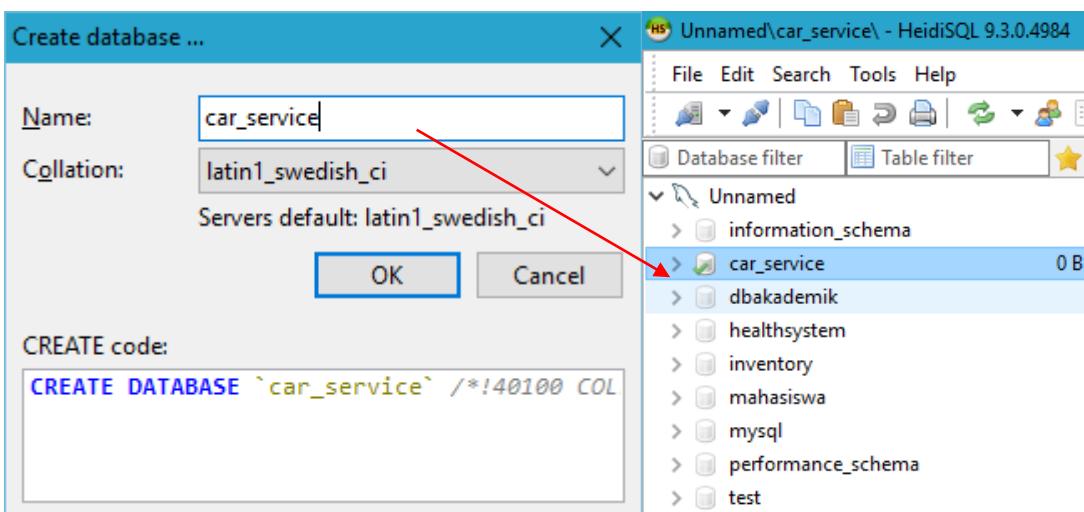
Perintah untuk membuat database yaitu:

```
CREATE DATABASE namadatabase;
```

Perintah ini digunakan pertama kali sebelum membuat tabel, view, maupun komponen lain dalam sistem database. Nama database dapat terdiri dari kombinasi huruf dan karakter namun sebaiknya tidak mengandung spasi dan tanda baca untuk memudahkan pengaturan database dikemudian hari. Contoh:

```
CREATE DATABASE car_service;
```





Gambar 3.1 create database car_service

Sedangkan perintah untuk melakukan penghapusan database, yaitu:

```
DROP DATABASE namadatabase;
```

Contoh query:

```
DROP DATABASE car_service;
```

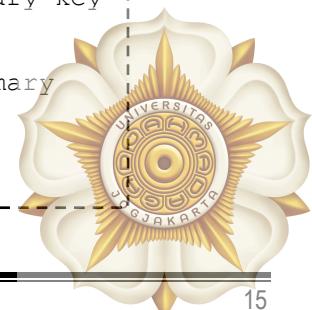
Untuk dapat mengelola sebuah database, pengguna terlebih dahulu harus masuk ke dalam lingkungan database yang akan diolah. Untuk dapat mengolah sebuah database sintaks yang digunakan adalah

```
USE nama_database;
```

3.2.2 Tabel

Perintah umum untuk membuat tabel pada database yaitu:

```
CREATE TABLE namabel (
    Field1 TipeData1(jumlahdigit) [keterangan primary key
    atau not null jika diperlukan],
    Field2 TipeData2 (jumlahdigit) [keterangan primary
    key atau not null jika diperlukan],
    Field3....);
```



Atau secara lebih lengkapnya:

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
(create_definition,...)
[table_option ...]
[partition options]
```

Field1 adalah atribut yang menjadi nama kolom pertama, Field2 adalah nama kolom kedua, dan seterusnya sejumlah kolom yang akan didefinisikan pada sebuah tabel. Tipedata1 adalah tipedata yang digunakan untuk mendefinisikan tipe data yang digunakan pada kolom yang bersangkutan, sedangkan jumlahdigit merupakan jumlah karakter yang dialokasikan untuk tipe data tersebut. Namun, terdapat juga pendeklarasi tipe data yang tidak perlu dinyatakan jumlahdigit-nya karena sudah secara otomatis didefinisikan oleh database tersebut.

Berikut adalah contoh pembuatan table :

The screenshot shows the MySQL Workbench interface. In the left sidebar, under the 'car_service' schema, a context menu is open with 'Create new' selected. A sub-menu for 'Table' is shown, containing options like 'Table copy', 'View', 'Stored routine', 'Trigger', and 'Event'. Below this, the main workspace shows the 'Basic' tab selected for creating a new table named 'items'. The 'Comment:' field is empty. At the bottom, the 'Columns:' section lists seven columns with their details:

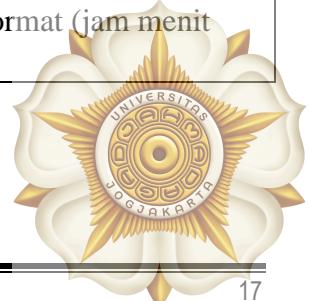
#	Name	Datatype	Length/Set	Unsign...	Allow N...	Zerofill	Default
1	ID	INT	11	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL
2	itemName	VARCHAR	50	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL
3	CategoryId	INT	11	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL
4	SupplierId	INT	11	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL
5	BuyPrice	INT	11	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL
6	SellPrice	INT	11	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL
7	Unit	VARCHAR	50	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL

Gambar 3.2 Pembuatan table : item

Berikut tipe data – tipe data yang didukung oleh MySQL:

Tabel 3.1 Tipe data yang didukung MySQL

Tipe Data	Deklarasi	Keterangan
Char	char(panjang)	Kolom bertipe char dapat diisi dengan data teks. Panjang maksimal hanya 255 karakter
Varchar	varchar(panjang)	Tipe ini berisi data teks dengan panjangmaksimal seperti yang dideklarasikan
Tinytext	Tinytext	Seperti varchar
Text	Text	Berisi data teks dengan panjangkarakter yang lebih luas dari tipe-tipe sebelumnya
Integer	Int(panjang) [unsigned]	Data berupa angka antara 0 – 4294967295 jika unsigned atau -2147483648 – 2147483647 jika signed. Panjang maksimal karakter sesuai yang dideklarasikan
Tinyint	Tinyint(panjang) [unsigned]	Data berupa angka antara 0 – 255 jika unsigned atau -128 – 127 jika signed. Panjang maksimal karakter sesuai yang dideklarasikan
Mediumint	mediumint(panjang) [unsigned]	Data berupa angka antara 0 – 1677215 jika unsigned atau -8388608 – 8388607 jika signed. Panjang maksimal karakter sesuai yang dideklarasikan
Bigint	bigint(panjang) [unsigned]	Data berupa angka antara 0 – 10^{20}
Float	float(panjang)	Data berupa angka real dengan presisi hingga pangkat 38. Panjang maksimal adalah 53
Double	double(panjang)	Data berupa angka real dengan presisi hingga pangkat 308.
Date	Date	Menyimpan nilai dalam format (tahun – bulan – tanggal)
Datetime	Datetime	Menyimpan nilai dalam format (tahun – bulan- tanggal – jam – menit – detik)
Timestamp	Timestamp (panjang)	Menyimpan date dan time berdasarkan panjang yang diberikan
Time	Time	menyimpan nilai dalam format (jam menit detik)



3.2.3 Foreign Key

Cara mendefinisikan foreign key ada bermacam-macam tergantung kasus. Berikut contoh pendefinisan foreign key.

```
create table (ID int (11) primary key not null, itemName  
varchar (50), CategoryId int (11) references category (id),  
SupplierId int (11) references supplier (id), buyprice int  
(11), sellprice int (11), unit varchar (50))
```

Kolom:								
#	Nama	Tipe data	Panjang/Batas	Tidak t...	Ijinkan ...	Zerofill	Default	
1	ID	INT	11	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	AUTO_INCREMENT	
2	itemName	VARCHAR	50	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	NULL	
3	CategoryId	INT	11	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL	
4	SupplierId	INT	11	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL	
5	BuyPrice	INT	11	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL	
6	SellPrice	INT	11	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL	
7	Unit	VARCHAR	50	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	NULL	

Gambar 3.3 Query untuk pembuatan *foreign key*

Biasanya pendefinisan foreign key diikuti dengan aksi yang akan berlaku untuk tabel yang direferensi maupun yang mereferensi. Aksi yang didukung oleh MySQL antara lain:

1. On Delete Restrict: kolom terkait pada tabel yang direferensi tidak dapat dihapus.
2. On Delete No Action: sama seperti On Delete Restrict.
3. On Delete Cascade: jika kolom pada tabel yang direferensi dihapus maka data pada kolom yang mereferensi juga turut dihapus.
4. On Delete Set Null: Jika kolom pada tabel yang direferensi dihapus maka data pada kolom yang mereferensi diubah menjadi null.
5. On Update Restrict: update pada kolom yang masih direferensi suatu tabel tidak diperbolehkan
6. On Update Cascade: jika kolom pada tabel yang direferensi diupdate maka data kolom tabel yang mereferensi juga turut diupdate.
7. On Update Set Null: jika kolom pada tabel yang direferensi diupdate maka data kolom tabel yang mereferensi akan diubah menjadi null.



Berikut contoh perintah penggunaan on update restrict dan on delete restrict serta on update casacase dan on delete cascade pada database car_service :

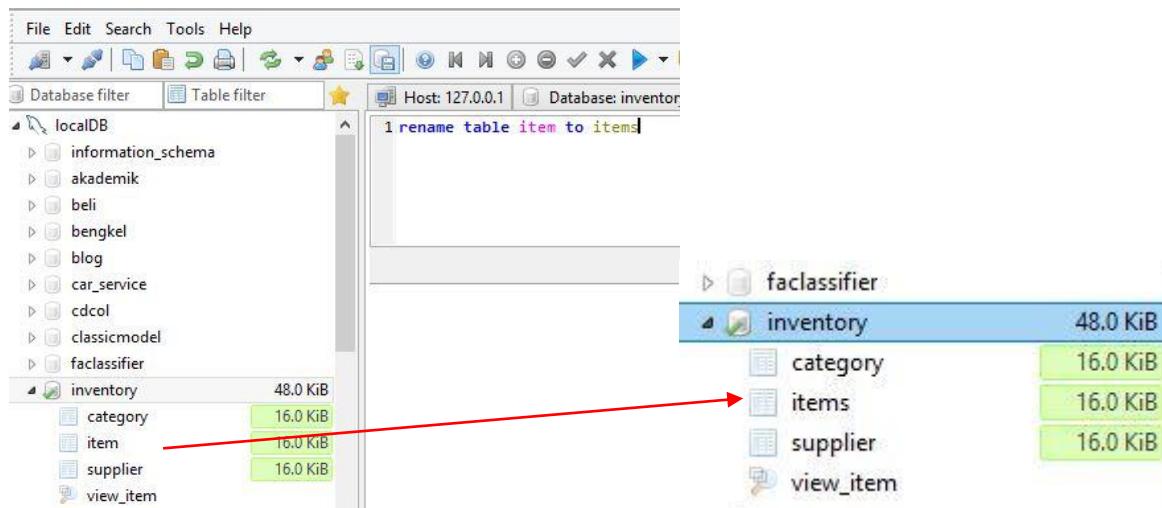
```
FOREIGN KEY (`CategoryId`) REFERENCES `category` (`id`) ON  
UPDATE RESTRICT ON DELETE RESTRICT;
```

```
FOREIGN KEY (`CategoryId`) REFERENCES `category` (`id`) ON  
UPDATE CASCADE ON DELETE CASCADE;
```

Perintah untuk mengganti nama tabel pada database yaitu:

```
RENAME TABLE nama_tabel_lama TO nama_tabel_baru;
```

Contoh implementasi dari sql ini ditunjukan pada Gambar 3.3 untuk mengganti tabel **item** menjadi tabel **items**.



Gambar 3.4 query untuk mengganti nama tabel item menjadi items



Perintah untuk menghapus tabel pada database yaitu:

```
DROP TABLE namatabel;
```

The screenshot shows the MySQL Workbench interface. On the left, the database tree is expanded to show 'beli' and its tables: 'bengkel', 'blog', 'car_service', 'cdcol', 'classicmodel', 'faclassifier', and 'inventory'. The 'inventory' node is expanded to show its tables: 'category', 'items', 'supplier', 'transaksi', and 'view_item'. In the main pane, a query window contains the command 'drop table transaksi'. To the right, the results pane shows the 'inventory' database with its tables: 'faclassifier', 'inventory', 'category', 'items', 'supplier', and 'view_item'. The 'transaksi' table is highlighted in blue.

Gambar 3.5 Query untuk menghapus tabel transaksi pada *database*

Untuk menampilkan tabel yang ada di dalam sebuah database sintaks yang digunakan adalah

```
SHOW TABLES nama_database;
```

Gambar 3.6 merupakan implementasi dari *syntax* show tables yang digunakan untuk menampilkan tabel-tabel yang ada di dalam *database* car_service.

```
1 show tables;
```

The screenshot shows the MySQL Workbench interface. A query window contains the command 'show tables'. Below it, a results grid titled 'TABLE_NAMES (1x3)' displays the following data:

Tables_in_car_service	category	items
	supplier	

Gambar 3.6 Query untuk menampilkan tabel yang ada pada *database* car_service



Untuk menampilkan struktur sebuah table digunakan sintaks,

```
DESCRIBE nama_tabel;
```

Untuk mengetahui cara penggunaan *syntax* describe dapat dilihat pada Gambar 3.7. Gambar 3.7 ini merupakan query untuk melihat struktur tabel dari tabel supplier.

The screenshot shows the MySQL Workbench interface with the following details:

- Host: 127.0.0.1
- Database: car_service
- Query window: `1 describe supplier`
- Result pane: Shows the table structure with 2 columns:

Field	Type	Null	Key	Default	Extra
Id	int(11)	NO	PRI	(NULL)	auto_increment
supplierName	varchar(50)	YES		(NULL)	
- Right sidebar: SQL keywords, Snippets, Query history, Query profile, Bind parameters.

Gambar 3.7 Query untuk menampilkan struktur tabel supplier

3.2.4 ALTER TABEL

Digunakan untuk mengubah struktur tabel. Beberapa alter tabel diantaranya yaitu menambah kolom, memodifikasi kolom, mengubah nama kolom, dst. Sintaks umum untuk alter tabel berupa :

```
ALTER TABLE namatabel jenis_alter kondisi;
```

Untuk menambah kolom pada sebuah tabel, digunakan sintaks.

```
ALTER TABLE namatabel add nama_kolom tipe  
AFTER/BEFORE nama_kolom_eksis;
```

The screenshot shows the MySQL Workbench interface with the following details:

- Host: 127.0.0.1
- Database: car_service
- Table: supplier
- Query window: `1 alter table supplier add column city varchar(50)`
- Result pane: Shows the updated table structure with the new column added:

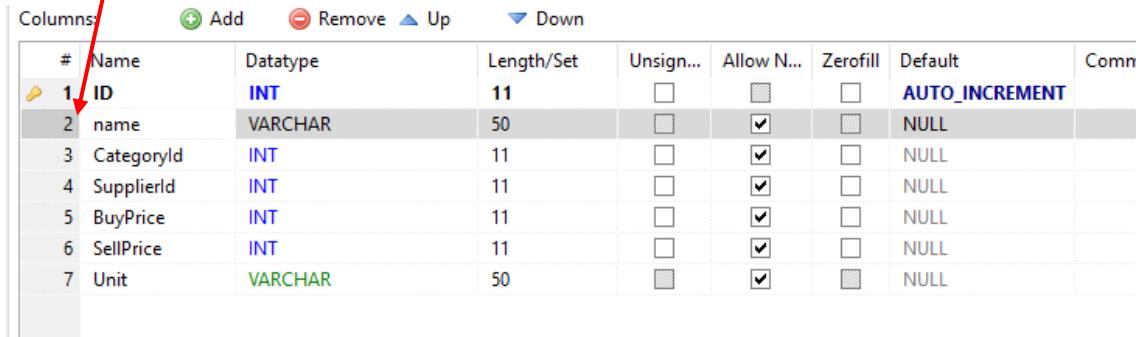
Id	supplierName	city
4	Meka Adi Pratama	(NULL)
5	Teruna Gema Nusa	(NULL)
6	Cv. Makin Jaya	(NULL)
7	PT. Enseval Putera Megatrading	(NULL)
8	Cv. Cahaya Mulia Lectari	(NULL)

Gambar 3.8 Query untuk menambah kolom city pada tabel supplier

Query selanjutnya adalah query untuk mengubah nama kolom yang sudah ada pada sebuah tabel, digunakan sintaks.

```
ALTER TABLE namatabel change nama_kolom_lama
nama_kolom_baru tipe_data_baru(panjang);
```

Penggunaan syntax ini ditunjukan pada Gambar 3.4 untuk mengganti nama kolom itemName menjadi name.



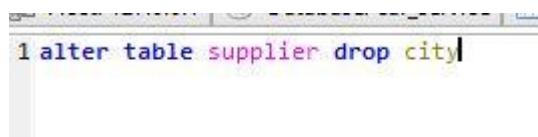
#	Name	Datatype	Length/Set	Unsign...	Allow N...	Zerofill	Default	Com...
1	ID	INT	11	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	AUTO_INCREMENT	
2	itemName	VARCHAR	50	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	NULL	
3	CategoryId	INT	11	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL	
4	SupplierId	INT	11	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL	
5	BuyPrice	INT	11	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL	
6	SellPrice	INT	11	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL	
7	Unit	VARCHAR	50	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	NULL	

Columns:								
+ Add - Remove ▲ Up ▼ Down								
#	Name	Datatype	Length/Set	Unsign...	Allow N...	Zerofill	Default	Com...
1	ID	INT	11	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	AUTO_INCREMENT	
2	name	VARCHAR	50	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	NULL	
3	CategoryId	INT	11	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL	
4	SupplierId	INT	11	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL	
5	BuyPrice	INT	11	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL	
6	SellPrice	INT	11	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL	
7	Unit	VARCHAR	50	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	NULL	

Gambar 3.9 Query untuk mengganti nama kolom dari itemName menjadi name Untuk menghapus kolom pada sebuah tabel, digunakan sintaks.

```
ALTER TABLE namatabel DROP nama_kolom;
```

Sebagai contoh,



```
1 alter table supplier drop city;
```

Gambar 3.10 Query untuk menghapus kolom city pada tabel supplier



Untuk mengubah/memodifikasi kolom yang ada pada sebuah tabel, digunakan sintaks.

```
ALTER TABLE namatabel MODIFY COLUMN nama_kolom  
kolom_definiton;
```

Untuk mengubah kolom yang ada pada sebuah tabel menjadi primary key digunakan sintaks.

```
ALTER TABLE namatabel ADD PRIMARY KEY(nama_kolom);
```

Untuk menghapus primary key pada sebuah tabel digunakan sintaks.

```
ALTER TABLE namatabel DROP PRIMARY KEY;
```

3.2.5 Index

Index adalah skema objek yang digunakan untuk meningkatkan kecepatan dalam mendapatkan baris data yang diinginkan dengan menggunakan pointer. Indeks dibuat dengan berbasiskan kolom. Index dapat dibuat pada saat tabel diciptakan maupun pada tabel yang sudah ada. Untuk membuat index pada saat tabel diciptakan, digunakan perintah :

```
CREATE TABLE nama_tabel(kolom type definition,  
kolom type definition,...,INDEX(column));
```

Untuk membuat index pada tabel yang sudah tercipta sebelumnya, digunakan perintah:

```
CREATE INDEX nama_index ON nama_tabel(nama_kolom);
```

Sedangkan untuk menghapus index menggunakan perintah:

```
DROP INDEX nama_index;
```

3.3 AKTIVITAS MAHASISWA

1. Mahasiswa mencoba contoh penggunaan query dalam DDL mulai dari Gambar 3.1 sampai 3.10.
2. Mahasiswa mengerjakan latihan 3.4.



3.4 LATIHAN/TUGAS

1. Buatlah basis data akademik, kemudian buat table mahasiswa mahasiswa (dengan atribut: NIM, nama, jenis kelamin, tempat lahir, tanggal lahir, telepon dan pembimbing akademik), dosen(terdiri dari atribut NIP, nama, jenis kelamin, jabatan, minat, dan telepon), mata kuliah(terdiri dari atribut kode mata kuliah, nama mata kuliah, sks, dosen pengampu, hari, jam, dan kode ruang), KRS (terdiri dari atribut id KRS, kode mata kuliah, NIM, tahun, semester dan nilai). Setiap field yang berupa primary key harus didefinisikan auto_increment dan not null.
2. Tampilkan semua tabel yang ada pada database.
3. Tampilkan struktur masing-masing tabel yang telah dibuat.
4. Buatlah query untuk melakukan hal-hal berikut:
 - 1) Hapus (drop) keterangan primary key pada “NIP”.
 - 2) Tambahkan kembali primary key pada “NIP”.
 - 3) Lihat field table “dosen”.
 - 4) Ganti nama table “dosen” menjadi “table_dosen”.
 - 5) Ganti nama atribut “Nama” dengan “Nama_Dosen”.
 - 6) Ubah tipe data “jenis_kelamin” menjadi enum {'pria', 'wanita'}.
 - 7) Ubah tipe data “telepon” menjadi int.



BAB IV

DATA MANIPULATION LANGUAGE (DML)

4.1. TUJUAN PRAKTIKUM

1. Mahasiswa dapat menulis statement SELECT yang mengakses data ke lebih dari satu tabel dengan menggunakan Operator Join, dan menampilkan data yang tidak memenuhi kondisi Join dengan menggunakan Operator Outer Join
2. Mahasiswa mampu menggambarkan tipe persoalan yang dapat dipecahkan oleh sub query, mendefinisikan sub query, memahami tipe-tipe sub query, menulis sub query baris tunggal dan baris berganda
3. Mahasiswa mampu menggunakan fungsi agregasi

4.2. MATERI, CONTOH DAN ILUSTRASI

4.2.1. Data Manipulation Language (DML)

Data manipulation language (DML) merupakan perintah SQL yang digunakan untuk memanipulasi database. Perintah DML diantaranya sebagai berikut :

1. Insert
Digunakan untuk menyisipkan data ke dalam tabel.
2. SELECT
Digunakan untuk memilih dan menampilkan data.
3. Update
Digunakan untuk mengubah data dalam tabel.
4. Delete
Digunakan untuk menghapus data dari tabel.

4.2.2. Insert

Tedapat beberapa sintaks umum yang digunakan untuk pengisian data ke dalam tabel, yaitu:

1. INSERT INTO nama_tabel VALUES (nilai1, nilai2,..)

Perintah ini berfungsi untuk mengisikan data ke dalam tabel secara urut dari kolom paling kiri hingga paling akhir. Values yang diberikan harus sama jumlahnya dengan kolom yang tersedia di dalam tabel.



```

Host: 127.0.0.1 Database: car_service Table: category
1 INSERT INTO category VALUES (10, 'elektronik')

```

Gambar 4.1 Query MySQL untuk insert kategori ‘ekeltronik’ pada tabel category

	id	categoryName
1	1	Oli
2	2	Parfum
3	3	Aksesories
4	4	Minuman
5	5	Lampu
6	6	Kanebo
7	7	Lain-Lain
8	8	Alarm
9	9	Hoods
10	10	elektronik

Gambar 4.2 Hasil insert elektronik

2. INSERT INTO nama_tabel (daftar kolom) VALUES (daftar nilai)

Pengisian seperti ini tidak menghiraukan apakah kolom yang seharusnya NOT NULL terisi atau tidak. Cara ini digunakan seperlunya, sebab bisa menimbulkan error seperti kolom primary key tidak terisi.

```

Host: 127.0.0.1 Database: car_service Table: supplier Data Query*
1 insert into category(id,categoryName) values(null, 'Hoods')

```

Gambar 4.3 Query MySQL untuk insert kategori ‘Hoods’ ke tabel category

	id	categoryName
1	1	Oli
2	2	Parfum
3	3	Aksesories
4	4	Minuman
5	5	Lampu
6	6	Kanebo
7	7	Lain-Lain
8	8	Alarm
9	9	Hoods

Gambar 4.4 Hasil insert kategori ‘Hoods’ ke tabel category



The screenshot shows two windows of MySQL Workbench. The top window displays the SQL query:

```
1 INSERT INTO `items` (`ID`, `itemName`, `CategoryId`, `SupplierId`, `BuyPrice`, `SellPrice`, `Unit`)
2 VALUES(64, 'Pocary Sweat', 4, 18, 4700, 6000, 'pcs')
```

The bottom window shows the resulting data in the 'items' table:

ID	itemName	CategoryId	SupplierId	BuyPrice	SellPrice	Unit
5	Mesran Super	1	5	108,000	160,000	galon
6	Fastron	1	5	232,000	285,000	galon
7	Rored EPA	1	5	32,000	55,000	Liter
10	Shell Helix Hx 7	1	4	287,500	340,000	Galon
21	Areon Nice	2	21	25,000	40,000	pcs
22	Areon Sport Lux	2	21	12,000	20,000	pcs
23	Dorfree Cair	2	9	24,000	35,000	pcs
24	Dorfree Paper	2	9	12,000	18,000	pcs
64	Pocary Sweat	4	18	4,700	6,000	pcs
70	Kanebo JAP	6	9	21,750	25,000	pcs
74	STP Oil treatment	1	22	35,000	45,000	Pcs

Gambar 4.5 Query MySQL untuk insert ‘Pocary Sweat’ ke tabel items

3. INSERT INTO nama_tabel SET (nama_kolom = nilai, ...)

The screenshot shows two windows of MySQL Workbench. The top window displays the SQL query:

```
1 INSERT INTO category SET id=11, categoryName='Gunting'
```

The bottom window shows the resulting data in the 'category' table:

id	categoryName
1	Oli
2	Parfum
3	Aksesoris
4	Minuman
5	Lampu
6	Kanebo
7	Lain-Lain
8	Alarm
9	Hoods
10	elektronik
11	Gunting

Gambar 4.6 query MySQL untuk insert ‘Gunting’ ke tabel items

4.2.3. Delete

Delete digunakan untuk menghapus record suatu tabel. Sedangkan, untuk menghapus data pada kolom tertentu menggunakan perintah update. Sintaks untuk menghapus data dari sebuah tabel secara umum berbentuk:

```
DELETE FROM namatabel
WHERE <kondisi>;
```



Berikut adalah contoh penghapusan data pada tabel items

```
Host: 127.0.0.1 Database: car_service Table: items Data Query*  
1 delete from items where id=64
```

Gambar 4.5. Sintak MySQL untuk delete pocary sweat

Host: 127.0.0.1 Database: car_service Table: items Data Query*

car_service.items: 10 rows total (approximately)

ID	itemName	CategoryId	SupplierId	BuyPrice	SellPrice	Unit
5	Mesran Super	1	5	108,000	160,000	galon
6	Fastron	1	5	232,000	285,000	galon
7	Rored EPA	1	5	32,000	55,000	Liter
10	Shell Helix Hx 7	1	4	287,500	340,000	Galon
21	Areon Nice	2	21	25,000	40,000	pcs
22	Areon Sport Lux	2	21	12,000	20,000	pcs
23	Dorfree Cair	2	9	24,000	35,000	pcs
24	Dorfree Paper	2	9	12,000	18,000	pcs
70	Kanebo JAP	6	9	21,750	25,000	pcs
74	STP Oil treatment	1	22	35,000	45,000	Pcs

Gambar 4.6. Hasil setelah pocary sweat dihapus dari tabel items

Filtering delete dapat berupa fungsi SELECT seperti

```
DELETE FROM <namatabel> WHERE <namakolom> IN (SELECT <kolom>  
FROM namatabel WHERE <kondisi>);
```

Select hanya dapat dilakukan pada satu kolom karena pencocokan nilai hanya bias dengan satu kolom.

Filtering dengan *aggregate*

```
DELETE FROM krs WHERE nilai < (SELECT AVG(nilai) FROM krs);
```

Yang harus diperhatikan dalam delete adalah fungsi ini hanya dapat melibatkan satu tabel. Terlihat dalam tabel setelah FROM hanya berisi sebuah tabel. Cara ini berbeda dengan FROM pada fungsi SELECT (dibahas selanjutnya) yang bisa melibatkan lebih dari satu tabel.



4.2.4. Update

Update merupakan perintah untuk mengubah data yang sudah ada didalam tabel. Perintah update memiliki sintaks secara umum sebagai berikut:

```
UPDATE namatabel  
SET <nama kolom> = <value> WHERE <kondisi>;
```

Contoh update pada tabel items

The screenshot shows two windows from MySQL Workbench. The top window is a query editor with the following SQL code:

```
1 update items set sellPrice=5000 where id=64;
```

The bottom window is a results grid for the 'items' table in the 'car_service' database. The table has columns: ID, itemName, CategoryId, SupplierId, BuyPrice, SellPrice, and Unit. The data includes rows for various items like Mesran Super, Fastron, Rored EPA, etc. A red arrow points from the highlighted row in the results grid to the 'id=64' part of the query in the editor.

ID	itemName	CategoryId	SupplierId	BuyPrice	SellPrice	Unit
5	Mesran Super	1	5	108,000	160,000	galon
6	Fastron	1	5	232,000	285,000	galon
7	Rored EPA	1	5	32,000	55,000	Liter
10	Shell Helix Hx 7	1	4	287,500	340,000	Galon
21	Areon Nice	2	21	25,000	40,000	pcs
22	Areon Sport Lux	2	21	12,000	20,000	pcs
23	Dorfree Cair	2	9	24,000	35,000	pcs
24	Dorfree Paper	2	9	12,000	18,000	pcs
64	Pocary Sweat	4	18	4,700	5,000	pcs
70	Kanebo JAP	6	9	21,750	25,000	pcs
74	STP Oil treatment	1	22	35,000	45,000	Pcs

Gambar 4.7 Query MySQL untuk update data sellPrice Pocary Sweat menjadi 5000

4.2.5. Select

Perintah Select digunakan untuk memilih dan menampilkan data dari suatu database, baik dari satu tabel atau lebih. Fungsi SELECT dalam bentuk sederhana hanya mempunyai pasangan kata FROM. Bentuk umumnya sebagai berikut:

```
SELECT <nama_kolom>  
FROM <nama_tabel>
```



The screenshot shows the MySQL Workbench interface. At the top, it says "Host: 127.0.0.1" and "Database: car_service". Below that is a code editor window containing the SQL query:

```
1 select itemName from items
```

Below the code editor is a results grid titled "items (1x9)". The grid contains the following data:

itemName
Mesran Super
Fastron
Rored EPA
Shell Helix Hx 7
Areon Nice
Areon Sport Lux
Dorfree Cair
Dorfree Paper
STP Oil treatment

Gambar 4.8 Query MySQL untuk menampilkan satu data pada tabel items

Sintak MySQL untuk menampilkan semua data pada salah satu tabel adalah

The screenshot shows the MySQL Workbench interface. A red circle highlights the query in the code editor, and a red arrow points from this circle to a red circle containing the number "1" in the top right corner of the interface.

The code editor contains the SQL query:

```
SELECT *
FROM <nama_tabel>
```

Below the code editor is a results grid titled "items (7x9)". The grid contains the following data:

ID	itemName	CategoryId	SupplierId	BuyPrice	SellPrice	Unit
5	Mesran Super	1	5	108,000	160,000	galon
6	Fastron	1	5	232,000	285,000	galon
7	Rored EPA	1	5	32,000	55,000	Liter
10	Shell Helix Hx 7	1	4	287,500	340,000	Galon
21	Areon Nice	2	21	25,000	40,000	pcs
22	Areon Sport Lux	2	21	12,000	20,000	pcs
23	Dorfree Cair	2	9	24,000	35,000	pcs
24	Dorfree Paper	2	9	12,000	18,000	pcs
74	STP Oil treatment	1	22	35,000	45,000	Pcs

Gambar 4.9 Query MySQL untuk menampilkan semua data dalam tabel items

Perintah di atas digunakan untuk memilih semua kolom dan seluruh record pada tabel items. Pemilihan seperti ini memungkinkan terjadinya redundansi data (data berulang).



Sintak MySQL untuk memilih beberapa kolom pada salah satu tabel adalah

```
SELECT <namakolom1>,<namakolom2>, ....  
      FROM <nama_tabel>
```

The screenshot shows the MySQL Workbench interface. At the top, it displays 'Host: 127.0.0.1', 'Database: car_service', and 'Table: supplier'. Below this, a query window contains the SQL command:

```
1 select itemName,categoryId,sellPrice from items
```

Below the query window is a results grid titled 'items (3x9)'. The data is as follows:

itemName	categoryId	sellPrice
Mesran Super	1	160,000
Fastron	1	285,000
Rored EPA	1	55,000
Shell Helix Hx 7	1	340,000
Areon Nice	2	40,000
Areon Sport Lux	2	20,000
Dorfree Cair	2	35,000
Dorfree Paper	2	18,000
STP Oil treatment	1	45,000

Gambar 4.10 Query MySQL untuk menampilkan tiga atribut dalam tabel items

4.2.5.1. Select Data dengan Fungsi Matematika

1. Menghitung jumlah baris pada suatu tabel

```
SELECT count(*) FROM nama_tabel;
```

The screenshot shows the MySQL Workbench interface. At the top, it displays 'Host: 127.0.0.1' and 'Database: car_service'. Below this, a query window contains the SQL command:

```
1 select count(*) from items
```

Below the query window is a results grid titled 'Result #1 (1x1)'. The data is as follows:

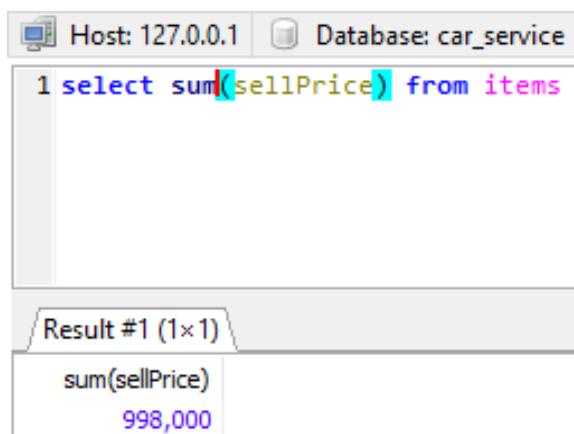
count(*)
9

Gambar 4.11 Query MySQL untuk menghitung jumlah baris



2. Menjumlahkan nilai dari beberapa record pada kolom tertentu

```
SELECT SUM(nama_kolom) FROM nama_tabel;
```



The screenshot shows the MySQL Workbench interface. The top bar displays 'Host: 127.0.0.1' and 'Database: car_service'. The SQL editor window contains the following query:

```
1 select sum(sellPrice) from items
```

The results window below shows the output:

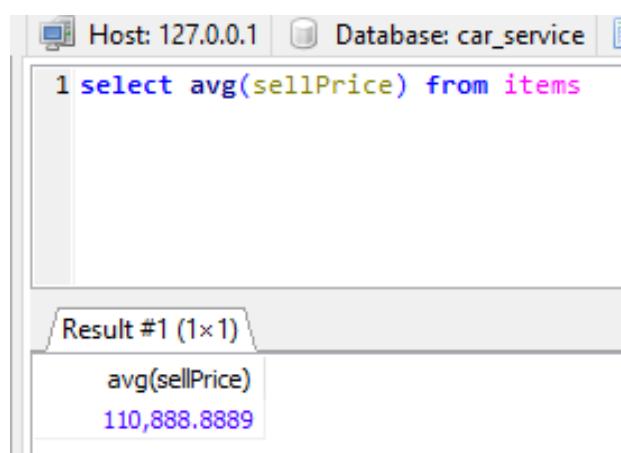
Result #1 (1x1)

sum(sellPrice)
998,000

Gambar 4.12 Query MySQL untuk menghitung jumlahan nilai pada kolom tertentu

3. Menghitung nilai rata-rata beberapa record pada kolom tertentu

```
SELECT AVG(nama_kolom) FROM nama_tabel;
```



The screenshot shows the MySQL Workbench interface. The top bar displays 'Host: 127.0.0.1' and 'Database: car_service'. The SQL editor window contains the following query:

```
1 select avg(sellPrice) from items
```

The results window below shows the output:

Result #1 (1x1)

avg(sellPrice)
110,888.8889

Gambar 4.13 Query MySQL untuk menghitung rata-rata pada kolom tertentu



4. Mencari nilai tertinggi pada kolom tertentu di sebuah tabel

```
SELECT MAX(nama_kolom) FROM nama_tabel;
```

The screenshot shows the MySQL Workbench interface. At the top, it displays the connection details: Host: 127.0.0.1, Database: car_service, and Table: items. Below this, the SQL editor contains the following query:

```
1 select itemName, max(sellPrice) from items
```

Below the editor is the results pane, titled "items (2x1)". It contains a single row of data:

itemName	max(sellPrice)
Mesran Super	340,000

Gambar 4.14 Query MySQL untuk mencari nilai maximum pada kolom tertentu

5. Mencari nilai terendah pada kolom tertentu di sebuah tabel

```
SELECT MIN(nama_kolom) FROM nama_tabel;
```

The screenshot shows the MySQL Workbench interface. At the top, it displays the connection details: Host: 127.0.0.1, Database: car_service, and Table: items. Below this, the SQL editor contains the following query:

```
1 select itemName,min(sellPrice) from items
```

Below the editor is the results pane, titled "items (2x1)". It contains a single row of data:

itemName	min(sellPrice)
Mesran Super	18,000

Gambar 4.14 Query MySQL untuk mencari nilai minimum pada kolom tertentu

4.2.5.2. Operator Relational dan Logika

Selain FROM pelengkap dari fungsi SELECT adalah where. Where berfungsi untuk memberikan kondisi pemilihan sehingga hasil perintah SELECT sesuai dengan yang dibutuhkan. Terdapat operator relasional dengan tanda =, <>, <, >, <=, >=, serta operasi logika AND, OR, XOR yang digunakan sebagai kondisi pada where. Sintaksnya sebagai berikut



```

SELECT <namakolom1, namakolom2, ....>
      FROM <nama tabel>
      WHERE <kondisi>

```

Kata SELECT diikuti oleh nama kolom yang akan dipilih. Kata FROM diikuti oleh nama tabel dimana asal kolom yang dipilih. Kata where diikuti oleh batasan record-record yang terpilih (kondisi). Contoh penggunaan where dalam SELECT, perhatikan contoh berikut:

The screenshot shows the MySQL Workbench interface. The top bar displays 'Host: 127.0.0.1', 'Database: car_service', 'Table: supplier', and 'Query*'. The main area contains a query window with the following code:

```

1 select itemName,categoryId,sellPrice from items where sellPrice>100000

```

Below the query window is a results table titled 'items (3x3)'. The table has three columns: 'itemName', 'categoryId', and 'sellPrice'. The data is as follows:

itemName	categoryId	sellPrice
Mesran Super	1	160,000
Fastron	1	285,000
Shell Helix Hx 7	1	340,000

Gambar 4.15 Query MySQL untuk menampilkan tiga atribut dengan sellPrice lebih dari 100000

5.2.5.3. Operator In dan Not In

Fungsi lain yang mengikuti where adalah in dan not in. Fungsi in atau not in digunakan untuk melakukan filtering terhadap record yang dipilih. Jika data pada suatu kolom sesuai dengan daftar in atau not in maka record yang mengandung data tersebut ditampilkan.

```

SELECT <nama kolom>
      FROM <nama tabel>
      WHERE <nama kolom> in (katakunci1,katakunci2,...)

```

The screenshot shows the MySQL Workbench interface. The top bar displays 'Host: 127.0.0.1', 'Database: car_service', 'Table: items', and 'Query*'. The main area contains a query window with the following code:

```

1 select ID, itemName, sellPrice From items
2 Where sellPrice in (40000,20000)

```

Below the query window is a results table titled 'items (3x2)'. The table has three columns: 'ID', 'itemName', and 'sellPrice'. The data is as follows:

ID	itemName	sellPrice
21	Areon Nice	40,000
22	Areon Sport Lux	20,000

Gambar 4.16 Query MySQL untuk menampilkan tiga atribut dengan sellPrice 40000 dan 20000



The screenshot shows the MySQL Workbench interface. At the top, it displays 'Host: 127.0.0.1', 'Database: car_service', and 'Table: items'. Below this is a code editor window containing the following SQL query:

```

1 select ID, itemName, sellPrice From items
2 Where sellPrice not in (40000,20000)

```

Below the code editor is a results grid titled 'items (3x7)'. The grid contains the following data:

ID	itemName	sellPrice
5	Mesran Super	160,000
6	Fastron	285,000
7	Rored EPA	55,000
10	Shell Helix Hx 7	340,000
23	Dorfree Cair	35,000
24	Dorfree Paper	18,000
74	STP Oil treatment	45,000

Gambar 4.17 Query MySQL untuk menampilkan tiga atribut dengan sellPrice selain 40000 dan 20000

4.2.5.4. Operator Like

Like hanya digunakan jika data berupa string atau karakter. Sebagai contoh perhatikan query berikut

```

SELECT <nama kolom>
FROM <nama tabel>
WHERE <nama kolom> LIKE (operator);

```

Karakter yang digunakan untuk fungsi like adalah persen (%) dan underscore(_).

1. % → string apapun
2. _ → character apapun

Contoh-contoh berikut akan lebih memudahkan pemahaman tentang pemakaian fungsi like.

1. "Rum%" → "Rumanystring" → kata yang awalnya dimulai dengan Rum.
2. "%fi%" → "anystringfianystring" → kata yang di tengah-tengahnya mengandung karakter fi.
3. " __ " → kata yang tepat dua karakter.
4. " ___ %" → kata yang minimum terdiri dari 4 karakter.



```

Host: 127.0.0.1 Database: car_service
1 select itemName From items
2 Where itemName like "____%"

items (1x9)
itemName
Mesran Super
Fastron
Rored EPA
Shell Helix Hx 7
Areon Nice
Areon Sport Lux
Dorfrees Cair
Dorfrees Paper
STP Oil treatment

```

Gambar 4.18 Query MySQL dengan menggunakan LIKE untuk menampilkan data dengan kata minimum terdiri dari 5 karakter

4.2.5.5. Null dan Not Null

Filtering yang lain menggunakan kata kunci is null atau is not null. Perhatikan query berikut

```

SELECT <nama kolom>

FROM <nama tabel>

WHERE <nama kolom> is (not null/null);

```

4.2.5.6. Renaming

Renaming adalah mengganti nama dari sebuah tabel atau memberikan alias. Pengganti nama ini hanya berlaku dalam query itu saja, sehingga setelah query selesai dijalankan, alias yang dibuat akan hilang. Renaming umumnya digunakan untuk menyingkat nama tabel.

```

SELECT <namakolom1, namakolom2,...>

AS <nama1, nama2,...>

```



Host: 127.0.0.1 Database: car_service Table: items Data

```
1 select itemName, max(sellPrice) as 'harga jual termahal'
2 From items
```

items (2x1)

itemName	harga jual termahal
Mesran Super	340,000

Gambar 4.19 Query MySQL untuk menampilkan itemName dengan harga jual termahal

4.2.5.7. Distinct

Distinct digunakan untuk menampilkan data unik yang ada pada suatu kolom.

Perhatikan contoh berikut.

```
SELECT DISTINCT <nama kolom> FROM <nama tabel>
```

Host: 127.0.0.1 Database: car_service Table: items Data

```
1 select unit From items
```

items (1x9)

unit
galon
galon
Liter
Galon
pcs
pcs
pcs
Pcs

Gambar 4.20 Query MySQL untuk menampilkan data pada kolom unit

Host: 127.0.0.1 Database: car_service Table: items Data

```
1 select distinct unit From items
```

items (1x3)

unit
galon
Liter
pcs

Gambar 4.21 Query MySQL untuk menampilkan data unik pada kolom unit



Perhatikan query pada Gambar 5.20, terlihat bahwa kolom unit dari seluruh row pada tabel krs ditampilkan. Sedangkan pada query Gambar 5.21, hanya nilai unik saja yang ditampilkan. Atau dengan kata lain, apabila menggunakan distinct, data yang berulang akan ditampilkan sekali saja.

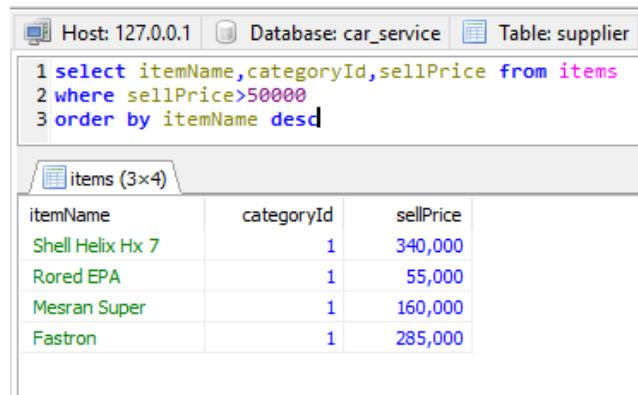
4.2.6. Order By

ORDER BY adalah keyword yang digunakan untuk mengurutkan data. Terdapat dua jenis pengurutan data, yaitu secara ascending dan descending. Default fungsi ORDER BY adalah mengurutkan secara ascending. Jika ORDER BY diikuti oleh kata DESC baru data akan diurutkan secara descending.

```
SELECT <nama kolom> FROM <nama tabel>
```

```
ORDER BY <nama kolom> (DESC/ASC);
```

Berikut adalah contoh query MySQL yang menggunakan order by secara desending



The screenshot shows a MySQL Workbench interface. The query window contains:

```
Host: 127.0.0.1 Database: car_service Table: supplier
1 select itemName,categoryId,sellPrice from items
2 where sellPrice>50000
3 order by itemName desc
```

The results window shows a table titled "items (3x4)" with the following data:

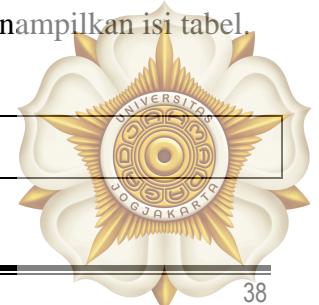
itemName	categoryId	sellPrice
Shell Helix Hx 7	1	340,000
Rored EPA	1	55,000
Mesran Super	1	160,000
Fastron	1	285,000

Gambar 4.22 Query MySQL untuk menampilkan data dengan sellPrice lebih dari 50000 terurut berdasarkan nama item secara descending

4.2.7. Limit

Fungsi LIMIT digunakan untuk membatasi tampilan record dari tabel MySQL kita menggunakan parameter LIMIT pada perintah SELECT. Parameter LIMIT ini nantinya berguna ketika membuat *pagination* atau pengaturan halaman untuk menampilkan isi tabel. Cara penggunaan LIMIT dengan menuliskan jumlah *record* adalah :

```
SELECT * FROM nama_tabel LIMIT jumlah_record;
```



Sedangkan penggunaan LIMIT dengan menuliskan awal *record* dan jumlah *record* yang ditampilkan adalah sebagai berikut.

```
SELECT * FROM nama_tabel LIMIT awal_record, jumlah_record;
```

Berikut ini adalah contoh query SELECT dengan fungsi LIMIT.

```
SELECT * FROM items order by itemName limit 2,3 ;
```

ID	itemName	CategoryId	SupplierId	BuyPrice	SellPrice	Unit
23	Dorfree Cair	2	9	24,000	35,000	pcs
24	Dorfree Paper	2	9	12,000	18,000	pcs
6	Fastron	1	5	232,000	285,000	galon

Gambar 4.23 hasil *running query* dengan menggunakan LIMIT

```
SELECT * FROM items order by itemName limit 2,3 ;
```

ID	itemName	CategoryId	SupplierId	BuyPrice	SellPrice	Unit
24	Dorfree Paper	2	9	12,000	18,000	pcs
6	Fastron	1	5	232,000	285,000	galon

Gambar 4.24 hasil *running query* dengan menggunakan LIMIT

Terdapat perbedaan hasil perbedaan dari *running query* pada kedua syntax. Penjelasan mengenai hal tersebut adalah:

1. Limit 2, 3: hasil *query* akan menampilkan data dimulai dari indeks baris ke-2, sebanyak 3 *record* data.
2. Limit 3, 2: hasil *query* akan menampilkan data dimulai dari indeks baris ke-3, sebanyak 2 *record* data.

Catatan : Indeks baris pada MySQL dimulai dari 0 (nol).

4.2.8. Cross Join

Cross join merupakan penggabungan dari beberapa tabel yang akan menghasilkan *Cartesian product*. Syntax umum dari CROSS JOIN adalah :

```
SELECT nama_kolom FROM nama_tabel1, nama_tabel2, nama_tabel-n
```

Contoh implementasi dari *cross join* yang telah dimodifikasi (menggunakan klausula *where*) adalah :



Host: 127.0.0.1 | Database: car_service | Table: supplier | Data | Query*

```
1 select * from items as a,category as b where a.categoryId=b.Id
```

Result #1 (9x9)

ID	itemName	CategoryId	SupplierId	BuyPrice	SellPrice	Unit	id	categoryName
5	Mesran Super	1	5	108,000	160,000	galon	1	Oli
6	Fastron	1	5	232,000	285,000	galon	1	Oli
7	Rored EPA	1	5	32,000	55,000	Liter	1	Oli
10	Shell Helix Hx 7	1	4	287,500	340,000	Galon	1	Oli
21	Areon Nice	2	21	25,000	40,000	pcs	2	Parfum
22	Areon Sport Lux	2	21	12,000	20,000	pcs	2	Parfum
23	Dorfree Cair	2	9	24,000	35,000	pcs	2	Parfum
24	Dorfree Paper	2	9	12,000	18,000	pcs	2	Parfum
74	STP Oil treatment	1	22	35,000	45,000	Pcs	1	Oli

Gambar 4.25 hasil *running query* menggunakan *cross join* dengan kondisi kolom categoryid pada tabel items harus sama dengan kolom id di tabel category

4.2.9. Inner Join

INNER JOIN akan mengambil semua row dari tabel asal dan tabel tujuan dengan kondisi nilai *key* yang terkait saja (jika ada). Apabila tidak ada, maka *row* tersebut tidak akan muncul. Kalau tidak terdapat kondisi *key* terkait antar tabel, maka semua *row* dari kedua tabel dikombinasikan. *Syntax* umum dari INNER JOIN adalah sebagai berikut :

```
SELECT nama_kolom1, nama_kolom2, nama_kolom-n FROM nama_tabel1 INNER JOIN
nama_tabel2 ON klausa
```

```
1 select * from items inner join category on categoryId=category.Id
```

Result #1 (9x9)

ID	itemName	CategoryId	SupplierId	BuyPrice	SellPrice	Unit	id	categoryName
5	Mesran Super	1	5	108,000	160,000	galon	1	Oli
6	Fastron	1	5	232,000	285,000	galon	1	Oli
7	Rored EPA	1	5	32,000	55,000	Liter	1	Oli
10	Shell Helix Hx 7	1	4	287,500	340,000	Galon	1	Oli
21	Areon Nice	2	21	25,000	40,000	pcs	2	Parfum
22	Areon Sport Lux	2	21	12,000	20,000	pcs	2	Parfum
23	Dorfree Cair	2	9	24,000	35,000	pcs	2	Parfum
24	Dorfree Paper	2	9	12,000	18,000	pcs	2	Parfum
74	STP Oil treatment	1	22	35,000	45,000	Pcs	1	Oli

Gambar 4.26 hasil *running query* menggunakan *inner join* dengan kondisi kolom categoryId pada tabel items harus sama dengan kolom Id di tabel category



4.2.10. Left (Outer) Join

Left outer join akan menampilkan semua baris dari tabel kiri, dan akan menampilkan hasil NULL di tabel kanan apabila tidak terdapat hubungan dengan tabel kiri .

Result #1 (9x18)								
ID	supplierName	ID	itemName	CategoryId	SupplierId	BuyPrice	SellPrice	Unit
5	Teruna Gema Nusa	5	Mesran Super	1	5	108,000	160,000	galon
5	Teruna Gema Nusa	6	Fastron	1	5	232,000	285,000	galon
5	Teruna Gema Nusa	7	Rored EPA	1	5	32,000	55,000	Liter
4	Meka Adi Pratama	10	Shell Helix Hx 7	1	4	287,500	340,000	Galon
9	JJ Cars Accessories	23	Dorfree Cair	2	9	24,000	35,000	pcs
9	JJ Cars Accessories	24	Dorfree Paper	2	9	12,000	18,000	pcs
6	Cv. Makin Jaya	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)
7	PT. Enseval Putera Megatrading	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)
8	Cv. Cahaya Mulia Lestari	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)
10	RS Accessories	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)
11	Padi Mili	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)
12	Aneka Jaya	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)
13	Sumber Rejeki	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)
16	Trimo Motor	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)
17	Wurth	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)
18	PT. Dimas Surya Aditama	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)
19	Toko Karunia	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)
20	Nabita Motor	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)

Gambar 4.27 hasil running query menggunakan *left outer join* dengan kondisi kolom supplierId pada tabel items harus sama dengan kolom Id di tabel supplier

4.2.11. Right (Outer) Join

Kebalikan dari *left outer join*, *right outer join* akan menampilkan semua baris dari tabel kanan, dan akan menampilkan hasil NULL di tabel kiri apabila tidak terdapat hubungan dengan tabel kanan.

Result #1 (9x15)								
ID	itemName	CategoryId	SupplierId	BuyPrice	SellPrice	Unit	ID	categoryName
5	Mesran Super	1	5	108,000	160,000	galon	1	Oli
6	Fastron	1	5	232,000	285,000	galon	1	Oli
7	Rored EPA	1	5	32,000	55,000	Liter	1	Oli
10	Shell Helix Hx 7	1	4	287,500	340,000	Galon	1	Oli
21	Areon Nice	2	21	25,000	40,000	pcs	2	Parfum
22	Areon Sport Lux	2	21	12,000	20,000	pcs	2	Parfum
23	Dorfree Cair	2	9	24,000	35,000	pcs	2	Parfum
24	Dorfree Paper	2	9	12,000	18,000	pcs	2	Parfum
74	STP Oil treatment	1	22	35,000	45,000	Pcs	1	Oli
(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	3	Aksesoris
(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	4	Minuman
(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	5	Lampu
(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	6	Kanebo
(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	7	Lain-Lain
(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	8	Alarm

Gambar 4.28 hasil running query menggunakan *right outer join* dengan kondisi kolom supplierId pada tabel items harus sama dengan kolom Id di tabel supplier

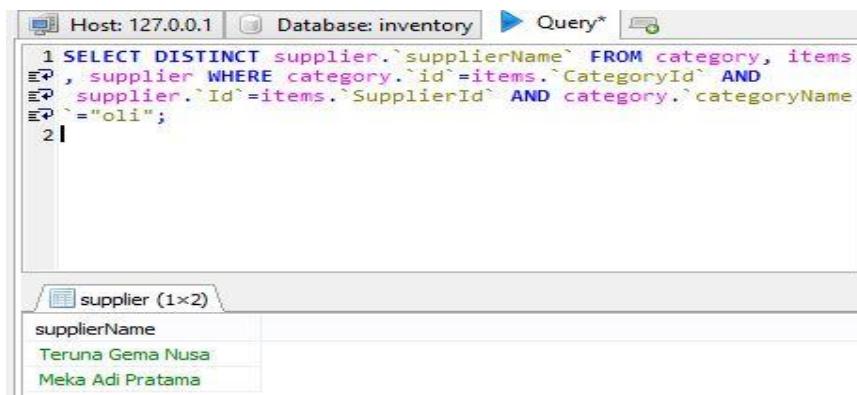


4.2.12. Select dengan banyak tabel

SQL tidak menentukan batasan untuk jumlah tabel yang mungkin digabungkan dalam statemen SELECT. Peraturan dasar untuk membuat join tetap sama. Pertama, daftarkan semua tabel dan kemudian tentaukan hubungan di antara setiap tabel. Contohnya sebagai berikut :

```
SELECT
    tabel1.column_name1,
    tabel1. column_name2,
    tabel2. column_name1,
FROM tabel1, tabel2 WHERE tabel1.PK=tabel2.FK
```

Untuk contoh penerapan select dengan banyak tabel ini dapat dilihat pada Gambar 4.29



The screenshot shows a MySQL Workbench interface. The query editor window displays the following SQL code:

```
1 SELECT DISTINCT supplier.supplierName FROM category, items
  , supplier WHERE category.id=items.CategoryId AND
  supplier.Id=items.SupplierId AND category.categoryName
  = "oli";
2 |
```

Below the query editor, the results pane shows a table named 'supplier' with two rows:

supplierName
Teruna Gema Nusa
Meka Adi Pratama

Gambar 4.28 Query dengan banyak tabel untuk menampilkan nama supplier yang men-supply barang dengan kategori oli

Contoh di atas merupakan query untuk menampilkan nama supplier yang men-supply barang dengan kategori oli. Pada query tersebut dilakukan penggabungan terhadap 3 tabel yaitu category, items dan supplier. Jumlah tabel yang dapat digabungkan dalam sebuah query dapat disesuaikan dengan kehendak praktikan.

4.2.13. Subquery

Nested query atau juga disebut dengan subquery adalah query yang minimal memiliki dua perintah SELECT. Format dari subquery adalah sebagai berikut:



```

SELECT nama_kolom [, nama_kolom]
FROM tabel1 [, tabel2 ]
WHERE column_name OPERATOR
  (SELECT nama_kolom [, nama_kolom]
   FROM tabel1 [, tabel2]
   [WHERE])

```

Untuk pengaplikasian sub query ini dapat di lihat pada contoh yang ditunjukan pada Gambar 4.29

The screenshot shows the MySQL Workbench interface. In the top bar, it says "Host: 127.0.0.1", "Database: inventory", "Table: supplier", and "Data". Below the bar, there is a SQL editor window containing the following code:

```

1 select supplierName from supplier where id in (select
2 SupplierID from items where CategoryId=1)
3

```

Below the editor, the "Data" tab is selected. It shows a table named "supplier (1x2)" with one row of data:

supplierName
Teruna Gema Nusa
Meka Adi Pratama

Gambar 4.29 Menampilkan nama supplier yang men-supply barang dengan kategori oli dengan menggunakan subquery

Gambar diatas menunjukan proses query untuk menampilkan supplier yang men-supply barang dengan kategori oli. Subquery ini akan menampilkan hasil yang sama dengan query menggunakan banyak tabel. Sebuah subquery digunakan untuk mengembalikan data yang akan digunakan dalam query utama sebagai syarat untuk lebih membatasi data yang akan diambil.

4.2.14. Fungsi Aggregat dan GROUP BY

Fungsi Aggregat adalah fungsi matematika sederhana dalam SQL. Fungsi Aggregat hanya bisa digunakan untuk kolom bertipe angka (misal integer, shortint). Fungsi Aggregat sebagai berikut:

SUM	<input type="checkbox"/>	nilai jumlah suatu kolom
AVG	<input type="checkbox"/>	nilai rata – rata suatu kolom
MAX	<input type="checkbox"/>	nilai maksimal suatu kolom
MIN	<input type="checkbox"/>	nilai minimum suatu kolom
COUNT	<input type="checkbox"/>	nilai cacah suatu kolom



Fungsi GROUP BY digunakan untuk mengelompokkan data berdasarkan menurut satu atau lebih kolom. Fungsi ini digunakan untuk memfasilitasi fungsi Aggregat.

```
SELECT column1, column2  
FROM table_name  
WHERE [ conditions ]  
GROUP BY column1, column2  
ORDER BY column1, column2
```

Untuk implementasi dari contoh query ini ditunjukkan pada Gambar 4.30 untuk menampilkan supplier dan jumlah item yang di *supply*.

The screenshot shows the MySQL Workbench interface. At the top, there are tabs for Host: 127.0.0.1, Database: inventory, Table: items, Data, and Query. The Query tab contains the following SQL code:

```
1 select supplierName, count(items.id) as item_amt from  
2 supplier, items where supplier.id=items.SupplierId group by  
3 supplier.id
```

Below the query, a results grid titled "supplier (2x3)" displays the following data:

supplierName	item_amt
Meka Adi Pratama	1
Teruna Gema Nusa	3
JJ Cars Accessories	2

Gambar 4.30 Menampilkan nama supplier dan jumlah item yang di *supply*

4.2.15. HAVING

Having merupakan fungsi pelengkap dari Aggregat dan digunakan bersama GROUP BY. Fungsinya menyeleksi data yang ditampilkan berdasarkan kondisi Aggregat tertentu.

```
SELECT column1, column2  
FROM table1, table2  
WHERE [ conditions ]  
GROUP BY column1, column2  
HAVING [ conditions ]  
ORDER BY column1, column2
```



Contoh penggunaan **having** dalam query dicontohkan pada Gambar 4.31.

The screenshot shows a MySQL Workbench interface. At the top, it displays the host as 127.0.0.1, database as inventory, table as items, and the current tab as Data. Below this, the SQL query is shown:

```
1 select supplierName, count(items.id) as item_amt from
2 supplier, items where supplier.id=items.SupplierId group by
3 supplier.id having item_amt>2
```

Below the query, the results are displayed in a table titled "supplier (2x1)". The table has two columns: "supplierName" and "item_amt". The data shows one row for "Teruna Gema Nusa" with a value of 3.

Gambar 4.31 Menampilkan nama supplier dengan jumlah item yang distok lebih dari 2

Gambar 4.31 merupakan contoh penggunaan having untuk menampilkan nama supplier dengan jumlah item yang distok lebih dari 2. Hasil dari query tersebut adalah Teruna Gema Nusa dengan jumlah item sama dengan 3.

4.2.16. Exist

Arti kata exist adalah ada. Kata kunci **exists** digunakan pada filtering setelah kata where, biasanya digunakan pada nested query. Perhatikan format umum dalam penggunaan exists.

```
-----  
SELECT column-names  
  
FROM table-name  
  
WHERE EXISTS  
  
(SELECT column-name  
  
FROM table-name  
  
WHERE condition)
```

Syntax exists ini digunakan untuk mengetahui keberadaan setiap record dalam subquery. Exists akan memberikan nilai kembalian benar jika subquery memberikan 1 atau lebih record data. Exists juga umumnya digunakan dengan subquery yang berhubungan. Implementasi penggunaan exists dapat dilihat pada contoh query untuk menampilkan nama supplier dengan item yang memiliki harga beli > 100000.



```

1 SELECT supplierName
2   FROM supplier
3 WHERE exists(SELECT distinct ItemName
4   FROM items where SupplierId=supplier.id and BuyPrice>
5 >100000)

```

supplier (1x2)	
	supplierName
1	Meka Adi Pratama
2	Teruna Gema Nusa

Gambar 4.32 Menampilkan nama supplier dengan item yang memiliki harga beli 100000

Dari hasil query didapatkan bahwa supplier yang memiliki harga beli item dengan harga > 100000 adalah Meka Adi Pratama dan Teruna Gama Nusa.

4.2.17. VIEW

View mulai ada pada MySQL Versi 5.0. View digunakan untuk menyederhanakan Query SQL dan membatasi field-field yang dibutuhkan pada saat mengakses tabel. View seperti tabel, tetapi datanya berasal dari tabel lain. View akan disimpan dalam basis data sesuai dengan nama yang dibuat. View sebenarnya adalah komposisi tabel dalam bentuk query yang telah ditetapkan.

Sintaks view adalah sebagai berikut:

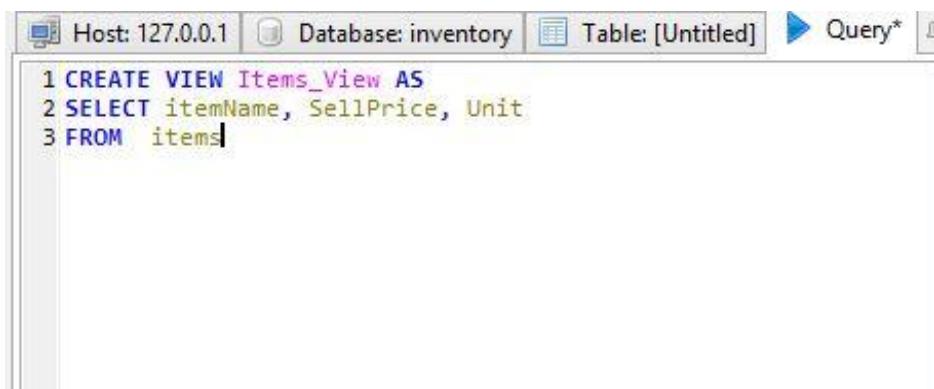
```

CREATE VIEW view_name AS
SELECT column1, column2.....
FROM table_name
WHERE [condition];

```

Berdasarkan dari sintaks view yang telah ada, maka akan dibuat sebuah view dengan nama view_items yang isinya terdiri dari Items_View, harga jual dan unitnya. Implementasi sintak view dapat dilihat pada contoh di bawah ini:





```
Host: 127.0.0.1 Database: inventory Table: [Untitled] Query*
```

```
1 CREATE VIEW Items_View AS
2 SELECT itemName, SellPrice, Unit
3 FROM items
```

Dari query tersebut akan menghasilkan tabel virtual yaitu dengan nama Items_View yang berisi data berupa nama item, harga jual dan unitnya, seperti di bawah ini:

inventory.view_item		
itemName	SellPrice	Unit
Mesran Super	160,000	galon
Fastron	285,000	galon
Rored EPA	55,000	Liter
Shell Helix Hx 7	340,000	Galon
Areon Nice	40,000	pcs
Areon Sport Lux	20,000	pcs
Dorfree Cair	35,000	pcs
Dorfree Paper	18,000	pcs
STP Oil treatment	45,000	Pcs

Gambar 4.32 Membuat view dengan nama view items_view untuk menampilkan nama item, harga jual dan unitnya.

View ini bisa dibuat dari satu atau banyak tabel tergantung pada query SQL yang ditulis untuk membuat tampilan. View ini memungkinkan pengguna untuk melakukan hal berikut:

- Membatasi akses ke data sehingga pengguna hanya dapat melihat atau mengubah apa yang dibutuhkan saja.
- Merangkum data dari berbagai tabel yang dapat digunakan untuk menghasilkan program.



4.3 AKTIVITAS MAHASISWA

1. Mahasiswa mencoba contoh penggunaan query dalam DML mulai dari Gambar 4.1 sampai 4.32.
2. Mahasiswa mengerjakan latihan 4.4.

4.4 LATIHAN TUGAS

Buat query MySQL untuk:

- a. Tampilkan semua item dengan nama suppliernya.
- b. Urutkan hasil pada (a) berdasarkan nama supplier.
- c. Tampilkan semua item dari supplier Meka Adi Pratama.
- d. Lakukan group dan count item berdasarkan supplier.
- e. Delete semua item dari supplier Teruna Gema Nusa.
- f. Delete semua item dengan sellPrice antara 100000 dan 150000.
- g. Tampilkan item dengan nama yang mengandung ‘lu’ .
- h. Tampilkan suppliers yang tidak melakukan supply items untuk toko car service.



BAB V

STORED ROUTINE : PROCEDURE

5.1 TUJUAN PRAKTIKUM

1. Memahami konsep *stored procedure* dan pemanfaatannya pada MySQL.
2. Mengetahui sintak untuk mendeklarasikan dan memanggil *stored procedure*.
3. Mampu mendeklarasikan (membuat), menghapus, dan memanggil *stored procedure* pada basis data MySQL

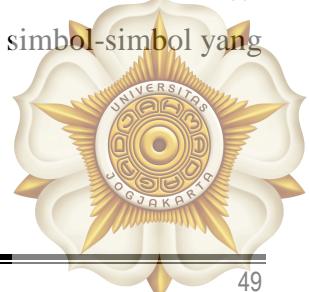
5.2 MATERI, CONTOH DAN ILUSTRASI

Stored routine adalah kumpulan dari perintah SQL yang disimpan di server. Ketika sebuah *stored routine* telah dibuat, *client* tidak perlu menjalankan kumpulan perintah tersebut perbarisnya kembali namun cukup memanggil *stored routine* yang telah disimpan. *Stored routine* dapat berupa *procedure* atau *function*.

Stored procedure merupakan prosedur (seperti subprogram di dalam bahasa pemrograman) yang disimpan di dalam basis data. Keuntungan menggunakan *stored procedure* antara lain :

1. Cepat, kompilasi dilakukan di basis data sehingga lalu lintas data dari/ke program aplikasi berkurang.
2. Program aplikasi menjadi lebih sederhana dan ringkas.
3. Berupa objek dalam basis data, sehingga menghilangkan ketergantungan terhadap bahasa pemrograman yang digunakan.
4. Bersifat *portable*, jika basis data dapat diinstal di manapun, maka dapat dipastikan *stored procedure* tersebut juga dapat dijalankan di manapun.

Penulisan sintak umum *stored procedure* yaitu CREATE PROCEDURE *nama_sp* ([*parameter_prosedur* [, ...]]) *isi_prosedur* dan ditutup dengan *delimiter*. *Delimiter* merupakan penanda akhir suatu *statement SQL*, *default*-nya adalah tanda titik koma (;). Sebelum membuat *stored procedure* sebaiknya *delimiter* diganti dengan simbol-simbol yang jarang digunakan seperti ^, #, dan lain-lain.



Penjelasan sintak umum tersebut adalah sebagai berikut. Bagian yang dicetak miring perlu disesuaikan dengan kebutuhan. Terdapat tiga parameter prosedur yaitu IN, OUT, dan INOUT yang diikuti dengan nama variabel dan tipe datanya. Tanda kurung siku berarti pilihan, dapat digunakan ataupun tidak. Sedangkan tanda [, ...] berarti dapat digunakan lebih dari satu parameter prosedur.

Tiga parameter prosedur yang telah disebutkan dapat dijelaskan sebagai berikut :

1. `IN (default)`: memasukkan nilai dari memori ke SP.
2. `OUT`: mengeluarkan nilai dari SP ke memori.
3. `INOUT`: memasukkan nilai dari memori ke SP dan memungkinkan nilai yang berbeda dari SP dikeluarkan ke memori menggunakan parameter yang sama.

Sintak lain yang sering digunakan yaitu sintak untuk melihat status *stored procedure*, sintak untuk melihat isi *stored procedure*, dan sintak untuk memanggil *stored procedure*. Penulisan sintak untuk melihat status *stored procedure* yaitu `SHOW PROCEDURE STATUS;`. Penulisan sintak untuk melihat isi *Stores procedure* yaitu `SHOW CREATE PROCEDURE nama_sp;`. Penulisan sintak untuk memanggil *stored procedure* yaitu `CALL nama_sp (nilai);`. Sedangkan penulisan sintak untuk menghapus *stored procedure* yaitu `DROP PROCEDURE nama_sp;`.

5.3 AKTIVITAS MAHASISWA

1. Skenario A: Membuat *Hello World!*

a. Langkah 1 : *Login* sebagai *root*

Buka *command prompt/terminal* lalu *login* ke dalam *MySQL* dengan mengetikkan sintak “`mysql -u root -p`” (tanpa tanda petik). Kemudian akan muncul prompt yang meminta untuk memasukkan *password*.

b. Langkah 2 : Menuliskan sintak *stored procedure*

Karena *stored procedure* disimpan di dalam basis data, maka terlebih dahulu masuk ke dalam basis data dengan sintak `USE nama_basisdata`. Karena isi prosedur ini hanya satu *statement SQL* saja, maka tidak perlu mengganti *delimiter*-nya. Lalu tuliskanlah sintak *stored procedure* untuk menampilkan *Hello World!* seperti berikut ini.



```
CREATE STORED PROCEDURE helloworld ()  
    SELECT "Hello World!";
```

c. Langkah 3 : Melihat status *stored procedure*

Lihatlah status *stored procedure* yang telah dibuat.

d. Langkah 4 : Melihat isi *stored procedure*

Lihatlah isi *stored procedure* dengan nama prosedur *helloworld* yang telah dibuat.

e. Langkah 5 : Memanggil *stored procedure*

Dengan sintak CALL, panggilah *stored procedure* untuk menampilkan *Hello World!* yang telah dibuat pada langkah 2.

2. Skenario B : *Stored procedure* dengan parameter IN

a. Langkah 1 : Membuat *stored procedure* dengan parameter IN

Buatlah *stored procedure* untuk menampilkan data mahasiswa dengan *input* kode program studi. Sintaknya sebagai berikut.

```
CREATE PROCEDURE spDataMhs (IN prodiID CHAR(3))  
    SELECT * FROM mahasiswa WHERE kode_prodi LIKE prodiID;
```

b. Langkah 2 : Memanggil *stored procedure*

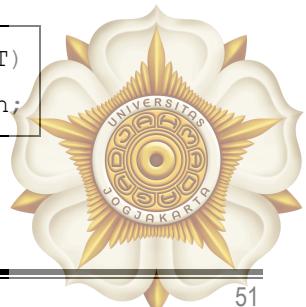
Dengan sintak CALL, panggilah *stored procedure* dengan nama *spDataMhs* yang telah dibuat pada langkah 1 dengan *input* kode program studi.

3. Skenario C : *Stored procedure* dengan parameter OUT

a. Langkah 1 : Membuat *stored procedure* dengan parameter OUT

Buatlah *stored procedure* untuk menghitung dan menampilkan jumlah yang ditawarkan pada tabel mata_kuliah. Sintaknya sebagai berikut.

```
CREATE PROCEDURE spJumlahSKS (OUT jumlah INT)  
    SELECT SUM(sks) INTO jumlah FROM mata_kuliah;
```



b. Langkah 2 : Memanggil *stored procedure*

Pemanggilan *stored procedure* dengan parameter OUT ini sedikit berbeda dengan parameter IN sebelumnya. Pada *stored procedure* dengan parameter OUT, sintak CALL digunakan untuk menyimpan hasil *query* pada *stored procedure* ke dalam variabel tertentu misalnya *jam* kemudian untuk memanggil variabel tersebut digunakan sintak SELECT. Sintak lengkapnya sebagai berikut.

```
CALL spJumlahSKS (@jam);
SELECT @jam AS 'Jumlah SKS';
```

4. Skenario D : *Stored procedure* dengan parameter IN dan OUT

a. Langkah 1 : Membuat *stored procedure* dengan parameter IN dan OUT

Pada skenario ini, isi *stored procedure* terdiri dari beberapa *statement SQL* sehingga perlu mengubah *delimiter default*-nya terlebih dahulu. Buatlah *stored procedure* untuk melihat data mahasiswa dengan *input* tempat lahir kemudian tampilkan jumlah mahasiswa yang lahir di tempat tersebut. Untuk menampilkan jumlah mahasiswa tersebut digunakan sintak FOUND_ROWS() yang akan menghitung banyaknya baris yang dihasilkan dari *statement SELECT* sebelumnya. Sintak lengkapnya sebagai berikut.

```
DELIMITER %%
CREATE PROCEDURE spTempatMhs (IN param1 varchar(20), OUT param2 INT)
BEGIN
    SELECT * FROM mahasiswa WHERE tempat_lahir LIKE param1;
    SELECT FOUND_ROWS() INTO param2;
END;%%
```

b. Langkah 2 : Memanggil *stored procedure*

Pemanggilan *stored procedure* ini menggabungkan antara pemanggilan parameter IN dan parameter OUT sehingga terdapat dua *input* misalnya ‘W%’ sebagai *input* untuk parameter IN yang akan meminta data mahasiswa yang dilahirkan di tempat dengan awalan huruf ‘W’ serta @jml sebagai *input* untuk parameter OUT yang akan menyimpan jumlah mahasiswa yang lahir di tempat tersebut. Sintak lengkapnya sebagai berikut.

```
CALL spTempatMhs('W%', @jml);
SELECT @jml;
```



5. Skenario E : *Stored procedure* dengan parameter INOUT

a. Langkah 1 : Membuat *stored procedure* dengan parameter INOUT

Buatlah *stored procedure* untuk mengubah format tanggal yang semula TTTT-BB-HH menjadi HH-BB-TTTT. Sintaknya sebagai berikut.

```
CREATE PROCEDURE spTanggal (INOUT tgl VARCHAR(10))
SELECT CONCAT(RIGHT(tgl,2) , '-' , SUBSTR(tgl,6,2) , '-' , LEFT(tgl,4))
INTO tgl;
```

b. Langkah 2 : Memanggil *stored procedure*

Pada skenario ini, terdapat tambahan sintak untuk memanggil *stored procedure* yaitu untuk memberikan nilai *input* dengan sintak SET. Sintak lengkapnya sebagai berikut.

```
SET @tanggal = '2020-10-20';
CALL spTanggal (@tanggal);
SELECT @tanggal;
```

6. Skenario F : Latihan membuat *stored procedure*

a. Langkah 1 : Membuat *stored procedure*

Buatlah *stored procedure* dengan nama *spFakultas* dan *input* berupa kode fakultas dan nama fakultas. Ketentuan :

- Jika kode fakultasnya ADA pada tabel, maka *stored procedure* akan melakukan proses UPDATE nama fakultas.
- Jika kode fakultasnya TIDAK ADA pada tabel, maka *stored procedure* akan melakukan proses INSERT data fakultas.

b. Langkah 2 : Memanggil *stored procedure*

Panggilah *stored procedure* tersebut dengan kondisi kode fakultas terdapat pada tabel dan kode fakultas tidak terdapat pada tabel.

5.4 LATIHAN/TUGAS

1. Buatlah sebuah *stored procedure* untuk menghitung jumlah data pada sebuah tabel!
2. Buatlah sebuah *stored procedure* untuk menghitung nilai rata-rata, nilai maksimal, nilai minimal, dan jumlah total sebuah kolom dalam suatu tabel dengan tipe data numeric!
3. Berapakah jumlah maksimal dan minimal parameter IN dan OUT pada sebuah *stored procedure*?



BAB VI

STORED ROUTINE : FUNCTION

6.1 TUJUAN PRAKTIKUM

1. Menerangkan apa itu *function* di dalam *MySQL*
2. Menerangkan cara membuat *function* di dalam *MySQL*
3. Menerangkan fungsi – fungsi dalam *function* di dalam *MySQL*

6.2 MATERI, CONTOH DAN ILUSTRASI

Stored function/user defined function (UDF) merupakan fasilitas dari MySQL mulai versi 5.0. Perbedaan antara *procedure* dan *function* pada MySQL hampir mirip dengan *procedure* dan *function* pada bahasa pemrograman. Fungsi dari *function* adalah mengembalikan suatu nilai skalar dan dapat dipanggil di dalam *statement procedure* atau *function* lain. *Procedure* dipanggil melalui perintah CALL dan dapat mengembalikan nilai melalui variabel *output*.

Cara *function* melalui beberapa langkah antara lain :

1. Membuat *database* terlebih dahulu karena *function* disimpan di dalam *database* bukan di DBMS.
2. Atau jika *database* sudah dibuat sebelumnya masuk ke dalam *database* yang bersangkutan.
3. Lakukan perintah *delimiter <karakter>*. Contoh *delimiter >>*, langkah ini akan mengubah karakter berhenti di *server MySQL* dari ; menjadi >>.

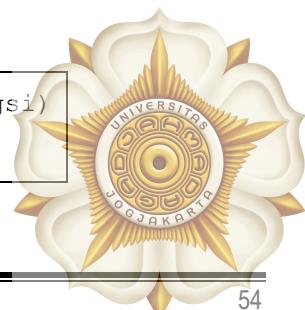
6.3 AKTIVITAS MAHASISWA

Sebelum membuat *function* terlebih dahulu pastikan bahwa Anda sudah masuk ke dalam suatu *database* (*use nama_database*).

1. *Create Function*

Bentuk umum dari perintah create function adalah

```
CREATE FUNCTION nama_function (parameter_fungsi)
RETURNS tipedata isi_fungsi
```



Keterangan

parameter_fungsi : nama_variabel tipe data
isi_fungsi : *statement* fungsi SQL yang benar

Contoh :

```
Delimiter | (enter)
CREATE FUNCTION coba (isi char(20))
RETURNS concat('hello', ',s,'!'); |
(enter)
Select hello ('world') | (enter)
Alter Function
```

Fungsi ini hanya untuk mengubah nama *procedure* atau function dan mengubah karakteristik. Bentuk umumnya sebagai berikut :

```
ALTER {procedure | function} nama [characteristic ...]
```

Fungsi *alter* tidak bisa untuk mengubah badan program, parameter *function*, dan lain-lain. Oleh karena itu apabila kita ingin mengganti badan program suatu *procedure* dapat dilakukan dengan menghapus *procedure* yang ingin diganti baru kemudian membuat *procedure* baru lagi.

2. Drop Function

Fungsi ini untuk menghapus *procedure* atau *function* yang sudah ada.

```
DROP {procedure | function} [if exist] nama
```

if exist bisa ditambahkan untuk menghindari munculnya pesan kesalahan MySQL karena *procedure/function* yang dimaksud tidak ada.

3. Show Function

Apabila diperlukan untuk melihat sintak suatu *procedure* atau *function* yang sudah ada, dapat dilakukan dengan perintah sbb :

```
SHOW CREATE {procedure | function} nama
```

4. Memanggil Function



Fungsi *call* digunakan untuk memanggil *procedure*. *Procedure* atau *function* dapat dipanggil di dalam bahasa SQL atau dipanggil secara langsung dengan fungsi ini. Perintahnya sebagai berikut :

```
CALL nama procedure atau function  
Badan program  
BEGIN...END  
Bentuk umumnya :  
[begin_label : ] begin  
statement1;  
statement2;  
...  
statementN;  
end [end label];
```

Badan program diperbolehkan memuat lebih dari satu blok program (*begin...end*). Namun harus diperhatikan label *begin* dan label *end*-nya harus sama (apabila diberi label).

5. *Declare*

Declare didefinisikan di dalam blok *begin...end* dan di awal baris sebelum *statement-statement* yang lain. Fungsinya adalah untuk mendefinisikan variabel lokal yang berlaku di dalam blok terkait. Ada tiga macam *declare*, yang pertama adalah *declare* untuk mendeklarasikan variabel, *declare handler*, dan *declare cursor*. Berikut bentuk umum deklarasi variabel. *declare nama_variabel <tipe data> [DEFAULT nilai]*.

Contoh :

declare jumlah int default 5 atau declare i,j int.

Contoh-contoh implementasi *stored function*:

Menampilkan jumlah data dari setiap kelas dari tabel tb_siswa yang *field*-nya adalah nim, nama dan kelas.



```

DELIMITER $$

CREATE FUNCTION sf_tampilan_siswa (siswa int)
RETURNS INT
BEGIN
DECLARE jml INT;
SELECT COUNT(*) AS jml_kelas INTO jml
FROM tb_siswa WHERE kelas = siswa;
RETURN jml;
END $$
```

Untuk menampilkan datanya adalah dengan sintak di bawah ini

```
SELECT sf_tampilan_siswa ('2');
```

Maka yang tampil adalah jumlah siswa yang ada di kelas 2.

Menggunakan kondisional IF

```

DELIMITER &&
CREATE FUNCTION tampil_gaji(gaji float(8,2))
RETURNS VARCHAR(20)
BEGIN
DECLARE uang_gaji varchar(20);
    if gaji < 4000 then set uang_gaji = 'Gaji Rendah';
        else set uang_gaji = 'Gaji Tinggi';
    end if;
RETURN uang_gaji;
END &&
DELIMITER ;
```

Untuk menampilkan datanya adalah dengan sintak di bawah ini

```
SELECT nama_depan, nama_belakang, tampil_gaji(gaji) FROM pekerja;
```

Maka yang akan tampil adalah nama depan pekerja nama belakang pekerja dan gaji pekerja tersebut. Apabila gaji di atas 4000 maka akan muncul sebagai pekerja yang bergaji tinggi, sebaliknya jika di bawah 4000 maka gajinya rendah.

Menggunakan kondisi CASE



```

DELIMITER ^^

CREATE FUNCTION sfFakultas (fakID CHAR(2)) RETURNS VARCHAR(20)
BEGIN
    DECLARE namaFak VARCHAR(20);
    CASE fakID
        WHEN 'PA' THEN SET namaFak = 'Matematika dan Ilmu
                      Pengetahuan
                      Alam';
        WHEN 'TK' THEN SET namaFak = 'Teknik';
        ELSE SET namaFak = 'Fakultas Tidak Terdaftar';
    END CASE;
    RETURN namaFak;
END;

DELIMITER ^^

```

Untuk menampilkan datanya adalah dengan sintak di bawah ini

```

SELECT kode_jurusan, nama, sfFakultas(kode_fakultas) FROM jurusan;

```

Maka akan tampil kode jurusan, nama jurusan yang kode jurusan mengikuti kodenya sesuai dengan kondisi CASE nya.

6.4 LATIHAN/TUGAS

1. Sebutkan dan jelaskan perbedaan *stored procedure* dan *user defined function*!
2. Dapatkan sebuah *function* mempunyai lebih dari satu nilai kembalian?
3. Buatlah sebuah *function* untuk mencari NIM mahasiswa dengan IPK tertinggi pada basis data akademik!



BAB VII

TRIGGER

7.1 TUJUAN PRAKTIKUM

1. Memahami prinsip *trigger*.
2. Mengenal sintak-sintak *trigger* pada MySQL.

7.2 MATERI, CONTOH DAN ILUSTRASI

Trigger adalah sebuah objek basis data berupa barisan perintah (*statement*) yang diasosiasikan dengan tabel dan akan aktif/dieksekusi secara otomatis ketika sebuah kejadian tertentu terjadi pada tabel tersebut. *Trigger* tidak dapat diasosiasikan dengan sebuah tabel sementara (*temporary table*) atau sebuah *view*. Berikut adalah sintak umum untuk membuat *trigger* pada DBMS MySQL:

```
CREATE  
    [DEFINER = {user | CURRENT_USER }]  
    TRIGGER trigger_name trigger_time trigger_event  
    ON tbl_name FOR EACH ROW trigger_body
```

Penjelasan:

trigger_time adalah waktu yang didefinisikan untuk mengaktifkan *trigger*. Nilainya dapat berupa BEFORE atau AFTER untuk mengindikasikan bahwa *trigger* aktif sebelum atau sesudah adanya perubahan pada tabel yang diacu sebagai pemicu *trigger*.

trigger_event mengindikasikan jenis *statement* yang membuat *trigger* berjalan. Dapat berupa *statement-statement* berikut:

1. INSERT: *Trigger* akan aktif ketika ada baris baru dimasukkan ke dalam tabel; sebagai contoh dijalankannya INSERT, LOAD DATA, atau REPLACE.
2. UPDATE: *Trigger* akan aktif ketika sebuah baris dimodifikasi; contohnya ketika dijalankan statemen UPDATE.
3. DELETE: *Trigger* akan dijalankan ketika sebuah baris dihapus dari tabel; sebagai contoh adalah apabila dilakukan eksekusi *statement* DELETE dan REPLACE. Perlu diketahui juga bahwa DROP TABLE dan TRUNCATE TABLE tidak akan mengaktifkan *trigger* karena tidak termasuk menggunakan *statement* DELETE.



Statemen SQL yang mengikuti FOR EACH ROW mendefinisikan isi dari trigger, yaitu barisan perintah SQL yang akan dieksekusi setiap kali trigger terpicu. Trigger akan aktif/terpicu ketika setiap baris (*row*) terkena dampak dari statemen pemicu yang dijalankan. Perintah pada isi trigger dimulai dengan kata kunci BEGIN dan diakhiri dengan END, kecuali jika perintah pada isi hanya satu baris perintah maka tidak wajib disertakan kata kunci BEGIN dan END.

```
mysql> delimiter //
mysql> CREATE TRIGGER upd_check BEFORE UPDATE ON account
      -> FOR EACH ROW
      -> BEGIN
      ->     IF NEW.amount < 0 THEN
      ->         SET NEW.amount = 0;
      ->     ELSEIF NEW.amount > 100 THEN
      ->         SET NEW.amount = 100;
      ->     END IF;
      -> END;//
mysql> delimiter ;
```

Contoh di atas membuat trigger bernama upd_check yang akan aktif apabila ada statemen SQL yang melakukan perubahan data pada tabel account. Ketika ada *statement* UPDATE dijalankan maka barisan kode di antara BEGIN dan END akan dijalankan. Seperti yang telah diungkap sebelumnya, kata kunci BEGIN...END digunakan apabila *statement* SQL yang dieksekusi trigger lebih dari satu.

Di dalam BEGIN...END sintak kondisional dan perulangan dapat digunakan. Seperti halnya stored procedure and stored function, apabila ingin menyertakan lebih dari satu statement MySQL maka harus mendefinisikan kembali statement delimiter di MySQL sehingga karakter; dapat digunakan dalam trigger tanpa memutus alur pendefinisian trigger.

7.3 TUGAS/LATIHAN

1. Buatlah sebuah *trigger* yang akan menyimpan data yang dihapus setiap ada penghapusan baris data pada sebuah tabel!
2. Buatlah sebuah *trigger* yang akan membatasi menjadi minimal 0 dan maksimal 100 ketika terjadi UPDATE atau INSERT pada sebuah kolom bertipe data numerik! Contoh: apabila ada masukan data -10 maka akan data yang dimasukkan menjadi 0, dan apabila ada masukan atau perubahan nilai menjadi lebih dari 100 maka nilai 100 yang akan disimpan.
3. Dapatkan dua buah *trigger* (atau lebih) pada sebuah basis data yang sama mempunyai *trigger time* dan *trigger event* yang sama?



BAB VIII

TRANSACTION

8.1 TUJUAN PRAKTIKUM

1. Mampu memahami prinsip-prinsip *transaction* dan kapan *transaction* perlu digunakan.
2. Mengetahui sintaks yang digunakan dalam *transaction*.

8.2 MATERI, CONTOH DAN ILUSTRASI

Transaction adalah kumpulan statemen-statementen SQL yang diperlakukan sebagai satu unit oleh *Relational Database Management System* (RDBMS). Statement-statementen SQL yang dijadikan satu unit sebagai *transaction* adalah statement-statementen yang mempunyai satu tujuan tertentu dan saling bergantung antara statement satu dengan statement yang lainnya. Ketergantungan yang dimaksudkan adalah *transaction* hanya dianggap berhasil apabila seluruh statement berhasil dijalankan.

Sebuah RDBMS yang mendukung *transaction* harus memenuhi sebuah aturan yang disebut *ACID rules*. *ACID (Atomicity, Consistency, Isolation, Durability) rules* berisi prinsip-prinsip dasar yang harus dipenuhi RDBMS untuk dapat menerapkan *transaction* yang aman.

Atomicity: *transaction* harus diperlakukan sebagai satu kesatuan, dan setiap perintah yang menyusun *transaction* harus berhasil dieksekusi seluruhnya barulah eksekusi *transaction* dianggap sukses. Ketika terjadi kegagalan, sistem harus dapat dikembalikan pada keadaan sebelum *transaction* dijalankan.

Consistency: ketika *transaction* selesai dijalankan, sistem haruslah tetap konsisten tanpa ada kesalahan, dengan setiap *constraint* yang ada menunjukkan integritas tabel-tabel dalam basis data. Setiap aturan yang terdapat pada basis data sebelum dijalankannya *transaction* harus tetap terpenuhi setelah dijalankannya *transaction*.



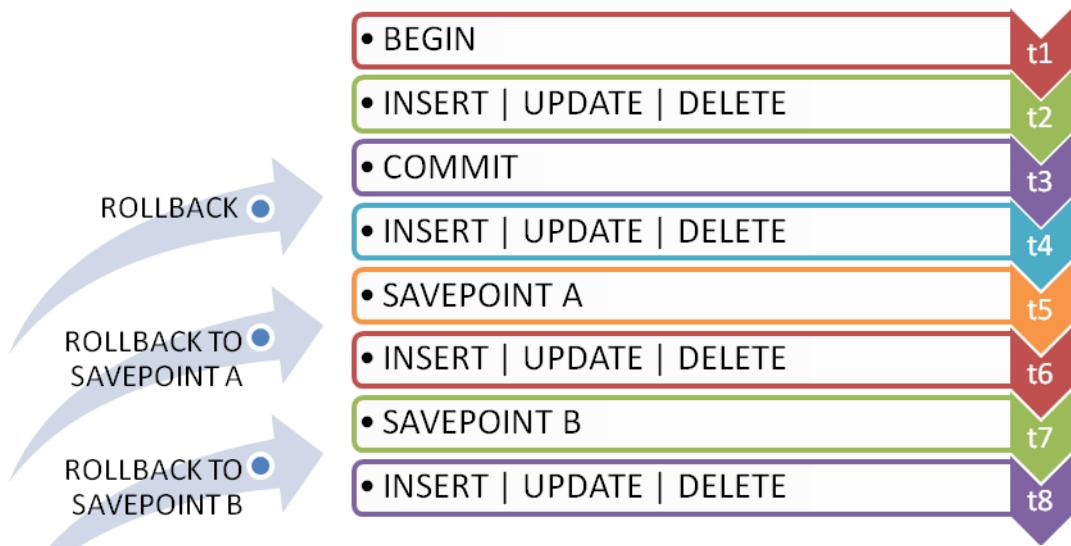
Isolation: perubahan yang diakibatkan oleh *transaction* harus terisolir dari *transaction* yang lain ketika *transaction* tersebut sedang berjalan. Apabila sebuah *transaction* tidak terisolir alias dapat disela oleh *transaction* atau *statement SQL* lain maka konsistensi dari RDBMS tidak terjamin.

Durability: ketika sebuah *transaction* selesai dijalankan, perubahan-perubahan yang dibuat harus diingat/dicatat oleh sistem bahkan pada saat terjadinya kegagalan pada sistem. Sehingga ada jaminan tidak terjadi kesalahan ketika sebuah *transaction* tidak berhasil dieksekusi.

Untuk dapat menjalankan *transaction*, fitur *autocommit* harus dalam keadaan tidak aktif. Pada *setting standar MySQL*, *autocommit* berada pada kondisi aktif. Hal ini dapat diartikan bahwa ketika sebuah *statement SQL* dijalankan untuk mengubah (*update*) tabel maka *MySQL* akan menyimpannya dalam *disk* agar perubahan terjadi secara permanen. Untuk menon-aktifkan *autocommit* bisa dilakukan dua cara yaitu secara implisit (dengan menjalankan *transaction*) atau secara eksplisit (dengan mematikan fitur *autocommit*).

Statement-statement SQL yang digunakan untuk mengontrol *transaction* antara lain BEGIN, COMMIT, ROLLBACK, dan SAVEPOINT. *Statement* BEGIN atau START TRANSACTION digunakan untuk memulai transaksi atau secara implisit digunakan untuk mematikan fitur *autocommit*. *Statement* COMMIT digunakan untuk mengeksekusi *transaction* sehingga perubahan-perubahan yang dibuat dapat diingat/dicatat oleh sistem. *Statement* ROLLBACK digunakan untuk membatalkan seluruh *transaction* yang telah dibuat. *Statement* SAVEPOINT digunakan untuk membuat titik aman dengan nama *identifier* tertentu sehingga dapat dilakukan ROLLBACK ke titik aman tersebut tanpa membatalkan seluruh transaksi. Penjelasan *statement-statement SQL* tersebut dapat diilustrasikan dengan Gambar 1.





Gambar 1 *Statement-statement SQL untuk Mengontrol Transaction*

Pada tipe tabel yang menggunakan *storage engine* yang mendukung *transaction* seperti *InnoDB*, tidak dapat dilakukan transaksi bersarang, yang berarti bahwa jika *Statement BEGIN* dijalankan sebelum *transaction* selesai dilakukan (*Statement COMMIT*), maka *Statement COMMIT* secara otomatis akan dijalankan terlebih dahulu. Pada tipe tabel ini juga, *transaction* diatur oleh klien, sehingga jika pada saat proses *transaction* terjadi gangguan koneksi antara klien dengan *server*, maka *MySQL* secara otomatis akan membatalkan seluruh *transaction* yang telah dibuat, sama seperti konsep *ROLLBACK*.

8.3 AKTIVITAS MAHASISWA

1. Skenario A: Memanipulasi Fitur *Autocommit*.

a. Langkah 1 : Melihat kondisi *autocommit*

Kondisi *default autocommit* bernilai 1 yang berarti bahwa *transaction* akan dicatat/disimpan di dalam sistem secara otomatis. Lihat dan catatlah kondisi *autocommit* dengan menuliskan sintak berikut.

```
SELECT @@AUTOCOMMIT;
```

b. Langkah 2 : Mengubah nilai *autocommit*

Ubahlah kondisi *default autocommit* tersebut dengan nilai 0 menggunakan sintak berikut.

```
SET AUTOCOMMIT = 0;
```



c. Langkah 3 : Memastikan nilai *autocommit* telah berubah

Lihatlah kembali kondisi *autocommit* setelah diubah dan pastikanlah nilainya telah sesuai dengan kondisi yang mana dimungkinkan dilakukan statement COMMIT, ROLLBACK, dan SAVEPOINT.

2. Skenario B: Pemanfaatan *statement COMMIT*

a. Langkah 1 : Membuka dua sesi klien MySQL

Buka *command prompt/terminal* lalu *login* ke dalam MySQL menggunakan *user root*, sesi ini kita sebut sebagai sesi klien pertama. Buka *command prompt/terminal* baru lalu *login* menggunakan *user* lain, sesi ini kita sebut sebagai sesi klien kedua.

b. Langkah 2 : Melihat isi tabel yang sama pada kedua sesi klien MySQL

Masuk ke dalam basis data dengan sintak USE *nama_basisdata* kemudian lihatlah isi tabel dengan sintak SELECT * FROM *nama_tabel* pada kedua sesi klien.

c. Langkah 3 : Memulai *transaction* dan menambahkan baris baru pada sesi klien pertama

Mulai *transaction* pada sesi klien pertama dengan sintak BEGIN atau START TRANSACTION lalu tambahkan baris baru ke dalam tabel dengan sintak INSERT INTO *nama_tabel* VALUES(*data*); kemudian lihatlah perubahan isi tabelnya.

d. Langkah 4 : Melihat isi tabel pada sesi klien kedua dan mencatat hasilnya

Gunakan kembali sintak SELECT * FROM *nama_tabel* pada sesi klien kedua kemudian catatlah hasilnya.

e. Langkah 5 : Melakukan *statement COMMIT* pada sesi klien pertama

Tuliskanlah sintak COMMIT pada sesi klien pertama kemudian lihatlah perubahan isi tabelnya.

f. Langkah 6 : Melihat isi tabel pada sesi klien kedua dan mencatat hasilnya

Gunakan kembali sintak SELECT * FROM *nama_tabel* pada sesi klien kedua kemudian catatlah hasilnya.

g. Langkah 7 : Membandingkan hasil pada Langkah 4 dan Langkah 6 dan menyimpulkan pemanfaatan *statement COMMIT*

h. Bandingkanlah isi tabel pada Langkah 4 (sebelum COMMIT) dan Langkah 6 (setelah COMMIT) kemudian analisis dan simpulkan pemanfaatan *statement COMMIT*



3. Skenario C: Pemanfaatan Statement ROLLBACK

- a. Langkah 1 : Melihat dan mencatat isi tabel sebelum *transaction*

Sebelum *transaction* dimulai, lihat dan catatlah terlebih dahulu isi tabel dengan sintak
SELECT * FROM *nama_tabel*;

- b. Langkah 2 : Memulai *transaction* dan mengubah isi kolom tertentu pada tabel

Mulai *transaction* dengan sintak BEGIN, lalu ubahlah salah satu isi kolom pada tabel dengan sintak UPDATE *nama_tabel* SET *nama_kolom* = *isi_kolom_baru* WHERE *kolom_primary_key* = *isi_kolom_yang_akan_diubah*; kemudian lihatlah perubahan isi tabel yang terjadi.

- c. Langkah 3 : Menghapus baris tertentu pada tabel

Hapus salah satu baris pada tabel dengan sintak DELETE FROM *nama_tabel* WHERE *kolom_primary_key* = *isi_kolom_yang_akan_dihapus*; lalu lihatlah perubahan isi tabel yang terjadi.

- d. Langkah 4 : Melakukan statement ROLLBACK

Tuliskan sintak ROLLBACK kemudian lihatlah perubahan isi tabel yang terjadi

- e. Langkah 5 : Membandingkan isi tabel sebelum dan setelah statement ROLLBACK dieksekusi dan menyimpulkan pemanfaatan statement ROLLBACK

Bandingkanlah isi tabel sebelum statement ROLLBACK dan setelah statement ROLLBACK kemudian analisis dan simpulkan pemanfaatan statement ROLLBACK.

4. Skenario D: Pemanfaatan Statement SAVEPOINT

- a. Langkah 1 : Melihat dan mencatat isi tabel sebelum *transaction*

Sebelum *transaction* dimulai, lihat dan catatlah terlebih dahulu isi tabel dengan sintak
SELECT * FROM *nama_tabel*;

- b. Langkah 2 : Memulai *transaction* dan mengubah isi kolom tertentu pada tabel

Mulai *transaction* dengan sintak START TRANSACTION, lalu ubahlah salah satu isi kolom pada tabel dengan sintak UPDATE *nama_tabel* SET *nama_kolom* = *isi_kolom_baru* WHERE *kolom_primary_key* = *isi_kolom_yang_akan_diubah*; kemudian lihatlah perubahan isi tabel yang terjadi.

- c. Langkah 3 : Menghapus baris tertentu pada tabel

Hapus salah satu baris pada tabel dengan sintak DELETE FROM *nama_tabel* WHERE *kolom_primary_key* = *isi_kolom_yang_akan_dihapus*; lalu lihatlah perubahan isi tabel yang terjadi.



- d. Langkah 4 : Melakukan statemen SAVEPOINT dengan nama A
Tuliskan *statement* SAVEPOINT A; yang berarti bahwa telah dibuat titik aman dengan nama/*identifier* A.
- e. Langkah 5 : Menambahkan baris baru pada tabel
Tambahkan baris baru ke dalam tabel dengan sintak INSERT INTO *nama_tabel* VALUES(*data*); kemudian lihatlah perubahan isi tabelnya.
- f. Langkah 6 : Melakukan *statement* SAVEPOINT dengan nama B
Tuliskan *statement* SAVEPOINT B; yang berarti bahwa telah dibuat titik aman dengan nama/*identifier* B.
- g. Langkah 7 : Melakukan *statement* ROLLBACK TO SAVEPOINT A
Tuliskan sintak ROLLBACK TO SAVEPOINT A; kemudian lihat dan catatlah isi tabel setelah sintak tersebut dieksekusi.
- h. Langkah 8 : Menyimpulkan pemanfaatan *statement* SAVEPOINT
Berdasarkan beberapa langkah tersebut, simpulkanlah pemanfaatan *statement* SAVEPOINT.

8.4 LATIHAN/TUGAS

1. Kombinasikanlah pemanfaatan *statement* COMMIT, SAVEPOINT, ROLLBACK, dan ROLLBACK TO SAVEPOINT pada sebuah *transaction*!
2. Lakukanlah *transaction* dengan tiga atau lebih sesi klien! Apa yang terjadi jika ada klien melakukan perubahan data dengan *transaction* ketika sebuah klien lain terlebih dulu melakukan *transaction* dan belum mengakhiri/mengeksekusinya dengan COMMIT atau ROLLBACK?



LAMPIRAN

PROSES INSTALASI MySQL

9.1. TUJUAN PRAKTIKUM

Tujuan dari praktikum ini adalah, mahasiswa dapat:

1. Memperkenalkan perangkat lunak MySQL
2. Mempelajari instalasi MySQL mulai dari pengunduhan hingga siap dipakai

9.2. MATERI, CONTOH DAN ILUSTRASI

Pada Oktober 2005, MySQL AB selaku produsen MySQL akhirnya mengumumkan bahwa MySQL 5.0 telah dirilis. Hingga sekarang, versi MySQL paling terbaru masih terus dikembangkan. Untuk mengetahui versi terbaru MySQL yang telah dirilis silahkan kunjungi website resminya di <http://www.mysql.com>.

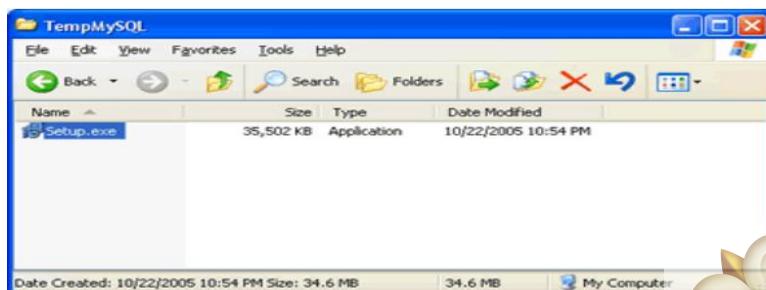
9.2.1. Download

Untuk dapat menginstal dan menggunakan MySQL, terlebih dahulu kita mengunduh *software* MySQL di website <http://www.mysql.com>. Selanjutnya, pilih platform yang sesuai untuk instalasi.

9.2.2. Langkah-Langkah Install MySQL

Setelah selesai mengunduh *software* MySQL, berikut langkah-langkah untuk instalasi MySQL: (Pada buku ini, menggunakan MySQL 5.0)

1. Setelah selesai mengunduh paket instalasi MySQL untuk Windows. Selanjutnya masuk ke direktori tempat menyimpan paket instalasi MySQL. Jalankan file bernama **Setup.exe**.



Gambar 9.1 Menjalankan file setup MySQL



- Untuk melanjutkan proses instalasi kemudian klik **Next >**



Gambar 9.2 Langkah awal instalasi MySQL

- Langkah berikutnya adalah memilih jenis instalasi (setup). Ada 3 jenis instalasi yaitu, *Typical*, *Complete* dan *Custom*. Tipe *typical* berarti memasang fitur-fitur umum dari MySQL. Tipe *Complete* berarti memasang semua fitur dari MySQL, tentunya akan membutuhkan ruang *disk* yang lebih besar. Tipe *Costum* berarti dapat memilih program apa saja yang akan diinstal. Pada tutorial kali ini, memilih tipe *typical*



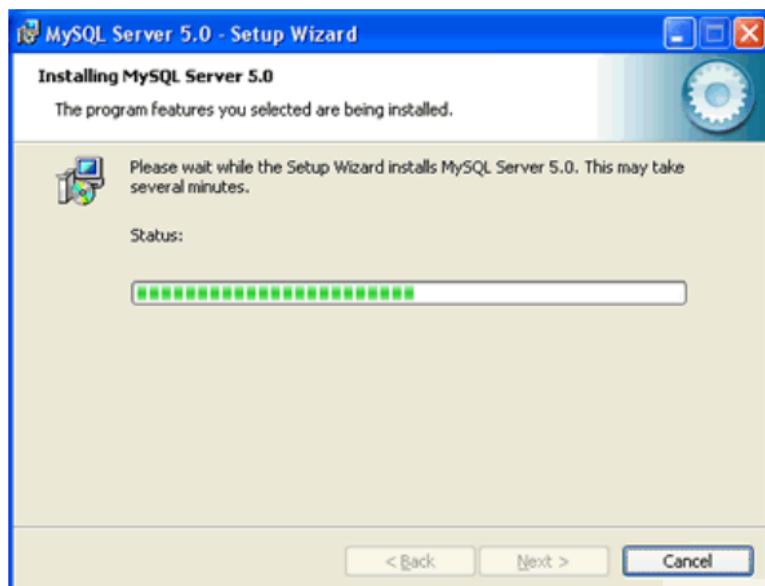
Gambar 9.3 Memilih tipe atau jenis instalasi

4. Langkah selanjutnya adalah konfirmasi direktori instalasi MySQL yaitu C:\Program Files\MySQL\MySQL Server 5.0\. Direktori instalasi tidak dapat diubah karena menggunakan jenis instalasi *typical*. Klik **Install** untuk mulai instalasi.



Gambar 9.4 Lokasi instalasi MySQL

Kemudian proses instalasi akan berlangsung



Gambar 9.5 Proses Instalasi MySQL

5. Kemudian, MySQL menawarkan Anda untuk bergabung secara gratis menjadi anggota di MySQL.com. Bila Anda berminat bergabung, silakan pilih **Create a**



new free MySQL.com account. Kalau sudah pernah memiliki account di MySQL.com, silakan pilih baris kedua (Login to MySQL.com), atau Skip Sign-Up . Pada modul ini kita pilih saja baris ke tiga. (Skip Sign-Up). Kemudian lanjutkan dengan menekan tombol **Next >**.



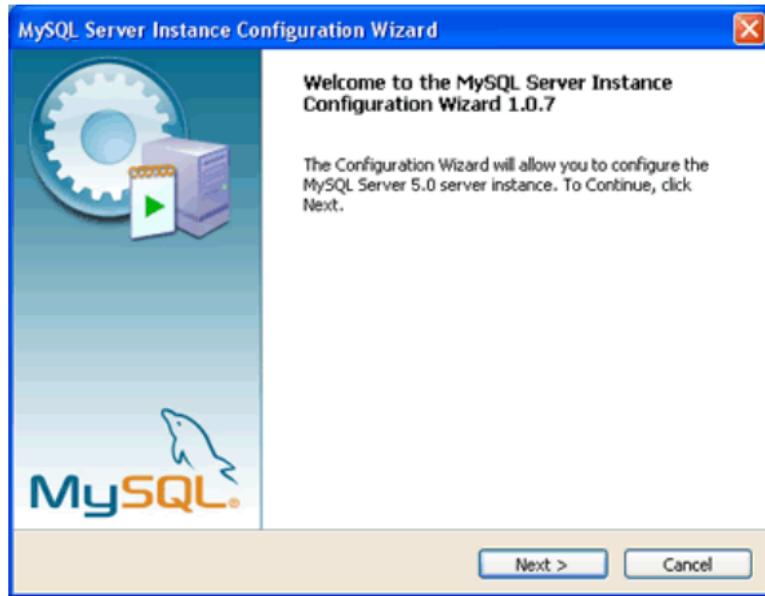
Gambar 9.6 Sign-Up Akun MySQL

6. Tunggu beberapa saat hingga proses instalasi selesai. Kemudian (disarankan) untuk melanjutkan ke proses konfigurasi MySQL server (Configure the MySQL Server now). Dan Anda bisa menekan tombol Finish untuk tahapan ini (tapi akan dilanjutkan dengan proses konfigurasi).



Gambar 9.7 Konfirmasi Konfigurasi MySQL server

7. Klik **Next** untuk melanjutkan konfigurasi



Gambar 9.8 Langkah awal konfigurasi MySQL server

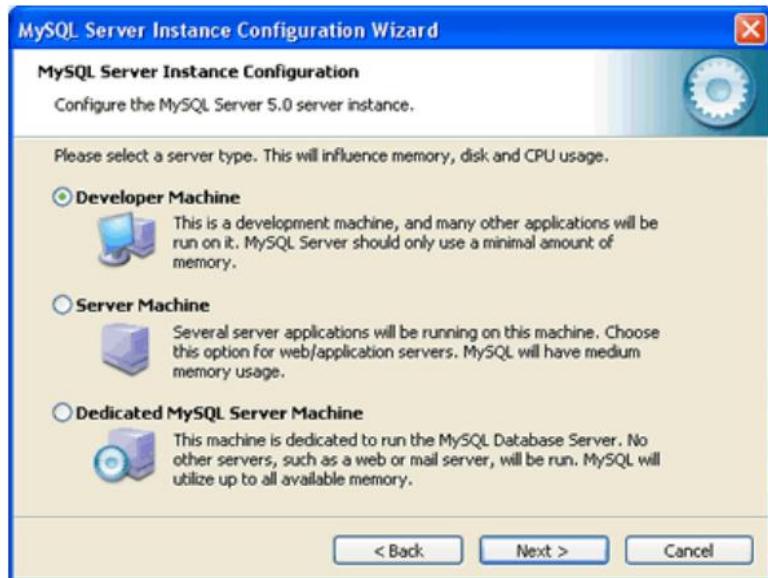
8. Ada dua pilihan konfigurasi, Detailed Configuration dan Standard Configuration. Pada modul ini, kita menggunakan **Detailed Configuration**, karena membuat server menjadi lebih optimal. Klik pada tombol **Next >** untuk melanjutkan.



Gambar 9.9 Tipe konfigurasi MySQL server

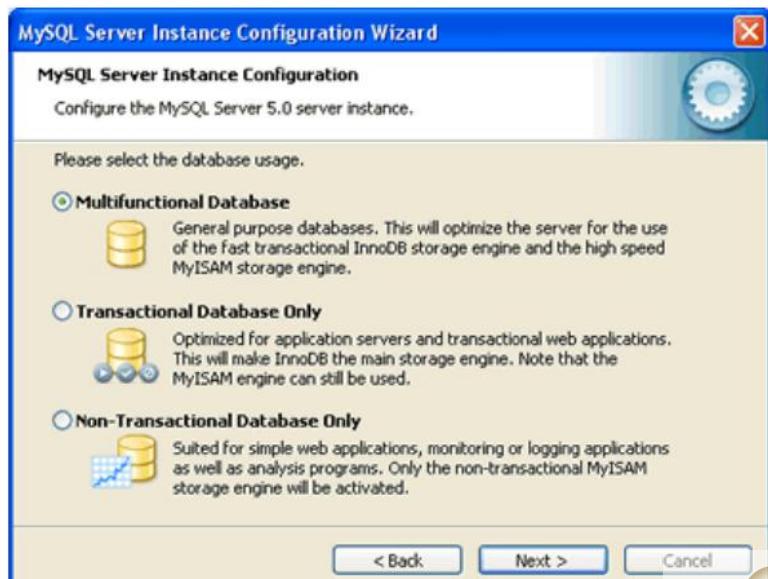
9. Langkah berikutnya terdapat 3 pilihan jenis server yaitu *Developer Machine*, *Server Machine* dan *Dedicated MySQL Server Machine*. Pada modul ini kita memilih **Developer Machine**. Dengan demikian MySQL tidak akan

memonopoli atau memberatkan kerja sistem lainnya yang ada di komputer. Klik pada tombol **Next >** untuk melanjutkan.



Gambar 9.10 Tipe server

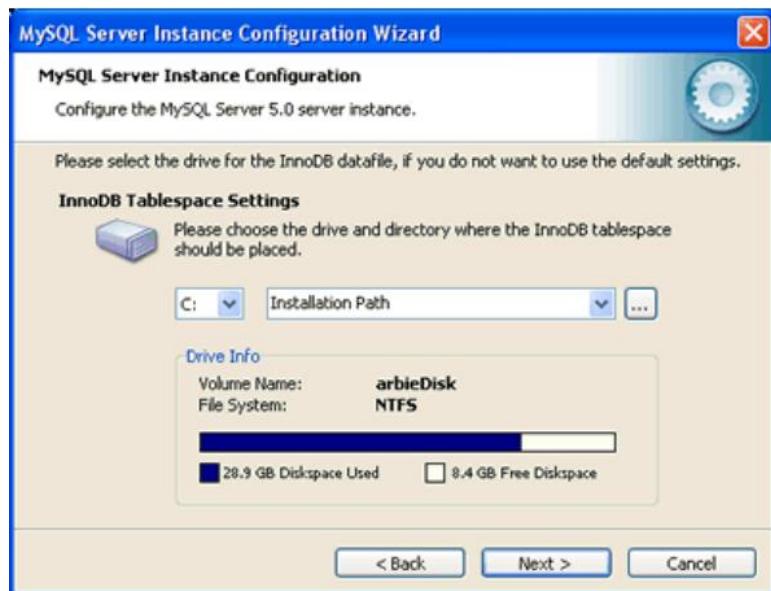
10. Langkah selanjutnya yaitu memilih kegunaan MySQL. Ada 3 pilihan, *Multifunctional Database*, *Transactional Database Only* atau *Non-Transactional Database Only*. Pada modul ini, menggunakan ***Multifunctional Database***. Klik pada tombol **Next >** untuk melanjutkan.



Gambar 9.11 Jenis Database

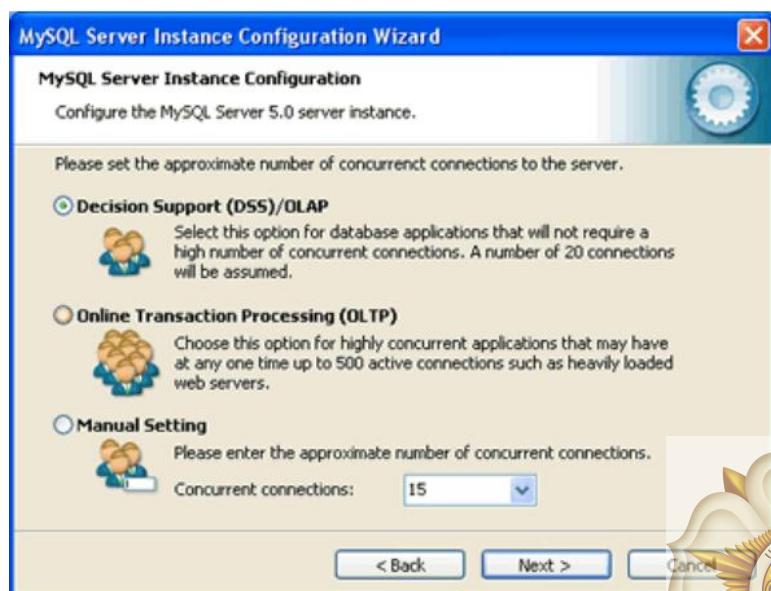
11. Tahap ini menampilkan drive yang akan diinstall InnoDB. Mulai MySQL 4.1.x, InnoDB sudah dimasukkan dalam satu paket. Dengan adanya InnoDB

memungkinkan Anda untuk melakukan *Transaction* dalam database. Dimana table MyISAM (yang merupakan jenis tabel default pada MySQL) tidak mendukung *Transaction*. Untuk tampilan ini gunakan pilihan standar yang ada, dan lanjutkan dengan meng-klik tombol Next.



Gambar 9.12 Setting InnoDB

12. Langkah selanjutnya adalah memilih jumlah maksimum koneksi yang diijinkan pada server MySQL, yaitu *Decission Support (DSS)/OLAP*, *Online Transaction Processing (OLTP)*, atau *Manual Setting*. Pada modul ini dipilih *Decission Support (DSS)/OLAP*. Klik **Next >** untuk melanjutkan konfigurasi.



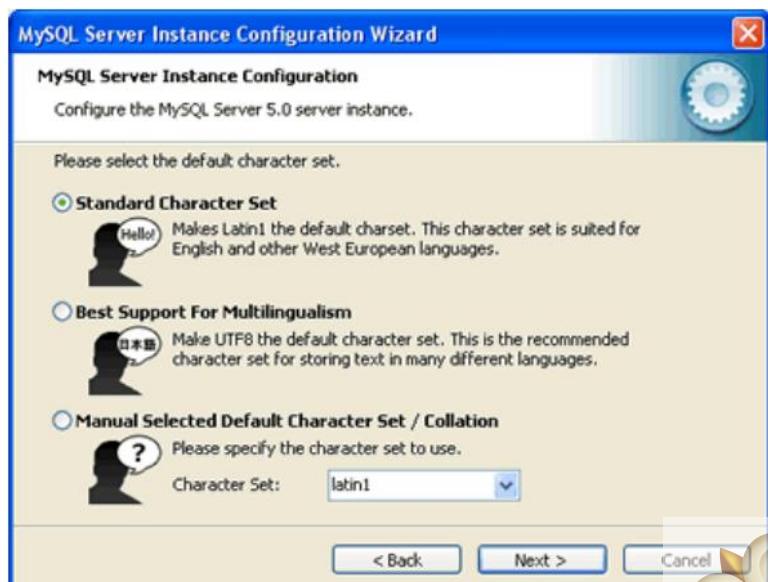
Gambar 9.13 Setting koneksi pada server MySQL

13. Langkah selanjutnya, menentukan apakah akan menggunakan fasilitas TCP/IP untuk mengakses server MySQL atau tidak. Disini ada sedikit perbedaan dibanding dengan proses instalasi MySQL 4.1.x. Ada tambahan pilihan *Enable Strict Mode*. Aktifkan pilihan *Enable Strict Mode* sesuai dengan yang disarankan oleh sistem. Klik **Next >** untuk melanjutkan.

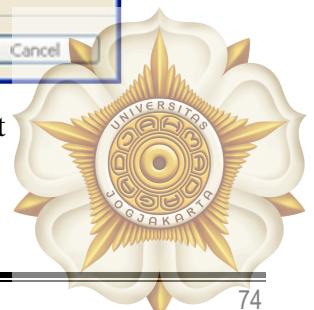


Gambar 9.14 Setting TCP/IP Networking

14. Pilihan berikutnya mengenai jenis karakter yang digunakan. Pilih karakter standar (*Standard Character Set*), kemudian Next.



Gambar 9.15 Setting default character set



15. Tampilan berikutnya, Aktifkan pilihan Install as Windows Service dan Launch the MySQL Server automatically . Dengan pilihan ini maka setiap komputer Anda dinyalakan, secara otomatis program MySQL server akan dijalankan. Begitupun sebaiknya aktifkan pilihan Include Bin directory in Windows Path. Program-program MySQL biasanya disimpan di dalam directory C:\Program Files\MySQL\MySQL Server 5.0\Bin. Dengan mengaktifkan pilihan ini, maka Anda dapat menjalankan atau memanggil program MySQL langsung dari DOS/Command Prompt.



Gambar 9.16 Setting Windows Service

16. Tampilan berikutnya, mengenai masalah keamanan server MySQL. Anda sebaiknya memberikan password khusus sebagai Root, dan tidak memberikan peluang kepada orang-orang untuk memasuki sistem anda tanpa password. Maka aktifkan pilihan Modify Security Setting dan masukkan password Root Anda dengan seksama. Tetapi, matikan pilihan Create An Anonymous Account. Dengan demikian tidak sembarang orang dapat masuk menggunakan MySQL server Anda.

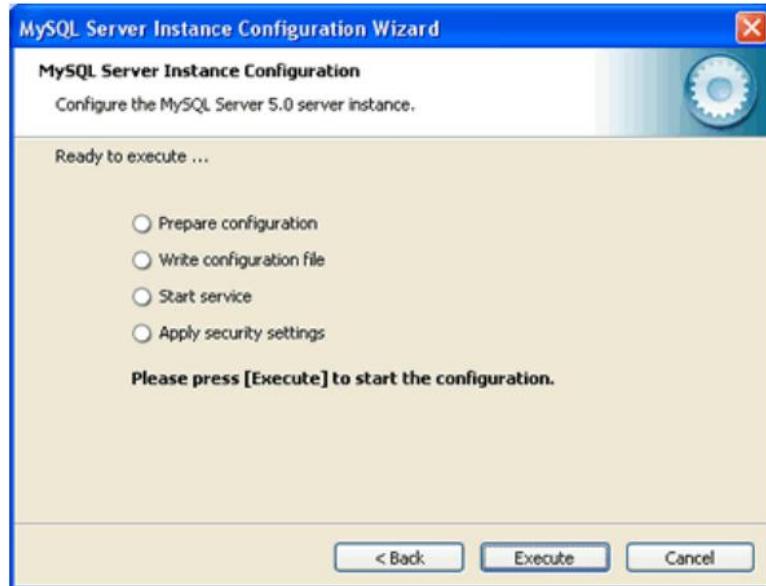
Satu hal lagi, disarankan mematikan juga pilihan Enable Root access from remote machines. Ini untuk mencegah celah-celah yang bisa dimasuki oleh orang-orang yang tidak bertanggungjawab menyelinap ke dalam sistem kita. Lanjutkan dengan menekan tombol Next >.





Gambar 9.17 Setting Security

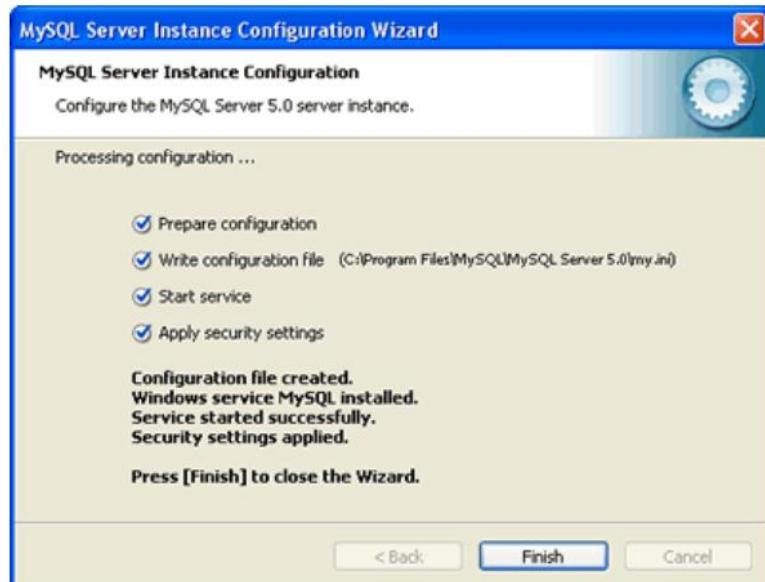
17. Bila Anda telah yakin untuk melanjutkan, klik pada tombol Execute. Dan Anda bisa santai sejenak sambil menunggu proses setting selesai.



Gambar 9.18 Langkah akhir konfigurasi

18. Bila tidak ada kendala apapun, maka seluruh proses instalasi dan setting program MySQL 5.0.x telah selesai. Klik tombol **Finish**.



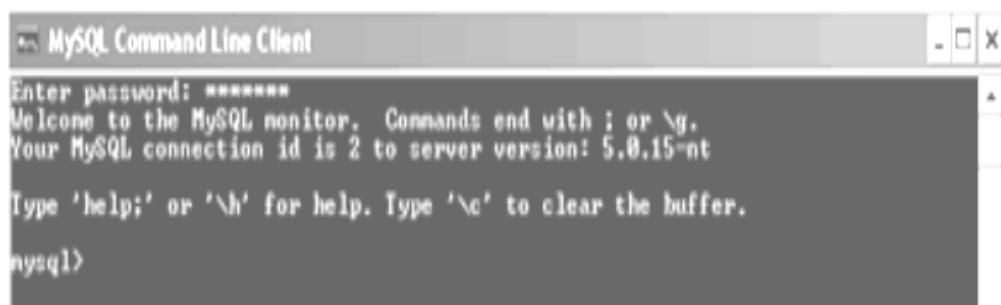


Gambar 9.19 Langkah akhir instalasi MySQL

19. Sekarang silahkan anda lakukan uji coba untuk mengakses MySQL dari Command Line. Ketikkan:

```
C:>mysql -u root -p
```

```
Enter password:*****
```



Gambar 9.20 Mengakses MySQL melalui Command Line

20. Coba juga untuk memasukkan beberapa perintah MySQL seperti show databases, dan sebagainya. Bila hal ini bisa berjalan sebagaimana mestinya. Instalasi MySQL 5.0.x telah selesai dilakukan.

9.3. AKTIVITAS MAHASISWA

Mahasiswa melakukan instalasi MySQL, sebagai contoh pada Gambar 9.1 sampai dengan Gambar 9.20.

