UNIVERSITAS
MERCU BUANA

MODUL PERKULIAHAN

PEMROGRAMAN WEB

Modul Standar untuk digunakan dalam Perkuliahan di Universitas Mercu Buana

Fakultas	Program Studi	Tatap Muka	Kode MK	Disusun Oleh
Ilmu Komputer	Sistem Informasi	01	87007	Team Dosen

Abstract

SDLC, RAD, Design Tersturktur, Class dan Object

Kompetensi

Memahami siklus pengembangan sistem, memahami metodhologi pengembangan sistem, mengenal UML

PENDAHULUAN

1.1. GAMBARAN UMUM

Pada saat ini berbagai macam sistem aplikasi mengakses jutaan data dalam berbagai bentuk format data. Sistem Aplikasi Rumah Sakit mencari data yang digunakan baik itu mencari data pasien ataupun dokter. Aplikasi Penerbangan menggunakan ribuan data untuk mengatur penerbangan dan penumpangnya. Sistem Perbankan juga menggunakan data untuk mencari data transaksi pelanggan. Saat ini berbagai situs internet menggunakan *Search Engine* yang digunakan untuk keperluan pencarian data dalam internet. Perusahaan besar dan kecil menggunakan database untuk mengontrol inventory, keuangan, mengatur sumber daya manusia dan kebutuhan analisa yang lain.

Peranan Database tidak saja mempunyai peranan yang penting dalam sistem arsitek komputer tapi juga sebagai salah satu pendorong kekuatan ekonomi Amerika Serikat dengan sumbangan pajak kurang lebih delapan juta dollar Amerika . Perusahan – perusahaan Amerika Serikat seperti Microsoft Inc, Oracle Corporation, Informix Corporation, Sybase Incooperated, Teradata Corporation mendominasi pasar database di dunia. Penggunaan database tidak saja digunakan perusahaan namun juga meliputi institusi pendidikan, militer , pemerintah ataupun kombinasi dari berbagai institusi tersebut.

Kebanyakan dari relational database tersebut tidak terlepas dari penemuan model yang diketemukan pada tahun 1960 – an dan 1970 – an. Perlu digaris bawahi pengembangan database tidak terlepas daripada pengaruh pemerintah (Amerika Serikat) dalam hal meningkatkan teknologi. Sebagai contoh walaupun pengembangan teknologi database dilakukan oleh perusahaan IBM, namun atas prakasa pemerintah Amerika Serikat melalui Universitas California di Berkeley (UC – Berkeley) untuk menyebarluaskan ide teknologi database dan memberikan hak intelektual secara baik komersial maupun akademik.

1.2. SEJARAH DATABASE

Pemerintah Amerika Serikat selalu melakukan pengumpulan, meng-urutkan, dan membuat laporan dari jumlah data yang besar. Pada tahun 1890, Departemen Sensus Penduduk Amerika menganjurkan majikan karyawan, Herman Hollerith, untuk membuat system peralatan proses informasi secara otomatis. Sebagai hasilnya, mesin kartu (Punch Card) digunakan untuk sensus pada tahun 1890 dan 1900. Pada tahun 1911, perusahaan Hollerith bergabung dengan perusahaan lain, dan juga didukung oleh Departemen Sensus Penduduk untuk membuat sebuah perusahaan baru yang kita kenal dengan nama International Business Machines (Anderson, 1988), sekarang dengan nama IBM.

Selama perang dunia 1, pemerintah Amerika Serikat menggunakan kartu (Punch – Card) untuk melakukan proses berbagai data yang digunakan untuk produksi, mengumpulkan pajak, dan klasifikasi wajib militer. Pada tahun 1935, Departemen Perlindungan Sosial (Social Security Act), menggunakan Kartu (Punch Card) untuk penempatan tenaga kerja dari 26 juta karyawan. Ini adalah pekerjaan administrasi terbesar di dunia, IBM membantu membuatkan mesin khusus untuk pengumpulan data tersebut. Departement Sensus Penduduk akhirnya membeli model pertama dari

komputer digital pertama di pasar komersial, dengan nama UNIVAC 1. Pada tahun 1959, Pentagon sendiri mempunyai 200 komputer untuk kebutuhan bisnis seperti audit biaya, audit peralatan, audit karyawan) dengan biaya pemeliharaan kurang lebih 70 juta dollar Amerika Serikat per tahun. Pada saat itu Amerika didominasi oleh kartu punch untuk pemrosesan data. Berawal dari kebutuhan pemerintah dan akhirnya banyak perusahaan – perusahaan menggunakannya.

Pada awal tahun 1960 –an, muncul keinginan untuk memindahkan tugas piranti keras dari tugas – tugas programmer. Istilah "database" muncul untuk menangkap pengertian bahwa informasi disimpan di dalam komputer dapat berupa konsepsual, terstruktur, dan dapat dimanipulasi secara mandiri tanpa interfensi piranti keras. Pada saat itu, kebanyakan pembuatan aplikasi database dibuat untuk keperluan lingkungan militer dan mata – mata (intelligence), tetapi konsep tersebut dengan cepat diadopsi pada pada pengguna komersial (System Development Corporation, 1964; Fry dan Sibley, 1974).

Pada saat komputer memasuki pasar komersial, sejumlah teknik muncul untuk memfasilitasi akses data, memastikan kualitas, memelihara privasi, dan bahkan menyediakan untuk kontrol data.

Pada tahun 1960, Konferensi System Bahasa Data (Data Systems Languages – Codasyl), yang diadakan oleh Departemen Pertahanan dan Keamanan Amerika Serikat membahas tentang standar aplikasi piranti lunak, membangun bahasa Common Business – Oriented Language (COBOL) untuk programming (ACM Sigplan, 1978), dan menggabungkan bahasa pendefinisian data (Fry dan Sibley, 1974). Magnetic disk drive yang dapat mengakses data secara random, mulai digantikan dengan magnetic tape drives, yang membutuhkan akses data yang berurutan, untuk online storage. Pada tahun 1961, Charles Bachman dari perusahaan General Electric memperkenalkan system Integrated Data Store (IDS) , yang merupakan pelopor system management database yang memanfaatkan teknologi penyimpanan (storage) terbaru ,meliputi novel schemas dan logging, dan kemampuan yang lain.

Pada awal tahun, trend inovasi teknologi praktis dilaksanakan / di buat oleh user group dan peneliti dari industri terkait, dengan pengaruh sedikit dari akademi (CSTB, 1982; Wiederhold, 1984). Pada pertengahan tahun 1960, Bachman dan yang lainnya, kebanyakan dari perusahaan dan pabrik, membuat Database Task Group (DBTG) dibawah Codasyl untuk menyatukan persamaan pendapat dari berbagai bidang . Group tersebut mengumumkan beberapa set spesifikasi untuk bahasa komputer, COBOL terutama sekali, yang mengatur database. Pada Tahun 1971, hal tersebut diumumkan sebagai standar umum, yang digunakan bahasa umum untuk berbagai industri dengan pendekatan Codasyl untuk managament database. Beberapa Codasyl berdasarkan produk diperkenalkan untuk komputer besar (mainframe) oleh Eckerct – Mauchly Computer Corporation (pembuat UNIVAC), Honeywell Incorporated, dan Siemens AG, dan, untuk komputer sedang (minicomputers), dilakukan oleh Digital Equipment Coporation (DEC) dan Prime Computer Corporation.

Secara praktis pemasok produk yang tidak ikut adalah IBM, yang pada awal (Pada Tahun 1968) memperkenalkan produknya sendiri, IMS, yang diambil dari bagian National Aeronautics and Space Administration (NASA) – Appolo Project. Dapat dijalankan pada mesin S/360. Sebaliknya Codasyl berdasarkan model jaringan data sedangkan IBM database menggunakan struktur hirarki. Baik

produk IBM maupun Codasyl, kadang – kadang disebut dengan nama navigasi database karena produk membutuhkan pengguna untuk membuat program atau menggunakan data set.

Pada akhirnya ada seorang ilmuwan dari IBM yang kurang puas dengan produk Codasyl ataupun produk paket database dari IBM. Edgar F. (Ted) Codd, seorang matematikawan, yang bergabung dengan IBM pada tahun 1949 dan pada akhirnya pindah ke IBM San Jose. Codd mendapatkan teknologi database yang sekarang ataupun yang baru harusnya dapat menggunakan satu perintah dan tidak perlu menggunakan beberapa perintah.

Pada seri laporan teknikal IBM, pada makalah " A Relational Model of Data for Large Shared Data Banks," Codd menunjukkan pendekatan baru untuk mengatur dan akses data. Apa yang dimaksud Codd adalah model relational yang menunjukkan 2 kata kunci :

" Yang mendukung pendefinisian data dengan hanya struktur biasa tanpa menambahkan tambahan struktur untuk mewakili representasi dari mesin. Menurutnya, sesuatu yang mendukung basis data tingkat tinggi yang menghasilkan secara maksimal antara program pada satu pihak dan mewakili mesin di satu pihak. (Codd, 1970)."

Dengan kata lain, model relasi mengandung

1. data yang berdiri sendiri dari perangkat keras dan implementasi penyimpanan (storage);
2. Navigasi secara otomatis pada tingkat tinggi, tanpa bahasa prosedur untuk mengakses data. Malahan proses satu record dalam satu waktu, programmer dapat menggunakan satu bahasa yang dapat mengakses keseluruhan data.

Model Codd mempunyai pengaruh yang luar biasa pada penelitian yang ada, banyak menelurkan beberapa prototype projek. Model Relasi Codd mendapat sambutan hangat di dalam masyarakat luas.

1.2.1. SYSTEM R

Pada awal tahun 1970, sejumlah programmer dari IBM pindah dari Yorktown ke San Jose. Group ini mendesain dan membangun prototype system untuk mendemonstrasikan ide – ide dari relational database. Dengan System R, prototype ini bertujuan untuk mendukung mutu yang tinggi, tanpa navigasi, data yang independen untuk di akses oleh secara terus – menerus, dengan integritas yang baik dan kuat (Astrahan et al, 1976). Fase dari Projek pertama, sekitar tahun 1974 – 1975, menghasilkan prototype yang cepat untuk didemonstrasikan apa yang telah dikerjakan, tetapi pembahasan source code ditinggalkan. Fase berikutnya menghasilkan fungsi secara penuh, versi multi pengguna, yang digunakan untuk mengevaluasi percobaan – percobaan berikutnya pada tahun 1978 – 1979. Boleh jadi pembuatan yang dihasilkan dari projek tersebut adalah Structure Query Language (SQL), sekarang menjadi standar akses database di Amerika maupun dunia internasional.

System R tidak mendapat menyakinkan managemen IBM untuk meninggalkan produk sekarang dan beralih ke relational database. IBM dan pelanggannya masih berkeyakinan untuk menggunakan

teknologi IMS. Memerlukan usaha dari yan dibiayai pemerintah untuk membuktikan bahwa relational database dapat menjadi produk yang dapat diperdagangkan.

1.2.2. INGRES

Pada tahun 1973, tentang kapan System R mulai di pakai di IBM, 2 orang ilmuwan dari Universitas California (UC – Berkeley). Michael Stonebraker dan Eugene Wong, menjadi tertarik untuk melakukan penelitian pada relational database. Awalnya, mereka mengumpulkan dana untuk mendisain system data grafikal untuk group ekonomi Berkeley (nama Ingres digunakan untuk mewakili sistem grafik interaktif dan pencarian data). Untuk mendapat dukungan lebih jauh, Stonebraker melakukan pendekatan pada *Advanced Reseach Projects Agency* (DARPA), yang sebelumnya merupakan sumber dana untuk penelitian dan pembangunan system komputer. Baik DARPA maupun Office of Naval Reseach (ONR) tidak mendukung Ingres, namun mereka telah mendukung penelitian database di tempat lain.

Stonebraker lalu memperkenalkan idenya ke perusahaan lain, dan atas pertolongan Wong dan teman dari Berkeley, Lotfi Zadeh, yang mendapat dukungan dari NSF dan 3 wakil dari militer (Air Force Office of Scientific Research, the Army Research Office, dan Navy Electronic Systems Command). Pengalaman dari pencarian dana yang baik untuk Ingres adalah pembelajaran dalam membuat proposal dan pengirimannya.

Setelah mendapat dana, Ingres dibangun, sekitar pertengahan 1970 an, prototype system relation database dibuat yang mempunyai kesamaan dengan System R (IBM) tetapi mempunyai perbedaan pada perangkat keras dan system operasi. Ingres mempunyai persamaan evolusi dengan Sistem R, pada fase pertama demonstrasi solusi pada tahun 1974 kemudian diikuti oleh beberapa revisi untuk membuat pemeliharaan kode lebih baik. Ingres kemudian disebarkanpada komunitas kecil pengguna, baik di dalam maupun di luar akademik, yang memberi input balik pada group pembuat. Proses penyebaran selanjutnya dengan menggunakan mesin DEC di universitas. Anggota dari team projek menulis ulang prototype Ingres secara berulang dalam tahun itu dengan menggabungkan beberapa pengalaman, pendpat balik dari pengguna, dan ide – ide baru. Ingres juga mempunyai bahasa sendiri, QUEL, yang mempunyai kesamaan, dan mempunyai perbedaan dari produk IBM SQL (Stonebraker, 1976, 1980).

Teknologi Ingres menyebar ke sektor komersial melalui 3 saluran utama yaitu kode, individu, dan pulikasi. Berbeda dengan projek IBM, Ingres source code tersedia di masyarakat umum, kurang lebih 1000 salinan telah didistribusikan ke seluruh dunia jadi ilmuwan komputer dan programmer dapat mempunyai pengalaman dengan system dan dapat dimodifikasi seusai dengan kebutuhan. Michael Stonebraker mendirikan Ingres Corporation (dibeli oleh Computer Associates pada tahun 1994) untuk memperdagangkan kode Berkeley secara langsung. Robert Epstein, kepala programer dari Ingres pada tahun 1970, pindah ke Britton- Lee Incorporated dan akhirnya Sybase. Baik Britton-Lee atau Sybase menggunakan ide dan pengalaman dari pemikiran orisinil Ingres, dan beberapa agen pemerintah merupakan pelanggan dari kedua perusahaan tersebut. Computer Associates mengeluarkan versi produk Ingres pada tahun 1980.

Mengikuti pergerakan ilmuwan Ingress melalui komunitas database yang menyebarkan teknologi lebih jauh. Jerry Held dan Carol Youseffi pindah dari UC-Berkeley ke Tandem Computer Incorporated, di mana mereka membuat system relational, yang merupakan awal dari NonStop SQL. Sampai bergabungnya Kleiner, Perkins, Caufield & Byers pada tahun 1988. Held adalah senior vice president teknikal di Oracle, di mana dia memimpin perjuangan perusahaan database tersebut. Paula Hawthorn pindah dari Ingres ke Britton Lee (sama seperti Michael Ubell) dan akhirnya menjadi pendiri dari Illustra Information Technology Incorporated, sekarang merupakan bagian dari Informix. Stonebraker sendiri bekerja untuk Ingres Corporation, Illustra dan Informix. Alumni Ingres yang lain pindah ke AT&T, Hewlett-Packard Company (HP), IBM dan Oracle, membawa semua pelajaran dari Ingres.

Kesuksesan dari SQL selalu menimbulkan kecemburuan di tubuh IBM, yang dapat mengambil teknologi bahasa SQL pada tahun berikutnya dan ini terjadi. **Oracle** yang didirikan oleh **Larry Ellison** (1977) membuat dan mula menjual produk SQL yang kompatibel sebelum IBM menjual produk SQL ke pasar. Ellison belajar dari penjualan SQL melalui publikasi yang dilakukan projek team System R. IBM memaksakan untuk membuat system SQL / DS di bawah ancaman produk kompetitif dari perusahaan database yang ada, seperti Software AG, perusahaan Jerman. Dan perusahan yang belum berpengalaman seperti Informix dan Ingres, yang juga memperkenalkan system relational.

Teknologi System R dan Ingres bukan satu satunya usaha relational database dari buah hasil kerja Cood. Penelitian lain di Universitas Toronto, IBM di Inggriss, Universitas Utah dan Universitas Wiconsin mempunyai kontribusi juga. Menjadi sebuah kenyataan bahwa model relational mempunyai limitasi, terutama pada data yang kompleks. Pada tahun 1982, projek Ingres berakhir dan pada tahun 1985 dirubah menjadi Postgress di Universita California – Berkeley, yang merupakan perpanjangan dari penambahan objek relation model. Perubahan ini bertepatan dengan DARPA yang merekrut program manager untuk database pertama kali, yang membiayai Postgres. Projek ini menjadi bagian dari usaha untuk perpustakaan digital dan penelitian database.

Tahun yang menonjol untuk model relational ada tahun 1980, ketika produk SQL/DS dari IBM memukul pasar untuk mainframe, pemasok kecil mulai menjual generasi kedua system relational dengan pemasaran yang sukses, dan Codd mendapat penghargaan dari Assocation for Computing Machinery's (ACM) . Sistem Relational memasuki jamannya.

Pemerintah Amerikas Serikat mempunyai peranan yang penting dalam perkembangan relational database. Pada awalnya pemerintah mempunyai misi untuk membuat pasar baru untuk teknologi dengan memberikan insentif pada inovasi produk. Kebutuhan Departemen Sensus Penduduk pada proses pengumpulan informasi akhirnya menghasilkan otomisasi dari sebuah komputer. Keterlibatan dunia academic juga mempunyai peranan yang besar dalam menyumbangkan atau memperbaharui kemajuan dari teknologi database.Pada akhirnya perkembangan database juga tidak luput dari perusahaan database yang bersaing satu dengan yang lain untuk mendapatkan pangsa pasarnya. Oracle database, sampai saat ini tetap memimpin teknologi database.

1.3. SEJARAH SINGKAT PERUSAHAAN ORACLE

Tahun	Keterangan

1964 Charles Bachman dari General Electric mengajukan model jaringan dengan record data dihubungkan secara bersama – sama,

1965 North American Aviation's Space Division dan IBM membuat pendekatan kedua berdasarkan model hirarki. Dalam hal ini, data ditampilkan dalam bentuk tree (pohon),

1969 Produk IBM berdasarkan model hirarki dipasarkan dengan nama Information Managemetn System (IMS)

1970 Untuk pertama kali konsep relational database dijabarkan oleh Dr. Edgar F. Codd melalui artikel penelitian IBM publikasi yang berjudul "System R4 Relational."

1979 Relational Software, Incorporated (RSI) didirikan dan me-release Oracle V.2 sebagai relasional database pertama.

1985 Oracle menyatakan 1000 relasional database di pakai pelanggannya.

1983 Relational Software Incorporated dirubah namanya menjadi Oracle Corporation.

1985 Oracle mengumumkan bahwa databasenya dapat berjalan lebih dari 30 platform.

1986 Oracle support model Client / Sever

1988 Oracle memasuki pasa ERP (Enterprise Resources Planning) – Oracle Financial

1989 Oracle database versi 6 diluncurkan. Untuk pertama kalinya PL / SQL diperkenalkan.

1993 Oracle database versi 7.0.

1997 Oracle pertama kali mendukung object pada database yang dikenal dengan nama Object Relational Database Management System) – Oracle 8

1999 Oralce memasuki model internet. Salah satu kemampuan yang besar adalah mendukung java pada oracle database.

2000 Perusahaan Oracle tidak mendukung database management system saja tetapi pembangunan semua lini aplikasi. Application Server dapat dijalankan di arsitektur 3 tier client / server

2001 Oracle 9iDB Direlease.

2003 Oracle Grid diperkenalkan.

Daftar Pustaka

Alan Denis, Barbara Haley Wixon, David Tagerden, *System Analys and Design : an Object – Oriented Approach with UML 2.0*, John Willey and Sons, 2005

Henry C. Lucas Jr., *The Analysis, Design and Implementation of Information Systems 4th*, McGraw Hill, 1992

Satzinger, Jackson, Burd, *Object-Oriented Analysis and Design with the Unified Process*, Course Technology, 2005

Simon Bennet, Steve McRobb and Ray Farmer, *Object-Oriented System Analysis and Design Using UML*, McGraw Hill, 2006

Wendy and Michael Boggs, *UML with Rational Rose 2002*, Sibex Inc., 2002

UNIVERSITAS MERCU BUANA

MODUL PERKULIAHAN

PEMROGRAMAN WEB

Modul Standar untuk digunakan dalam Perkuliahan di Universitas Mercu Buana

Fakultas	Program Studi	Tatap Muka	Kode MK	Disusun Oleh
Ilmu Komputer	Sistem Informasi	02	87007	Team Dosen

Abstract

SDLC, RAD, Design Tersturktur, Class dan Object

Kompetensi

Memahami siklus pengembangan sistem, memahami metodhologi pengembangan sistem, mengenal UML

DATABASE

2.1. PENGENALAN DATABASE

Istilah database mungkin telah umum di dengar kalangan akademik maupun masyarakat teknologi informasi. Database pada awalnya timbul karena adanya kebutuhan seperti pencatatan data dalam jumlah besar (Lihat Sejarah Database). Jikalau hal itu dilakukan oleh seorang manusia tentu memakan waktu yang lebih lama daripada kalau kita menggunakan teknologi komputer. Database dapat diibaratkan sebuah lemari besar di mana di dalamnya terdapat banyak rak – rak yang berisikan dokumen – dokumen. Sebagai Ilustrasi adalah jikalau kita pergi ke perpustakaan maka pertama kali yang kita lakukan adalah melihat daftar buku yang hendak kita cari, kemudian di dalam daftar buku tersebut terdapat nomor yang mengidentifikasikan letak buku tersebut. Berdasarkan no. buku tersebut kita menuju lemari dan rak yang berisikan buku yang kita inginkan. Ilustrasi yang lain adalah ketika kita hendak mencari nama pelanggan pada buku catatan pelanggan. Biasanya catatan tersebut terdiri dari no. pelanggan, nama, no. telpon dsb. Berdasarkan catatan tersebut dapatlah kita mencari pelanggan yang kita inginkan. Sebenarnya jika kita teliti lebih dalam, biasanya pencatatan itu hanya berupa 2 dimensi yaitu baris dan kolom kemudian yang kita kenal dengan nama tabel.

Berawal dari konsep yang di atas, timbul pemikiran bagaimana jikalau di digitalisasi yaitu meminta bantuan mesin untuk membantu manusia melakukan pencatatan. Berawal dari hal yang sederhana akhirnya terciptalah mesin komputer pada tahun 1890 yang melakukan pencatatan dengan menggunakan teknologi komputer (punch card).

Jadi secara harafiah arti dari database adalah *sistem penyimpanan data secara elektronik.* Di sini terlihat dari proses manual ke otomatisasi. Sedangkan sistem di sini menunjukkan sesuatu yang diolah secara teratur sehingga memudahkan seseorang / pengguna untuk mencari data yang diinginkannya. Jadi kesimpulannya dari penggunaan database adalah memudahkan manusia untuk melakukan proses pengolahan data.

Dalam proses selanjutnya pengguna dapat membuat tabel baru, mengisi data baru ke dalam tabel, mengedit data, menghapus data bahkan menghapus tabel tersebut. Intinya adalah tergantung dari kebutuhan pengguna.

Pada awal perkembangan database,data yang disimpan ke dalam database dalam bentuk tabel namun data tersebut berdiri sendiri – sendiri seperti tabel karyawan, tabel pelanggan, dll. Pencarian data dalam tabel dilakukan dengan membaca secara berurutan. Jikalau data banyak maka proses pencarian akan sangat lama. Oleh karena itu tabel yang berdiri sendiri – sendiri tersebut kita sebut dengan istilah flat – file.

2.2. DATA, DATABASE, DATABASE MANAGEMENT SYSTEM

2.2.1. DATA

Data adalah mewakili kenyataan pada suatu enitity (obyek).
Entities adalah sangat umum, mewakili sebuah obyek atau barang yang sudah diidentifikasikan sebelumnya. Sebagai contoh sebuah buku terdiri penulis, no. penerbit, tgl. penerbitan, halaman. Kesemuanya itu mewakili entity yang bernama buku. Sedangkan penerbit, no. buku, diterbitkan oleh, halaman adalah data yang mewakili kenyataan sebuah buku. Data – data tersebut di sebut attribute dalam sebuah entity.
Sebuah entity dapat dihubungkan dengan entity lain yang disebut dengan nama **relationship.** Sebagai contoh entity book mungkin terhubung dengan entity toko melalui hubungan "no.buku". Dalam hal ini hanya ada 2 entity yang terhubung yang disebut dengan **binary relationship**. Klasifikasi dari entity, attribute, relationship bersifat subyektif artinya ketentuan berdasarkan kebutuhan. Sebagai contoh entity buku mempunyai attribute penulis. Namun kita bisa mempertimbangkan penulis sebagai sebuah entity di mana data (attribute) dapat berupa no. induk penulis, nama penulis, alamat penulis. Hubungan antara entity dapat didefinisikan dalam berbagai jalan. Sebagai contoh sebuah buku mungkin hanya mempunya satu penerbit namun penerbit bisa mempunyai banyak buku yang dipublikasikan. Hal ini disebut dengan istilah **cardinality** dari hubungan penerbit.

Ada beberapa hubungan yang dapat dijelaskan

1. One to One : 1 entity mempunyai hubungan hanya dengan 1 entity
2. One to Many : 1 entity mempunyai hubungan dengan banyak entity. Sebagai contoh Setiap penerbit mempunyai banyak buku yang diterbitkan. Namun setiap buku hanya diterbitkan oleh 1 penerbit. Hubungan antara buku dan penerbit di sebut one to many.
3. Many to Many : Seorang penulis mungkin mempunyai banyak buku yang ditulisnya, dan sebuah buku mungkin ditulis oleh beberapa penulis. Hubungan antara penulis dan buku disebut many to many.

Sebuah entity mungkin mempunyai 1 attribute yang unik dari entity yang lainnya. Sebagai contoh no.penerbit untuk menunjukkan sebuah buku. No. Penerbit disebut dengan key attributes.

2.2.2. DATABASE

Database mempunyai 2 tujuan utama yaitu

- Untuk mendapatkan data tentang entity dan hubungannya.
- Untuk memberikan informasi kepada pengguna dari database.

Secara umum definisi dari **database** adalah berbagi (shared), penyimpanan yang saling terhubung (integration collection) , kumpulan (repository) yang terintegrasi dari data. 2 kata yang menitik – beratkan pada pendefinisian dari database adalah berbagi (shared) dan yang saling terhubung (integrated). **Berbagi (shared)** berarti beberapa pengguna dapat meng-akses data di dalam database secara bersama – sama. **Yang terhubung (integrated)** berarti menghilangkan data yang terduplikasi (informasi yang ditulis lebih dari 1 kali). Sebagai contoh informasi tentang data

mahasiswa mungkin saja tersebar di berbagai komputer namun jikalau diletakkan dalam 1 server data mahasiswa itu di akses secara bersama – sama oleh pengguna maka dapat dikatakan database.

2.2.3.DATABASE MANAGEMENT SYSTEM (DBMS)

DBMS adalah sebuah piranti lunak yang mengatur penyimpanan dan akses data di dalam database. Tentunya hal ini di atur secara otomotis – berdasarkan sistem informasi management. Kata management meliputi beberapa fungsi yang meliputi keamanan, akses secara bersama – sama, dan perbaikan. Pengguna dan program meng-akses dan menyimpan data dengan berinteraksi melalui DBMS.

Kebanyakan dari DBMS mempunyai beberapa cara untuk menyimpan data pada alat penyimpanannya, seperti bagaimana data di atur dalam disk, bagaimana cara meng-aksesnya, dan seterusnya dan semuanya berinteraksi dengan level internal dari database.
DBMS bertanggungjawab dalam membagi aplikasi atau pengguna (sesuai dengan hak akses) dari keruwetan level internal dan menampilkan secara mudah, dalam tampilan logika dari sebuah data. Tujuan dari semua database adalah memberikan dukungan data fisik yang berdiri sendiri (physical data independent). Istilah lain adalah penampilan luar (eksternal) yaitu bagaimana menampilkan data kepada masing – masing pengguna.

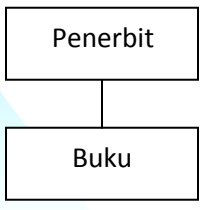
Database juga mempunyai sebuah metadata (data dictionary) yang berfungsi untuk menyimpan informasi tentang database dan objek – objek yang ada di database.

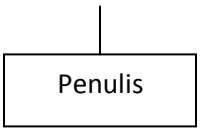
2.3. DATABASE MODEL

Penampilan konsepsual data di mana pengguna melihat adalah berdasarkan model yang disebut dengan database model. Database model mendefinisikan cara menampilkan entity, attribute dan relationship (hubungan) secara terstruktur. Menghubungkan antara konsepsual data dengan data fisik ada tanggungjawab dari DBMS.

Ada 3 data model database yang sampai saat ini digunakan yaitu

a) **Hirarki Model (Hierarcy Model)**: menampilkan data secara hirarki. Sebagai contoh yang mudah adalah ketika melihat struktur organisasi ataupun jikalau kita membuka window eksplorer.





Ada beberapa keterbatasan dari model hirarki, yaitu :

- Karena berstruktur hirarki maka jikalau kita hendak melihat data penulis maka terlebih dahulu kita harus membuka data penerbit,
- Entity anak hanya bisa mempunyai 1 induk entity
- Sebuah entity tidak bisa dihubungkan lebih dari 1 entity

Jadi secara natural hubungan hirarki adalah berdasarkan one to one dan one to many

b) **Model Jaringan (network model)** mempunyai sedikit kelebihan dari model hirarki. Pada model jaringan, sebuah entity dapat mempunyai lebih dari 1 entity induk, dan beberapa buah entity anak dapat mempunyai beberapa buah entity induk. Namun demikian model jaringan masih mempunyai masalah hubungan many to many secara langsung. Sebagai contoh : jikalau kita hendak mencari semua buku yang diterbitkan oleh Bill Gates, lalu dengan menggunakan pointer yang mewakili hubungan penerbit untuk mencari buku yang diterbitkan oleh penerbit. Hal ini memberikan limitasi pada data fisik secara independen. Ketika hubungan berubah, aplikasi program jika ikut berubah.

c) Model Hubungan (Relational Model) diusulkan oleh E.F Codd dalam jurnal ilmiahnya "A Relational model of data for large shared data banks" pada tahun 1970. Codd menjelaskan penampilan data dipaksakan dalam bentuk konsep hubungan matemateka (berdasarkan group). Pada model Codd, data ditampilkan dalam bentuk tabel, dimana setiap tabel mempunyai kolom dan baris. Hubungan yang lain adalah berdasarkan kolom unik (primary key) dan kolom bukan unik (foreign key). Setiap table harus mempunyai 1 kolom unik (primary key) untuk membedakan 1 entity dengan entity lainnya **(entity integrity)**. Kolom bukan unik (foreign key) tidak boleh kosong (null) – **Referential Integrity**.

2.4. BAHASA DATABASE

Digunakan untuk membuat dan memanipulasi objek – objek dan data di dalam database. Interaksi bahasa langsung dengan database disebut **query language.** Sedangkan bahasa yang digunakan di dalam program di sebut dengan **sublanguage.** Ada beberapa komponen bahasa database yaitu

a. Data Definition Language (DDL) seperti CREATE, ALTER, DROP, RENAME, TRUNCATE, ANALZE, GRANT, REVOKE.
b. Data Manipulation Language (DML) seperti DELETE, INSERT, SELECT, UPDATE.
c. Transaction Control Statements seperti COMMIT, ROLLBACK, SAVEPOINT, SET TRANSACTION
d. Session Control Statements seperti ALTER SESSION, SET ROLE
e. System Control Statements seperti ALTER SYSTEM

f. Embedded SQL Statements merupakan kombinasi dari DDL, DML, Transacation Control Statements dalam sebuah bahasa data sub langauge seperti DECLARE, CURSOR, OPEN, CLOSE, DECLARE DATABASE, CONNECT, DECLARE STATEMENT, DESCRIBE, WHENEVER,FETCH, PREPARE,EXECUTE, EXECUTE IMMEDIATE.

2.5. BAHASA RELASIONAL DATABASE (RELATIONAL DATABASE LANGUAGES)

Ada beberapa bahasa relational database yang dapat didefinisikan

a. **Relational Algebra** yaitu bahasa yang berdasarkan kumpulan operator yang meliputi tradisional operator (UNION, INTERSECTION, DIFFERENCE) dan operator khusus sepert (PROJECTION,RESTRICTION, JOIN, PRODUCT, DIVISION).

b. **Relation Calculus** berdasarkan predikat kalkulus yang digunakan untuk hubungan data . Gagasan dasar dari relational calculus adalah query variable yaitu variabel yang menghubungkan beberapa baris dalam tabel. Query ini berdasarkan target yang mendefinisikan spesifik kolom sebagai hasil dan kualifikasi yaitu memilih baris dari target tabel berdasarkan ekspresi dari query variabel tersebut. Contoh dalam kalimat adalah carilah buku yang mempunyai halaman lebih dari 500. Penulisan bahasa databasenya adalah sebagai berikut

```
RANGE OF B IS BOOKS.
RETRIEVE ( B.ISBN,B.TITLE ) WHERE  B.LENGTH  > 500
```

Bahasa database ini disebut QUEL yang diimplementasikan Ingres System. Produk IBM adalah QBE (Query By Example) dalam bahasa SQL.

c. **Mapping Oriented Languages**.

Bahasa ketiga diusulkan oleh Boyce pada tahun 1974. Digunakan untuk pengguna yang tidak mengerti tentang query variabel. Contoh penulisan dalam bahasa databasenya adalah

```
SELECT ISBN, TITLE FROM BOOKS WHERE LENGTH > 500
```

d. Relational Completeness

Ketika pertama kali Codd mengusulkan relational kalkulus. Beliau mendefinisikan bahasa relational databasenya menjadi relationally complete. Relational Complete mempunyai kelebihan untuk mengekspresikan relational kalkulus. Kerapkali, definisi ini merupakan perpanjangan yang meliputi pengambilan informasi dari database dengan 1 kalimat. Dengan kata lain, jika informasi dapat di ambil dari tabel database, query yang digunakan untuk mengambil informasi dapat diformulasikan dengan menggunakan bahasa yang bebas. Codd, juga memperlihatkan bahwa relational algebra merupakan relationally complete. Kebanyakan dari bahasa yang dituju baik itu SQL dan QUEL adalah relationally complete.

Ada 12 aturan yang diajukan oleh Codd, yaitu :

Aturan	Keterangan
0	Relational DBMS harus dapat mengatur database secara keseluruhan dengan menggunakan kapasitas relational.
1	**The Information Rule** Semua informasi yang ada di relational database (meliputi tabel dan kolom) harus mewakili nilai dalam tabel.
2	**Guarantedd Access** Setiap nilai di dalam relational database harus dapat di akses dengan menggunakan kombinasi dari nama tabel, nilai dari kolom unik (primary key), dan kolom yang lain
3	**Systematic Null Value Support** DBMS harus mendukung secara sistematis untuk nilai kosong (null), nilai default, dan berdiri sendiri.
4	**Active, On Line Relational Catalog** Keterangan dari database dan semua isinya harus mewakili level logika sebagai tabel, dan dapat dilihat dengan menggunakan bahasa database.
5	**Comprehensive Data Sublanguage** Harus sekurangnya 1 bahasa yang mendukung definisi data, manipulasi, aturan integritas, hak akses dan transaksi.
6	**View Updating Rule** Semua penampilan data dapat di edit melalui sistem.
7	**Set – Level Insertion, Update dan Deletion** DBMS tidak hanya mendukung level penampilan informasi, tetapi juga mendukung penambahan, peng-editan dan penghapusan informasi.
8	**Physical Data Independence** Aplikasi program dan program ad – hoc secara logika tidak dipengaruhi metode akses fisik atau struktur penyimpanan dirubah.
9	**Logical Data Independence** Aplikasi program dan program ad –hoc secara logika tidak dipengaruhi jika terjadi penambahan struktur tabel.
10.	**Integrity Independence** Bahasa database harus bisa mendefinisikan aturan integritas informasi, harus di simpan pada metadata secara on line, dan tidak bisa dilewatkan.
11.	**Distibution Independence** Permintaan program aplikasi dan ad –hoc secara logika tidak dipengaruhi ketika data pertama kali didistribusikan, atau ketika tidak didistribusikan.

12. **Non – Subversion**
Harus tidak bisa dilewatkan aturan integritas yang didefinisikan melalui bahasa database dengan menggunakan level bahasa terendah.

2.6. STRUCTURED QUERY LANGUAGE (SQL)

Bahasa SQL adalah sebuah bahasa tingkat tinggi, beriorientasi pada group data, non prosedur untuk relational database. Disebut bahasa tingkat tinggi karena memakai bahasa Inggris. Disebut orientasi pada group data (tabel) karena berbeda dengan bahasa lain yang hanya mempunyai kapabilitas record. Bersifat non prosedur karena memberikan apa yang hendak kita inginkan dari informasi, bukan berdasarkan dengan cara apa kita harus mengambil informasi.

SQL dibuat pada tahun 1970 – an di IBM San Jose Research Laboratories yang merupakan bagian dari projek System R. Bahasa SQL menjadi sangat penting dalam penggunaannya karena ada 2 alasan

a. Bersifat portabel karena aplikasi SQL yang dibuat di minikomputer bisa langsung digunakan di PC,
b. Prosedur bahasa database yang dibuat dibuat di PC, bisa dipakai untuk mengambil data di minikomputer karena adanya persamaan bahasa.

Standarisasi dari bahasa SQL dimulai dari tahun 1986 dari American National Standards Institute (ANSI) – X3.135-86 dan pada tahun 1987 oleh International Standard Organization (ISO). Tahun 1989 , ISO dan ANSI mempublikasikan SQL Standard yaitu SQL-89 (SQL 1) diteruskan SQL-92 dan SQL-99.

Bahasa SQL didasarkan pada penggunaan untuk mendefinisikan dan memanipulasi objek – objek pada database namun supaya dapat dimanfaatkan pada aplikasi dibuat pendekatan yang kita kenal dengan nama **Host Language Interface** yang digunakan untuk bahasa mesin mainframe seperti COBOL, C, PL/1 sedangkan dalam minikomputer kita kenal dengan nama **Applications Programming Interface** (API). Ada beberapa pendekatan untuk interface (layer), yaitu

- **Embedded Approach**. Setiap perintah dalam program untuk mengakses database di tulis dalam satu kesatuan dengan program. Biasanya dengan menggunakan karakter khusus seperti $.Kemudian program tersebut di compile.
- **Library Approach**. Dalam hal ini database telah menyediakan fungsi – fungsi khusus untuk memanggil perintah – perintah SQL yang diinginkan programmer.
- **Approaches Compared**. Menggunakan kelebihan – kelebihan dari Embedded dan Library Approach.
  Kelebihan dari pendekatan Embedded adalah

⇒ Spesifikasi SQL yang dibuat oleh programer dalam source code adalah sama dengan SQL yang digunakan pada level interface ad – hoc. Ini berarti query yang digunakan akan berjalan dengan baik karena sama.
⇒ Kemampuan untuk dipakai dalam beberapa platform sejauh alat compile yang dipakai sama.

Kekurangan dari pendekatan Embedded adalah

⇒ Proses compile dan precompile yang dibutuhkan prosedur pada aplikasi. Hal ini tentunya tidak praktis bagi programer dan programer diharuskan untuk terbiasa dengan source code (precompiler). Karena original source code sering dirubah maka berbeda jauh dengan original source code.
⇒ Dibutuhkan tambahan gagasan (SQL dan bahasa programming) untuk mendukung akses database. Karena SQL adalah berdasarkan group data dan kebanyakan aplikasi memerlukan akses baris maka harus menggunakan cursors pada Embedded SQL.

Sedangkan kelebihan dan kekurangan dari pendekatan Library adalah

⇒ Tidak portabel, sehingga tidak multi platform. Namun jikalau aplikasi menggunakan DBMS dengan O/S dan platform yang sama maka dengan mudah dipindahkan
⇒ Tidak perlu dicompile ulang seperti yang digunakan pendekatan Embedded.
⇒ Penggunaan call function memudahkan dalam penulisan aplikasi.

2.7. OLTP (Online Transaction Processing) dan OLAP (Online Analytical Processing)

Di dalam database di kenal 2 konsep database yaitu

2.7.1. OLTP (Online Transaction Processing) adalah proses data yang digunakan untuk transaksi sehari – hari. Ciri – Ciri dari database OLTP adalah

- Akses data bersifat read-write, seperti insert,update dan delete,
- Orientasi data pada aplikasi, yaitu data yang diambil dari aktivitas bisnis.
- Karakter data tidak dipentingkan. Di pakai oleh keseluruhan sistem (Tidak terintegrasi)
- Aktivitas data konsisten.

2.7.2. OLAP (Online Analtycal Processing) adalah proses data yang digunakan analisa data seperti trend penjualan, umur, dll. Ciri – Ciri dari database OLAP adalah

- Akses data bersifat read-only,
- Orientasi data pada subjek bisnis,
- Data terintegrasi,
- Data bersifat History (2 – 5 tahun).
- Aktivitas data kadang – kadang tinggi (tidak rata).

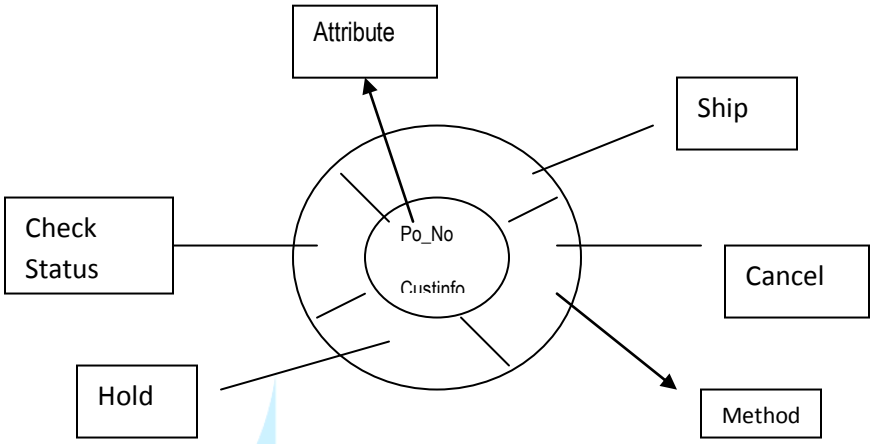
2.8. ORIENTASI OBJEK (OBJECT ORIENTED)

Sebuah objek kerapkali dipertimbangkan sebagai presentasi dari benda – benda nyata di dunia. Objek – Objek mengetahui apa yang harus dikerjakan, seperti objek karyawan dapat berbicara untuk menghitung potongan gaji. Objek Grafikal dapat berbicara untuk mengetahui bagaimana menggambar, dll. Contoh objek – objek yang lain dalam dunia nyata seperti pelanggan,perusahaan, sekolah, dll.

Beberapa definisi dari objek adalah

" **Sebuah Objek adalah sebuah paket piranti lunak yang mengandung sekumpulan prosedur – prosedur (metodologi) yang saling berhubungan dan data (variables).** ". --- *David Taylor's Object Oriented Technology: A Manager's Guide [Massachusetss: Addison-Wesley Publishing Company, 1981].*

"Sebuah Objek mempunyai bagian, karakter, dan identitas; Struktur dan karakter dari objek yang mirip didefinisikan pada objek umum kelas; dan dapat dipertukarkan." --- *Grady Booch's Object Oriented Analysis and Design with Applications*.



2.8.1. Ciri – Ciri sebuah Objek

- Mempunyai kemampuan untuk memberikan informasi tentang dirinya sendiri (instantion),
- Mempunyai metadata yang dapat digunakan aplikasi untuk berhubungan dengan objek tersebut.
- Mempunyai kemampuan untuk mewarisi kemampuan - kemampuan dari tipe objek (inheritance).

- Mempunyai kemampuan untuk menyembunyikan data dan proses yang berada di dalam objek tersebut (encapsulation),
- Mempunyai kemampuan untuk berinteraksi dengan objek yang lain (messages, interfaces).
- Mampu mengetahui perbedaan dengan objek yang lain tanpa analysis,
- Mampu memterjemahkan perintah yang sama dengan cara yang berbeda (messages, method overloading) – polymorphism.

2.8.2. Beberapa keuntungan dari teknologi Objek

- Mudah dalam pemeliharaan aplikasi,
- Cepat dalam membangun sebuah produk,
- Kualitas bagus biaya murah,
- Kode program dapat digunakan lagi.

2.8.3. Karakter dari Objek Relational Databases

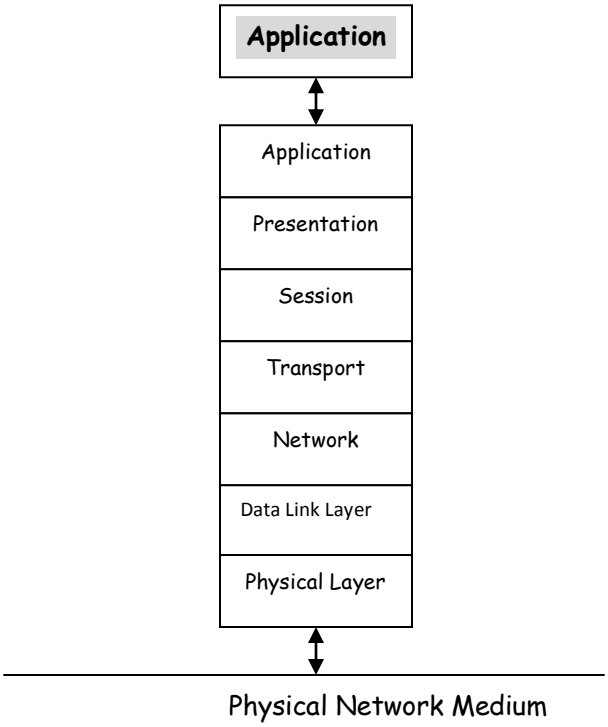
- Dapat membuat user-defined datatypes,
- Mendukung multimedia dan data objek yang besar,
- Sesuai dengan standar objek SQL,
- Mempunyai kemampuan database server dengan kualitas yang tinggi.

2.9. JARINGAN

Salah satu komponen penting di dalam teknologi database adalah jaringan. Tanpa jaringan maka data antara 1 komputer dengan komputer yang lain tidak dapat dihubungkan. Di dalam teknologi jaringan secara umum kita kenal dengan 2 istilah

- **LAN (Local Area Network)**, yaitu komunkasi jaringan yang memungkinkan komunikasi data antara peralatan elektronik di dalam suatu area geografi yang kecil. Peralatan elektronik di sini dapat berbentuk komputer, telepon,dll. Tipikal penggunaan biasanya antara gedung.
- **WAN (Wide Area Network),** yaitu komunikasi jaringan yang memungkinkan komunikasi data antara peralatan elektronik di dalam area geografi yang luas, seperti antara daerah atau antar negera.

Kerangkan umum yang biasa digunakan dalam jaringan adalah menggunakan model 7 layers - OSI (Open Standard International) yang dibuat oleh International Standards Organization (ISO) yang bermarkas di Paris. Tujuan dari OSI adalah untuk mendefinisikan berbagai fungsi dalam berbagai layer yang menjembati fasilitas komunikasi dari berbagai peralatan elektronik dari berbagai pemasok. Tujuan dari layer adalah untuk melindungi layer yang lebih tinggi dari layer yang lebih rendah. Sebagai contoh kabel optik yang bermain pada layer fisik. Secara teori tidak akan mempengaruhi layer yang di atasnya. Ke 7 layer tersebut adalah



Keterangan

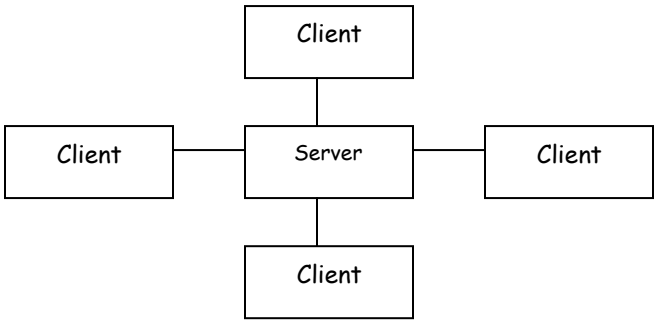
- **Physical Layer** adalah layer yang paling rendah yang berhubungan dengan hubungan elektris,signal yang memungkinkan komunikasi data terjadi. Layer yang biasa digunakan adalah RS-232C, yang mendefinisikan berbagai macam pin dan signal dari fisik konektor. Standar yang lain adalah RS-449 dan X.21 (Eropa).
- **Data Link Layer** adalah layer yang berkonsentrasi pada format pesan data, mengatur arus data dari dalam dan keluar node, dan menambahkan alamat yang benar dari sumber dan node yang menjadi tujuan. Protokol yang biasa digunakan meliputi High – Level Data Link Control (HDLC) dan Bisynchronous. Pada mesin mikrokomputer kita kenal dengan istilah Token Ring, Ethernet dan Arcnet. Biasanya di install pada Client (workstation).
- **Network Layer** digunakan untuk memberi solusi pada konflik alamat dan memutuskan bagaimana data paket dikirim melalui jaringan berdasarkan kondisi dari berbagai macam jaringan yang ada.
- **Transport Layer** bertanggungjawab untuk memastikan bahwa pesan alamat dimasukkan atau dibuka ke dalam atau keluar dari paket sebelum diteruskan ke layer yang lebih tinggi. Jikalau terjadi erro maka Transport Layer akan mencari jalur alternatif. Protokol yang kita kenal di sini adalah TCP / IP (Transmission Control Program / Internet Protocol) dan ISO Transport Class 4 (TP 4).
- **Session Layer** bertanggung jawab untuk membangun hubungan antara komputer dalam sebuah jaringan yang memungkinkan aplikasi berkomunikasi.Fungsi yang di handle dalam layer ini dalah pengenalan nama, logging, keamanan , administrasi , dll.
- **Presentation Layer** berkonsentarsi pada presentasi data seperti printer dan pada komputer.Fungsi yang terkait pada layer ini seperti format file, mail, management database, dll.

2.9.1. TOPOLOGI JARINGAN

Tujuan mempelajari topologi jaringan adalah untuk mengetahui bagaimana sebuah ditransmisikan baik itu dari server ataupun ke client. Beberapa tipikal topologi jaringan adalah sebagai berikut :

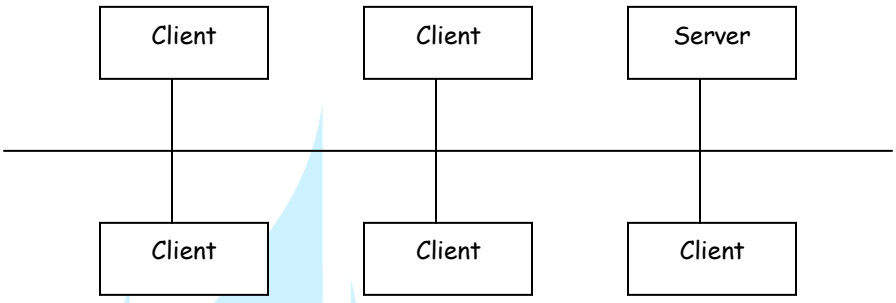
Star Networks

Star Networks adalah jaringan yang paling tua dan umum dipakai. Sebagai contoh penggunaan Star Networks adalah pada penggunaan telpon. Pada saat kita telpon dari / ke telpon yang lain. Maka nada akan dikirimkan ke pusat lalu baru dikirim ke telpon yang lain. Kekurangan dari jaringan ini adalah jika pusat rusak maka sistem tidak bisa berjalan.



Bus Networks

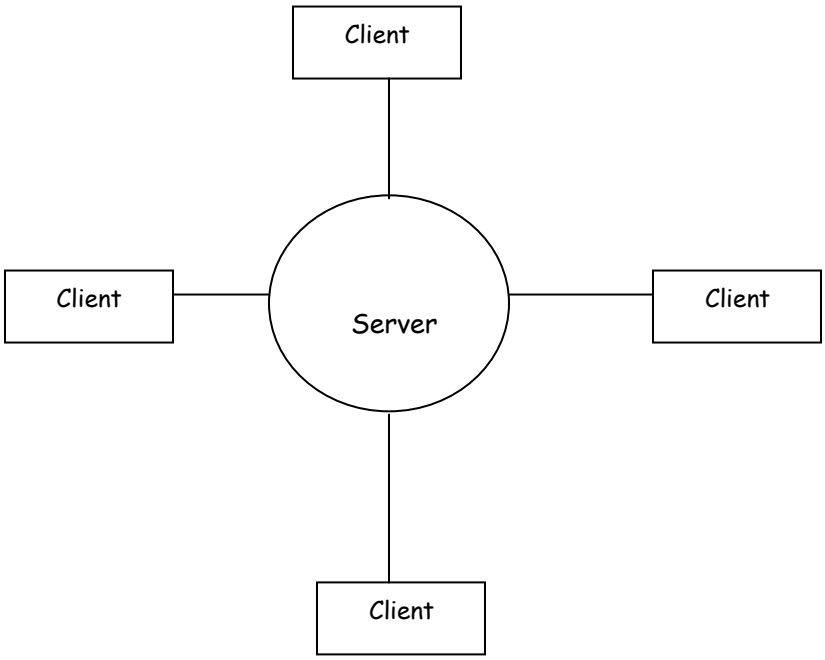
Bus Networks adalah berbagi medium untuk semua client. Setiap client akan memeriksa apakah medium dipakai oleh client yang lain, jika tidak maka medium akan di pakai oleh client tersebut. Keunggulan dari topologi star networks adalah jika terjadi kerusakan pada salah satu client, tidak akan merusak sistem.



Ring Networks

Ring Network menggunakan sistem star maupun bus. Sebagai ilustrasi, jika sebuah client mengirim data maka data tersebut akan dikirim ke jaringan ring. Selanjutnya dikirim ke client yang dituju

dengan cara membuat dupikasi data. Jika data telah selesai dikirim maka data akan kembali ke client yang mengirim.



2.9.2 EVOLUSI CLIENT / SERVER

Menurut Business Research Group ada 3 fase evolusi dari client/server

1. Proto Client / Server, PC pengguna menjalankan aplikasi melalui terminal emulasi. Pengguna tidak mempunyai kuasa untuk merubah bentuk tampilan. Semua data di ambil dari mesin mainframe atau mini. PC pengguna tidak pandai.
2. Generasi Client / Server pertama. Generasi yang pertama ini di rancang untuk mengijinkan pengguna dalam menjalankan dan memanipulasi data dari GUI pada PC. Kebanyakan aplikasi generasi pertama menempatkan prosedur logika pada client dan menggunakan server hanya sebagai tempat menyimpan data.
3. Generasi Client / Server ketiga. Ciri – ciri dari generasi kedua ini adalah mempunyai pengguna yang banyak, proses aplikasinya terdistribusi, datanya terdistribusi dan mempunyai sifat multi-tier arsitektur dimana aplikasi bersifat modular dan fleksibel untuk dirubah.

2.10. PENGGUNA DATABASE

Pengguna database sangat bervariasi. Namun jika digolongkan dalam tipikal pengguna dapat didefinisikan sebagai berikut :

a. Database Administrator

Tugas dari Administrator adalah

- Installasi dan Upgrade Oracle Server .
- Menyediakan sistem penyimpanan (storage) dan rencana ke depan untuk kebutuhan penyimpanan sistem database .
- Membuat penyimpanan struktur database (tablespaces) yang utama setelah pembuat aplikasi selesai mendesian sebuah aplikasi.
- Membuat objek utama (tables, views, indexes) setelah pembuat aplikasi selesai mendesain aplikasi.
- Modifikasi struktur database, jikalau diperlukan, dari informasi yang diberikan oleh pembuat aplikasi.
- Mendaftarkan pengguna dan memelihara sistem keamanan.
- Memastikan pemenuhan perjanjian database License.
- Kontrol dan monitor akses pengguna ke database.
- Monitor dan memaksimalkan performance dari database.
- Rencana Backup dan Recovery untuk informasi database.
- Memelihara pemasukan data ke ke tape.
- Backup dan Restore database.
- Menghubungi Database Teknikal Support jikalau ada masalah.

b. Security Officers

Dalam kasus tertentu, Petugas Keamanan Data bisa lebih dari satu. Tugas utama dari petugas keamanan data adalah mendaftarkan pengguna database, kontrol dan monitoring akses pengguna database, dan memelihara sistem keamanan data.

c. Application Developers

Pembuat aplikasi mendesain dan mengimplementasikan aplikasi database. Ada tugas dari pembuat aplikasi adalah

- Desain dan membuat aplikasi database.
- Desain struktur database untuk aplikasi.
- Estimasi kebutuhan penyimpanan data untuk aplikasi.
- Spesifikasi modifikasi struktur database untuk aplikasi.
- Menyampaikan informasi di atas kepada database administrator.
- Memperbaiki aplikasi selama masa pengembangan.
- Menyusun dan memastikan keamanan database untuk aplikasi selama masa pengembangan.

d. Application Administrators

Tugas yang dilakukan adalah melakukan tugas administrasi pada aplikasi tertentu.

e. Database Users

Pengguna database berinteraksi dengan database melalui aplikasi dan alat bantu yang lain (utility). Beberapa tipikal tugas yang dilakukan adalah

- Menambah, Meng-Edit, dan Menghapus data, yang diizinkan.
- Membuat Laporan

f. Network Adminstrators

Untuk beberapa perusahaan mungkin mempunyai lebih dari 1 Network Administrator. Network Administrator bertugas untuk menghubungkan database dengan pengguna.

2.11. KEMAMPUAN ORACLE DATABASE

Ada beberapa keunggulan dari Oracle Database ini yaitu

a. Scalability
kemampuan menangani banyak user yang melakukan akses secara simultan tanpa berkurangnya performance secara signifikan. Dalam dokumentasi Oracle disebutkan bahwa database Oracle sanggup melayani puluhan ribu pengguna secara simultan.
b. Reliability
kemampuan untuk melindungi data dari kerusakan jika terjadi kegagalan fungsi pada sistem seperti disk failure.
c. Servicebility
kemampuan untuk mendeteksi masalah, kecepatan dalam mengoreksi kesalahan, dan kemampuan melakukan konfigurasi ulang struktur data.
d. Stability
Kemampuan untuk tidak crash karena beban kerja yang tinggi. Hal ini berkaitan dengan scability.
e. Availibility
Kemampuan dalam penanganan crash atau failure agar service dapat tetap berjalan. Misalnya saja dengan tersedianya fasilitas pendistribusian database pada beberapa data server dan juga pemulihan database (recovery).
f. Multiplatform
Dapat digunakan pada banyak sistem operasi seperti Windows, Unix, Linux dan Solaris
g. Mendukung data yang sangat besar. Menurut dokumentasi Oracle, dapat menampung sampai 512 petabytes (1 petabytes = 1.000.000 gigabytes).
h. Sistem keamanan data yang cukup handal
i. Mendukung database beriorientasi objek.
j. Dapat menampung hampir semua tipe data seperti teks, image, sound, video dan time series.

Daftar Pustaka

Alan Denis, Barbara Haley Wixon, David Tagerden, *System Analys and Design : an Object – Oriented Approach with UML 2.0*, John Willey and Sons, 2005

Henry C. Lucas Jr., *The Analysis, Design and Implementation of Information Systems 4th*, McGraw Hill, 1992

Satzinger, Jackson, Burd, *Object-Oriented Analysis and Design with the Unified Process*, Course Technology, 2005

Simon Bennet, Steve McRobb and Ray Farmer, *Object-Oriented System Analysis and Design Using UML*, McGraw Hill, 2006

Wendy and Michael Boggs, *UML with Rational Rose 2002*, Sibex Inc., 2002

UNIVERSITAS MERCU BUANA

MODUL PERKULIAHAN

PEMROGRAMAN WEB

Modul Standar untuk digunakan dalam Perkuliahan di Universitas Mercu Buana

Fakultas	Program Studi	Tatap Muka	Kode MK	Disusun Oleh
Ilmu Komputer	Sistem Informasi	03	87007	Team Dosen

Abstract

SDLC, RAD, Design Tersturktur, Class dan Object

Kompetensi

Memahami siklus pengembangan sistem, memahami metodhologi pengembangan sistem, mengenal UML

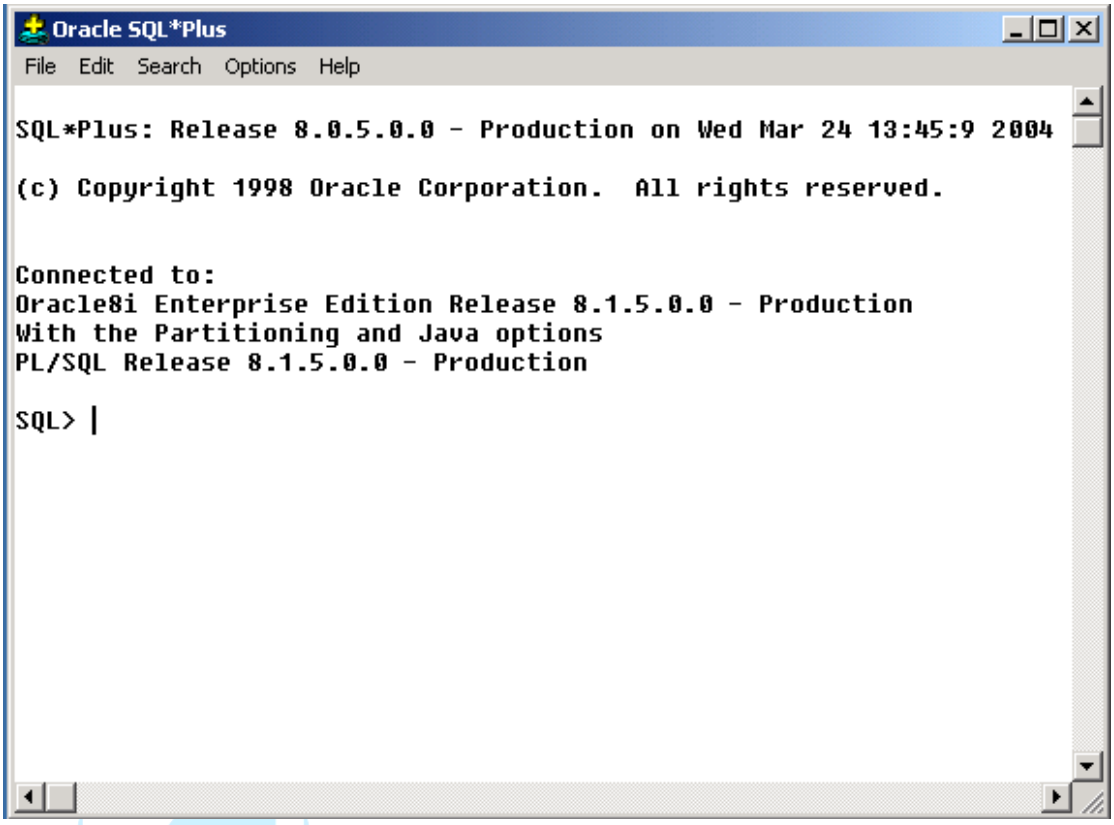
STRUCTURE QUERY LANGUAGE (SQL)

3.1. PENGENALAN ALAT BANTU UNTUK MENJALANKAN PERINTAH – PERINTAH SQL (Structured Query Language).

Ada berbagai cara untuk menjalankan perintah – perintah SQL pada produk Oracle database, yaitu :

3.1.1. SQL Plus (Windows).

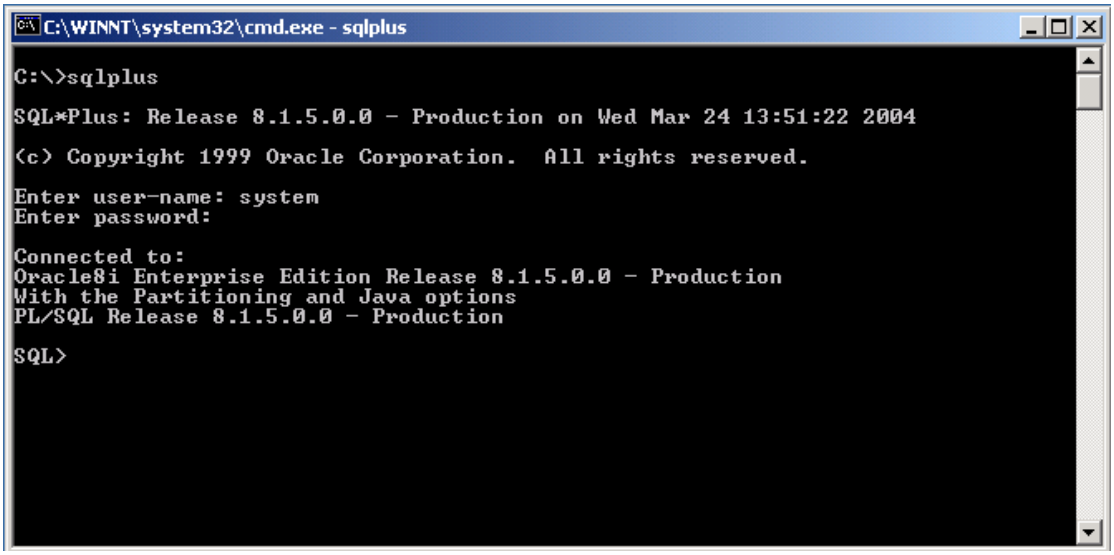
SQL Plus adalah alat system di dalam database Oracle yang digunakan untuk menjalankan perintah – perintah SQL. Produk ini merupakan bagian dari produk database ataupun produk development seperti Developer / 2000, Designer / 2000. SQL Plus hanya dapat dijalankan pada system operasi Microsoft Window XP/98/2000. Cara Menjalankan SQPLus adalah Start -> Program -> Oracle For Windows NT -> SQL Plus 8.0



3.1.2. SQLPLUS (WINDOWS – DOS)

SQLPlus juga dapat dijalankan melalui DOS dengan cara :

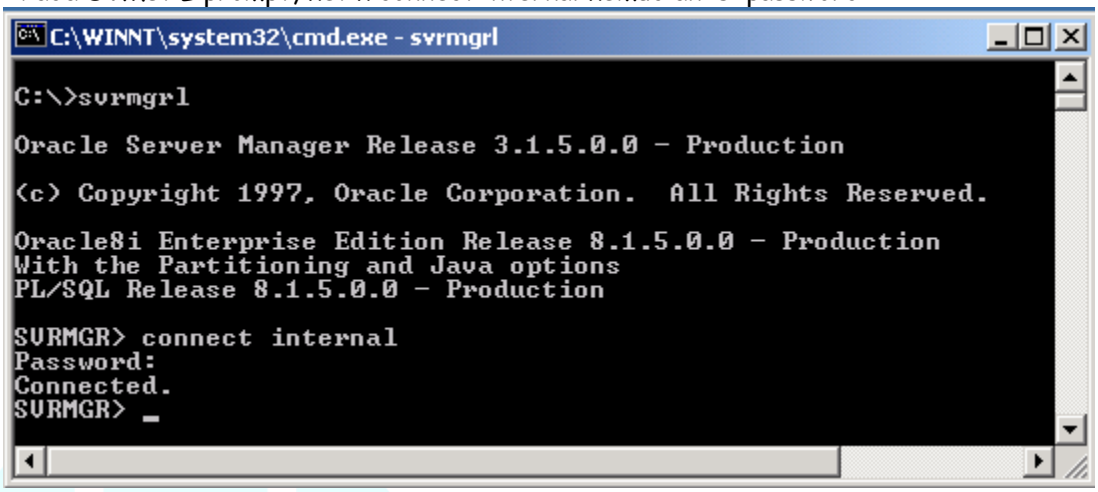
a. Start -> Run. Kemudian ketik cmd.
b. Ketik sqlplus kemudian tekan enter.
c. Masukkan User ID dan Password
d. Setelah itu maka akan tampak SQL prompt.



3.1.3. SVRMGRL (WINDOWS – DOS)

Penggunaan SVRMGRL biasanya digunakan untuk mengatur atau mengolah system database. Cara menggunakannya adalah sebagai berikut :

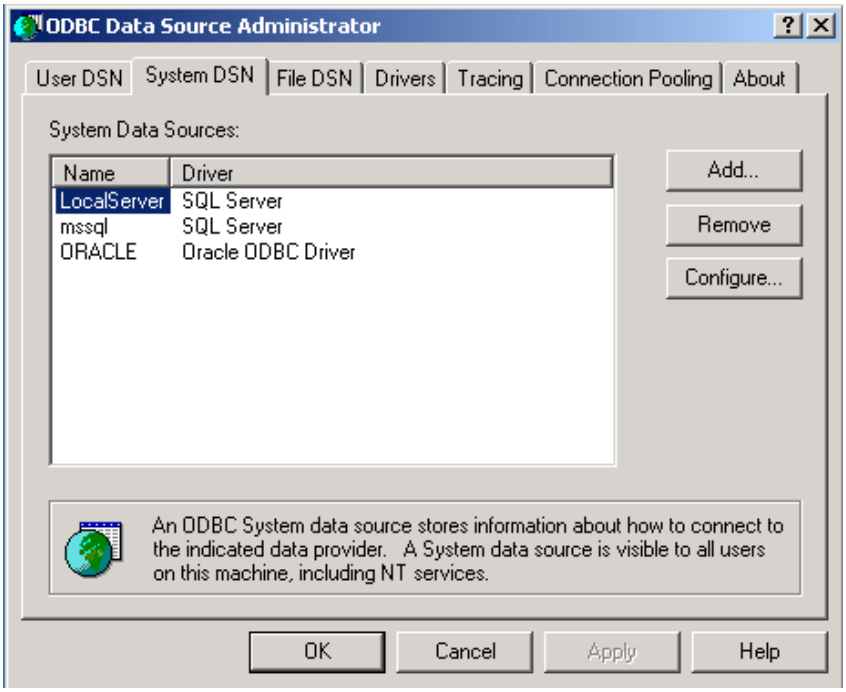
a. Start -> Run. Kemudian ketik cmd.
b. Ketik svrmgrl kemudian tekan enter.
c. Pada SVMGRL prompt, ketik connect internal kemudian isi password



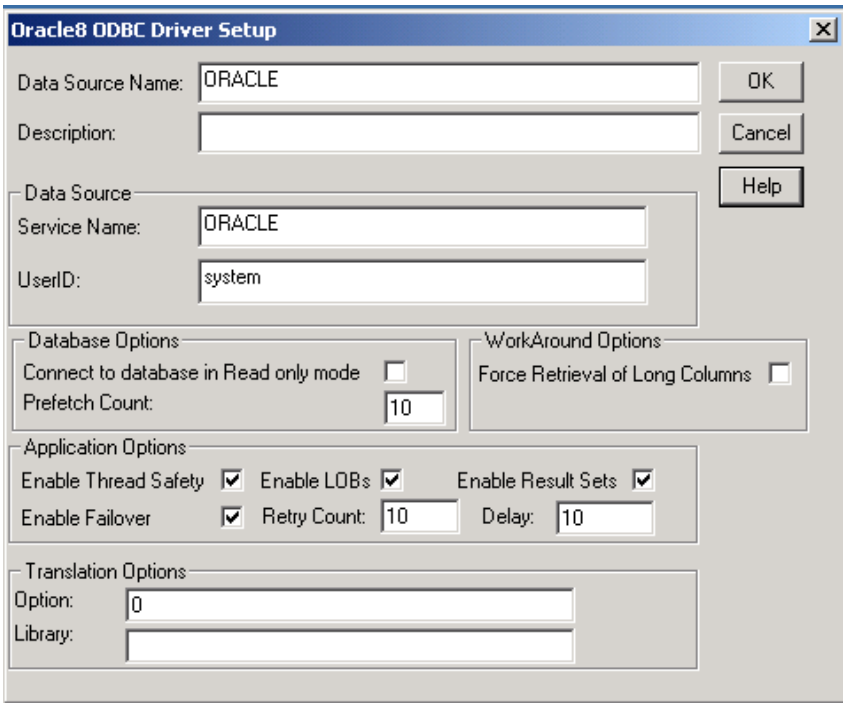
3.2. ODBC (Open Database Connectivity)

Cara lain adalah dengan menggunakan ODBC (Open Database Connectivity). ODBC merupakan standarisasi dari berbagai database agar dapat dijalankan oleh aplikasi lainnya. Cara Menggunakannya

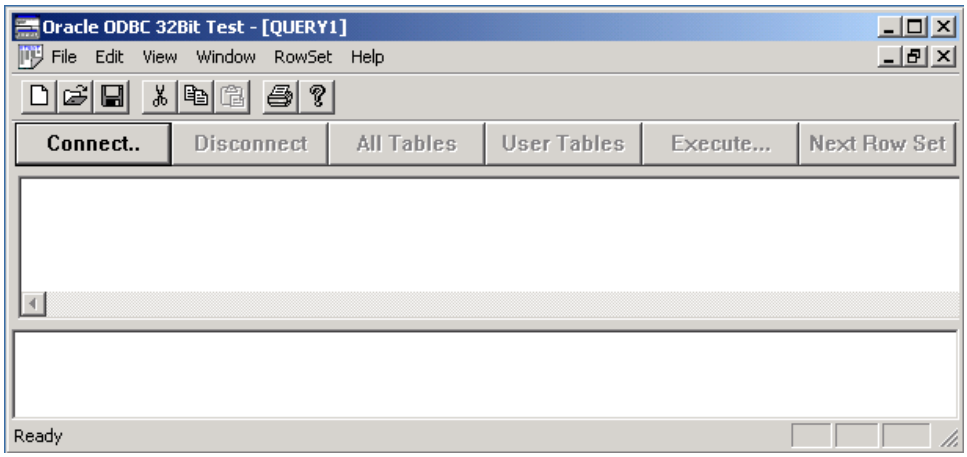
a. Start -> Settings -> Control Panel -> Administrative Tools -> Data Sources (ODBC).



b. Tekan tombol Add -> Pilih Driver Oracle ODBC Driver



c. Ada berbagai cara untuk mengakses ODBC. Tergantung dari penggunaan aplikasi. Pada Oracle Developer 6i dapat menggunakan Oracle ODBC Test. Adapun layar bentuk layar dari Oracle ODBC Test adalah sbb :



3.3. Alat bantu yang lain.

Pada saat ini tersedia di pasar berbagai produk alat bantu yang dapat digunakan untuk mengakses oracle database seperti TOAD, Quest Software,dll.

3.4. Entity Relationship Diagram (ERD)

Dalam pembuatan sistem informasi, salah satu komponen terpenting adalah dokumentasi. Semakin kompleks sebuah sistem informasi maka dokumentasi sangat diperlukan untuk membantu pembuat aplikasi melihat struktur proses dari sebuah aplikasi. Ada beberapa keuntungan yang lain dari adanya sistem dokumentasi, yaitu tidak perlunya seorang pembuat aplikasi untuk berpikir dari awal dalam membuat sistem informasi, dapat digunakan untuk pengembangan aplikasi ataupun mempercepat pembuat aplikasi untuk menganalisa sistem informasi jika terjadi kesalahan.

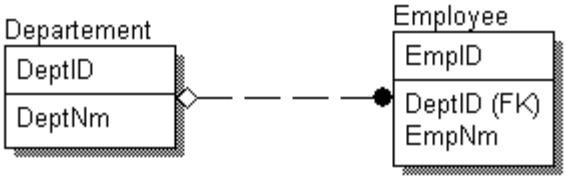
Ada 3 unsur di dalam pembuatan ataupun pengembangan sistem informasi yang kemudian kita kenal dengan nama segitiga sukses (the triangle of success), yaitu metode permodelan (notation), proses (process) dan tool (alat bantu) yang digunakan.

Tujuan dalam permodelan data adalah untuk mengambarkan struktur dari informasi dan menangkap peraturan dari bisnis yang dispesifikasikan ke dalam kebutuhan pengguna. Sebuah model data mewakili keseimbangan antara kebutuhan implementasi dari database dan kebutuhan umum dari area bisnis yang dituju. Di dalam pasar piranti lunak, ada beberapa alat permodelan data yang dapat digunakan, seperti ER-Win, Relational Rose, Designer / 2000, dll.

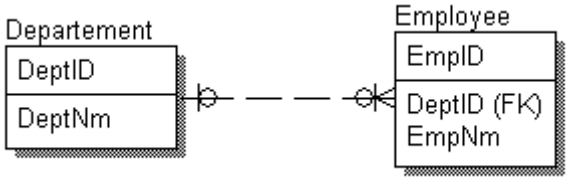
Aada beberapa metodologi pendekatan yang dapat digunakan untuk mengambarkan hubungan antar entity (data), yaitu

- IDEF1X. Metodologi ini dibuat oleh U.S Air Force. Sekarang banyak digunakan oleh pemerintah, penerbangan dan industri keuangan dan beberapa industri lainnya,
- IE (Information Engineering). Di buat oleh James Martin, Clive Finkelstein. Metodologi ini banyak digunakan dalam berbagai macam industri.

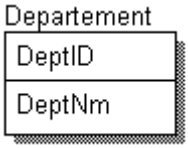
Contoh gambar dari metodologi IDEF1X adalah



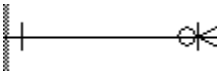
Contoh gambar dari metodologi IE (Information Engineering) adalah



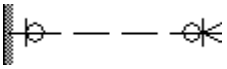
Dalam pelajaran kuliah selanjutnya, maka metode IE (Information Engineering) akan dijadikan acuan sebagai bentuk penggambaran hubungan antara 1 entity dengan entity lainnya.
Beberapa petunjuk gambar dalam metodologi IE adalah sebagai berikut :



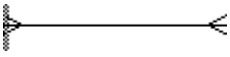
-> Menggambarkan entitas (model data)



-> Menggambarkan hubungan entity dari 1 ke 0 / 1 / banyak



-> Menggambarkan hubungan entity dari 1 / 0 ke 0 / 1 / banyak.



-> Menggambarkan hubungan entity many to many

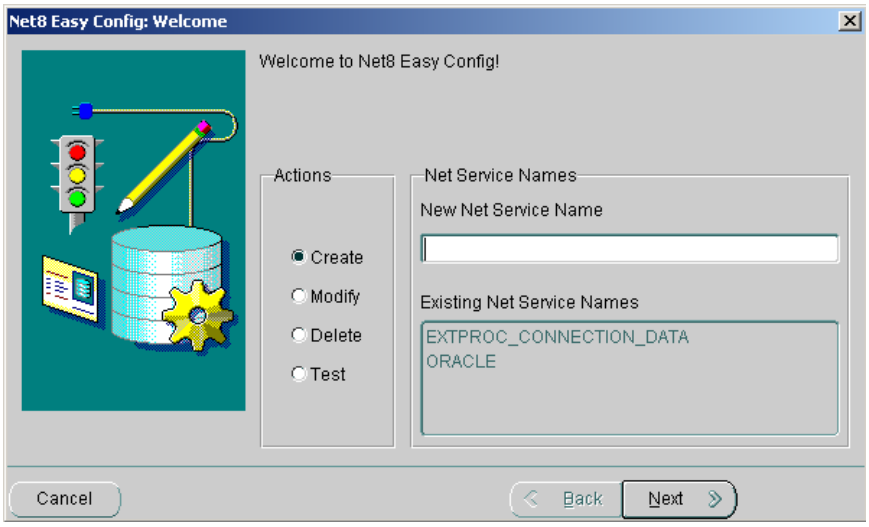
Beberapa keuntungan dari penggunaan data model adalah

- Implementasi dari sebuah data model bersifat berdiri sendiri.
- Spesifikasi data model mewakili data yang diinginkan.
- Istilah data model menggunakan bahasa bisnis.

3.4.1. Net8 Easy Config (Windows)

Net8 Easy Config adalah aplikasi oracle yang digunakan untuk menghubungkan aplikasi client dengan oracle database server. Cara menggunakannnya adalah

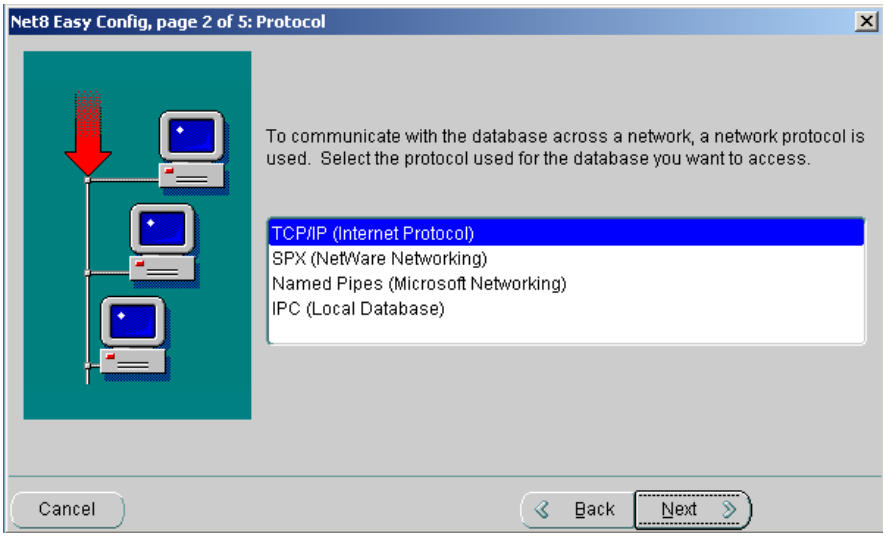
1. Jalankan Start -> Program -> Oracle – OraHome81 -> Network Administration -> Net8 Easy Config.
2. Setelah itu layar yang akan ditampilkan adalah sebagai berikut



Keterangan

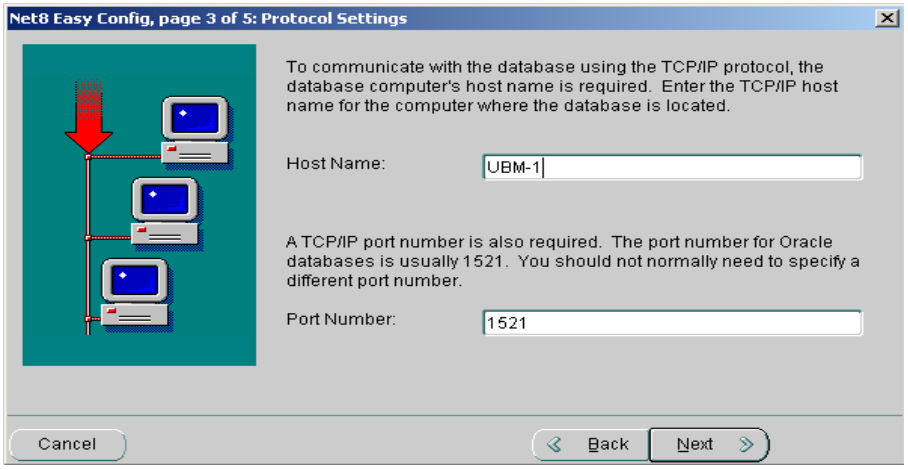
No.	Menu	Keterangan
01.	Action	Create - > untuk membuat services baru. Modify -> untuk mengedit services yang sudah ada. Delete -> untuk menghapus services Test -> digunakan untuk mentest services yang sudah ada.
02.	New Net Service Name	Isi nama service yang diinginkan
03.	Existing Net Service Names	Berisikan services yang sudah ada.

2. Klik Next,

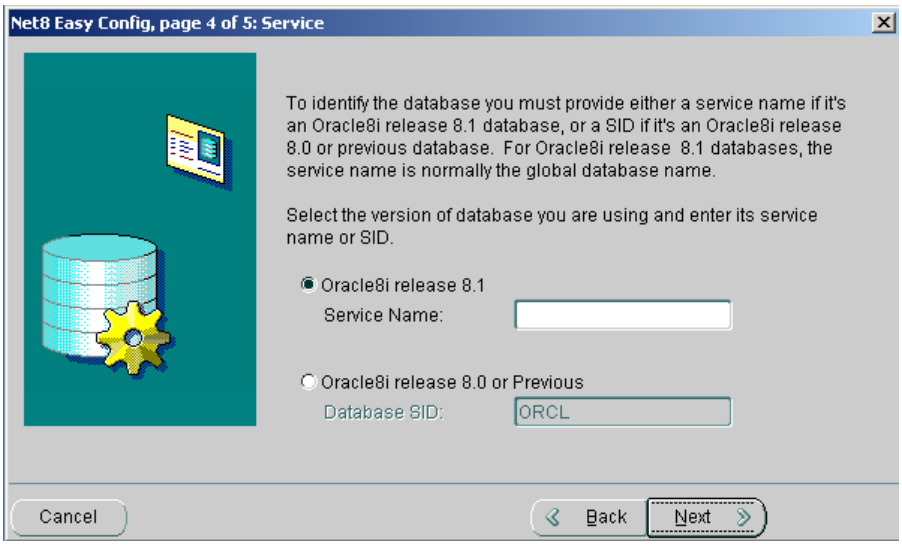


Pilih protokol yang akan digunakan.

3. Klik Next, Isi Host Name (Nama Server untuk oracle database).



4. Klik Next,



Keterangan

No.	Menu	Keterangan
01.	Oracle8i release 8.1. Service Name	Digunakan untuk memberikan nama service yang diinginkan.
02.	Oracle8i release 8.0 or Previous. Database SID	Digunakan untuk mengisi nama SID. Digunakan untuk database yang lama.

5. Klik Next, untuk melakukan test. Setelah Test selesai tekan finish untuk mengakhiri wizard net8 easy config.

3.5. SQL Plus

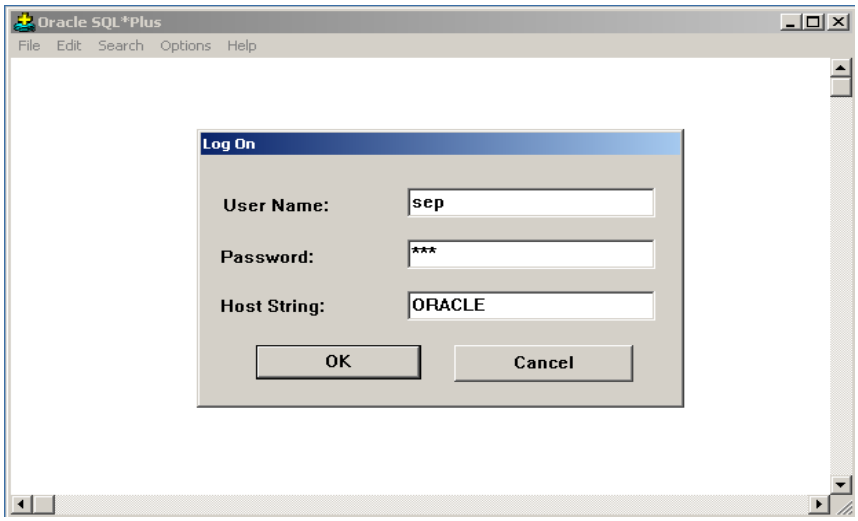
SQL Plus adalah alat bantu yang dibuat perusahaan Oracle untuk menjalankan perintah – perintah SQL. SQL Plus biasanya sudah merupakan bagian dari produk database server maupun produk development tools yang dikeluarkan oleh perusahaan Oracle.

Beberapa keuntungan SQL Plus, yaitu

- Dapat menerima perintah – perintah khusus,
- Dapat membaca file yang berisi perintah SQL dan kemudian mengeksekusinya,
- Menyediakan editor baris untuk memodifikasi perintah SQL,
- Kontrol terhadap environment settings,
- Dapat menformat hasil query untuk membuat laporan sederhana,
- Dapat mengakses database baik lokal maupun remote,
- Dapat membuat file script yang menyimpan perintah SQL untuk penggunaan di kemudian hari.

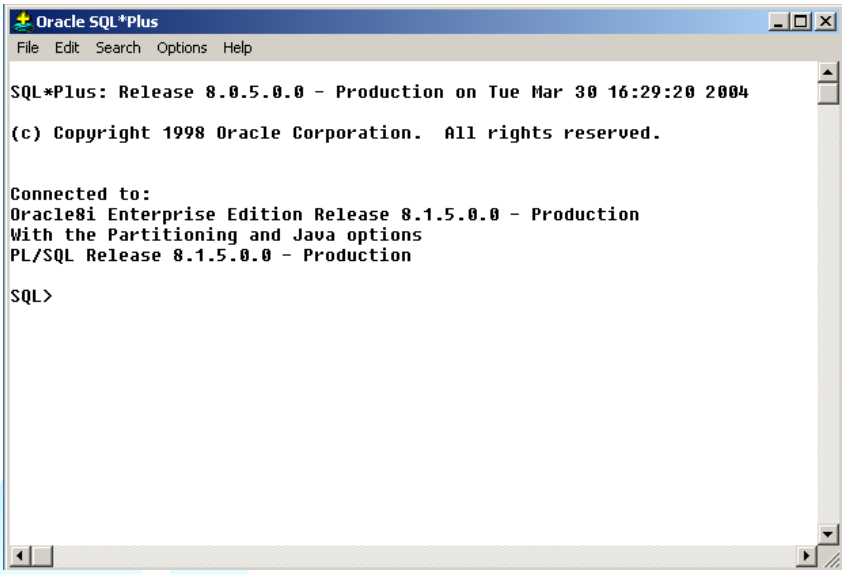
Cara penggunaannya adalah sebagai berikut :

1. Start -> Program -> Oracle – OraHome81 -> Application Development -> SQLPlus
2. Maka layar yang ditampilkan adalah



No.	Menu	Keterangan
01.	User Name	Isi nama user yang diinginkan
02.	Password	Isi password yang diinginkan
03.	Host String	Adalah nama services yang dibuat dengan Net8 Easy Config

3. Klik Ok,



SQLPlus siap digunakan untuk menjalankan perintah SQL.

3.6. Tipe Data Oracle Database

Dibawah ini adalah beberapa tipe data yang digunakan pada oracle database versi 8, dengan penjelasan sbb :

No.	Tipe Data	Keterangan
01.	VARCHAR2	Panjang maksimum dari penggunaan tipe data ini adalah 32, 767 characters.
02.	NVARCHAR2	Multibyte karakter
03.	NUMERIC	Digunakan untuk numerik data.
04.	NUMBER	Digunakan untuk numerik data. Panjang maksimum 38 dengan pecahan antara 84 s / d 125 digits.
05.	LONG	Panjang maksimum 2GB
06.	DATE	Digunakan untuk mengisi tipe data tanggal
07.	RAW	Digunakan untuk binary data. Panjang maksimum 2000 .
08.	LONG RAW	Digunakan untuk binary data. Panjang maksimum 2 GB.
09.	CHAR	Tipe data karakter. Panjang maksimum 254 karakter.
10.	NCHAR	Multibye karakter.
11.	MLSLABEL	Digunakan untuk Trusted Oracle yang mewakili sistem operasi label.
12.	CLOB	Panjang maksimum 4GB karakter.
13.	NCLOB	Lebih dari 4 GB karakter.
14.	BLOB	Binary Data. Panjang Maksimum 4 GB.
15.	BFILE	Digunakan sebagai pointer untuk database yang besar.

Daftar Pustaka

Alan Denis, Barbara Haley Wixon, David Tagerden, *System Analys and Design : an Object – Oriented Approach with UML 2.0*, John Willey and Sons, 2005

Henry C. Lucas Jr., *The Analysis, Design and Implementation of Information Systems 4th*, McGraw Hill, 1992

Satzinger, Jackson, Burd, *Object-Oriented Analysis and Design with the Unified Process*, Course Technology, 2005

Simon Bennet, Steve McRobb and Ray Farmer, *Object-Oriented System Analysis and Design Using UML*, McGraw Hill, 2006

Wendy and Michael Boggs, *UML with Rational Rose 2002*, Sibex Inc., 2002

UNIVERSITAS MERCU BUANA

MODUL PERKULIAHAN

PEMROGRAMAN WEB

Modul Standar untuk digunakan dalam Perkuliahan di Universitas Mercu Buana

Fakultas	Program Studi	Tatap Muka	Kode MK	Disusun Oleh
Ilmu Komputer	Sistem Informasi	04	87007	Team Dosen

Abstract

SDLC, RAD, Design Tersturktur, Class dan Object

Kompetensi

Memahami siklus pengembangan sistem, memahami metodhologi pengembangan sistem, mengenal UML

STRUCTURE QUERY LANGUAGE (SQL)

3.7. Data Definition Language (DDL)

Data Definition Language (DDL) adalah bahasa SQL yang digunakan untuk mendefinisikan, menghapus, memberi hak akses ataupun menganalisa objek – objek database.

a. MEMBUAT TABEL

Definisi tabel adalah sebagai berikut :

```
CREATE TABLE nama tabel
    ( nama kolom   tipe data DEFAULT ekspresi ,
      nama kolom   tipe data DEFAULT ekspresi,
      ……..                                  )
    PCTFREE number   PCTUSED number
    STORAGE ( INITIAL   number
              NEXT      number
              PCTINCREASE number
              MINEXTENTS number
              MAXEXTENTS number )
    TABLESPACES  nama tablespace;
```

Keterangan

No.	Menu	Keterangan
01.	Default	Untuk mengisi data default
02.	PCTFREE	Jumlah tempat yang diberikan untuk setiap blok pada baris untuk pertumbuhan panjang. Berikan nilai PCTFREE rendah jika penggunakan update rendah.
03.	PCTUSED	Menentukan batas bawah dari tempat yang digunakan dalam blok sebelum digunakan untuk memasukkan baris (insert row). Berikan nilai PCTUSED jika aktivitas penambahan dan penghapusan tinggi.
04.	STORAGE	Mendefinisikan untuk menentukan bagaimana penambahan penyimpanan yang dialokasikan untuk tabel.
05.	INITIAL	Mendefinisikan size untuk extent pertama dalam bytes.
06.	NEXT	Mendefinisikan size untuk next extent dalam bytes.
07.	MINEXTENTS	Permulaan extent di buat
08.	MAXEXTENTS	Total extent yang dialokasikan
09.	PCTINCREASE	Menentukan dasar pertumbuhan untuk extent berikutnya.
10.	TABLESPACES	Tempat di mana table akan dibuat

Contoh

```
CREATE TABLE employee (
empid                   varchar2(10) not null,
empnm                   varchar2(30) default 'kosong',
salary                  number(5,2))
PCTFREE 5               PCTUSED 65
STORAGE  (  INITIAL  5M
            NEXT     5M
            PCTINCREASE 0
            MINEXTENTS 2
            MAXEXTENTS 50)
TABLESPACE users;
```

b. MERUBAH STRUKTUR TABEL

Definisi ALTER TABLE adalah sebagai berikut

```
ALTER TABLE nama table
ADD nama kolom  tipe data DEFAULT ekspresi
MODIFY nama kolom tipe data DEFAULT ekspresi
DROP COLUMN nama kolom tipe data DEFAULT ekspresi
PCTFREE number
PCTUSED number
STORAGE ( INITIAL   number
          NEXT      number
          PCTINCREASE number
          MINEXTENTS number
          MAXEXTENTS number )
```

Keterangan

No.	Menu	Keterangan
01.	ADD	Untuk menambahkan kolom pada sebuah tabel
02.	DEFAULT	Untuk memberikan nilai default pada sebuah kolom tabel
03.	MODIFY	Untuk merubah nama kolom pada sebuah tabel
04.	DROP COLUMN	Untuk menghapus nama kolom pada sebuah tabel
02.	PCTFREE	Jumlah tempat yang diberikan untuk setiap blok pada baris untuk pertumbuhan panjang. Berikan nilai PCTFREE rendah jika penggunakan update rendah.
03.	PCTUSED	Menentukan batas bawah dari tempat yang digunakan dalam blok sebelum digunakan untuk memasukkan baris (

		insert row). Berikan nilai PCTUSED jika aktivitas penambahan dan penghapusan tinggi.
04.	STORAGE	Mendefinisikan untuk menentukan bagaimana penambahan penyimpanan yang dialokasikan untuk tabel.
05.	INITIAL	Mendefinisikan size untuk extent pertama dalam bytes.
06.	NEXT	Mendefinisikan size untuk next extent dalam bytes.
07.	MINEXTENTS	Permulaan extent di buat
08.	MAXEXTENTS	Total extent yang dialokasikan
09.	PCTINCREASE	Menentukan dasar pertumbuhan untuk extent berikutnya.

Contoh :

ALTER TABLE emp ADD address varchar2(60);

c. MENGHAPUS TABEL

Digunakan untuk menghapus tabel dari database. Definisinya adalah sebagai berikut :

DROP TABLE nama table CASCADE CONSTRAINTS.

Keterangan

No.	Menu	Keterangan
01.	CASCADE CONSTRAINTS	
	Menghapus tabel dan constraint yang berhubungan dengan tabel tersebut.	

Contoh :

DROP TABLE emp CASCADE CONSTRAINTS;

d. MEMBUAT TABEL DENGAN SUBQUERY

Definisinya adalah :

CREATE TABLE nama tabel AS subquery

Contoh :

CREATE TABEL emp_fin AS select empid,empnm from emp;

e. MENGOSONGKAN ISI TABEL

Definisi : TRUNCATE TABLE nama tabel

Contoh :

TRUNCATE TABLE emp;

f. MENGGANTI NAMA TABEL

Definisi : RENAME nama tabel yang lama TO nama tabel yang baru;

Contoh : **RENAME** emp_fin **TO** emp_hrd;

g. PEMBERIAN KOMENTAR PADA TABEL ATAU KOLOM

Definisi :

COMMENT ON TABLE nama tabel **IS** 'text';

COMMENT ON COLUMN nama table.nama kolom **IS** 'text';

Contoh :

COMMENT ON TABLE emp IS ' tabel ini digunakan untuk karyawan' ;

COMMENT ON COLUMN emp.empid IS ' Nomor Induk Karyawan';

h. CONSTRAINT

Definisi pembuatan CONSTRAINT dapat dilakukan pada saat pembuatan tabel atau pada saat merubah struktur tabel.

Adapun definisi pada saat pembuatan tabel adalah

```
CREATE TABLE nama tabel
( nama kolom    tipe data  DEFAULT ekspresi,
  nama kolom    tipe data  DEFAULT  ekspresi,
  CONSTRAINT  nama constraint tipe constraint
);
```

Sedangkan definisi pada saat merubah struktur tabel adalah

ALTER TABLE nama tabel ADD nama constraint ekspresi constraint

Di dalam CONSTRAINT ada tipe – tipe constraint yang dapat digunakan

- **NOT NULL**

  Tipe constraint ini hanya dapat digunakan pada kolom.

  Definisi : [**CONSTRAINT** nama constraint] **NOT NULL**;

  Contoh :

  ALTER TABLE emp **MODIFY** address varchar2(100) **NOT NULL**;

- **UNIQUE**

  Definisi : [**CONSTRAINT** nama constraint] **UNIQUE** (kolom 1, kolom 2, …)]

  Contoh :

  ALTER TABLE emp **ADD CONSTRAINT** c_1 **UNIQUE** (empnm,address);

- **PRIMARY KEY**

  Definisi : [CONSTRAINT nama constraint PRIMARY KEY (kolom 1, kolom 2,..)]

  Contoh :

  ALTER TABLE emp **ADD CONSTRAINT** c_2 **PRIMARY KEY** (empid);

- **FOREIGN KEY**

  Definisi : [**CONSTRAINT** nama constraint **FOREIGN KEY** nama kolom **REFERENCES** nama tabel (nama kolom 1 **) ON DELETE CASCADE | SET NULL**]

  Contoh :

  ALTER TABLE emp **ADD CONSTRAINT** c_3 **FOREIGN KEY** dept_id **REFERENCES** dept (dept_id);

- **CHECK**

  Definisi [CONSTRAINT nama constraint CHECK (kondisi)

Kondisi di sini dapat berupa kondisi Perbandingan, kondisi BETWEEN, kondisi NULL, kondisi LIKE, kondisi EXISTS

Contoh :

ALTER TABLE emp **ADD CONSTRAINT** c_4 **CHECK** (salary > 0);

i. MENGAKTIFKAN / MENONAKTIFKAN CONSTRAINT

Definisi : [ALTER TABLE nama tabel MODIFY CONSTRAINT nama constraint < ENABLE | DISABLE >

Contoh :

ALTER TABLE emp **MODIFY CONSTRAINT** c_1 **DISABLE**;

j. MEMBUAT INDEX

Definisi :

CREATE [UNIQUE | BITMAP] INDEX nama index ON nama tabel (nama kolom 1, nama kolom 2…) ASC | DESC;

Keterangan :

No.	Menu	Keterangan
01.	UNIQUE	Data tabel harus unik ,artinya tidak boleh sama.
02.	BITMAP	Digunakan jikalau kondisi WHERE kompleks, Melibatkan jumlah data yang sangat besar.
03.	ASC	Data akan diurutkan dari atas ke bawah
04.	DESC	Data akan diurutkan dari bawah ke atas.

k. ANALISA INDEX

Definisi : ANALYZE INDEX nama index VALIDATE STRUCTURE;

Digunakan untuk statistik sebuah index.

Contoh :

SQL > ANALYZE INDEX idx_emp VALIDATE STRUCTURE;
SQL > SELECT btree_space, used_space, pct_used FROM index_stats WHERE name = 'idx_emp';

l. DROP INDEX

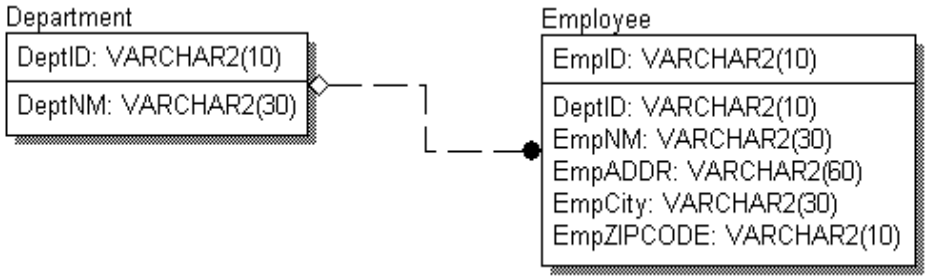
Definisi : DROP INDEX nama index

Digunakan untuk menghapus sebuah index.

Contoh : SQL > DROP INDEX idx_emp;

M. PEMETAAN KONSEPTUAL KE ITERNAL

- **Konseptual**



- **Internal**

```
CREATE TABLE Department
( DeptID        varchar2(10) not null,
  DeptNM        varchar2(30),
  Constraint c_1  primary key ( DeptID ));

 CREATE TABLE  Employee
( EmpID         varchar2(10),
  DeptID        varchar2(10),
  EmpNM         varchar2(30),
  EmpADDR       varchar2(60),
  EmpCity       varchar2(30),
  EmpZIPCODE  varchar2(10),
  Constraint  c_2 primary key( EmpID),
  Constraint  c_3 foreign key( DeptID ) REFERENCES Department(DeptID));
```

N. AKSES USER

Perintah – perintah SQL di bawah ini adalah digunakan untuk membuat, memberi hak akses ataupun menghapus pengguna database.

a. Membuat User

Definisi : CREATE USER nama user IDENTIFIED BY password;

b. Mengubah Password User

Definisi : ALTER USER nama user IDENTIFIED BY new_password;

c. Mengunci dan Membuka User

Definisi : ALTER USER nama user ACCOUNT < LOCK | UNLOCK > atau dapat juga dibuat dengan perintah ALTER USER nama user PASSWORD EXPIRE;

d. Menghapus User

Definisi : DROP USER nama user [CASCASDE]

e. Perintah GRANT

Definisi : GRANT { system_privilige } | { role_name } | < all_privilleges > TO < { user name } | { role_name } | PUBLIC > [WITH ADMIN OPTION]

Untuk melihat semua hak akses yang dapat diberikan user dapat dilhat pada tabel SYSTEM_PRIVILEGE_MAP dengan cara :

SQL > SELECT name FROM system_privilege_map;

Contoh dari pemberian grant kepada user adalah sebagai berikut :

SQL > CREATE USER ubm IDENTIFIED BY ubm;

SQL > GRANT CREATE USER, CREATE VIEW to ubm;

Untuk melihat hak akses apa saja yang diberikan dapat diliha dengan menggunakan perintah :

SQL > SELECT * FROM USER_SYS_PRIVS;

Hak Akses untuk objek database terdiri dari

No.	Hak Akses	Tabel	View	Sequence	Procedure, Function, Package
01.	ALTER	Yes	-	Yes	-
02.	DELETE	Yes	Yes	-	-
03.	EXECUTE	-	-	-	Yes
04.	INDEX	Yes	-	-	-
05.	INSERT	Yes	Yes	-	-
06.	REFERENCES	Yes	-	-	-
07.	SELECT	Yes	Yes	Yes	-
08.	UPDATE	Yes	Yes	-	-

Contoh :

SQL > GRANT INSERT,DELETE, UPDATE ON EMP TO ubm;

Untuk melihat hasil grant dapat dilihat pada tabel USER_TAB_PRIVS_RECD dengan perintah sbb :

SQL > SELECT * FROM USER_TAB_PRIVS_RECD;

f. Perintah ROLE

Adalah kumpulan koleksi dari berbagai hak akses. Beberapa role telah dibuat oleh oracle database seperti

No.	Role	Kegunaan
01.	CONNECT,RESOURCE, DBA	-
02.	DELETE_CATALOG_ROLE	Untuk pengaksesan data dictionary views dan packages
03.	EXECUTE_CATALOG_ROLE	
04.	SELECT_CATALOG_ROLE	
05.	EXP_FULL_DATABASE	Untuk penggunaan utilitas import dan export
06.	IMP_FULL_DATABASE	
07.	AQ_USER_ROLE	Untuk Advanced Queueing
08.	AQ_ADMINISTRATION_ROLE	
09.	SNMPAGEN	Untuk Enterprise Manager / Intelligent Agent.
10.	RECOVERY_CATALOG_OWNER	Untuk membuat user tyang memiliki sebuah recovery catalog.
11.	HS_ADMIN_ROLE	Digunakan untuk hetereogeneous feature.

Untuk melihat hak akses pada sebuah ROLE dapat melihat tabel DBA_SYS_PRIVS dengan cara :

SQL > SELECT * FROM DBA_SYS_PRIVS WHERE GRANTEE IN ('CONNECT','RESOURCE');

- Membuat atau Menghapus ROLE

  Definisi : CREATE ROLE nama role [IDENTIFIED BY password];

  Contoh :

  SQL > CREATE ROLE emp_role;

  SQL > GRANT DBA TO emp_role;

  Untuk menghapus Role dapat menggunakan perintah

  Definisi : DROP ROLE nama role

  Contoh : DROP ROLE emp_role;

  Untuk Mengetahui apakah sebuah ROLE mempunyai password dengan cara

  SQL > SELECT * FROM DBA_ROLES;

g. Perintah REVOKE

Digunakan untuk membatalkan hak privileges, objek atau role pada suatu user atau role tertentu.

Definisi :

REVOKE { system_privilege } | { role_name } | < all privileges > FROM < {user_name } | { role_name } | PUBLIC >;

Contoh :

SQL > REVOKE emp_role from ubm;

Daftar Pustaka

Alan Denis, Barbara Haley Wixon, David Tagerden, *System Analys and Design : an Object – Oriented Approach with UML 2.0*, John Willey and Sons, 2005

Henry C. Lucas Jr., *The Analysis, Design and Implementation of Information Systems 4th*, McGraw Hill, 1992

Satzinger, Jackson, Burd, *Object-Oriented Analysis and Design with the Unified Process*, Course Technology, 2005

Simon Bennet, Steve McRobb and Ray Farmer, *Object-Oriented System Analysis and Design Using UML*, McGraw Hill, 2006

Wendy and Michael Boggs, *UML with Rational Rose 2002*, Sibex Inc., 2002

UNIVERSITAS MERCU BUANA

MODUL PERKULIAHAN

PEMROGRAMAN WEB

Modul Standar untuk digunakan dalam Perkuliahan di Universitas Mercu Buana

Fakultas	Program Studi	Tatap Muka	Kode MK	Disusun Oleh
Ilmu Komputer	Sistem Informasi	05	87007	Team Dosen

Abstract

SDLC, RAD, Design Tersturktur, Class dan Object

Kompetensi

Memahami siklus pengembangan sistem, memahami metodhologi pengembangan sistem, mengenal UML

SQL PLUS

4.1 FASILITAS DALAM SQL PLUS

4.1.1 SLASH (/)

Digunakan untuk menjalankan perintah SQL atau PL / SQL yang disimpan dalam buffer.

4.1.2 EXEC[UTE]

Digunakan untuk menjalankan sebuah perintah PL/SQL.

4.1.3 TIMI[ING] [START text|SHOW|STOP]

Record timing data for an elapsed periode of time, lists the current timer's name and timing data, or lists the number of active timers.

4.1.4 HELP [topic]

Digunakan untuk memberikan informasi tentang perintah – perintah SQL*Plus.

4.1.5 HO[ST] [command]

Digunakan untuk menjalankan perintah sistem operasi tanpa meninggalkan SQL*Plus.

4.1.6 A[PPEND] text

Menambahkan spesifikasi text pada akhir perintah SQL di buffer.

Contoh :

```
SQL > SELECT * FROM employee;
SQL > APPEND  WHERE nip='E0001';
```

4.1.7 C[HANGE] sepchar old [sepchar [new [sepchar]]]

Digunakan untuk merubah karakter / string yang lama dengan yang baru. Contoh

SQL > SELECT * FROM employee WHERE nip='E0001';
SQL > c/E0001/E0002;

4.1.8 DEL [n|n m|n |n LAST||* n|* LAST|LAST]

Untuk menghapus beberapa baris perintah atau keseluruhan. Sebagai Contoh :

SQL > SELECT *
SQL > FROM employee
SQL > WHERE nip = 'E0001';
Setelah dijalankan
SQL > DEL 3;

4.1.9 I[NPUT] [text]

Untuk menambahkan satu atau lebih perintah SQL setelah perintah SQL yang ada pada buffer.

Contoh :

SQL > SELECT * FROM employee;

Setelah dijalankan

SQL > input where nip = 'E0001';

4.1.10 [IST] [n|n m|n |n LAST|*|* n|* LAST|LAST]

Untuk melihat perintah SQL pada buffer.

Contoh :

SQL > SELECT * FROM employee
SQL > WHERE nip = 'E0001';
SQL > LIST;

4.1.11 file_name[.ext] [arg …]

Digunakan untuk menjalankan perintah SQL pada sebuah file.

4.1.12 @@ file_name[.ext]

Digunakan untuk dalam file yang memanggil file lainnya.

4.1.13 ED[IT] [file_name[.ext]]

Digunakan untuk mengedit perintah SQL pada buffer atau pada file.

4.1.14 REM[ARK]

Digunakan untuk memberikan keterangan pada sebuah perintah SQL di dalam sebuah file.

4.1.15 SAV[E] file_name [.ext] [CRE[ATE] | REP[LACE] | APP[END]]

Digunakan untuk menyimpan isi dari buffer ke dalam sebuah file.

4.1.16 STORE {SET} file_name[.ext] [CRE[ATE] | REP[LACE] | APPE[END]]

Untuk menyimpan atribut setup SQL*Plus ke sebuah file

4.1.17 STA[RT] file_name[.ext] [arg….]

Digunakan untuk menjalankan isi dari sebuah file yang berisikan perintah SQL.

4.2 DATA MANIPULATION LANGUAGE (DML)

Data Manipulation Language adalah bahasa yang digunakan untuk memanipulasi objek – objek yang ada di database. Dibawah ini adalah contoh tabel yang akan dipergunakan untuk menjalankan perintah SQL, sebagai penggambaran penggunaan perintah SQL dalam manipulasi data di database. Untuk membuat tabel lihat pada pelajaran Data Definition Language (DDL).

Nama Tabel : employee
Primary Key : NIP
Index : idx_employee

NIP	NAMA	AlAMAT	SEX	NO_TELP	NO_FAX	GAJI
E0001	Josephine	Jl. Surya No.1 DKI Jakarta	F	654232	878623	100.000
E0002	Piter	Jl. Garuda No.2 DKI Jakarta	M	653322	676777	200.000

E0003	Surya	Jl. Manggis No.3 DKI Jakarta	M	873022	736322	300.000

PERINTAH INSERT (INSERT STATEMENT)

Perintah INSERT digunakan untuk memasukkan data baru ke dalam database.

Definisi Klausa :

INSERT INTO <table | view(column_1, column_2, ..) > VALUES (values_clause) | subquery.

Contoh :

SQL > INSERT INTO employee(nip,nama,alamat,sex,no_telp,no_fax,gaji)
VALUES('E0004','Libra','Jl. Bali No. 23 DKI Jakarta','M','768323','345233','100000');

Atau

SQL > INSERT INTO employee SELECT * FROM karyawan;

Asumsi bahwa tabel Karyawan sudah ada. Dengan atribut yang sama dengan tabel karyawan.. Untuk contoh kedua, data di ambil dari tabel Karyawan.

PERINTAH UPDATE (UPDATE STATEMENT)

Perintah Update digunakan untuk merubah data yang sudah ada di dalam database.

Definisi Klausa :

UPDATE table_name set_clause [where_clause] ;

Contoh :

SQL> UPDATE employee SET alamat = 'Jl. Garuda No.1 DKI' Jakarta WHERE nip = 'E0001';

Atau

SQL> UPDATE employee set alamat = (SELECT alamat FROM karyawan WHERE nip = 'E0002');

PERINTAH DELETE (DELETE STATEMENT)

Perintah Delete digunakan untuk menghapus data di dalam database.

Definisi Klausa :

DELETE [FROM] table_name [WHERE condition];

Contoh :

SQL > DELETE FROM employee WHERE nip='E0003';

Atau

SQL > DELETE FROM employee WHERE nip = (SELECT nip FROM karyawan WHERE nip='E0003');

PERINTAH SELECT (SELECT Statement)

Perintah SELECT adalah perintah SQL yang digunakan untuk menampilkan data dari database baik itu menggunakan kondisi ataupun tidak.

Bentuk Klausa :

SELECT [DISTINCT | UNIQUE | ALL]
{ [< schema.][table.][view.] > | t_alias.]<*|column> [AS] c_aliases]}
FROM { [schema.] < table | view > [t_aliases] }
[WHERE where_condition]
[GROUP BY group_by_expression]
[HAVING having_condition]
[UNION | UNION ALL | INTERSECT | MINUS select_statement]
[ORDER BY {expression [ASC | DESC] [NULLS <FIRST | LAST > }];

Secara umum data di dalam database dapat ditampilkan dengan menggunakan perintah SELECT dan FROM yang merupakan 2 perintah mandatory dalam perintah query. Sebagai contoh :

4.3 MENAMPILKAN SEMUA ATRIBUT PADA TABEL

SQL > SELECT * FROM employee;

Keterangan :

Perintah di atas akan menampilkan semua data dari tabel employee.

4.4 MENAMPILKAN BEBERAPA ATRIBUT PADA TABEL

SQL > SELECT nip,nama FROM employee;

Keterangan :

Perintah di atas akan menampilkan semua data dari field **nip** dan **nama** dari tabel Employee.

4.5 MENAMPILKAN ALIAS UNTUK ATRIBUT TABEL

SQL > SELECT nip "N.I.P", nama "NAMA KARYAWAN" FROM employee atau
SQL > SELECT nip AS "N.I.P", nama AS "NAMA KARYAWAM" FROM employee

Keterangan :

Untuk menampilkan alias dari sebuah atribut tabel dapat menggunakan AS atau langsung dengan menggunakan tanda kutip dua.

MENYARING DATA ATRIBUT DENGAN MENGGUNAKAN DISTINCT ATAU UNIQUE

SQL > SELECT DISTINCT * FROM employee atau SELECT UNIQUE * FROM employee

Keterangan

Dengan penambahan kata DISTINCT atau UNIQUE maka data yang sama hanya akan ditampilkan 1 kali saja.

MENAMPILKAN DATA ATRIBUT DENGAN MENGGUNAKAN OPERATOR ARITMATIKA

Operator aritmatika yang dipakai dalam perintah SELECT adalah perkalian (*), penambahan (+), pengurangan (-), pembagian (/) . Penambahan tanda kurung pada operator aritmatika maka prioritas perhitungan akan diutamakan pada tanda kurung tersebut. Sebagai Contoh :

SQL > SELECT nip,nama, gaji + (gaji * 0,1) FROM employee;

4.6 MENAMPILKAN DATA ATRIBUT DENGAN MENGGUNAKAN KLAUSA WHERE

Penambahan klausa WHERE pada perintah SELECT adalah untuk memberikan data dengan kondisi yang diinginkan.

4.7 KLAUSA WHERE dengan OPERATOR PEMBANDING

Di dalam klausa WHERE terdapat operator pembanding yang sering digunakan, seperti

- Tanda sama dengan (=), contoh :

SQL > SELECT * FROM employee WHERE nip = 'E0001';

- Tanda lebih besar (>), contoh :

SQL > SELECT * FROM employee WHERE nip > 'E0001';

- Tanda lebih kecil (<), contoh :

SQL > SELECT * FROM employee WHERE nip < 'E0001';

- Tanda lebih kecil atau sama dengan (<=), contoh :

SQL > SELECT * FROM employee WHERE nip <= 'E000';

- Tanda tidak sama dengan (X <> Y, X != Y, X ^= Y), contoh :

SQL > SELECT * FROM employee WHERE nip < > ' E0001';

- LIKE ..... [ESCAPE]

Digunakan untuk menentukan data atribut sesuai dengan karakter yang diinginkan. Penggunaan WILDCARD (%) atau "_" sering digunakan untuk klausa LIKE.Sebaliknya penggunaan klausa ESCAPE digunakan untuk mencari data atribut yang mempunyai karakter "_" atau "%". Sebagai contoh :

SQL > SELECT * FROM employee WHERE nip LIKE 'E%_001';

Contoh penggunaan klausa ESCAPE

SQL > SELECT * FROM emp WHERE nip LIKE 'E#%_001' ESCAPE '#';

- BETWEEN.....AND.....

Operator ini digunakan untuk mencari data yang mempunyai nilai antara. Contoh :

SQL > SELECT * FROM employee WHERE nip BETWEEN 'E0001' AND 'E0002';

- IN

Digunakan untuk memasukkan nilai yang mempunyai kriteria dalam Klausa IN.
Contoh :

SQL > SELECT * FROM employee WHERE nip IN ('E0001','E0002');

- NULL

NULL di dalam database, mewakili data yang kosong alias tidak ada data. Sebaliknya jikalau di dalam perintah SELECT kita menambahkan perintah NOT NULL, maka data yang bukan NULL akan ditampilkan. Sebagai contoh :

SQL > SELECT nip,nama FROM employee WHERE nip is not null;

4.8 KLAUSA WHERE dengan OPERATOR LOGIKA

- Operator Logika AND.

Pengertian operator logika AND dapat dijabarkan sebagai berikut

No.	X	Y	AND
01.	True	True	True
02.	True	False	False
03.	False	False	False
04.	False	True	False

Sebagai Contoh :

SQL > SELECT * FROM employee WHERE nip = 'E0001' AND nip = 'E0002';

- Operator Logika OR

Pengertian operator logika OR dapat dijabarkan sebagai berikut

No.	X	Y	OR
01.	True	True	True
02.	True	False	True
03.	False	False	False
04.	False	True	True

Sebagai Contoh :

SQL > SELECT * FROM employee WHERE nip = 'E0001' OR nip = 'E0002';

- Operator Logika NOT

Pengertian operator logika NOT dapat dijabarkan sebagai berikut

No.	X	NOT

01.	True	False
02.	FaLse	True

Sebagai Contoh :

SQL > SELECT * FROM employee WHERE nip NOT LIKE 'E0001' OR nip = 'E0002';

- ORDER BY [ASC | DESC]

Penambahan klausa ORDER BY pada perintah SELECT digunakan untuk mengurutkan data atribut berdasarkan atribut yang diletakan setelah klausa ORDER BY. Penambahan klausa ASC adalah untuk mengurutkan dari yang terkecil sampai yang terbesar sedangkan penggunaan klausa DESC adalah kebalikannya. Contoh :

SQL > SELECT * FROM employee WHERE nip LIKE 'E%' ORDER BY nip ASC;

- NULLS FIRST dan NULLS LAST

Penggunaan klausa NULLS FIRST akan menampilkan data yang bernilai null pada awal penampilan data sedangkan NULLS LAST maka data yang bernilai null akan ditampilkan pada akhir baris data yang ditampilkan. Contoh :

SQL > SELECT * FROM employee WHERE gaji > 100000 ORDER BY empno DESC NULLS LAST, 1;

- DECODE

Penambahan klausa DECODE pada perintah SELECT sama dengan melakukan IF THEN ELSE pada prosedur. Sebagai Contoh :

SQL> select nip,nama, decode(sex,'M','MALE','F','FEMALE') sex FROM employee ORDER BY nip DESC;

- TABEL DUAL

Tabel digunakan untuk melakukan validasi pada sebuah atribut tabel, seperti penggunaan operator aritmatika ataupun untuk menampilkan nilai default yang dapat diambil dari sistem database. Sebagai Contoh :

SQL > SELECT user,sysdate FROM dual;

Atau

SQL > SELECT 1 + 2 FROM dual;

- NVL atau NVL2

Klausa NVL Digunakan untuk menggantikan nilai NULL dengan nilai yang diinginkan. Sedangkan NVL2 mempunyai pengertian jikalau ekspresi 1 tidak null maka nilainya adalah ekspresi 2, jikalau ekspresi 1 null maka nilanya adalah ekspresi 3.

Contoh :

SQL > SELECT (nama,NVL(nama,'***')) FROM employee;

Atau

SQL > SELECT(nama, NVL2(nama,'Aktif','Berhenti')) status FROM employee;

4.9 FUNGSI – FUNGSI KARAKTER

- Klausa UPPER

Digunakan untuk merubah huruf menjadi besar. Contoh :

SQL > SELECT nip,UPPER(nama) FROM employee;

- Klausa LOWER

Digunakan untuk merubah huruf menjadi kecil. Contoh :

SQL > SELECT nip,LOWER(nama) FROM employee;

- Klausa INITCAP

Digunakan untuk merubah sebuah kalimat di mana awal kalimat huruf besar diikuti dengan huruf kecil. Contoh :

SQL > SELECT nip,INITCAP(nama) FROM employee;

- Klausa CONCAT

Digunakan untuk menggabungkan 2 karakter (atribut) dalam klausa SELECT.
Contoh :

SQL > SELECT CONCAT(nip,nama) FROM employee;

- Klausa SUBSTR

Digunakan untuk mengambil sebuah atau beberapa karakter dari sebuah kalimat (string) dan menentukan awal karakter dan banyak karakter yang mau di ambil. Contoh :

SQL > SELECT nip, substr(nama, 2, 3) FROM employee;

- Klausa LENGTH

Digunakan untuk menentukan panjang dari sebuah kalimat. Contoh :

SQL > SELECT nip,nama,LENGTH(nama) FROM employee;

- Klausa INSTR

Digunakan untuk menentukan sebuah substring di dalam string. Contoh :

SQL > SELECT INSTR('SQL dan PL/SQL', 'SQL') posisi FROM dual;

Posisi
12

- Klausa LTRIM dan RTRIM

LTRIM digunakan untuk menghapus karakter dari kiri. Sedangkan RTRIM digunakan untuk menghapus karakter dari kanan.

Contoh :

SQL > SELECT LTRIM('xyxXxyLAST WORD','xy') "LTRIM example" FROM dual;

LTRIM example
XxyLAST WORD

SQL> SELECT RTRIM('BROWNINGyxXxy','xy') "RTRIM e.g." FROM dual;

RTRIM e.g
BROWNINGyxX

- Klausa TRIM

Merupakan penggabungan dari RTRIM dan LTRIM.Digunakan untuk menghapus sebuah karakter dalam string. Contoh :

SQL > SELECT TRIM(" oracle ') trim FROM dual;

TRIM
oracle

SQL > SELECT TRIM('-' FROM '—Oracle –') trim FROM dual;

TRIM
Oracle

SQL> SELECT TRIM(LEADING '-' FROM '—oracle –') trim FROM dual;

TRIM
oracle--

- Klausa REPLACE

Digunakan untuk menganti sebuah atau beberapa karakter dengan sebuah atau beberapa karakter.

Contoh :

SQL > SELECT REPLACE ('JACK and JUE','J','BL') "Changes" FROM dual;

Changes
BLACK and BLUE

- Klausa TRANSLATE

Digunakan untuk mengganti beberapa karakter dengan karakter baru. Contoh :

SQL > SELECT TRANSLATE (nip,'E','I') from employee;

Maka karakter 'E" akan diganti dengan 'I'

- Klausa ASCII dan CHR

Klausa ASCII digunakan untuk menentukan nilai ASCII pada sebuah karakter. Sedangkan CHR merupakan kebalikan dari nilai ASCII. Contoh :

SQL > SELECT ASCII('A'),CHR(65) from dual;

ASCII('A') CHR(65)
65 A

4.10 FUNGSI – FUNGSI NUMERIK

- Klausa ROUND

Definisi klausa : ROUND(x, [,n])

Digunakan untuk melakukan pembulatan nilai decimal di belakang koma. Contoh :

SQL > SELECT ROUND(25.5255 , 3) FROM dual;

ROUND(25.5255)
25.523.

- Klausa TRUNC

Definisi Klausa : TRUNC(x [, n])

Fungsi numerik ini digunakan untuk melakukan penghapusan nilai decimal yang tidak diinginkan. Contoh :

SQL > SELECT TRUNC(25.345, 2) FROM dual;

TRUNC(25.345,2)
25.3

- Klausa CEIL dan FLOOR

Digunakan untuk melakukan pembulatan ke bilangan bulat (integer). Fungsi CEIL untuk pembulatan ke atas sedangkan FLOOR untuk pembulatan ke bawah.

Definis klausa : CEIL(x), FLOOR(x)

SQL > SELECT CEIL(18.95), FLOOR(18.95) FROM DUAL;

CEIL(18.95)	FLOOR(18.95)
19	18

- Klausa MOD

Definis klausa : MOD(x, y)

Nilai balikan dari rumusan di atas adalah sebuah angka yang merupakan sisa dari operasi pembagian x/y. Contoh :

SQL > SELECT MOD(15, 4) mod1, MOD(15.35, 3.5) mod2 FROM DUAL;

MOD1	MOD2
3	1.35

- Klausa ABS

Definisi klausa : ABS(x)

Nilai balikan dari fungsi ini adalah nilai absolut dari x. Contoh :

SQL > SELECT ABS(-2.75) x, ABS(0) y, ABS(2.75) z FROM DUAL;

X	Y	Z
2.75	0	2.75

- Klausa SIGN

Definisi klausa : SIGN(x)

Untuk suatu nilai apakah positif,negatif atau 0.

Contoh :

SQL > SELECT SIGN(-2.75) x, SIGN(0) y, SIGN(2.75) z FROM DUAL;

X	Y	Z
-1	0	1

- Klausa SQRT

Definisi Klausa : SQRT(x)

Untuk menentukan nilai akar kuadrat dari argumen x. Contoh :

SQL > SELECT SQRT(10) FROM DUAL;

SQRT(10)
3.162277

- Klausa POWER

Definisi klausa : POWER (x, y)

Digunakan untuk pemangkatan nilai numerik. Contoh :

SQL > SELECT POWER(2,2) FROM DUAL;

POWER(2,2)
4

- Klausa EXP

Definisi klausa : EXP(x). Digunakan untuk pemangkatan bilangan natural (e). Contoh :

SQL > SELECT EXP(3) FROM DUAL;

EXP(3)
20.0855369

- Klausa LOG dan LN

Definisi klausa : LOG(x,y) atau LN(y)

LOG digunakan untuk menghasilkan nilai logaritma berbasis x untuk nilai y. Sedangkan fungsi LN memberikan nilai logaritma natural untuk nilai y.Contoh :

SQL > SELECT LOG(10, 100), LN(5) FROM DUAL

LOG(5,2)	LN(2)
.430676558	1.09861229

- Klausa COS, SIN dan TAN

Definisi Klausa : COS(x), SIN(x), TAN(x). Digunakan untuk fungsi trigonometri. Contoh :

SQL > SELECT COS(30*3.14159265/180) A, SIN(30*3.14159265/180) B, TAN (30*3.14159265/ 180) C FROM DUAL;

A	B	C
.866025404	.499999999	.577350268

- Klausa COSH, SINH, TANH

Definisi Klausa : COSH(x), SINH(x), TANH(x). Digunakan untuk fungsi hiperbolik.

Contoh :

- Klausa ACOS,ASIN, ATAN dan ATAN2

Definisi Klausa : ACOS(x), ASIN(x), ATAN(x), ATAN2(x,y). Merupakan fungsi kosinus, sinus, dan tangen. Contoh:

SQL > SELECT ASIN(0.5), ACOS(1), ACOS(1/3), ATAN2(1,3) FROM DUAL;

ASIN(0.5)	ACOS(1)	ATAN(1/3)	ATAN2(1,3)
.523598776	0	.321750554	.321750554

- Klausa BITAND

Definisi klausa : BITAND(n1, n2)

Digunakan untuk melakukan operasi AND untuk nilai – nilai bit antara dua argumen.

Contoh :

SQL > SELECT TO_NUMBER(BITAND(2,3)) FROM DUAL;

TO_NUMBER(BITAND(2,3))
2

4.11 FUNGSI TANGGAL DAN WAKTU

- Klausa ADD_MONTHS

Definisi klausa : ADD_MONTHS(d, n)

Digunakan untuk mendefinisikan sebuah tanggal yang berjarak n bulan setelah tanggal d. Contoh :

SQL > SELECT ADD_MONTHS(SYSDATE,1) FROM DUAL;

- Klausa MONTHS_BETWEEN

Klausa Definisi : MONTHS_BETWEEN(d1,d2)

Digunakan untuk menentukan jumlah bulan antara tgl d1 dan d2. Jika d1 < d2 maka hasilnya adalah negatif. Jika d1 > d2 maka hasilnya adalah positif. Contoh :

SQL > SELECT MONTHS_BETWEEN(sysdate,'22-JAN-2004') tgl from dual;

TGL
2.50789725

- Klausa NEW_TIME

Klausa Definisi : NEW_TIME(d, zone 1, zone 2)

Tanggal dan waktu pada zone 2 ketika tanggal dan waktu zone 1 adalah d.

Tabel Zona Waktu

AST,ADT	Atlantic Standard / Daylight Time
BST, BDT	Bering Standard / Daylight Time
CST, CDT	Central Standard / Daylight Time
EST, EDT	Eastern Standard / Daylight Time
HST, HDT	Alaska – Hawaii Standard / Daylight Time
GMT	Greenwich Mean Time
MST, MDT	Mountain Standard / Daylight Time
NST	Newfoundland Standard / Daylight Time
PST, PDT	Pacific Standard / Daylight Time
YST, YDT	Yukon Standard / Daylight Time

Contoh :

```
SQL > ALTER SESSION SET NLS_DATE_FORMAT = 'DD-MON-YYYY HH24:MI';
SQL > SELECT SYSDATE, NEW_TIME(SYSDATE, 'AST', 'PST' ) new_time FROM DUAL;
```

- Klausa NEXT_DAY

Definisi Klausa : NEXT_DAY(d, day_name)

Digunakan untuk menghasilkan tanggal yang jatuh pada nama hari (day_name) yang terdekat setelah tanggal d.

Contoh :

```
SQL > SELECT NEXT_DAY(SYSDATE, 'WEDNESDAY')  Tgl FROM dual;
```

Tgl
07-APR-2004

- Klausa LAST_DAY

Definisi Klausa : LAST_DAY(d)

Menentukan tanggal terakhir pada bulan yang sama dengan bulan argumen. d harus bernilai tanggal.

Contoh:

```
SQL > SELECT LAST_DAY(SYSDATE) Tgl FROM dual;
```

Tgl
30-APR-2004

4.12 FUNGSI KONVERSI

Fungsi konversi digunakan untuk merubah suatu nilai ke dalam format yang lain. Misalnya dari karakter (CHAR) ke tanggal (DATE) atau dari tanggal (DATE) ke bilangan bulat (NUMBER). Beberapa nilai konversi yang dikenal di Oracle adalah sebagai berikut :

- TO_CHAR

Digunakan untuk merubah sebuah data ke dalam bentuk karakter.

Contoh :

SQL > SELECT TO_CHAR(gaji) from employee;

Maka atribut gaji akan berubah menjadi karakter.

- TO_DATE

Digunakan untuk merubah sebuah data atribut ke dalam bentuk format tanggal.

Contoh :

SQL > SELECT TO_DATE('20040214','YYYYMMDD') FROM dual;

TO_DATE('
14-FEB-2004

- TO_NUMBER

Digunakan untuk merubah sebuah data atribut ke dalam bentuk bilangan.

Contoh :

SQL > SELECT TO_NUMBER('10') + 50 FROM dual;

TO_NUMBER('
60

4.13 TRANSACTION CONTROL STATEMENTS

- **PERINTAH COMMIT**

Perintah Commit di dalam database digunakan untuk menyimpan data secara permanen.

Definisi Klausa : COMMIT

Contoh :

SQL>**INSERT INTO Employee(nip,nama,alamat,sex.no_telp,no_fax)VALUES('E0005','Mahendra','Jl.Kepak No.23 – DKI Jakarta','M','765876','987600','200000');**

SQL > **COMMIT;**

- PERINTAH ROLLBACK

Perintah Rollback di dalam database digunakan untuk membatalkan data yang baru saja di masukkan, dirubah ataupun dihapus.

Definisi Klausa : ROLLBACK;

Contoh :

SQL > **DELETE FROM employee;**
4 rows deleted.
SQL > **ROLLBACK;**
Rollback complete.

- PERINTAH SAVEPOINT

Perintah ini digunakan untuk menyimpan transaksi yang sedang berjalan.

Definisi Klausa : SAVEPOINT savepoint;

Contoh :

SQL **>INSERT INTO Employee(nip,nama,alamat,sex.no_telp,no_fax)VALUES('E0006','Jupiter','Jl.Kepak No.40 – DKI Jakarta','M','765876','987600','300000');**

SQL > **SAVEPOINT A;**

SQL > **SELECT * FROM employee;**

SQL > **SAVEPOINT B;**

SQL > **DELETE FROM employee;**

SQL > **ROLLBACK TO SAVEPOINT A;**

- PERINTAH SET TRANSACTION

Digunakan untuk membuat transaksi sekarang sebagai READ ONLY atau READ WRITE. Perintah SET TRANSACTION tidak berlaku lagi jika perintah COMMIT atau ROLLBACK dijalankan.

Definisi Klausa : SET TRANSACTION param.

Daftar Pustaka

Alan Denis, Barbara Haley Wixon, David Tagerden, *System Analys and Design : an Object – Oriented Approach with UML 2.0*, John Willey and Sons, 2005

Henry C. Lucas Jr., *The Analysis, Design and Implementation of Information Systems 4th*, McGraw Hill, 1992

Satzinger, Jackson, Burd, *Object-Oriented Analysis and Design with the Unified Process*, Course Technology, 2005

Simon Bennet, Steve McRobb and Ray Farmer, *Object-Oriented System Analysis and Design Using UML*, McGraw Hill, 2006

Wendy and Michael Boggs, *UML with Rational Rose 2002*, Sibex Inc., 2002

UNIVERSITAS MERCU BUANA

MODUL PERKULIAHAN

PEMROGRAMAN WEB

Modul Standar untuk digunakan dalam Perkuliahan di Universitas Mercu Buana

Fakultas	Program Studi	Tatap Muka	Kode MK	Disusun Oleh
Ilmu Komputer	Sistem Informasi	06	87007	Team Dosen

Abstract

SDLC, RAD, Design Tersturktur, Class dan Object

Kompetensi

Memahami siklus pengembangan sistem, memahami metodhologi pengembangan sistem, mengenal UML

DATA MANIPULATION LANGUAGE

5.1 FASILITAS DALAM DML

5.1.1. RUN

Fasilitas ini dapat menampilkan perintah SQL dalam buffer sebelum dijalankan. Hal ini berbeda dengan perintah slash (/) yang langsung menjalankan perintah SQL dari buffer. Contoh :

SQL > SELECT * FROM employee;
No row selected.
SQL > RUN
SQL> SELECT * FROM employee;
No row selected.

5.1.2. CLEAR

Fasilitas ini digunakan untuk menghapus semua perintah yang ada di buffer SQLPlus.

Contoh :

SQL > Cl Buff
Buffer Cleared.

5.1.3. CONNECT [user name / password]

Digunakan untuk menghubungkan aplikasi dengan database. Contoh :

SQL > connect sep/sep;

5.1.4. DISCONNECT

Digunakan untuk memutuskan hubungan aplikasi dengan database.

Contoh :

SQL > Dicsonnect;

5.1.5. SPOOL < file.ext >

Digunakan untuk merekam seluruh aktivitas perintah SQLPlus dan hasilnya ke dalam sebuah file.
Contoh :

SQL > Spool c:\hasil.txt;

5.1.6. SPOOL OFF

Digunakan untuk menghentikan perintah SPOOL. Contoh :

SQL > Spool Off;

5.1.7. PASSWORD [username]

Digunakan untuk merubah password

SQL > passw sep
Changing password for sep
Old password : ....
New password : .....
Retype new password : .....
Password changed

5.2. PERINTAH SELECT (SELECT STATEMENT)

5.2.1. ORDER BY ASC | DESC

Order By digunakan untuk mengurutkan data (sorting) dalam suatu tabel. Contoh :

SQL > SELECT * FROM employee WHERE nip IS NOT NULL ORDER by nip DESC;

5.2.2. PSEUDOCOLUMN (KOLOM VARIABEL)

Kolom variabel bertindak seperti kolom yang ada di tabel., namun sebenarnya tidak disimpan di dalam tabel. Perintah SQL seperti INSERT,UPDATE,DELETE tidak dapat dilakukan dengan menggunakan kolom variabel namun dapat menggunakan perintah SELECT untuk melihat data yang dihasilkan. Yang dimaksud PSEUDOCOLUMN adalah

- **CURRVAL dan NEXTVAL**

  Kolom ini dapat digunakan untuk SEQUENCE number.

- **LEVEL**

  Digunakan untuk query hirarki.

- **ROWID**

  Kolom ini menunjukkan alamat baris dalam query. Informasi dari ROWID adalah :

  - Nomor objek data
  - Data Blok yang digunakan di dalam data file
  - Baris yang digunakan dalam data blok (baris pertama adalah 0)

- Data File yang digunakan (File pertama adalah 1).

Nilai sebuah ROWID adalah unik untuk mengidentifikasikan sebuah baris di dalam database.

Keuntungan menggunakan ROWID adalah :

- Mempercepat proses pencarian baris data.
- Dapat memperlihatkan bagaimana baris data di simpan dalam tabel.
- Unik Identifikasi baris pada sebuah tabel.

ROWID tidak dapat dipergunakan sebagai primary key.

- **ROWNUM**

Digunakan untuk menampilkan urutan data.

5.2.3. SYS_CONTEXT dan USERENV

Baik SYS_CONTEXT maupun USERENV digunakan untuk memberikan informasi di dalam session.

Klausa SYS_CONTEXT adalah : SYS_CONTEXT('USERENV','parameter') sedangkan klausa USERENV adalah USERENV('parameter');

Adapun parameter yang dapat digunakan adalah :

Paramater	Keterangan
USER ENV Paramater	
CLIENT_INFO	Memberikan informasi tentang client
INSTANCE	Digunakan untuk memberikan informasi tentang jumlah instance yang dipakai.
ISDBA	TRUE jika hak akses user adalah DBA
LANG	Informasi tentang bahasa standardISO yang digunakan.
LANGUAGE	Informasi tentang bahasa teritori dan karakter yang digunakan dalam database.
SESSIONID	Identifikasi tentang audit session
SYS_CONTEXT Parameter	
TERMINAL	Informasi tentang terminal yang digunakan
CURRENT_SCHEMA	Informasi tentang Schema yang digunakan. Informasi schema dapat dirubah dengan menggunakan ALTER SESSION SET CURRENT_SCHEMA.
CURRENT_SCHEMAID	Informasi tentang ID Schema yang digunakan.
CURRENT_USER	Informasi tentang user
CURRENT_USERID	Informasi tentang User ID
DB_DOMAIN	Informasi tentang domain database.
HOST	Informasi tentang host

IP_ADDRESS	Informasi tentang IP Address yang digunakan
NLS_CALENDAR	Informasi tentang setup sistem kalendar yang digunakan
NLS_CURRENCY	Informasi tentang setup mata uang yang digunakan
NLS_DATE_FORMAT	Informasti tentang setup tanggal yang digunakan
NLS_SORT	Informasi tentang pengurutan. DEFAULT Binary
NLS_TERRITORY	Informasi tentang nama teritori
SESSION_USER	Informasi tentang nama user
SESSION_USERID	Informasi tentang id user

5.2.4. SYSDATE,USER,UID

Digunakan untuk memberikan informasi tentang tanggal yang digunakan sistem (SYSDATE), user yang sedang aktif (USER) dan ID user yang sedang aktif (UID).

Contoh :

SQL > SELECT SYSDATE,USER,UID FROM dual;

SYSDATE USER UID
13-APR-04 SEP 38

5.2.5. FUNGSI SQL UNTUK KELOMPOK DATA (AGREGAT)

Fungsi ini digunakan untuk melakukan perhitungan pada sekelompok data di dalam database. Umumnya data hasil lebih sedikit dari jumlah data yang masuk. Beberapa fungsi Antara lain :

a. SUM

Digunakan untuk melakukan perhitungan pada suatu tabel. Contoh :

SQL > SELECT SUM(Gaji) FROM employee;

b. AVG

Memberikan nilai rata – rata pada suatu tabel. Contoh :

SQL > SELECT AVG(Gaji), SUM(Gaji) / COUNT(Gaji) FROM employee;

c. MAX – MIN

MAX Digunakan untuk Memberikan nilai maksimum pada sebuah tabel sedangkan MIN digunakan untuk memberikan nilai minimum pada sebuah tabel. Contoh :

SQL > SELECT MAX(Gaji), MIN(Gaji) FROM employee;

d. COUNT

Digunakan untuk menghitung jumlah baris pada sebuah tabel. Contoh :

SQL > SELECT COUNT(*) FROM employee;

5.2.6. GROUP BY

Digunakan untuk mengelompokkan baris data.Dalam perintah select jikalau diisi dengan fungsi agregat maka tidak boleh ada single data kecuali dengan menggunakan perintah GROUP BY.

Contoh :

SQL > SELECT empid,COUNT(Empid,SUM(Gaji) FROM emp GROUP BY empid;

➔ Klausa CUBE & ROLLUP

Penambahan CUBE , ROLLUP pada klausa GROUP BY digunakan untuk perluasan query.

Contoh CUBE :

SQL > SELECT tgl_masuk,empid,SUM(gaji),COUNT(empid) FROM employee GROUP BY CUBE (tgl_masuk,empid);

Maka hasil query yang pertama adalah berdasarkan tgl_masuk+empid; kedua berdasarkan empid sedangkan yang terakhir berdasarkan keseluruhan data.

Contoh ROLLUP

SQL > SELECT tgl_masuk,empid,SUM(gaji),COUNT(empid) FROM employee GROUP BY ROLLUP (tgl_masuk,empid);

Maka hasil query yang pertama berdasarkan tgl_masuk+empid dan keseluruhan data.

5.2.7. HAVING

Penggunaan HAVING digunakan untuk pembatasan group data. Oleh karena berdasakan group data maka pada perintah SELECT harus ada perintah GROUP BY. Sedangkan perbedaan dengan klausa WHERE adalah digunakan untuk pembatasan baris data.

Contoh :

SQL > SELECT tgl_masuk,empid,sum(gaji) FROM employee WHERE empid BETWEEN 'E0001' AND 'E0005' GROUP BY tgl_masuk,empid HAVING sum(gaji) < 1000000;

5.2.8. MENGGABUNGKAN BEBERAPA TABEL (JOIN TABLE)

Penggabungan beberapa tabel sering digunakan untuk menghubungkan sebuah data dari tabel yang satu dengan yang lainnya. Ada beberapa istilah dalam menggabungkan tabel.

a. Product Cartesian

Menggabungkan beberapa tabel tanpa suatu kondisi tertentu. Akibatnya seluruh data akan ditampilkan baik itu yang berasal dari tabel pertama maupun tabel kedua. Ciri – ciri dari produk cartesian adalah :

- Terbentuk bila suatu penggabungan kondisi tabel tidak digunakan;
- Penggabungan Kondisi tabel salah;
- Baris yang ada pada tabel pertama ditampilkan dengan semua baris tabel yang kedua.

Contoh :

SQL > SELECT dept.deptid,employee.empid FROM dept,employee;

b. Equijoin

Equijoin atau simple join atau inner join adalah menampilkan data dengan penggabungan antara 2 atau lebih tabel dengan menggunakan operator sama dengan (=). Biasanya menggunakan primary key dan foreign key untuk menghubungkan 2 atau lebih tabel.

Contoh

SQL > SELECT A.empid,A.empnm,B.dept FROM Employee A, dept B WHERE A.deptid = B.deptid;

c. Non-Equijoin

Non-Equijoin adalah menampilkan data dengan penggabungan 2 atau lebih selain menggunakan operator sama dengan (=) juga menggunakan operator lainnya seperti lebih besar atau sama dengan (>=), BETWEEN … AND …

Contoh :

SQL > SELECT A.empid,A.empnm,B.dept FROM Employee A, dept B WHERE A.deptid = B.deptid and A.dept BETWEEN '01' AND '02';

d. Self Join

Menampilkan data dengan menggabungkan tabel dengan dirinya sendiri.

Contoh :

SQL > SELECT A.empid || ' no ID Karyawan '|| B.empnm FROM emp A,emp B where A.empid = B.empid;

e. Outer Join

Digunakan untuk menampilkan penggabungan tabel dengan menggunakan operator sama dengan (=) serta data yang tidak sesuai dengan equijoin.

Beberapa aturan di dalam menggunakan outer join

1. Operator outer joing (+) hanya dapat dipakai pada klausa WHERE.Dan hanya dapat digunakan kolom dalam tabel atau view ;
2. Jika tabel A join dengan tabel B dengan berbagai kondisi, maka operator outer join harus digunakan pada semua kondisi;
3. Operator outer join hanya bisa digunakan pada kolom, tidak pada ekspresi keputusan (artbitrary expression) ;
4. Kondisi yang menggunakan operator join tidak bisa digunakan pada operator logika OR;
5. Konsidi yang menggunakan operator IN tidak bisa digunakan untuk operator outer join (+);
6. Tidak bisa digunakan untuk subquery.

Contoh :

SQL > select a.empno,a.hiredate,b.deptno from emp a,dept b where a.deptno(+) = b.deptno and a.hiredate(+)= '17-DEC-80'

EMPNO	HIREDATE	DEPTNO
		10
7369	17-DEC-80	20
		30
		40

5.2.9. Operator Himpunan

Operator himpunan digunakan untuk menggabungkan hasil data dari 2 query atau yang dikenal dengan nama compound query. Ada beberapa ketentuan yang harus dipatuhi dalam menggunakan operator himpunan

- ➔ Jumlah kolom harus sama antara query yang pertama dengan yang kedua. Tipe data antara query yang satu dengan query yang kedua juga harus sama.
- ➔ Penggunaan tanda kurung untuk memperioritaskan urutan hasil query.

5.2.9.1. UNION | ALL

Digunakan untuk menggabungkan 2 buah query di mana duplikasi data akan ditampilkan. Penggunaan All berarti data query pertama ditampilkan terlebih dahulu sesudah itu baru data query kedua.

Contoh :

SQL > SELECT deptno,dname FROM dept UNION SELECT deptno,ename FROM Employee;

5.2.9.2. MINUS

Data yang akan ditampilkan merupakan data hasil query yang pertama.

Contoh :

SQL > SELECT deptno,dname FROM dept MINUS SELECT deptno,ename FROM Employee;

5.2.9.3. INTERSECT

Data yang ditampilkan merupakan data yang sama dari kedua query.

Contoh :

SQL > SELECT deptno FROM dept INTERSECT SELECT deptno FROM employee;

5.2.10. SUBQUERY

Subquery adalah sebuah query yang terdapat di dalam query(inner query). Sedangkan induk dari query di sebut dengan nama outer query. Manfaat subquery sangat banyak ,terutama jikalau kita hendak menyimpan data yang mempunyai kondisi tertentu. Beberapa aturan pemakaian subquery adalah sebagai berikut :

- ✓ Harus menggunakan tanda kurung
- ✓ Letakkan setelah tanda operator
- ✓ Subquery tidak dapat berisi klausa ORDER BY, karena anda diperkenankan hanya menggunakan satu klausa ORDER BY pada perintah SELECT, dan jika dipergunakan untuk mengurutkan data yang ditampilkan maka klausa ORDER BY diletakkan pada akhir perintah SELECT yang pertama.
- ✓ Operator pembanding yang dapat digunakan dalam subquery meliputi : operator single – row dan operator multi – row.

Subquery dapat ditempatkan pada Klausa setelah FROM, WHERE, HAVING.

Berdasarkan perbandingan kolom dan baris subquery dapat dibagi

- **Perbandingan Operasi Satu Baris (Single – Row Comparison Operation – Scalar Query)**

  Data yang ditampilkan merupakan hasil dari subquery yang mempunyai nilai satu baris. Tipe subquery ini menggunakan satu operator single – row seperti

  - Tanda sama dengan (=),
  - Tanda lebih besar (>) ,
  - Tanda lebih besar atau sama dengan (>=),
  - Ttanda lebih kecil (<).
  - Tanda lebih kecil atau sama dengan (<=),
  - Tanda tidak sama dengan (<>)).

  Jika hasil dari subquery tidak merupakan satu baris maka sistem database akan memberikan error "*single-row subquery returns more than one row*".

  Contoh :

  SQL > SELECT deptno,empno,empnm FROM employee where deptno = (SELECT deptno from Dept where deptno = '10');

  Contoh Penggunaan HAVING

  SQL > SELECT deptno,empno,empnm FROM employee GROUP BY empno HAVING min (Gaji) = (SELECT MIN(GAJI) from employee where deptno = '20');

- **Perbandingan Operasi Banyak Baris (Multi Row Comparison Operation)**

  Subquery yang menghasilkan nilai lebih dari satu baris. Operator yang digunakan adalah :

  - Operator IN

    Baris yang dihasilkan harus dalam operator IN. Operator "=ANY' atau "=SOME" mempunyai pengertian sama dengan operator IN.

    Contoh :

    SQL > SELECT deptno,empno,empnm FROM employee WHERE deptno IN (SELECT deptno FROM Dept);

Atau

SQL > SELECT deptno,empno,empnm FROM employee WHERE deptno =ANY (SELECT deptno FROM dept);

- Operator ANY atau SOME

Penggunaan operator ANY atau SOME harus di dahului operator (=, <=, !=, >=, <>, > , <). Contoh :

SQL > SELECT empno,ename, job FROM emp WHERE sal < ANY (SELECT sal FROM emp WHERE job = 'CLERK') AND job < > 'CLERK';

Atau

SQL > SELECT empno,ename, job FROM emp WHERE sal < SOME (SELECT sal FROM emp WHERE job = 'CLERK') AND job < > 'CLERK';

- ALL

Operator ALL membandingkan nilai semua hasil daripada subquery. Contoh penggunaan :

SQL > SELECT deptno,ename,sal FROM emp where sal > ALL (SELECT min(sal) FROM emp);

- **Perbandingan Operasi Banyak Baris (Multi Column Subquery – Non Pairwise Comparison)**

Operator yang digunakan adalah IN dan sama dengan (=).

SQL > SELECT deptno,empno,empnm FROM employee where deptno IN (SELECT deptno from Dept) AND gaji > (SELECT MIN(gaji) from employee);

- **Klausa FROM**

Contoh penggunaan subquery di dalam FROM.

SQL > SELECT a.deptno,a.ename FROM (select deptno,ename from emp) a

- **Operator EXISTS**

Operator EXISTS harus diletakkan setelah klausa WHERE. Penggunaan operator EXISTS menghasilkan paling tidak satu baris.

SQL > SELECT deptno,ename FROM emp WHERE EXISTS (select deptno from dept);

- **Uncorrelated and Correlated Subquery**

  Hasil dari inner query biasanya tidak berhubungan dengan outer query (Uncorrelated Subquery). Namun ada juga hasil dari inner query dipengaruhi oleh nilai outer query (Correlated Subquery).

  Contoh penggunaan uncorrelated subquery adalah

  SQL > SELECT deptno,empno,empnm FROM employee where deptno = (SELECT deptno from Dept where deptno = '10');

  Contoh penggunaan correlated subquery adalah

  SQL > SELECT a.deptno,b.ename FROM (SELECT deptno FROM dept) a, emp b WHERE a.deptno = b.deptno;

- **Penggunaan Optimizer pada Query**

  Penggunaan Optimizer harus di dahulu dengan tanda /*+ dan di akhiri dengan tandan */. Beberapa optimizer adalah

  1. USE_HASH

     Hanya berlaku untuk equijoin atau inner join. Untuk non – equijoin dapat menggunakan USE_MERGE. Dengan menggunakan optimizer ini maka sistem oracle database akan membuat hash tabel di memory kemudian membandingkan baris data dari tabel yang kecil dengan tabel lainnya.

     Contoh :

     SQL > select /*+USE_HASH(a,b) */ a.deptno,b.deptno from emp a,dept b where a.deptno=b.deptno;

  2. USE_MERGE

     Optimizer ini akan mengurutkan semua baris data pada tabel kecuali kolom pada klausa WHERE di pakai untuk index.

     Contoh :

     SQL > select /*+USE_MERGE(a,b) */ a.deptno,b.deptno from emp a,dept b where a.deptno=b.deptno;

3. USE_NL

   Hint ini memaksa menggabungkan suatu tabel dengan tabel lain (atau subquery) dengan menggabungkan inner tabel dan neested loop.

   Contoh :

   SQL > select /*+USE_NL(a,b) */ a.deptno,b.deptno from emp a,dept b where a.deptno=b.deptno;

4. HASH_AJ dan MERGE_AJ

   Digunakan untuk operator NOT. Contoh penggunaan :

   SQL > SELECT deptno,empno,ename FROM emp WHERE deptno NOT IN (SELECT /*+HASH_AJ*/ deptno FROM dept WHERE deptno = '10')

   Atau

   SQL > SELECT deptno,empno,ename FROM emp WHERE deptno NOT IN (SELECT /*+MERGE_AJ*/ deptno FROM dept WHERE deptno = '10');

5. HASH_SJ dan MERGE_SJ

   Digunakan untuk operator EXISTS. Contoh penggunaan :

   SQL > SELECT deptno,empno,ename FROM emp WHERE EXISTS (SELECT /*+HASH_SJ*/ deptno FROM dept WHERE deptno = '10');

   Atau

   SQL > SELECT deptno,empno,ename FROM emp WHERE EXISTS (SELECT /*+MERGE_SJ*/ deptno FROM dept WHERE deptno = '10');

   Beberapa ketentuan dalam penggunaan hint adalah :
   - Hanya boleh ada satu tabel pada subquery,
   - Tabel outer query bukanlah tabel yang sama dengan tabel pada subquery,
   - Korelasi subquery harus menggunakan operator "=",
   - Subquery tidak mengandung klausa GROUP BY, CONNECT BY, dan ROWNUM.

- **Query Hirarki**

  Sebuah tabel dapat digunakan sebagai query hirarki jika informasi baris dari tabel tersebut mempunyai hubungan dengan baris yang lain dari tabel tersebut. Hirarki yang terbentuk pada dijalankan tidak disimpan di dalam database

Definisi Klausa

SELECT [LEVEL], column, expr….
FROM table
[WHERE condition(s)]
[START WITH condition(s)]
[CONNECT BY PRIOR condition(s)] ;

Keterangan :

START WITH digunakan untuk menentukan root dari sebuah tabel
CONNECT BY digunakan untuk menghubungkan antara root dengan child.
PRIOR digunakan untuk menentukan sebuah ekspresi

Contoh Hirarki :

Tabel sebelum query hirarki dijalankan :

EMPNO	MGR	ENAME	JOB
7369	7902	SMITH	CLERK
7499	7698	ALLEN	SALESMAN
7521	7698	WARD	SALESMAN
7566	7839	JONES	MANAGER
7654	7698	MARTIN	SALESMAN
7698	7839	BLAKE	MANAGER
7782	7839	CLARK	MANAGER
7788	7566	SCOTT	ANALYST
7839		KING	PRESIDENT
7844	7698	TURNER	SALESMAN
7876	7788	ADAMS	CLERK
7900	7698	JAMES	CLERK
7902	7566	FORD	ANALYST
7934	7782	MILLER	CLERK

SQL > SELECT empno,mgr,ename,job FROM emp START WITH mgr IS NULL CONNECT BY PRIOR empno = mgr;

Hasil setelah perintah SQL dijalankan :

EMPNO	MGR	ENAME	JOB
7839		KING	PRESIDENT
7566	7839	JONES	MANAGER
7788	7566	SCOTT	ANALYST

7876	7788	ADAMS	CLERK
7902	7566	FORD	ANALYST
7369	7902	SMITH	CLERK
7698	7839	BLAKE	MANAGER
7499	7698	ALLEN	SALESMAN
7521	7698	WARD	SALESMAN
7654	7698	MARTIN	SALESMAN
7844	7698	TURNER	SALESMAN
7900	7698	JAMES	CLERK
7782	7839	CLARK	MANAGER
7934	7782	MILLER	CLERK

Penggunaan LPAD atau RPAD

Bentuk Klausa :

LPAD (str1, n [, str2])

RPAD (str1, n [, str2])

Berguna untuk menambahkan sejumlah str2 ke dalam str1 sehingga panjang gabungan string menjadi n karakter. LPAD menambahkan sebelah kiri sedangkan RPAD menambahkan sebelah kanan. Contoh penggunaan :

SQL > SELECT '|'||LPAD(ename,length(ename)+(level*2)+2,'_') AS org_chart FROM emp START WITH mgr IS NULL CONNECT BY PRIOR empno = mgr;

Maka hasilnya adalah :

```
ORG_CHART
|____KING
|______JONES
|________SCOTT
|__________ADAMS
|________FORD
|__________SMITH
|______BLAKE
|________ALLEN
|________WARD
|________MARTIN
|________TURNER

ORG_CHART
|________JAMES
```

|_____CLARK
|_____MILLER

Daftar Pustaka

Alan Denis, Barbara Haley Wixon, David Tagerden, *System Analys and Design : an Object – Oriented Approach with UML 2.0*, John Willey and Sons, 2005

Henry C. Lucas Jr., *The Analysis, Design and Implementation of Information Systems 4th*, McGraw Hill, 1992

Satzinger, Jackson, Burd, *Object-Oriented Analysis and Design with the Unified Process*, Course Technology, 2005

Simon Bennet, Steve McRobb and Ray Farmer, *Object-Oriented System Analysis and Design Using UML*, McGraw Hill, 2006

Wendy and Michael Boggs, *UML with Rational Rose 2002*, Sibex Inc., 2002

UNIVERSITAS MERCU BUANA

MODUL PERKULIAHAN

PEMROGRAMAN WEB

Modul Standar untuk digunakan dalam Perkuliahan di Universitas Mercu Buana

Fakultas	Program Studi	Tatap Muka	Kode MK	Disusun Oleh
Ilmu Komputer	Sistem Informasi	07	87007	Team Dosen

Abstract

SDLC, RAD, Design Tersturktur, Class dan Object

Kompetensi

Memahami siklus pengembangan sistem, memahami metodhologi pengembangan sistem, mengenal UML

PEMROGRAMAN PL/SQL

6.1 FUNGSI VALIDASI PADA SQL PLUS

Penggunaan fasilitas SQLPlus Lanjutan di bawah ini dapat menggunakan notepad untuk melihat hasil query yang optimal.

a. Tanda Ampersand (&) dan double Ampersand (&&)

Tanda Ampersand (&) digunakan untuk membuat sebuah variabel pada layar. Pada saat dijalankan, maka variabel tersebut harus diisi sebelum hasil query ditampilkan.

Contoh penggunaan tanda ampersand (&) ,

```
SQL > SELECT * FROM DEPT WHERE DEPTNO = &DEPTNO;
```

Sedangkan tanda double ampersand (&&) digunakan untuk menampilkan sebuah variabel dengan nilai yang sama pada variabel sebelumnya.

Contoh :

```
SQL > SELECT * FROM DEPT WHERE DEPTNO = &&DEPTNO;
SQL > SELECT * FROM EMP WHERE DEPTNO = &DEPTNO;
```

b. PROMPT

Fungsi PROMPT digunakan untuk menampilkan sebuah kalimat dalam layar SQLPlus.

Contoh penggunaan :

```
SQL > prompt Hello The World;
```

c. PAUSE

Fungsi PAUSE digunakan untuk menghentikan sementara waktu sebelum menjalankan perintah SQL selanjutnya.

Contoh penggunaan :

```
SQL > PAUSE < Tekan enter untuk melanjutkan proses. >
```

d. ACCEPT.

Fungsi ACCEPT digunakan untuk menampung nilai sebuah variabel di dalam layar.

Bentuk Klausanya :

ACC [EPT] variable_name [NUM[BER] | CHAR | DATE] [FOR[MAT] format] [DEF [AULT] default] [PROMPT text | NOPR [OMPT] [HIDE]

Keterangan :

- [NUM[BER] | CHAR | DATE] : digunakan untuk menentukan tipe data dari variabel.
- FORMAT digunakan untuk menentukan format variabel.
- DEFAULT digunakan untuk menentukan nilai default dari variabel.
- PROMPT digunakan untuk menampilkan text sedangkan NOPROMPT jikalau tidak mau ditampilkan.
- HIDE digunakan untuk menyembunyikan nilai masukan. Biasanya digunakan untuk password.

Contoh penggunaan :

SQL > ACCEPT v_user DEFAULT MANAGER PROMPT ' Masukkan User ID anda : ';

e. DEFINE

Fungsi dapat digunakan untuk mendefinisikan sebuah nilai pada variabel.

Contoh :

SQL > DEFINE v_user = 'Josephine';

f. BREAK

Fungsi BREAK dapat digunakan untuk :

- Menampilkan hanya satu nilai untuk beberapa nilai yang sama pada kolom yang ditentukan.
- Menambahkan baris kosong setiap kali nilai dari kolom yang ditentukan berubah.
- Mencetak nilai hasil perhitungan untuk kelompok baris dengan nilai kolom yang sama pada kolom yang ditentukan.

Bentuk Klausa :

BRE[AK] [ON < column > | expr | ROW | REPORT >
[SKIP n] | [SKIP] PAGE] [NODU[PLICATES] | DUP[LICATES]

Keterangan :

- Nama column tidak boleh disertai dengan nama tabel / schema,
- Expr jika digunakan, hanya berlaku jika expr di pakai pada perintah SQL,

- ON ROW jika digunakan, maka break terjadi setiap kali terjadi penamabahan baris,
- ON REPORT biasanya digunakan pada kalkulasi keseluruhan data,
- SKIP n, digunakan untuk baris kosong sebanyak n baris,
- SKIP PAGE, hasil query berikutnya akan dicetak pada halaman baru,
- NODUP[LICATES] digunakan jika data duplikasi tidak ingin dicetak,
- DUP[LICATES] adalah kebalikan dari NODUP[LICATES].

Contoh penggunaan :

SQL > break on sal skip 1
SQL > select empno,ename,sal from emp order by sal asc;

g. COMPUTE

Fungsi COMPUTE digunakan untuk melakukan kalkulasi pada kolom.

Bentuk Klausa :

COMPUTE [function [LAB[EL] text] OF <expr | column > ON <expr | column | REPORT | ROW >]

Keterangan

- Function yang dapat digunakan adalah :

Function	Keterangan	Tipe Data
AVG	Nilai rata – rata. Baris yang diperhitungkan hanya baris dengan nilai yang tidak NULL	number
COU[NT]	Menghitung jumlah baris yang nilai column atau expr-nya tidak null	Semua
MIN[IMUM]	Nilai minimum	Number, char, varchar2
MAX[IMUM]	Nilai maksimum	Number, Char, Varchar2
NUM[BER]	Menghitung jumlah baris, termasuk yang bernilai NULL.	Semua
SUM	Jumlah nilai	Number
STD	Standar Deviasi dari nilai column atau expr yang tidak null	Number
VAR[IANCE]	Variance dari nilai column atau expr yang tidak NULL	Number

- LABEL text dapat digunakan sebagai hasil dari sebuah kalkulasi.Jika diberikan, default text adalah nama fungsi.
- OF untuk menentukan expr atau kolom apa yang dikalkulasi.
- ON digunakan untuk menentukan kriteria BREAK untuk satu perhitungan COMPUTE.

Contoh penggunaan :

```
SQL > BREAK ON empno SKIP 1
SQL > COMPUTE avg max min Of sal ON empno
SQL > SELECT empno,ename,sal FROM emp ORDER BY empno;
```

h. SET

Digunakan untuk menentukan nilai sebuah variabel. Contoh :

```
SQL > SET HEADING OFF;
```

i. SHOW

Digunakan untuk melihat sebuah nilai variabel.Contoh :

```
SQL > SHOW HEADING;
```

6.2 MENGATUR TAMPILAN LAYAR

a. COLUMN

Beberapa atribut dibawah ini berkaitan dengan format kolom yang akan ditampilkan pada layar :

Klausa	Keterangan
ALI[AS] alias	Memberi nama alias yang dapat digunakan untuk mengantikan nama column pada perintah BREAK,COMPUTE ataupun COLUMN.
CLE[AR]	Mengembalikan atribut column ke nilai defaultnya sesuai dengan tipe data, panjang data, dan sebagainya.
FOLD_A[FTER]	Menambahkan karakter baris baru (carriage return) setelah judul *column* dan setelah nilai *column* dari setiap baris. Tidak berlaku untuk kolom terakhir pada klausa SELECT.
FOLD_B[EFORE]	Menambahkan karakter baris baru (carriage return) setelah judul column dan setelah nilai column dari setiap baris. Tidak berlaku untuk kolom terakhir pada klausa SELECT.
FOR[MAT] format	Menentukan format tampilan nilai column. Format untuk kolom karakter berupa 'An' di mana n adalah lebar kolom. Format untuk kolom DATE mengikuti NLS_DATE_FORMAT namun lebar kolomnya dapat ditentukan dengan 'An'. Format untuk kolom NUMBER sama dengan format yang digunakan pada fungsi TO_CHAR (NUMBER).
HEA[DING] text	Menentukan text sebagai judul kolom untuk menggantikan judul kolom yang diberikan di dalam SELECT statement. String karakter text dapat mengandung karakter HEADSEP untuk menampilkan judul kolom dalam dua baris atau lebih.
JUS[TIFY] <L[EFT] | C[ENTER] | R[IGHT] >	Menentukan apakah judul kolom dan nilai kolom ditampilkan rata kiri, di tengah – tengah atau rata kanan. Jika tidak disebutkan maka kolom NUMBER ditampilkan rata kanan dan kolom lainnya rata kiri.
LIKE [expr | alias]	Menyalin atribut tampilan dari kolom lain. Atribut yang disalin hanya atribut yang tidak ditentukan secara eksplisit.
NEWL[INE]	Sama efeknya dengan FOLD_BEFORE, yakni menampilkan baris barus sebelum nilai column ditampilkan.
NEW_V[ALUE] variabel	Digunakan untuk membuat variabel yang nilainya adalah nilai kolom. Variabel tersebut biasanya digunakan pada perintah TTITLE untuk membuat judul atas pada laporan master detail. Agar pengaruhnya terlihat, kolom pada perintah ini harus digunakan juga pada perintah "BREAK…SKIP PAGE".

NOPRI[NT] | PRI[NT]	Mengatur apakah judul dan nilai column dicetak di layar [ON] atau tidak [OFF]. Default adalah OFF.
NUL[L] text	Menentukan text sebagai nilai yang dicetak pada kolom yang ditentukan.
OLD_V[ALUE] variable	Digunakan untuk membuat variabel yang nilainya adalah nilai kolom.Variabel tersebut biasanya digunakan pada perintah BTITLE untuk membuat judul – bawah pada laporan master detail. Kolom pada perintah ini harus digunakan juga pada perintah "BREAK…SKIP PAGE".
ON | OFF	Mengaktifkan (ON) atau menonaktifkan (OFF) atribut – atribut kolom yang sudah didefinisikan.
WRAP[PPED] | WOR[D_WRAPPED |TRU[NCATE]	Perintah ini khusus untuk kolom DATE atau karakter yang lebarnya ditentukan namun nilai kolom tersebut melebihi lebar kolom yang disediakan. Pilihan WRAPPED akan melanjutkan bagian yang tidak mencukupi ke baris yang baru pada kolom yang sama. Pilihan WORD_WRAPPED akan melanjutkan kata – kata yang tidak mencukupi ke baris yang baru pada kolom yang sama. Pilihan TRUNCATE akan memotong bagian yang tidak mencukupi.

Contoh penggunaan :

SQL > COLUMN Karyawan FORMAT A30 HEADING "NAMA KARYAWAN"
SQL > SELECT empno "KARYAWAN" FROM emp;

b. CLEAR

Fungsi Clear digunakan untuk menghapus perintah SQL yang ada pada memory.

Bentuk klausa :

CL[EAR] BRE[AKS] | BUFF[ER] | COL[UMNS] |SCR[EEN] | SQL | TIMI [NG] | COMPUTES

Contoh penggunaan :

SQL > CL BUFF;

c. TTITLE dan BTITLE

Digunakan untuk memberi judul atas atau judul bawah pada sebuah query.

Bentuk Klausa :

TITLE [printspec [text] | variabel] ON | OFF
BTITLE [printspec [text] | variabel] ON | OFF

Komponen printspec yang dapag digunakan :

PrintSpec	Keterangan
COL n	Menentukan posisi cetak dimulai dari kolom ke n.
S[KIP] [n]	Memulai baris baru n baris berikutnya dari baris sebelumnya. Jika n tidak disertakan, n=1. Jika n=0, text atau variabel dicetak pada baris berikutnya.
TAB n	Menambahkan tab maju sebanyak n. Jika n negatif, yang dicetak adalah tab mundur.
LE[FT] | CE[NTER] | R[IGHT]	Menentukan apakah judul dicetak rapat kiri, tengah, atau kanan.
BOLD	Mencetak text dengan huruf tebal.
FORMAT format	Menentukan format. Format yang valid adalah format yang berlaku untuk perintah COLUMN.

Sedangkan komponen parameter yang dapat digunakan adalah

Parameter	Keterangan
SQL.LNO	Untuk menampilkan nomor dari current line.
SQL.PNO	Untuk menampilkan nomor dari current page.
SQL.RELEASE	Untuk menampilkan nomor release Oracle.
SQL. ERRORCODE	Untuk menampilkan kode kesalahan.
SQL.USER	Untuk menampilkan nama current user.

Contoh :

```
SQL > TTITLE CENTER "DAFTAR DEPARTEMEN" SKIP 1 –
SQL > RIGHT "Hal : " FORMAT 999 SQL.NO –
SQL > LEFT "Pengguna  : " SQL.USER SKIP " "

SQL > BTITLE HEADER  **** Akhir Halaman *****
```

d. REPHEADER dan REPFOOTER

Fungsi REPHEADER dan REPFOOTER hampir sama dengan fungsi TTITLE dan BTITLE. Perbedaannya, REPHEADER dan REPFOOTER hanya dicetak sekali saja pada perintah SELECT.

Bentuk Klausa :

```
REPH [EADER] [ PAGE ] [ printspec [text] | variabel ] [ ON | OFF ]
REPF [OOTER] [PAGE ]  [ printspec [text] | variabel ] [ ON | OFF ]
```

Contoh :

SQL > REPHEADER CENTER "Awal Laporan"
SQL > REPFOOTER CENTER "Akhir Laporan"

e. MENGATUR LAYOUT

Beberapa parameter yang dapat digunakan untuk mengatur format laporan pada layar.

Parameter	Keterangan
COLSEP text	Menentukan text pembatas antara kolom yang dipilih. Default adalah spasi tunggal.
HEA[DING] < ON | OFF >	Judul kolom dicetak jika nilai ON (default). Tidak dicetak jika OFF.
HEADS[EP] < c | ON | OFF >	Menentukan karakter pembatas judul kolom. Default c adalah karakter garis vertikal ("|"). Jika ON, karakter HEADSEP dikembalikan ke default dan diaktifkan. Jika OFF, karakter HEADSET menjadi tidak aktif. Karakter HEADSEP biasa digunakan bersama perintah "COLUMN….HEADING text ".
LIN[SIZE] n	Menentukan jumlah karakter maksimum dalam satu baris.Nilai Default adalah 80.
NEWP[AGE] < n | NONE >	Menentukan jumlah baris kosong sebelum judul atas. Nilai default adalah 1.
NULL text	Nilai NULL dicetak sebagai text untuk semua kolom yang dipilih pada perintah SELECT. Apabila perintah "COLUMN….NULL.." diberikan pada kolom tertentu maka nilai text ini digantikan oleh text pada perintah COLUMN untuk kolom yang disebutkan pada perintah COLUMN itu saja.
NUMF[ORMAT] format	Menentukan format cetak untuk tipe data NUMBER
NUM[WIDTH] n	Menentukan lebar kolom untuk tipe data NUMBER. Nilai default adalah 10.
PAGES[IZE] n	Menentukan angka maksimum dari jumlah baris dalam satu halaman. Nilai default adalah 14. Jika n=0, judul atas halaman, judul kolom, dan nilai yang lainnya tidak dicetak.
RECSEP < WR[APPED] | EA[CH] | OFF >	Menentukan apakah karakter pemisah record (RECSEPCHAR) dicetak atau tidak. Pilihan WRAPPED berarti pemisah record dicetak hanya jika ada baris data yang panjangnya lebih dari satu baris [wrapped]. Pilihan EACH berarti dicetak untuk setiap baris data. OFF berarti tidak dicetak sama sekali. Default adalah WRAPPED.

RECSEPCHAR c	Menentukan karakter c sebagai pemisah record. Dicetak tidaknya pemisah record ini ditentukan oleh parameter RECSEP. Default c adalah spasi tunggal.
UNDERLINE < c | ON | OFF >	Menentukan c sebagai garis bawah dari judul kolom. ON untuk mengembalikan nilai c menjadi default., yaitu tanda sambung (-). OFF berarti garis bawah tidak dicetak.

Gunakan perintah SET untuk mensetup parameter. Untuk menampilkan parameter gunakan perintah SHOW.

f. PARAMETER UNTUK EFESIENSI OUTPUT

Beberapa paramater yang dapat digunakan untuk membantu penampilan layar agar lebih baik.

Parameter	Keterangan
SET ECHO < ON | OFF >	Menentukan apakah perintah yang ada pada file script yang sedang dijalankan (termasuk perintah SQL dan blok SQL)
SET TERMOUT < ON | OFF >	Menentukan apakah keluaran perintah yang ada pada file script yang sedang dijalankan diperlihatkan pada layar (ON) atau tidak (OFF). Default adalah ON.
SET TRIMOUT < ON | OFF >	Menentukan apakah spasi atau blank pada akhir baris dari sebuah perintah di SQL*Plus diabaikan (ON) atau tidak (OFF). Default ON.
SET VERIFY < ON | OFF >	Menentukan apakah baris perintah SQL atau PL / SQL yang mengandung substitution variable diperlihatkan (ON) atau tidak (OFF). Jika ON, baris yang diperlihatkan adalah sebelum dan setelah variabel diberi nilai, dan ini menjadi default.

Daftar Pustaka

Alan Denis, Barbara Haley Wixon, David Tagerden, *System Analys and Design : an Object – Oriented Approach with UML 2.0*, John Willey and Sons, 2005

Henry C. Lucas Jr., *The Analysis, Design and Implementation of Information Systems 4th*, McGraw Hill, 1992

Satzinger, Jackson, Burd, *Object-Oriented Analysis and Design with the Unified Process*, Course Technology, 2005

Simon Bennet, Steve McRobb and Ray Farmer, *Object-Oriented System Analysis and Design Using UML*, McGraw Hill, 2006

Wendy and Michael Boggs, *UML with Rational Rose 2002*, Sibex Inc., 2002

UNIVERSITAS MERCU BUANA

MODUL PERKULIAHAN

PEMROGRAMAN BASIS DATA DAN SQL

Modul Standar untuk digunakan dalam Perkuliahan di Universitas Mercu Buana

Fakultas	Program Studi	Tatap Muka	Kode MK	Disusun Oleh
Ilmu Komputer	Sistem Informasi	08	87007	Team Dosen

Abstract

SDLC, RAD, Design Tersturktur, Class dan Object

Kompetensi

Memahami siklus pengembangan sistem, memahami metodhologi pengembangan sistem, mengenal UML

Data Definition Languange (DDL)

Data Definition Language (DDL)

Data Definition Language (DDL) adalah bahasa SQL yang digunakan untuk mendefinisikan, menghapus, memberi hak akses ataupun menganalisa objek – objek database.

A. MEMBUAT TABEL

Definisi tabel adalah sebagai berikut :

```
CREATE TABLE nama tabel
    ( nama kolom   tipe data DEFAULT ekspresi ,
      nama kolom   tipe data DEFAULT ekspresi,
      ……..                                  )
    PCTFREE number   PCTUSED number
    STORAGE ( INITIAL   number
              NEXT      number
              PCTINCREASE number
              MINEXTENTS number
              MAXEXTENTS number )
    TABLESPACES  nama tablespace;
```

Keterangan

No.	Menu	Keterangan
01.	Default	Untuk mengisi data default
02.	PCTFREE	Jumlah tempat yang diberikan untuk setiap blok pada baris untuk pertumbuhan panjang. Berikan nilai PCTFREE rendah jika penggunakan update rendah.
03.	PCTUSED	Menentukan batas bawah dari tempat yang digunakan dalam blok sebelum digunakan untuk memasukkan baris (insert row). Berikan nilai PCTUSED jika aktivitas penambahan dan penghapusan tinggi.
04.	STORAGE	Mendefinisikan untuk menentukan bagaimana penambahan penyimpanan yang dialokasikan untuk tabel.
05.	INITIAL	Mendefinisikan size untuk extent pertama dalam bytes.
06.	NEXT	Mendefinisikan size untuk next extent dalam bytes.
07.	MINEXTENTS	Permulaan extent di buat
08.	MAXEXTENTS	Total extent yang dialokasikan
09.	PCTINCREASE	Menentukan dasar pertumbuhan untuk extent berikutnya.
10.	TABLESPACES	Tempat di mana table akan dibuat

Contoh

```
CREATE TABLE employee (
empid                   varchar2(10) not null,
empnm                   varchar2(30) default 'kosong',
salary                  number(5,2))
PCTFREE 5               PCTUSED 65
STORAGE  (  INITIAL  5M
            NEXT     5M
            PCTINCREASE 0
            MINEXTENTS 2
            MAXEXTENTS 50)
TABLESPACE users;
```

B. CONSTRAINT

Definisi pembuatan CONSTRAINT dapat dilakukan pada saat pembuatan tabel atau pada saat merubah struktur tabel.

Adapun definisi pada saat pembuatan tabel adalah

```
CREATE TABLE nama tabel

( nama kolom    tipe data  DEFAULT ekspresi,

  nama kolom    tipe data  DEFAULT  ekspresi,

  CONSTRAINT  nama constraint tipe constraint

);
```

Sedangkan definisi pada saat merubah struktur tabel adalah

```
ALTER TABLE nama tabel ADD nama constraint  ekspresi constraint
```

Di dalam CONSTRAINT ada tipe – tipe constraint yang dapat digunakan

- NOT NULL
  Tipe constraint ini hanya dapat digunakan pada kolom.

Definisi : [**CONSTRAINT** nama constraint] **NOT NULL**;

Contoh :

ALTER TABLE emp **MODIFY** address varchar2(100) **NOT NULL**;

- **UNIQUE**

Definisi : [**CONSTRAINT** nama constraint] **UNIQUE** (kolom 1, kolom 2, …)]

Contoh :

ALTER TABLE emp **ADD CONSTRAINT** c_1 **UNIQUE** (empnm,address);

- **PRIMARY KEY**

Definisi : [CONSTRAINT nama constraint PRIMARY KEY (kolom 1, kolom 2,..)]

Contoh :

ALTER TABLE emp **ADD CONSTRAINT** c_2 **PRIMARY KEY** (empid);

- **FOREIGN KEY**

Definisi : [**CONSTRAINT** nama constraint **FOREIGN KEY** nama kolom **REFERENCES** nama tabel (nama kolom 1 **) ON DELETE CASCADE | SET NULL**]

Contoh :

ALTER TABLE emp **ADD CONSTRAINT** c_3 **FOREIGN KEY** dept_id **REFERENCES** dept (dept_id);

- **CHECK**

Definisi [CONSTRAINT nama constraint CHECK (kondisi)

Kondisi di sini dapat berupa kondisi Perbandingan, kondisi BETWEEN, kondisi NULL, kondisi LIKE, kondisi EXISTS

Contoh :

ALTER TABLE emp **ADD CONSTRAINT** c_4 **CHECK** (salary > 0);

PRAKTEK 1:

- **Soal :**
  Buatlah Tabel Barang, Supplier, Customer, Pembelian, Pasok

- **Program SQL :**

```
Drop table Pasok;
Drop table Pembelian;
Drop table Barang;
Drop table Supplier;
Drop table Customer;

create table Barang
        (Kode_Barang char (6),
        Nama_Barang varchar2 (25),
        Satuan_Barang varchar2 (20),
        Stok_Barang number (4),
        constraint pk_Barang primary key (Kode_Barang)
        );

create table Supplier
        (Kode_Supplier char (5),
        Nama_Supplier varchar2 (30),
        Alamat_Supplier varchar2 (30),
        Kota_Supplier varchar2 (15),
        Telepon_Supplier varchar2 (15),
        constraint pk_Supplier primary key (Kode_Supplier)
        );

create table Customer
        (Kode_Customer char (6),
        Nama_Customer varchar2 (30),
        Alamat_Customer varchar2 (30),
        Kota_Customer varchar2 (15),
        Telepon_Customer varchar2 (15),
```

```
        Constraint pk_Customer primary key (Kode_Customer)
        );

create table Pasok
        (Kode_Pasok char (9),
        Kode_Barang char (6),
        Kode_Supplier char (5),
        Tanggal_Pasok date,
        Jumlah_Pasok number (4),
        constraint pk_Pasok primary key (Kode_Pasok,Kode_Barang,Kode_Supplier),
        constraint fk_Pasok_Barang foreign key (Kode_Barang) references Barang
(Kode_Barang),
        constraint fk_Pasok_Supplier foreign key (Kode_Supplier) references Supplier
(Kode_Supplier)
        );
create table pembelian
        (Kode_Pembelian char (9),
        Kode_Barang char (6),
        Kode_Customer char (6),
        Tanggal_Pembelian date,
        Jumlah_Pembelian number (4),
        constraint pk_Pembelian primary key
(Kode_Pembelian,Kode_Barang,Kode_Customer),
        constraint fk_Pembelian_barang foreign key (Kode_Barang) references Barang
(Kode_Barang),
        constraint fk_Pembelian_customer foreign key (Kode_Customer) references
Customer (Kode_Customer)
        );
```

- **Tampilan :**

```
SQL> desc Barang;
 Name                                      Null?    Type
 ----------------------------------------- -------- ------------------------------
 KODE_BARANG                               NOT NULL CHAR(6)
 NAMA_BARANG                                        VARCHAR2(25)
 SATUAN_BARANG                                      VARCHAR2(20)
 STOK_BARANG                                        NUMBER(4)

SQL> desc Supplier;
 Name                                      Null?    Type
 ----------------------------------------- -------- ------------------------------
 KODE_SUPPLIER                             NOT NULL CHAR(5)
 NAMA_SUPPLIER                                      VARCHAR2(30)
 ALAMAT_SUPPLIER                                    VARCHAR2(30)
 KOTA_SUPPLIER                                      VARCHAR2(15)
 TELEPON_SUPPLIER                                   VARCHAR2(15)

SQL> desc Customer;
 Name                                      Null?    Type
 ----------------------------------------- -------- ------------------------------
 KODE_CUSTOMER                             NOT NULL CHAR(6)
 NAMA_CUSTOMER                                      VARCHAR2(30)
 ALAMAT_CUSTOMER                                    VARCHAR2(30)
 KOTA_CUSTOMER                                      VARCHAR2(15)
 TELEPON_CUSTOMER                                   VARCHAR2(15)

SQL> desc Pasok;
 Name                                      Null?    Type
 ----------------------------------------- -------- ------------------------------
 KODE_PASOK                                NOT NULL CHAR(9)
 KODE_BARANG                               NOT NULL CHAR(6)
 KODE_SUPPLIER                             NOT NULL CHAR(5)
 TANGGAL_PASOK                                      DATE
 JUMLAH_PASOK                                       NUMBER(4)

SQL> desc Pembelian;
 Name                                      Null?    Type
 ----------------------------------------- -------- ------------------------------
 KODE_PEMBELIAN                            NOT NULL CHAR(9)
 KODE_BARANG                               NOT NULL CHAR(6)
 KODE_CUSTOMER                             NOT NULL CHAR(6)
 TANGGAL_PEMBELIAN                                  DATE
 JUMLAH_PEMBELIAN                                   NUMBER(4)
```

Daftar Pustaka

Alan Denis, Barbara Haley Wixon, David Tagerden, *System Analys and Design : an Object – Oriented Approach with UML 2.0*, John Willey and Sons, 2005

Henry C. Lucas Jr., *The Analysis, Design and Implementation of Information Systems 4th*, McGraw Hill, 1992

Satzinger, Jackson, Burd, *Object-Oriented Analysis and Design with the Unified Process*, Course Technology, 2005

Simon Bennet, Steve McRobb and Ray Farmer, *Object-Oriented System Analysis and Design Using UML*, McGraw Hill, 2006

Wendy and Michael Boggs, *UML with Rational Rose 2002*, Sibex Inc., 2002

UNIVERSITAS
MERCU BUANA

MODUL PERKULIAHAN

PEMROGRAMAN BASIS DATA DAN SQL

Modul Standar untuk digunakan dalam Perkuliahan di Universitas Mercu Buana

Fakultas	Program Studi	Tatap Muka	Kode MK	Disusun Oleh
Ilmu Komputer	Sistem Informasi	09	87007	Team Dosen

Abstract

SDLC, RAD, Design Tersturktur, Class dan Object

Kompetensi

Memahami siklus pengembangan sistem, memahami metodhologi pengembangan sistem, mengenal UML

Data Manipulation Languange (DML)

Praktek 2

- **Data Manipulation language (DML)**
  - ***Insert***

➢ **Soal :**
Masukkan data pada Tabel Barang, Customer, Supplier, Pembelian, Pasok

➢ **Program :**

```
insert into Barang values ('ELK-02','Lemari Es','Unit','8');

insert into Barang values ('ELK-01','Rice cooker','Buah','20');

insert into Barang values ('ELK-03','Televisi','','');

insert into Barang values ('ELK-04','Radio/Tape','Buah','35');


insert into Supplier values ('EJ-01','PT.Acron','Jl.Thamrin 12','Jakarta','(021)8551048');

insert into Supplier values ('EJ-02','PT.Mulya','Jl.Sudirman 10','Jakarta','(021)5430146');

insert into Supplier values ('EB-01','PT.UltraSound','Jl.Sukarno 21','Bandung','(022)641347');

insert into Supplier values ('EB-02','PT.Supertron','Jl.Industri 37','Bandung','(022)660110');


insert into Customer values ('J-001','Toko Karisma','Jl.Cimanggis  5','Jakarta','(021)8761481');

insert into Customer values ('J-002','Toko Ayu','Jl.Depok 6','Jakarta','(021)77141');

insert into Customer values ('B-001','Toko Surya','Jl.ABC 27','Bandung','(022)515411');

insert into Customer values ('B-002','Toko Warna','Jl.XYZ 30','Bandung','(022)646121');


insert into Pasok values ('PAS-E001','ELK-01','EJ-01','01-jan-02','8');

insert into Pasok values ('PAS-E002','ELK-01','EJ-02','01-jan-02','5');
```

```
insert into Pasok values ('PAS-E003','ELK-02','EJ-01','01-feb-02','2');

insert into Pasok values ('PAS-E004','ELK-02','EJ-02','02-feb-02','3');

insert into Pasok values ('PAS-E005','ELK-02','EB-02','01-jan-02','2');

insert into Pasok values ('PAS-E006','ELK-03','EJ-01','03-Mar-02','5');

insert into Pasok values ('PAS-E007','ELK-03','EJ-01','04-Mar-02','2');

insert into Pasok values ('PAS-E008','ELK-03','EJ-02','03-Mar-02','3');

insert into Pasok values ('PAS-E009','ELK-03','EB-01','13-Mar-02','4');

insert into Pasok values ('PAS-E010','ELK-03','EB-02','13-Mar-02','3');

insert into Pasok values ('PAS-E011','ELK-04','EB-01','22-Apr-02','12');

insert into Pasok values ('PAS-E012','ELK-04','EB-02','30-jan-02','9');


insert into Pembelian values ('Bel-E001','ELK-01','J-001','20-May-02','3');

insert into Pembelian values ('Bel-E002','ELK-01','J-001','21-May-02','4');

insert into Pembelian values ('Bel-E003','ELK-01','J-002','20-May-02','2');

insert into Pembelian values ('Bel-E004','ELK-01','B-001','20-May-02','2');

insert into Pembelian values ('Bel-E005','ELK-01','B-002','22-May-02','3');

insert into Pembelian values ('Bel-E006','ELK-02','J-001','24-Jun-02','1');

insert into Pembelian values ('Bel-E007','ELK-02','J-002','24-Jun-02','1');

insert into Pembelian values ('Bel-E008','ELK-02','B-001','25-Jun-02','2');

insert into Pembelian values ('Bel-E009','ELK-02','B-002','25-Jun-02','2');

insert into Pembelian values ('Bel-E010','ELK-03','J-001','20-Jun-02','5');

insert into Pembelian values ('Bel-E011','ELK-03','J-002','02-Jul-02','4');

insert into Pembelian values ('Bel-E012','ELK-03','B-001','04-Jul-02','6');

insert into Pembelian values ('Bel-E013','ELK-03','B-001','10-Jul-02','5');

insert into Pembelian values ('Bel-E014','ELK-04','J-002','15-Jul-02','12');
```

insert into Pembelian values ('Bel-E015','ELK-04','B-002','17-Jul-02','15');

- **Tampilan :**

```
SQL> select * from Barang;

KODE_B NAMA_BARANG            SATUAN_BARANG       STOK_BARANG
------ ---------------------- ------------------- -----------
ELK-02 Lemari Es              Unit                          8
ELK-01 Rice cooker            Buah                         20
ELK-03 Televisi
ELK-04 Radio/Tape             Buah                         35

SQL> select * from Supplier;

KODE_ NAMA_SUPPLIER               ALAMAT_SUPPLIER             KOTA_SUPPLIER   TELEPON_SUPPLIE
----- --------------------------- --------------------------- --------------- ---------------
EJ-01 PT.Acron                    Jl.Thamrin 12               Jakarta         (021)8551048
EJ-02 PT.Mulya                    Jl.Sudirman 10              Jakarta         (021)5430146
EB-01 PT.UltraSound               Jl.Sukarno 21               Bandung         (022)641347
EB-02 PT.Supertron                Jl.Industri 37              Bandung         (022)660110

SQL> select * from Customer;

KODE_C NAMA_CUSTOMER               ALAMAT_CUSTOMER              KOTA_CUSTOMER   TELEPON_CUSTOME
------ --------------------------- ---------------------------- --------------- ---------------
J-001  Toko Karisma                Jl.Cimanggis 15              Jakarta         (021)8761481
J-002  Toko Ayu                    Jl.Depok 6                   Jakarta         (021)77141
B-001  Toko Surya                  Jl.ABC 27                    Bandung         (022)515411
B-002  Toko Warna                  Jl.XYZ 30                    Bandung         (022)646121
```

```
SQL> select * from Pasok;

KODE_PASO KODE_B KODE_ TANGGAL_P JUMLAH_PASOK
--------- ------ ----- --------- ------------
PAS-E001  ELK-01 EJ-01 01-JAN-02             8
PAS-E002  ELK-01 EJ-02 01-JAN-02             5
PAS-E003  ELK-02 EJ-01 01-FEB-02             2
PAS-E004  ELK-02 EJ-02 02-FEB-02             3
PAS-E005  ELK-02 EB-02 01-JAN-02             2
PAS-E006  ELK-03 EJ-01 03-MAR-02             5
PAS-E007  ELK-03 EJ-01 04-MAR-02             2
PAS-E008  ELK-03 EJ-02 03-MAR-02             3
PAS-E009  ELK-03 EB-01 13-MAR-02             4
PAS-E010  ELK-03 EB-02 13-MAR-02             3
PAS-E011  ELK-04 EB-01 22-APR-02            12

KODE_PASO KODE_B KODE_ TANGGAL_P JUMLAH_PASOK
--------- ------ ----- --------- ------------
PAS-E012  ELK-04 EB-02 30-JAN-02             9

12 rows selected.

SQL> select * from Pembelian;

KODE_PEMB KODE_B KODE_C TANGGAL_P JUMLAH_PEMBELIAN
--------- ------ ------ --------- ----------------
Bel-E001  ELK-01 J-001  20-MAY-02                3
Bel-E002  ELK-01 J-001  21-MAY-02                4
Bel-E003  ELK-01 J-002  20-MAY-02                2
Bel-E004  ELK-01 B-001  20-MAY-02                2
Bel-E005  ELK-01 B-002  22-MAY-02                3
Bel-E006  ELK-02 J-001  24-JUN-02                1
Bel-E007  ELK-02 J-002  24-JUN-02                1
Bel-E008  ELK-02 B-001  25-JUN-02                2
Bel-E009  ELK-02 B-002  25-JUN-02                2
Bel-E010  ELK-03 J-001  20-JUN-02                5
Bel-E011  ELK-03 J-002  02-JUL-02                4

KODE_PEMB KODE_B KODE_C TANGGAL_P JUMLAH_PEMBELIAN
--------- ------ ------ --------- ----------------
Bel-E012  ELK-03 B-001  04-JUL-02                6
Bel-E013  ELK-03 B-001  10-JUL-02                5
Bel-E014  ELK-04 J-002  15-JUL-02               12
Bel-E015  ELK-04 B-002  17-JUL-02               15

15 rows selected.
```

Praktek 3

- ***Membuat view***

  - **Soal :**
    Membuat view dari tabel Customer

  - **Program dan Tampilan :**

```
SQL> create view v_customer as
  2  select * from Customer;

View created.

SQL> select * from v_customer;

KODE_C NAMA_CUSTOMER                ALAMAT_CUSTOMER               KOTA_CUSTOMER   TELEPON_CUSTOME
------ ---------------------------- ----------------------------- --------------- ---------------
J-001  Toko Karisma                 Jl.Cimanggis 15               Jakarta         (021)8761481
J-002  Toko Ayu                     Jl.Depok 6                    Jakarta         (021)77141
B-001  Toko Surya                   Jl.ABC 27                     Bandung         (022)515411
B-002  Toko Warna                   Jl.XYZ 30                     Bandung         (022)646121
```

  - **Soal :**
    Membuat view dari tabel Barang. Tapi yang ditampilkan hanya Nama Barang dan Stok barang

  - **Program dan Tampilan :**

```
SQL> create view v_barang as
  2  select Nama_Barang,Stok_Barang from Barang;

View created.

SQL> select * from v_barang;

NAMA_BARANG                STOK_BARANG
------------------------- -----------
Lemari Es                           8
Rice cooker                        20
Televisi
Radio/Tape                         35
```

❖ Pengelompokkan data (group by) dan Aggregat function

- **Soal :**
  Tampilkan minimum pembelian, maksimum pembelian, rata-rata pembelian, jumlah pembelian dari tabel Pembelian berdasarkan kode barang.

- **Program dan Tampilan :**

```
SQL> select Kode_Barang as kd_brg,min(Jumlah_Pembelian) as minimum_pembelian,max(Jumlah_Pembelian) a
s maksimum_pembelian,avg(Jumlah_Pembelian) as average_pembelian,count(Jumlah_Pembelian) as count_pem
belian,sum(Jumlah_Pembelian) as sum_pembelian from Pembelian
  2  group by Kode_Barang;

KD_BRG MINIMUM_PEMBELIAN MAKSIMUM_PEMBELIAN AVERAGE_PEMBELIAN COUNT_PEMBELIAN SUM_PEMBELIAN
------ ----------------- ----------------- ----------------- --------------- -------------
ELK-01                 2                 4               2.8               5            14
ELK-02                 1                 2               1.5               4             6
ELK-03                 4                 6                 5               4            20
ELK-04                12                15              13.5               2            27
```

- **Soal :**

Tampilkan minimum pasok, maksimum pasok, rata-rata pasok, jumlah pasok dari tabel Pasok berdasarkan kode barang.

- **Program dan Tampilan :**

```
SQL> select Kode_Barang as kd_brg,min(Jumlah_Pasok) as minimum_pasok,max(Jumlah_Pasok) as maksimum_p
asok,avg(Jumlah_Pasok) as average_pasok,count(Jumlah_Pasok) as count_pasok,sum(Jumlah_Pasok) as sum_
pasok from Pasok
  2  group by Kode_Barang;

KD_BRG MINIMUM_PASOK MAKSIMUM_PASOK AVERAGE_PASOK COUNT_PASOK  SUM_PASOK
------ ------------ ------------- ------------ ----------- ----------
ELK-01            5             8          6.5           2         13
ELK-02            2             3   2.33333333           3          7
ELK-03            2             5          3.4           5         17
ELK-04            9            12         10.5           2         21
```

Daftar Pustaka

Alan Denis, Barbara Haley Wixon, David Tagerden, *System Analys and Design : an Object – Oriented Approach with UML 2.0*, John Willey and Sons, 2005

Henry C. Lucas Jr., *The Analysis, Design and Implementation of Information Systems 4th*, McGraw Hill, 1992

Satzinger, Jackson, Burd, *Object-Oriented Analysis and Design with the Unified Process*, Course Technology, 2005

Simon Bennet, Steve McRobb and Ray Farmer, *Object-Oriented System Analysis and Design Using UML*, McGraw Hill, 2006

Wendy and Michael Boggs, *UML with Rational Rose 2002*, Sibex Inc., 2002

UNIVERSITAS
MERCU BUANA

MODUL PERKULIAHAN

PEMROGRAMAN BASIS DATA DAN SQL

Modul Standar untuk digunakan dalam Perkuliahan di Universitas Mercu Buana

Fakultas	Program Studi	Tatap Muka	Kode MK	Disusun Oleh
Ilmu Komputer	Sistem Informasi	10	87007	Team Dosen

Abstract

SDLC, RAD, Design Tersturktur, Class dan Object

Kompetensi

Memahami siklus pengembangan sistem, memahami metodhologi pengembangan sistem, mengenal UML

Data Manipulation Languange (DML)

Praktek 4

❖ Pengurutan data (order by)

- **Soal :**
  Tampilkan semua data customer, tampilan diurut berdasarkan nama customer!

- **Program dan Tampilan :**

```
SQL> select * from Customer
  2  order by Nama_Customer;

KODE_C NAMA_CUSTOMER                ALAMAT_CUSTOMER              KOTA_CUSTOMER   TELEPON_CUSTOME
------ ---------------------------- ---------------------------- --------------- ---------------
J-002  Toko Ayu                     Jl.Depok 6                   Jakarta         (021)77141
J-001  Toko Karisma                 Jl.Cimanggis 15              Jakarta         (021)8761481
B-001  Toko Surya                   Jl.ABC 27                    Bandung         (022)515411
B-002  Toko Warna                   Jl.XYZ 30                    Bandung         (022)646121
```

- **Soal** :
  Tampilkan data pasok berdasarkan urutan jumlah pasok!

- **Program dan Tampilan :**

```
SQL> select * from Pasok
  2  order by Jumlah_Pasok;

KODE_PASO KODE_B KODE_ TANGGAL_P JUMLAH_PASOK
--------- ------ ----- --------- ------------
PAS-E003  ELK-02 EJ-01 01-FEB-02             2
PAS-E005  ELK-02 EB-02 01-JAN-02             2
PAS-E007  ELK-03 EJ-01 04-MAR-02             2
PAS-E004  ELK-02 EJ-02 02-FEB-02             3
PAS-E008  ELK-03 EJ-02 03-MAR-02             3
PAS-E010  ELK-03 EB-02 13-MAR-02             3
PAS-E009  ELK-03 EB-01 13-MAR-02             4
PAS-E002  ELK-01 EJ-02 01-JAN-02             5
PAS-E006  ELK-03 EJ-01 03-MAR-02             5
PAS-E001  ELK-01 EJ-01 01-JAN-02             8
PAS-E012  ELK-04 EB-02 30-JAN-02             9

KODE_PASO KODE_B KODE_ TANGGAL_P JUMLAH_PASOK
--------- ------ ----- --------- ------------
PAS-E011  ELK-04 EB-01 22-APR-02            12

12 rows selected.
```

- **Soal** :

  Tampilkan data pasok berdasarkan urutan jumlah pasok secara menurun!

- **Program dan Tampilan :**

```
SQL> select * from Pasok
  2  order by Jumlah_Pasok desc;

KODE_PASO KODE_B KODE_ TANGGAL_P JUMLAH_PASOK
--------- ------ ----- --------- ------------
PAS-E011  ELK-04 EB-01 22-APR-02            12
PAS-E012  ELK-04 EB-02 30-JAN-02             9
PAS-E001  ELK-01 EJ-01 01-JAN-02             8
PAS-E002  ELK-01 EJ-02 01-JAN-02             5
PAS-E006  ELK-03 EJ-01 03-MAR-02             5
PAS-E009  ELK-03 EB-01 13-MAR-02             4
PAS-E004  ELK-02 EJ-02 02-FEB-02             3
PAS-E010  ELK-03 EB-02 13-MAR-02             3
PAS-E008  ELK-03 EJ-02 03-MAR-02             3
PAS-E003  ELK-02 EJ-01 01-FEB-02             2
PAS-E005  ELK-02 EB-02 01-JAN-02             2

KODE_PASO KODE_B KODE_ TANGGAL_P JUMLAH_PASOK
--------- ------ ----- --------- ------------
PAS-E007  ELK-03 EJ-01 04-MAR-02             2

12 rows selected.
```

- **Soal :**

Tampilkan data supplier yang nama suppliernya bukan berakhiran tron!

- **Program dan Tampilan :**

```
SQL> select * from Supplier
  2  where Nama_Supplier not like '%tron';

KODE_ NAMA_SUPPLIER                ALAMAT_SUPPLIER              KOTA_SUPPLIER   TELEPON_SUPPLIE
----- ---------------------------- ---------------------------- --------------- ---------------
EJ-01 PT.Acron                     Jl.Thamrin 12                Jakarta         (021)8551048
EJ-02 PT.Mulya                     Jl.Sudirman 10               Jakarta         (021)5430146
EB-01 PT.UltraSound                Jl.Sukarno 21                Bandung         (022)641347
```

- **Soal :**

  Tampilkan kode barang dan kode supplier yang jumlah pasoknya 2,4,6,12!

- **Program dan Tampilan :**

```
SQL> select Kode_Barang,Kode_Supplier from Pasok
  2  where Jumlah_Pasok in (2,4,6,12)
  3  order by Jumlah_Pasok;

KODE_B KODE_
------ -----
ELK-02 EJ-01
ELK-02 EB-02
ELK-03 EJ-01
ELK-03 EB-01
ELK-04 EB-01
```

- **Soal** :

  Tampilkan customer yang mempunyai kode ‘j-001’ atau ‘B-0002;!

- **Program dan Tampilan :**

```
SQL> select * from Customer
  2  where Kode_Customer in ('J-001','B-002');

KODE_C NAMA_CUSTOMER              ALAMAT_CUSTOMER             KOTA_CUSTOMER   TELEPON_CUSTOME
------ -------------------------- --------------------------- --------------- ---------------
B-002  Toko Warna                 Jl.XYZ 30                   Bandung         (022)646121
J-001  Toko Karisma               Jl.Cimanggis 15             Jakarta         (021)8761481
```

```
SQL>  select * from Customer
  2   where Kode_Customer= 'J-001' or Kode_Customer='B-002';

KODE_C NAMA_CUSTOMER              ALAMAT_CUSTOMER             KOTA_CUSTOMER   TELEPON_CUSTOME
------ -------------------------- --------------------------- --------------- ---------------
B-002  Toko Warna                 Jl.XYZ 30                   Bandung         (022)646121
J-001  Toko Karisma               Jl.Cimanggis 15             Jakarta         (021)8761481
```

Praktek 5

- **Soal :**

  Tampilkan berapa kali pembelian (cacah beli), jumlah pembelian berdasarkan kode barang, kode customer . diurutkan berdasarkan kode barang dan kode customer!

- **Program dan Tampilan :**

```
SQL> select kode_barang as kdbrg,kode_customer as kdcus,count (jumlah_pembelian) as cacah_pembelian,
sum (jumlah_pembelian) as jumlah_beli from pembelian group by kode_barang,kode_customer,jumlah_pembe
lian order by kode_barang,kode_customer;

KDBRG  KDCUS  CACAH_PEMBELIAN JUMLAH_BELI
------ ------ --------------- -----------
ELK-01 B-001                1           2
ELK-01 B-002                1           3
ELK-01 J-001                1           3
ELK-01 J-001                1           4
ELK-01 J-002                1           2
ELK-02 B-001                1           2
ELK-02 B-002                1           2
ELK-02 J-001                1           1
ELK-02 J-002                1           1
ELK-03 B-001                1           5
ELK-03 B-001                1           6

KDBRG  KDCUS  CACAH_PEMBELIAN JUMLAH_BELI
------ ------ --------------- -----------
ELK-03 J-001                1           5
ELK-03 J-002                1           4
ELK-04 B-002                1          15
ELK-04 J-002                1          12

15 rows selected.
```

❖ Quey dan view

- **Soal :**

  Tampilkan nama barang, tanggal pasok, dan jumlah barang yang dipasok!

- **Program dan Tampilan :**

```
SQL> select barang.nama_barang,pasok.tanggal_pasok as tgl_pasok,pasok.jumlah_pasok
  2  from barang,pasok
  3  where barang.kode_barang=pasok.kode_barang;

NAMA_BARANG  TGL_PASOK JUMLAH_PASOK
---------------------- -------------------------- ------------
Rice cooker            01-JAN-02          8
```

Rice cooker 01-JAN-02 5

Lemari Es 01-FEB-02 2

Lemari Es 02-FEB-02 3

Lemari Es 01-JAN-02 2

Televisi 03-MAR-02 5

Televisi 04-MAR-02 2

Televisi 03-MAR-02 3

Televisi 13-MAR-02 4

Televisi 13-MAR-02 3

Radio/Tape 22-APR-02 12

NAMA_BARANG TGL_PASOK JUMLAH_PASOK

---------- --------- ----------

Radio/Tape 30-JAN-02 9

12 rows selected.

❖ Seleksi dengan subquery

- **Soal :**

  Tampilkan seluruh transaksi pembelian (tanggal pembelian dan jumlah pembelian).

- **Program dan Tampilan :**

SQL> select tanggal_pembelian,jumlah_pembelian from pembelian;

TANGGAL_P JUMLAH_PEMBELIAN

--------- ----------

20-MAY-02 3

21-MAY-02 4

20-MAY-02 2

20-MAY-02 2

22-MAY-02 3

24-JUN-02 1

24-JUN-02 1

25-JUN-02 2

25-JUN-02 2

20-JUN-02 5

02-JUL-02 4

TANGGAL_P JUMLAH_PEMBELIAN

--------- ----------

04-JUL-02 6

10-JUL-02 5

15-JUL-02 12

17-JUL-02 15

15 rows selected.

- **Soal :**

  Tampilkan rata – rata pembelian pada transaksi pembelian..

- **Program dan Tampilan :**

SQL> select avg(jumlah_pembelian)as rata2_pembelian from pembelian;

RATA2_PEMBELIAN

4.46666667

- **Soal :**

  Tampilkan transaksi pembelian yang jumlahnya di atas rata-rata.

- **Program dan Tampilan :**

```
SQL> select tanggal_pembelian, jumlah_pembelian from pembelian
  2  where jumlah_pembelian>(select avg(jumlah_pembelian)from pembelian);

TANGGAL_P JUMLAH_PEMBELIAN
--------- ----------------
20-JUN-02                5
04-JUL-02                6
10-JUL-02                5
15-JUL-02               12
17-JUL-02               15
```

- **Soal :**

  Tampilkan kode barang, kode customer, dan cacah pembelian from pembelian!

- **Program dan Tampilan :**

```
SQL> select kode_barang as kdb,kode_customer as kdc,count(jumlah_pembelian)as
cacah_pembelian
```

```
 2  from pembelian group by kode_barang,kode_customer;

KDB    KDC    CACAH_PEMBELIAN
------ ------ ---------------
ELK-01 B-001              1
ELK-01 B-002              1
ELK-01 J-001              2
ELK-01 J-002              1
ELK-02 B-001              1
ELK-02 B-002              1
ELK-02 J-001              1
ELK-02 J-002              1
ELK-03 B-001              2
ELK-03 J-001              1
ELK-03 J-002              1

KDB    KDC    CACAH_PEMBELIAN
------ ------ ---------------
ELK-04 B-002              1
ELK-04 J-002              1

13 rows selected.
```

Daftar Pustaka

Alan Denis, Barbara Haley Wixon, David Tagerden, *System Analys and Design : an Object – Oriented Approach with UML 2.0*, John Willey and Sons, 2005

Henry C. Lucas Jr., *The Analysis, Design and Implementation of Information Systems 4th*, McGraw Hill, 1992

Satzinger, Jackson, Burd, *Object-Oriented Analysis and Design with the Unified Process*, Course Technology, 2005

Simon Bennet, Steve McRobb and Ray Farmer, *Object-Oriented System Analysis and Design Using UML*, McGraw Hill, 2006

Wendy and Michael Boggs, *UML with Rational Rose 2002*, Sibex Inc., 2002

UNIVERSITAS
MERCU BUANA

MODUL PERKULIAHAN

PEMROGRAMAN BASIS DATA DAN SQL

Modul Standar untuk digunakan dalam Perkuliahan di Universitas Mercu Buana

Fakultas	Program Studi	Tatap Muka	Kode MK	Disusun Oleh
Ilmu Komputer	Sistem Informasi	11	87007	Team Dosen

Abstract

SDLC, RAD, Design Tersturktur, Class dan Object

Kompetensi

Memahami siklus pengembangan sistem, memahami metodhologi pengembangan sistem, mengenal UML

Data Manipulation Languange (DML)

Praktek 6

❖ Seleksi labih dari satu tabel

- **Soal :**
  Tampilkan cacah pembelian (berapa kali pembelian) dan jumlah pembelian per nama customer.

- **Program dan Tampilan :**

```
SQL> select C.Nama_Customer,count (L.Jumlah_Pembelian) as Cacah_Pembelian,sum (L.Jumlah_Pembelian) a
s Jumlah_Pembelian
  2  from Customer C,Pembelian L
  3  where C.Kode_Customer = L.Kode_Customer
  4  group by Nama_Customer;

NAMA_CUSTOMER               CACAH_PEMBELIAN JUMLAH_PEMBELIAN
--------------------------- --------------- ---------------
Toko Ayu                                  4              19
Toko Karisma                              4              13
Toko Surya                                4              15
Toko Warna                                3              20
```

- **Soal** ;

  Tampilkan seluruh transaksi pembelian (tanggal pembelian dan jumlah pembelian)

- **Program dan Tampilan :**

```
SQL> select Tanggal_Pembelian,Jumlah_Pembelian
  2  from Pembelian;

TANGGAL_P JUMLAH_PEMBELIAN
--------- ----------------
20-MAY-02                3
21-MAY-02                4
20-MAY-02                2
20-MAY-02                2
22-MAY-02                3
24-JUN-02                1
24-JUN-02                1
25-JUN-02                2
25-JUN-02                2
20-JUN-02                5
02-JUL-02                4

TANGGAL_P JUMLAH_PEMBELIAN
--------- ----------------
04-JUL-02                6
10-JUL-02                5
15-JUL-02               12
17-JUL-02               15

15 rows selected.
```

❖ Pengelompokkan data (group by) dan Aggregat function

- **Soal :**

  Tampilkan jumlah pembelian rata-rata pada transaksi pembelian.

- **Program dan Tampilan :**

```
SQL> select avg(Jumlah_Pembelian) as Rata_Rata_Pembelian
  2  from Pembelian;

RATA_RATA_PEMBELIAN
-------------------
         4.46666667
```

- **Soal :**

  Tampilkan transaksi pembelian yang jumlahnya di atas rata-rata.

- **Program dan Tampilan :**

```
SQL> select Tanggal_Pembelian,Jumlah_Pembelian
  2  from Pembelian
  3  where Jumlah_Pembelian > (select avg(Jumlah_Pembelian) from Pembelian);

TANGGAL_P JUMLAH_PEMBELIAN
--------- ----------------
20-JUN-02                5
04-JUL-02                6
10-JUL-02                5
15-JUL-02               12
17-JUL-02               15
```

- **Soal :**

  Tampilkan transaksi pembelian yang jumlahnya paling kecil

- **Program dan Tampilan :**

```
SQL> select Tanggal_Pembelian,min(Jumlah_Pembelian) as Minimum_Pembelian
  2  from Pembelian
  3  where Jumlah_Pembelian = (select min(Jumlah_Pembelian) from Pembelian)
  4  group by Tanggal_Pembelian;

TANGGAL_P MINIMUM_PEMBELIAN
--------- -----------------
24-JUN-02                 1
```

- **Soal :**

  Tampilkan nama barang yang mempunyai stok barang kurang dari rata-rata

- **Program dan Tampilan :**

```
SQL> select Nama_Barang,Stok_Barang
  2  from Barang
  3  where Stok_Barang < (select avg(Stok_Barang) from Barang);

NAMA_BARANG                STOK_BARANG
-------------------------- -----------
Lemari Es                            8
Rice cooker                         20
```

❖ Seleksi lebih dari satu tabel

- **Soal :**
  Tampilkan supplier yang paling besar pasoknya

- **Program dan Tampilan :**

```
SQL> select S.Nama_Supplier, P.Jumlah_Pasok
  2  from Supplier S, Pasok P
  3  where S.Kode_Supplier = P.Kode_Supplier and P.Jumlah_Pasok = (select max(Jumlah_Pasok) from Pas
ok);

NAMA_SUPPLIER              JUMLAH_PASOK
-------------------------- -----------
PT.UltraSound                       12
```

- **Soal:**
  Buat view bernama v_barang yang menampilkan nama barang dan stok barang.

- **Program dan Tampilan :**

```
SQL> create view v_Barang as
  2  select Nama_Barang,Stok_Barang
  3  from Barang;

View created.
```

- **Soal :**
  Tampilkan semua data yang ada pada view v_Barang

- **Program dan Tampilan :**

```
SQL> select * from v_Barang;

NAMA_BARANG                STOK_BARANG
-------------------------- -----------
Lemari Es                            8
Rice cooker                         20
Televisi
Radio/Tape                          35
```

- **Soal :**

  Tampilkan nama barang dari view v_Barang

- **Program dan Tampilan :**

```
SQL> select Nama_Barang from v_Barang;

NAMA_BARANG
--------------------------
Lemari Es
Rice cooker
Televisi
Radio/Tape
```

- **Soal :**

  Hapuslah view v_Barang

- **Program dan Tampilan :**

```
SQL> drop view v_Barang;

View dropped.
```

- **Soal :**

  Tampilkan view v_Barang (apakah masih ada?)

- **Program dan Tampilan :**

```
SQL> select * from v_Barang;
select * from v_Barang
              *
ERROR at line 1:
ORA-00942: table or view does not exist
```

- **Soal :**

  Buatlah view dengan nama V_Barang_Pasok yang menampilkan jumlah pasok per nama barang untuk tanggal pasok tertentu.

- **Tampilan :**

```
SQL> create view V_Barang_Pasok as
  2  select B.Nama_Barang, P.Tanggal_Pasok, P.Jumlah_Pasok
  3  from Barang B, Pasok P
  4  where B.Kode_Barang = P.Kode_Barang;

View created.
```

- **Soal :**

  Tampilkan jumlah pasok per nama barang per tanggal pasok (dari view tersebut)

- **Program dan Tampilan :**

```
SQL> select * from V_Barang_Pasok;

NAMA_BARANG                TANGGAL_P JUMLAH_PASOK
-------------------------- --------- ------------
Rice cooker                01-JAN-02             8
Rice cooker                01-JAN-02             5
Lemari Es                  01-FEB-02             2
Lemari Es                  02-FEB-02             3
Lemari Es                  01-JAN-02             2
Televisi                   03-MAR-02             5
Televisi                   04-MAR-02             2
Televisi                   03-MAR-02             3
Televisi                   13-MAR-02             4
Televisi                   13-MAR-02             3
Radio/Tape                 22-APR-02            12

NAMA_BARANG                TANGGAL_P JUMLAH_PASOK
-------------------------- --------- ------------
Radio/Tape                 30-JAN-02             9

12 rows selected.
```

- **Soal :**

Tampilkan jumlah pasok per nama barang per tanggal pasok. Nama barang yang ditampilkan adalah televise, radio/tape dan lemari es serta mempunyai jumlah antara 5 dan 10 (Tampilkan dari view V_Barang_Pasok)

- **Program dan Tampilan :**

```
SQL> select * from V_Barang_Pasok
  2  where Nama_Barang in ('Televisi','Radio/Tape','Lemari Es') and
  3  Jumlah_Pasok between 5 and 10;

NAMA_BARANG               TANGGAL_P JUMLAH_PASOK
------------------------- --------- ------------
Televisi                  03-MAR-02            5
Radio/Tape                30-JAN-02            9
```

- **Soal :**

  Buatlah view dengan nama V_Supplier_Pasok yang menampilkan jumlah pasok per nama barang untuk tanggal pasok tertentu.

- **Program dan Tampilan :**

```
SQL> create view V_Supplier_Pasok as
  2  select S.Nama_Supplier, P.Tanggal_Pasok, P.Jumlah_Pasok
  3  from Supplier S, Pasok P
  4  where S.Kode_Supplier = P.Kode_Supplier;

View created.
```

- **Soal :**

  Tampilkan jumlah pasok per nama supplier per tanggal pasok (dari view tersebut)

- **Program dan Tampilan :**

```
SQL> select * from V_Supplier_Pasok;

NAMA_SUPPLIER                  TANGGAL_P JUMLAH_PASOK
------------------------------ --------- ------------
PT.Acron                       01-JAN-02            8
PT.Mulya                       01-JAN-02            5
PT.Acron                       01-FEB-02            2
PT.Mulya                       02-FEB-02            3
PT.Supertron                   01-JAN-02            2
PT.Acron                       03-MAR-02            5
PT.Acron                       04-MAR-02            2
PT.Mulya                       03-MAR-02            3
PT.UltraSound                  13-MAR-02            4
PT.Supertron                   13-MAR-02            3
PT.UltraSound                  22-APR-02           12

NAMA_SUPPLIER                  TANGGAL_P JUMLAH_PASOK
------------------------------ --------- ------------
PT.Supertron                   30-JAN-02            9

12 rows selected.
```

- **Soal :**

  Tampilkan jumlah pasok per nama supplier per tanggal pasok. Nama supplier yang ditampilkan adalah yang berakhiran tron dan mempunyai jumlah pasok 3 dan 5 (Tampilkan dari view V_Supplier_Pasok).

- **Program dan Tampilan :**

```
SQL> select * from V_Supplier_Pasok
  2  where Nama_Supplier like '%tron' and
  3  Jumlah_Pasok in (3,5);

NAMA_SUPPLIER                  TANGGAL_P JUMLAH_PASOK
------------------------------ --------- ------------
PT.Supertron                   13-MAR-02            3
```

- **Soal :**

  Buatlah view dengan nama V_Supplier yang menampilkan kode supplier, nama supplier, dan telepon supplier.Supplier yang ditampilkan adalah supplier yang berasal dari kota Jakarta.

- **Program dan Tampilan :**

```
SQL> create view v_Supplier as
  2  select Kode_Supplier,Nama_Supplier,Telepon_Supplier
  3  from Supplier
  4  where Kota_Supplier = 'Jakarta';

View created.
```

- **Soal :**

  Tampilkan semua data yang ada pada view v_Supplier

- **Program dan Tampilan :**

```
SQL> select * from v_Supplier;

KODE_ NAMA_SUPPLIER                  TELEPON_SUPPLIE
----- ------------------------------ ---------------
EJ-01 PT.Acron                       (021)8551048
EJ-02 PT.Mulya                       (021)5430146
```

Daftar Pustaka

Alan Denis, Barbara Haley Wixon, David Tagerden, *System Analys and Design : an Object – Oriented Approach with UML 2.0*, John Willey and Sons, 2005

Henry C. Lucas Jr., *The Analysis, Design and Implementation of Information Systems 4th*, McGraw Hill, 1992

Satzinger, Jackson, Burd, *Object-Oriented Analysis and Design with the Unified Process*, Course Technology, 2005

Simon Bennet, Steve McRobb and Ray Farmer, *Object-Oriented System Analysis and Design Using UML*, McGraw Hill, 2006

Wendy and Michael Boggs, *UML with Rational Rose 2002*, Sibex Inc., 2002

UNIVERSITAS MERCU BUANA

MODUL PERKULIAHAN

PEMROGRAMAN BASIS DATA DAN SQL

Modul Standar untuk digunakan dalam Perkuliahan di Universitas Mercu Buana

Fakultas	Program Studi	Tatap Muka	Kode MK	Disusun Oleh
Ilmu Komputer	Sistem Informasi	12	87007	Team Dosen

Abstract

SDLC, RAD, Design Tersturktur, Class dan Object

Kompetensi

Memahami siklus pengembangan sistem, memahami metodhologi pengembangan sistem, mengenal UML

ALJABAR RELASIONAL

Praktek 7

❖ **Aljabar Relasional**

- ***Union (gabungan)***

➢ **Soal :**

Tampilkan Kode supplier, berapa kali pemasokkan (cacah pasok) dan jumlah pasok, atau total pasok per supplier. Kode supplier yang ditampilkan adalah 'EJ-01'

➢ **Program dan Tampilan :**

```
SQL> select Kode_Supplier, count (Jumlah_Pasok) as cacah_pasok, sum (Jumlah_Pasok) as Total_Pasok
  2  from Pasok
  3  where Kode_Supplier = 'EJ-01'
  4  Group by Kode_Supplier;

KODE_ CACAH_PASOK TOTAL_PASOK
----- ---------- ----------
EJ-01          4         17
```

➢ **Soal :**

Tampilkan Kode supplier, berapa kali pemasokkan (cacah pasok) dan jumlah pasok, atau total pasok per supplier. Kode supplier yang ditampilkan adalah 'EJ-02'

➢ **Program dan Tampilan :**

```
SQL> select Kode_Supplier, count (Jumlah_Pasok) as cacah_pasok, sum (Jumlah_Pasok) as Total_Pasok
  2  from Pasok
  3  where Kode_Supplier = 'EJ-02'
  4  Group by Kode_Supplier;

KODE_ CACAH_PASOK TOTAL_PASOK
----- ---------- ----------
EJ-02          3         11
```

- **Soal :**

Tampilkan Kode supplier, berapa kali pemasokkan (cacah pasok) dan jumlah pasok, atau total pasok per supplier. Kode supplier yang ditampilkan adalah ‘EB-01’ dan ‘EB-02’

- **Program dan Tampilan :**

```
SQL> select kode_supplier, count (Jumlah_Pasok) as cacah_pasok, sum (Jumlah_Pasok) as Total_Pasok
  2  from Pasok
  3  where Kode_Supplier in ('EB-01','EB-02')
  4  Group by Kode_Supplier;

KODE_ CACAH_PASOK TOTAL_PASOK
----- ---------- ----------
EB-01          2         16
EB-02          3         14
```

- **Soal :**

Tampilkan kode supplier, berapa kali pemasokkan (cacah pasok), jumlah pasok atau total pasok per supplier, kode supplier yang ditampilkan adalah ‘EJ-01’ dan ‘EJ-02’

- **Program dan Tampilan :**

```
SQL>  select Kode_Supplier, count (Jumlah_Pasok) as cacah_pasok, sum (Jumlah_Pasok) as Total_Pasok
  2   from Pasok
  3   where Kode_Supplier = 'EJ-01'
  4   Group by Kode_Supplier
  5   Union
  6  select Kode_Supplier, count (Jumlah_Pasok) as cacah_pasok, sum (Jumlah_Pasok) as Total_Pasok
  7  from Pasok
  8  where Kode_Supplier = 'EJ-02'
  9  Group by Kode_Supplier;

KODE_ CACAH_PASOK TOTAL_PASOK
----- ---------- ----------
EJ-01          4         17
EJ-02          3         11
```

- **Soal :**

  Tampilkan kode supplier, berapa kali pemasokkan, jumlah pasok atau total pasok per supplier. Kode supplier yang ditampilkan ‘EJ-01’, ‘EJ-02’, ‘EB-01’, ‘EB-02’

- **Program dan Tampilan :**

```
SQL> select Kode_Supplier, count (Jumlah_Pasok) as cacah_pasok, sum (Jumlah_Pasok) as Total_Pasok
  2  from Pasok
  3  where Kode_Supplier = 'EJ-01'
  4  Group by Kode_Supplier
  5  Union
  6 select Kode_Supplier, count (Jumlah_Pasok) as cacah_pasok, sum (Jumlah_Pasok) as Total_Pasok
  7 from Pasok
  8 where Kode_Supplier = 'EJ-02'
  9 Group by Kode_Supplier
 10 Union
 11 select kode_supplier, count (Jumlah_Pasok) as cacah_pasok, sum (Jumlah_Pasok) as Total_Pasok
 12 from Pasok
 13 where Kode_Supplier in ('EB-01','EB-02')
 14 Group by Kode_Supplier;

KODE_ CACAH_PASOK TOTAL_PASOK
----- ----------- -----------
EB-01           2          16
EB-02           3          14
EJ-01           4          17
EJ-02           3          11
```

- ***Intersection***

- **Soal :**

  Tampilkan Kode barang dan kode supplier yang memiliki jumlah pasok lebih besar atau sama dengan 4.

- **Program dan Tampilan :**

```
SQL>  select Kode_Barang, Kode_Supplier, sum(Jumlah_Pasok) as Total_Pasok from Pasok
  2   where Jumlah_Pasok >= 4
  3   Group by Kode_Barang, Kode_Supplier;

KODE_B KODE_ TOTAL_PASOK
------ ----- -----------
ELK-01 EJ-01           8
ELK-01 EJ-02           5
ELK-03 EB-01           4
ELK-03 EJ-01           5
ELK-04 EB-01          12
ELK-04 EB-02           9

6 rows selected.
```

- **Soal :**

  Tampilkan Kode barang dan kode supplier yang memiliki jumlah pasok kurang dari atau sama dengan 8.

- **Program dan Tampilan :**

```
SQL>  select Kode_Barang, Kode_Supplier, sum(Jumlah_Pasok) as Total_Pasok from Pasok
  2   where Jumlah_Pasok <= 8
  3   Group by Kode_Barang, Kode_Supplier;

KODE_B KODE_ TOTAL_PASOK
------ ----- -----------
ELK-01 EJ-01           8
ELK-01 EJ-02           5
ELK-02 EB-02           2
ELK-02 EJ-01           2
ELK-02 EJ-02           3
ELK-03 EB-01           4
ELK-03 EB-02           3
ELK-03 EJ-01           7
ELK-03 EJ-02           3

9 rows selected.
```

- **Soal :**

  Tampilkan Kode barnag dan kode supplier yang memiliki jumlah pasok lebih besar atau sama dengan 4 dan lebih kecil sama dengan 8.

- **Program dan Tampilan :**

```
SQL> select Kode_Barang, Kode_Supplier, sum(Jumlah_Pasok) as Total_Pasok from Pasok
  2  where Jumlah_Pasok >= 4
  3  Group by Kode_Barang, Kode_Supplier
  4  intersect
  5  select Kode_Barang,Kode_Supplier,sum(Jumlah_Pasok) as Total_Pasok from Pasok
  6  where Jumlah_Pasok <= 8
  7  Group by Kode_Barang,Kode_Supplier;

KODE_B KODE_ TOTAL_PASOK
------ ----- -----------
ELK-01 EJ-01           8
ELK-01 EJ-02           5
ELK-03 EB-01           4
```

- **Soal :**

  Tampilkan kode barang dan berapa kali terjadi pembelian untuk setiap kode barang tertentu (cacah pembelian)

- **Program dan Tampilan :**

```
SQL> select Kode_Barang, count (Jumlah_Pembelian) from Pembelian
  2  Group by Kode_Barang;

KODE_B COUNT(JUMLAH_PEMBELIAN)
------ ----------------------
ELK-01                      5
ELK-02                      4
ELK-03                      4
ELK-04                      2
```

- **Soal :**

  Tampilkan kode barang dan berapa kali terjadi pemasokkan untuk setiap kode barang tertentu (cacah pemasokkan)

- **Program dan Tampilan :**

```
SQL> select Kode_Barang, count (Jumlah_Pasok) from Pasok
  2  Group by Kode_Barang;

KODE_B COUNT(JUMLAH_PASOK)
------ -------------------
ELK-01                   2
ELK-02                   3
ELK-03                   5
ELK-04                   2
```

- **Soal :**

  Tampilkan kode barang dimana banyaknya pembelian sama dengan banyaknya pemasokkan (cacah_beli = cacah_pasok)

- **Program dan Tampilan :**

```
SQL> select Kode_Barang, count (Jumlah_Pembelian) from Pembelian
  2  Group by Kode_Barang
  3  intersect
  4  select Kode_Barang, count (Jumlah_Pasok) from Pasok
  5  Group by Kode_Barang;

KODE_B COUNT(JUMLAH_PEMBELIAN)
------ -----------------------
ELK-04                       2
```

- ***Difference / Minus***

  - **Soal :**

    (Query 1) Tampilkan kode barang, banyaknya pasokan dan jumlah pasokan untuk semua kode barang

  - **Program dan Tampilan :**

```
SQL> select Kode_Barang, count(Jumlah_Pasok) as Cacah_Pasok, sum(Jumlah_Pasok) as Total_Pasok
  2  from Pasok
  3  Group by Kode_Barang;

KODE_B CACAH_PASOK TOTAL_PASOK
------ ---------- ----------
ELK-01          2         13
ELK-02          3          7
ELK-03          5         17
ELK-04          2         21
```

- **Soal :**

  (Query 2) Tampilkan kode barang, banyaknya pasokan dan jumlah pasokan untuk semua kode barang yang jumlah pemasokannya > 15

- **Program dan Tampilan :**

```
SQL> select Kode_Barang, count(Jumlah_Pasok) as Cacah_Pasok, sum(Jumlah_Pasok) as Total_Pasok
  2  from Pasok
  3  having sum(Jumlah_Pasok) > 15
  4  Group by Kode_Barang;

KODE_B CACAH_PASOK TOTAL_PASOK
------ ---------- ----------
ELK-03          5         17
ELK-04          2         21
```

- **Soal :**

  (Quey 1 minus Query 2) Tampilkan kode barang, banyaknya pasokan dan jumlah pasokan untuk kode barang yang jumlah pasokannya lebih kecil atau sama dengan 15. Gunakan operasi minus

- **Program dan Tampilan :**

```
SQL> select Kode_Barang, count(Jumlah_Pasok) as Cacah_Pasok, sum(Jumlah_Pasok) as Total_Pasok
  2  from Pasok
  3  Group by Kode_Barang
  4  Minus
  5  select Kode_Barang, count(Jumlah_Pasok) as Cacah_Pasok, sum(Jumlah_Pasok) as Total_Pasok
  6  from Pasok
  7  having sum(Jumlah_Pasok) > 15
  8  Group by Kode_Barang;

KODE_B CACAH_PASOK TOTAL_PASOK
------ ---------- ----------
ELK-01          2         13
ELK-02          3          7
```

- ***Proyeksi / Distinct***

  - **Soal :**

    Tampilkan jumlah pembelian berdasarkan kode barang dan tanggal pembelian

  - **Program dan Tampilan :**

```
SQL> Select Kode_Barang, Tanggal_Pembelian, Jumlah_Pembelian from Pembelian;

KODE_B TANGGAL_P JUMLAH_PEMBELIAN
------ --------- ----------------
ELK-01 20-MAY-02                3
ELK-01 21-MAY-02                4
ELK-01 20-MAY-02                2
ELK-01 20-MAY-02                2
ELK-01 22-MAY-02                3
ELK-02 24-JUN-02                1
ELK-02 24-JUN-02                1
ELK-02 25-JUN-02                2
ELK-02 25-JUN-02                2
ELK-03 20-JUN-02                5
ELK-03 02-JUL-02                4

KODE_B TANGGAL_P JUMLAH_PEMBELIAN
------ --------- ----------------
ELK-03 04-JUL-02                6
ELK-03 10-JUL-02                5
ELK-04 15-JUL-02               12
ELK-04 17-JUL-02               15

15 rows selected.
```

  - **Soal :**

    Tampilkan jumlah pembelian berdasarkan kode barang dan tanggal pembelian. Setelah ada perintah Distinct

➢ **Program dan Tampilan :**

```
SQL> Select Distinct Kode_Barang, Tanggal_Pembelian, Jumlah_Pembelian from Pembelian;

KODE_B TANGGAL_P JUMLAH_PEMBELIAN
------ --------- ----------------
ELK-01 20-MAY-02                2
ELK-01 20-MAY-02                3
ELK-01 21-MAY-02                4
ELK-01 22-MAY-02                3
ELK-02 24-JUN-02                1
ELK-02 25-JUN-02                2
ELK-03 20-JUN-02                5
ELK-03 02-JUL-02                4
ELK-03 04-JUL-02                6
ELK-03 10-JUL-02                5
ELK-04 15-JUL-02               12

KODE_B TANGGAL_P JUMLAH_PEMBELIAN
------ --------- ----------------
ELK-04 17-JUL-02               15
```

- ***Joint / Normal joint / Equal joint (=)***

➢ **Soal :**

Tampilkan nama barang, tanggal pasokan dan jimlah pasokan barang.

➢ **Program dan Tampilan :**

```
SQL> select Barang.Nama_Barang, Pasok.Tanggal_Pasok, Pasok.Jumlah_Pasok
  2  from Barang,Pasok
  3  where Barang.Kode_Barang = Pasok.Kode_Barang;

NAMA_BARANG               TANGGAL_P JUMLAH_PASOK
------------------------- --------- ------------
Rice cooker               01-JAN-02            8
Rice cooker               01-JAN-02            5
Lemari Es                 01-FEB-02            2
Lemari Es                 02-FEB-02            3
Lemari Es                 01-JAN-02            2
Televisi                  03-MAR-02            5
Televisi                  04-MAR-02            2
Televisi                  03-MAR-02            3
Televisi                  13-MAR-02            4
Televisi                  13-MAR-02            3
Radio/Tape                22-APR-02           12

NAMA_BARANG               TANGGAL_P JUMLAH_PASOK
------------------------- --------- ------------
Radio/Tape                30-JAN-02            9

12 rows selected.
```

- **Soal :**

  Tampilkan nama supplier, tanggal pasokan dan jumlah pemasokan barang

- **Program dan Tampilan**

```
SQL> select Supplier.Nama_Supplier, Pasok.Tanggal_Pasok, Pasok.Jumlah_Pasok
  2  from Supplier,Pasok
  3  where Supplier.Kode_Supplier = Pasok.Kode_Supplier;

NAMA_SUPPLIER                  TANGGAL_P JUMLAH_PASOK
------------------------------ --------- ------------
PT.Acron                       01-JAN-02             8
PT.Mulya                       01-JAN-02             5
PT.Acron                       01-FEB-02             2
PT.Mulya                       02-FEB-02             3
PT.Supertron                   01-JAN-02             2
PT.Acron                       03-MAR-02             5
PT.Acron                       04-MAR-02             2
PT.Mulya                       03-MAR-02             3
PT.UltraSound                  13-MAR-02             4
PT.Supertron                   13-MAR-02             3
PT.UltraSound                  22-APR-02            12

NAMA_SUPPLIER                  TANGGAL_P JUMLAH_PASOK
------------------------------ --------- ------------
PT.Supertron                   30-JAN-02             9

12 rows selected.
```

- **Soal :**

  Tampilkan nama barang, nama supplier, pasokan, dan jumlah pasokan barang

- **Program dan Tampilan :**

```
SQL> select Barang.Nama_Barang, Supplier.Nama_Supplier, Pasok.Tanggal_Pasok, Pasok.Jumlah_Pasok
  2  from Barang,Supplier,Pasok
  3  where Barang.Kode_Barang = Pasok.Kode_Barang and Supplier.Kode_Supplier = Pasok.Kode_Supplier;

NAMA_BARANG              NAMA_SUPPLIER                TANGGAL_P JUMLAH_PASOK
------------------------ ---------------------------- --------- ------------
Rice cooker              PT.Acron                     01-JAN-02            8
Rice cooker              PT.Mulya                     01-JAN-02            5
Lemari Es                PT.Acron                     01-FEB-02            2
Lemari Es                PT.Mulya                     02-FEB-02            3
Lemari Es                PT.Supertron                 01-JAN-02            2
Televisi                 PT.Acron                     03-MAR-02            5
Televisi                 PT.Acron                     04-MAR-02            2
Televisi                 PT.Mulya                     03-MAR-02            3
Televisi                 PT.UltraSound                13-MAR-02            4
Televisi                 PT.Supertron                 13-MAR-02            3
Radio/Tape               PT.UltraSound                22-APR-02           12

NAMA_BARANG              NAMA_SUPPLIER                TANGGAL_P JUMLAH_PASOK
------------------------ ---------------------------- --------- ------------
Radio/Tape               PT.Supertron                 30-JAN-02            9

12 rows selected.
```

- ***Outer joint (left outer joint / right outer joint)***

Left outer joint

- **Soal :**

  Tampilkan kode barang, nama barang, keterangan, tanggal pasok, dan jumlah pasok. Keterangan yang kosong harus ditampilkan untuk menunjukkan barang apa saja yang tidak memiliki keterangan

- **Program dan Tampilan :**

```
SQL> select B.Kode_Barang, B.Nama_Barang, B.Keterangan, P.Tanggal_Pasok, P.Jumlah_Pasok
  2  from Barang B, Pasok P
  3  where B.Kode_Barang(+) = P.Kode_Barang;

KODE_B NAMA_BARANG            KETERANGAN     TANGGAL_P JUMLAH_PASOK
------ ---------------------- -------------- --------- ------------
ELK-01 Rice cooker            1 Rusak        01-JAN-02            8
ELK-01 Rice cooker            1 Rusak        01-JAN-02            5
ELK-02 Lemari Es                             01-FEB-02            2
ELK-02 Lemari Es                             02-FEB-02            3
ELK-02 Lemari Es                             01-JAN-02            2
ELK-03 Televisi                              03-MAR-02            5
ELK-03 Televisi                              04-MAR-02            2
ELK-03 Televisi                              03-MAR-02            3
ELK-03 Televisi                              13-MAR-02            4
ELK-03 Televisi                              13-MAR-02            3
ELK-04 Radio/Tape             Bagus Semua    22-APR-02           12

KODE_B NAMA_BARANG            KETERANGAN     TANGGAL_P JUMLAH_PASOK
------ ---------------------- -------------- --------- ------------
ELK-04 Radio/Tape             Bagus Semua    30-JAN-02            9

12 rows selected.
```

Right outer joint

- **Soal :**

  Tampilkan kode barang, nama barnag, tanggal pasok dan jumlah pasok. Jumlah barnag yang kosong pun harus ditampilkan untuk menunjukkan barnag apa saja yang tidak di pasok.

- **Program dan Tampilan :**

```
SQL> select B.Kode_Barang, B.Nama_Barang, P.Tanggal_Pasok, P.Jumlah_Pasok
  2  from Barang B, Pasok P
  3  where B.Kode_Barang = P.Kode_Barang(+);

KODE_B NAMA_BARANG                TANGGAL_P JUMLAH_PASOK
------ -------------------------- --------- ------------
ELK-01 Rice cooker                01-JAN-02            8
ELK-01 Rice cooker                01-JAN-02            5
ELK-02 Lemari Es                  01-FEB-02            2
ELK-02 Lemari Es                  02-FEB-02            3
ELK-02 Lemari Es                  01-JAN-02            2
ELK-03 Televisi                   03-MAR-02            5
ELK-03 Televisi                   04-MAR-02            2
ELK-03 Televisi                   03-MAR-02            3
ELK-03 Televisi                   13-MAR-02            4
ELK-03 Televisi                   13-MAR-02            3
ELK-04 Radio/Tape                 22-APR-02           12

KODE_B NAMA_BARANG                TANGGAL_P JUMLAH_PASOK
------ -------------------------- --------- ------------
ELK-04 Radio/Tape                 30-JAN-02            9

12 rows selected.
```

- **Soal :**

  Tampilkan kode supplier, nama supplier, alamat supplier, tanggal pasok dan jumlah pasok. Alamat yang kosong pun harus ditampilkan untuk menunjukkan supplier mana saja yang tidak mempunyai alamat.

- **Program dan Tampilan :**

```
SQL> select S.Kode_Supplier, S.Nama_Supplier, S.Alamat_Supplier, P.Tanggal_Pasok, P.Jumlah_Pasok
  2  from Supplier S, Pasok P
  3  where S.Kode_Supplier (+) = P.Kode_Supplier;

KODE_ NAMA_SUPPLIER               ALAMAT_SUPPLIER               TANGGAL_P JUMLAH_PASOK
----- ---------------------------- ------------------------------ --------- ------------
EJ-01 PT.Acron                    Jl.Thamrin 12                 01-JAN-02            8
EJ-02 PT.Mulya                    Jl.Sudirman 10                01-JAN-02            5
EJ-01 PT.Acron                    Jl.Thamrin 12                 01-FEB-02            2
EJ-02 PT.Mulya                    Jl.Sudirman 10                02-FEB-02            3
EB-02 PT.Supertron                Jl.Industri 37                01-JAN-02            2
EJ-01 PT.Acron                    Jl.Thamrin 12                 03-MAR-02            5
EJ-01 PT.Acron                    Jl.Thamrin 12                 04-MAR-02            2
EJ-02 PT.Mulya                    Jl.Sudirman 10                03-MAR-02            3
EB-01 PT.UltraSound               Jl.Sukarno 21                 13-MAR-02            4
EB-02 PT.Supertron                Jl.Industri 37                13-MAR-02            3
EB-01 PT.UltraSound               Jl.Sukarno 21                 22-APR-02           12

KODE_ NAMA_SUPPLIER               ALAMAT_SUPPLIER               TANGGAL_P JUMLAH_PASOK
----- ---------------------------- ------------------------------ --------- ------------
EB-02 PT.Supertron                Jl.Industri 37                30-JAN-02            9

12 rows selected.
```

Right outer joint

- **Soal :**

  Tampilkan kode supplier, nama supplier, tanggal pasok dan jumlah pasok. Jumlah pembelian yang kosong pun harus ditampilkan untuk menunjukkan supplier mana saja yang tidak mempunyai alamat.

- **Program dan Tampilan :**

```
SQL> select S.Kode_Supplier, S.Nama_Supplier, P.Tanggal_Pasok, P.Jumlah_Pasok
  2  from Supplier S, Pasok P
  3  where S.Kode_Supplier = P.Kode_Supplier(+);

KODE_ NAMA_SUPPLIER                  TANGGAL_P JUMLAH_PASOK
----- ------------------------------ --------- ------------
EB-01 PT.UltraSound                  13-MAR-02            4
EB-01 PT.UltraSound                  22-APR-02           12
EB-02 PT.Supertron                   01-JAN-02            2
EB-02 PT.Supertron                   30-JAN-02            9
EB-02 PT.Supertron                   13-MAR-02            3
EJ-01 PT.Acron                       01-JAN-02            8
EJ-01 PT.Acron                       01-FEB-02            2
EJ-01 PT.Acron                       03-MAR-02            5
EJ-01 PT.Acron                       04-MAR-02            2
EJ-02 PT.Mulya                       01-JAN-02            5
EJ-02 PT.Mulya                       03-MAR-02            3

KODE_ NAMA_SUPPLIER                  TANGGAL_P JUMLAH_PASOK
----- ------------------------------ --------- ------------
EJ-02 PT.Mulya                       02-FEB-02            3

12 rows selected.
```

Daftar Pustaka

Alan Denis, Barbara Haley Wixon, David Tagerden, *System Analys and Design : an Object – Oriented Approach with UML 2.0*, John Willey and Sons, 2005

Henry C. Lucas Jr., *The Analysis, Design and Implementation of Information Systems 4th*, McGraw Hill, 1992

Satzinger, Jackson, Burd, *Object-Oriented Analysis and Design with the Unified Process*, Course Technology, 2005

Simon Bennet, Steve McRobb and Ray Farmer, *Object-Oriented System Analysis and Design Using UML*, McGraw Hill, 2006

Wendy and Michael Boggs, *UML with Rational Rose 2002*, Sibex Inc., 2002

UNIVERSITAS
MERCU BUANA

MODUL PERKULIAHAN

PEMROGRAMAN BASIS DATA DAN SQL

Modul Standar untuk digunakan dalam Perkuliahan di Universitas Mercu Buana

Fakultas	Program Studi	Tatap Muka	Kode MK	Disusun Oleh
Ilmu Komputer	Sistem Informasi	13	87007	Team Dosen

Abstract

SDLC, RAD, Design Tersturktur, Class dan Object

Kompetensi

Memahami siklus pengembangan sistem, memahami metodhologi pengembangan sistem, mengenal UML

PENGENALAN PL/SQL

Praktek 8

❖ Data Control Language

- ***Commit***

➢ **Soal :**

Melakukan transaksi update pada table Barang, mengubah nama Televisi menjadi TV. Kemudian tambahkan perintah commit setelah anda selesai melakukan transaksi terhadap isi database.

➢ **Program dan Tampilan :**

```
SQL> update Barang
  2  set Nama_Barang ='Tv'
  3  where upper(Nama_Barang) = 'TELEVISI';

1 row updated.

SQL> commit;

Commit complete.
```

- ***Rollback***

➢ **Soal :**

Lakukan penghapusan data pada table pasok yang kodenya ‘PAS-E012. Kemudian setelah di hapus, anda ingin mengembalikan baris yang sudah dihapus tersebut seperti semula.Gunakan perintah rollback.

➢ **Program dan Tampilan :**

```
SQL>  delete from Pasok where Kode_Pasok = 'PAS-E012';

1 row deleted.

SQL> select * from Pasok;

KODE_PASO KODE_B KODE_ TANGGAL_P JUMLAH_PASOK
--------- ------ ----- --------- ------------
PAS-E001  ELK-01 EJ-01 01-JAN-02             8
PAS-E002  ELK-01 EJ-02 01-JAN-02             5
PAS-E003  ELK-02 EJ-01 01-FEB-02             2
PAS-E004  ELK-02 EJ-02 02-FEB-02             3
PAS-E005  ELK-02 EB-02 01-JAN-02             2
PAS-E006  ELK-03 EJ-01 03-MAR-02             5
PAS-E007  ELK-03 EJ-01 04-MAR-02             2
PAS-E008  ELK-03 EJ-02 03-MAR-02             3
PAS-E009  ELK-03 EB-01 13-MAR-02             4
PAS-E010  ELK-03 EB-02 13-MAR-02             3
PAS-E011  ELK-04 EB-01 22-APR-02            12

11 rows selected.


SQL> rollback;

Rollback complete.

SQL> select * from Pasok;

KODE_PASO KODE_B KODE_ TANGGAL_P JUMLAH_PASOK
--------- ------ ----- --------- ------------
PAS-E001  ELK-01 EJ-01 01-JAN-02             8
PAS-E002  ELK-01 EJ-02 01-JAN-02             5
PAS-E003  ELK-02 EJ-01 01-FEB-02             2
PAS-E004  ELK-02 EJ-02 02-FEB-02             3
PAS-E005  ELK-02 EB-02 01-JAN-02             2
PAS-E006  ELK-03 EJ-01 03-MAR-02             5
PAS-E007  ELK-03 EJ-01 04-MAR-02             2
PAS-E008  ELK-03 EJ-02 03-MAR-02             3
PAS-E009  ELK-03 EB-01 13-MAR-02             4
PAS-E010  ELK-03 EB-02 13-MAR-02             3
PAS-E011  ELK-04 EB-01 22-APR-02            12

KODE_PASO KODE_B KODE_ TANGGAL_P JUMLAH_PASOK
--------- ------ ----- --------- ------------
PAS-E012  ELK-04 EB-02 30-JAN-02             9

12 rows selected.
```

❖ Pengenalan PL / SQL

- **Soal :**
  Pengenalan blok PL / SQL

- **Program dan Tampilan :**

```
SQL> set set severoutput on
SP2-0158: unknown SET option "set"
SQL> set serveroutput on
SQL> declare
  2  x integer;
  3  Begin
  4  x:='TEST';
  5  Exception
  6  when VALUE_ERROR then
  7  dbms_output.put_line('Nilai yang dimasukkan tidak sesuai dengan tipe data');
  8  end;
  9  /
Nilai yang dimasukkan tidak sesuai dengan tipe data

PL/SQL procedure successfully completed.
```

- **Soal :**

  Pengenalan blok PL / SQL bila pada kasus di atas tidak dituliskan bagian eksepsi maka akan terdapat kesalahan yang diberikan pada oracle.

- **Program dan Tampilan :**

```
SQL> set serveroutput on
SQL> declare
  2  x integer;
  3  begin
  4  x:='TEST';
  5  end;
  6  /
declare
*
ERROR at line 1:
ORA-06502: PL/SQL: numeric or value error: character to number conversion error
ORA-06512: at line 4
```

❖ Percabangan

• *Satu kondisi*

- **Soal :**

  Buatlah sebuah blik PL / SQL yang menampilkan teks PERSEDIAAN BARANG TELAH HABIS dari tabel barang yang nama barangnya LEMARI ES dan jika stok dari barang tersebut sama dengan 0.

- **Program dan Tampilan :**

```
SQL> set serveroutput on
SQL> declare
  2   vstok Barang.Stok_Barang % type;
  3  begin
  4   select Stok_Barang into vstok from Barang
  5   where upper (Nama_Barang) = upper('Lemari Es');
  6   if vstok = 0 then
  7    dbms_output.put_line ('PERSEDIAAN BARANG TELAH HABIS');
  8   end if;
  9  end;
 10  /

PL/SQL procedure successfully completed.
```

- ***Dua Kondisi***

  - **Soal :**

    Buatlah sebuah blik PL / SQL yang menampilkan teks PERSEDIAAN BARANG TELAH HABIS jika jumlah stok dari LEMARI ES sama dengan 0. jika tidak, maka teks yang ditampilkan adalah PERSEDIAAN BARANG MASIH ADA.

  - **Program dan Tampilan :**

```
SQL> set serveroutput on
SQL> declare
  2   vstok Barang.Stok_Barang % type;
  3  begin
  4   select Stok_Barang into vstok from Barang
  5   where upper (Nama_Barang) = upper('Lemari Es');
  6   if vstok = 0 then
  7    dbms_output.put_line ('PERSEDIAAN BARANG TELAH HABIS');
  8   else
  9    dbms_output.put_line ('PERSEDIAAN BARANG MASIH ADA');
 10   end if;
 11  end;
 12  /
PERSEDIAAN BARANG MASIH ADA

PL/SQL procedure successfully completed.
```

- ***Tiga Kondisi atau lebih***

  - **Soal :**

    Buatlah sebuah blik PL / SQL yang menampilkan teks PERSEDIAAN BARANG TELAH HABIS jika jumlah stok dari LEMARI ES sama dengan 0, PERSEDIAAN BARANG KURANG DARI 10 jika jumlah stoknya kurang dari 10, sedangkan jika jumlah stok lebih dari 10 atau sama dengan 10, teks tang ditampilkan adalah PERSEDIAAN MASIH BANYAK.

➢ **Program dan Tampilan :**

```
SQL> set serveroutput on
SQL> declare
  2   vstok Barang.Stok_Barang % type;
  3  begin
  4   select Stok_Barang into vstok from Barang
  5   where upper (Nama_Barang) = upper('Lemari Es');
  6   if ( vstok >= 10 ) then
  7    dbms_output.put_line ('PERSEDIAAN BARANG MASIH BANYAK');
  8   elsif ( vstok > 0 ) then
  9    dbms_output.put_line ('PERSEDIAAN BARANG KURANG DARI 10');
 10   else
 11    dbms_output.put_line ('PERSEDIAAN BARANG HABIS');
 12   end if;
 13  end;
 14  /
PERSEDIAAN BARANG KURANG DARI 10

PL/SQL procedure successfully completed.
```

❖ Pengulangan

- ***Loop ... End Loop***

➢ **Soal , Program, dan Tampilan :**

```
SQL> set serveroutput on
SQL> declare
  2  k integer;
  3  begin
  4  k:=-1;
  5  loop
  6  dbms_output.put_line('Baris ke-' || to_char(k));
  7  k:=K+1;
  8  exit when k>5;
  9  end loop;
 10  end;
 11  /
Baris ke--1
Baris ke-0
Baris ke-1
Baris ke-2
Baris ke-3
Baris ke-4
Baris ke-5

PL/SQL procedure successfully completed.
```

- ***While ... End Loop***

➢ **Soal, Program, dan Tampilan :**

```
SQL> set serveroutput on
SQL> declare
  2  k integer;
  3  begin
  4  k:=1;
  5  while k<=5 loop
  6  dbms_output.put_line('Baris ke-' || to_char(k));
  7  k:=K+1;
  8  end loop;
  9  end;
 10  /
Baris ke-1
Baris ke-2
Baris ke-3
Baris ke-4
Baris ke-5

PL/SQL procedure successfully completed.
```

- ***For ... End Loop***

  - **Soal, Program, dan Tampilan :**

```
SQL> set serveroutput on
SQL> declare
  2  k integer;
  3  begin
  4  for k in 1..5 loop
  5  dbms_output.put_line('Baris ke-' || to_char(k));
  6  end loop;
  7  end;
  8  /
Baris ke-1
Baris ke-2
Baris ke-3
Baris ke-4
Baris ke-5

PL/SQL procedure successfully completed.
```

Daftar Pustaka

Alan Denis, Barbara Haley Wixon, David Tagerden, *System Analys and Design : an Object – Oriented Approach with UML 2.0*, John Willey and Sons, 2005

Henry C. Lucas Jr., *The Analysis, Design and Implementation of Information Systems 4th*, McGraw Hill, 1992

Satzinger, Jackson, Burd, *Object-Oriented Analysis and Design with the Unified Process*, Course Technology, 2005

Simon Bennet, Steve McRobb and Ray Farmer, *Object-Oriented System Analysis and Design Using UML*, McGraw Hill, 2006

Wendy and Michael Boggs, *UML with Rational Rose 2002*, Sibex Inc., 2002

UNIVERSITAS
MERCU BUANA

MODUL PERKULIAHAN

PEMROGRAMAN BASIS DATA DAN SQL

Modul Standar untuk digunakan dalam Perkuliahan di Universitas Mercu Buana

Fakultas	Program Studi	Tatap Muka	Kode MK	Disusun Oleh
Ilmu Komputer	Sistem Informasi	14	87007	Team Dosen

Abstract

SDLC, RAD, Design Tersturktur, Class dan Object

Kompetensi

Memahami siklus pengembangan sistem, memahami metodhologi pengembangan sistem, mengenal UML

Praktek 9

❖ Fasilitas dalam SQLPlus

- ***Fungsi validasi pada SQLPlus***

➢ **Soal Lat 01:**

```
set termout on
set echo off
set verify off
prompt ******************
prompt *** LATIHAN 1 ***
prompt ******************
select a.dname,b.ename,b.job from dept a, emp b where a.deptno = b.deptno;
prompt *** SELESAI ***
```

➢ **Program dan Tampilan :**

```
SQL> set termout on
SQL> set echo off
SQL> set verify off
SQL> prompt ******************
******************
SQL> prompt *** LATIHAN 1 ***
*** LATIHAN 1 ***
SQL> prompt ******************
******************
SQL> select a.dname,b.ename,b.job from dept a, emp b where a.deptno = b.deptno;

DNAME          ENAME      JOB
-------------- ---------- ---------
RESEARCH       SMITH      CLERK
SALES          ALLEN      SALESMAN
SALES          WARD       SALESMAN
RESEARCH       JONES      MANAGER
SALES          MARTIN     SALESMAN
SALES          BLAKE      MANAGER
ACCOUNTING     CLARK      MANAGER
RESEARCH       SCOTT      ANALYST
ACCOUNTING     KING       PRESIDENT
SALES          TURNER     SALESMAN
RESEARCH       ADAMS      CLERK

DNAME          ENAME      JOB
-------------- ---------- ---------
SALES          JAMES      CLERK
RESEARCH       FORD       ANALYST
ACCOUNTING     MILLER     CLERK

14 rows selected.

SQL>
SQL> prompt *** SELESAI ***
*** SELESAI ***
```

- **Soal Lat 02 :**

```
set termout on
set echo off
set verify off
prompt ******************************************************
prompt *** LATIHAN KE 2 DENGAN MEMAKAI SINGLE AMPERSAND ***
prompt ******************************************************
prompt ****** ketik nama variabel v_ename = 'SMITH' *******
select * from emp where ename = &v_ename;

prompt *** SELESAI ***
```

- **Program dan Tampilan :**

```
set termout on
set echo off
set verify off
prompt ******************************************************
prompt *** LATIHAN KE 2 DENGAN MEMAKAI SINGLE AMPERSAND ***
```

```
prompt ***************************************************
prompt ****** ketik nama variabel v_ename = 'SMITH' *******
select * from emp where ename = &v_ename;
prompt *** SELESAI ***
```

```
************************************************
*** LATIHAN KE 2 DENGAN MEMAKAI SINGLE AMPERSAND ***
************************************************
****** ketik nama variabel v_ename = 'SMITH' *******
Enter value for v_ename: 'SMITH'

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7369 SMITH      CLERK           7902 17-DEC-80        800                    20

Input truncated to 22 characters
*** SELESAI ***
```

- **Soal Lat 03 :**

```
set termout on
set echo off
set verify off
prompt *******************************************************
prompt **** LATIHAN KE 3 DENGAN MEMAKAI DOUBLE AMPERSAND
****
prompt *******************************************************
prompt ********* PERHATIKAN BENTUK PERINTAH SQLNYA **********
select dname,loc from dept where deptno = &v_deptno;

prompt *** SELESAI ***
```

- **Program dan Tampilan :**

```
set termout on
set echo off
set verify off
prompt *******************************************************
prompt **** LATIHAN KE 3 DENGAN MEMAKAI DOUBLE AMPERSAND
****
prompt *******************************************************
prompt ********* PERHATIKAN BENTUK PERINTAH SQLNYA
**********
select dname,loc from dept where deptno = &v_deptno;

prompt *** SELESAI ***
```

```
*********************************************************
**** LATIHAN KE 3 DENGAN MEMAKAI DOUBLE AMPERSAND ****
*********************************************************
********** PERHATIKAN BENTUK PERINTAH SQLNYA ***********
Enter value for v_deptno: 10

DNAME          LOC
-------------- -------------
ACCOUNTING     NEW YORK

Input truncated to 22 characters
*** SELESAI ***
```

- **Soal Lat 04 :**

```
set termout on
set echo off
set verify off
prompt ************************************************
prompt *** LATIHAN KE 4 MENGGUNAKAN FUNGSI ACCEPT ***
prompt ************************************************
accept v_dept prompt 'masukkan nomor deptno:'
select deptno,dname from dept where deptno = &v_deptno;

prompt *** SELESAI ***
```

- **Program dan Tampilan :**

```
set termout on
set echo off
set verify off
prompt ************************************************
prompt *** LATIHAN KE 4 MENGGUNAKAN FUNGSI ACCEPT ***
prompt ************************************************
accept v_dept prompt 'masukkan nomor deptno:'
select deptno,dname from dept where deptno = &v_deptno;

prompt *** SELESAI ***
```

```
************************************************
*** LATIHAN KE 4 MENGGUNAKAN FUNGSI ACCEPT ***
************************************************
masukkan nomor deptno:10
Enter value for v_deptno: 10

    DEPTNO DNAME
---------- --------------
        10 ACCOUNTING

Input truncated to 22 characters
*** SELESAI ***
```

- **Soal Lat 05 :**

```
set termout on
set echo off
set verify off
prompt ******************************************************
prompt *** LATIHAN KE 5 MENGGUNAKAN BEBERAPA FUNGSI LAYOUT ***
prompt ******************************************************
prompt ** Perhatikan penggunaan fungsi layout di bawah ini  **
set colsep '|'
set underline =
set pagesize 40
set numformat 999,999
set null kosong
select empno, ename, job, hiredate, sal, comm, deptno from emp;

prompt *** SELESAI ***
```

- **Program dan Tampilan :**

```
set termout on
set echo off
set verify off
prompt ******************************************************
prompt *** LATIHAN KE 5 MENGGUNAKAN BEBERAPA FUNGSI LAYOUT ***
prompt ******************************************************
prompt ** Perhatikan penggunaan fungsi layout di bawah ini  **
set colsep '|'
set underline =
set pagesize 40
set numformat 999,999
set null kosong
select empno, ename, job, hiredate, sal, comm, deptno from emp;

prompt *** SELESAI ***
```

```
**********************************************************
*** LATIHAN KE 5 MENGGUNAKAN BEBERAPA FUNGSI LAYOUT ***
**********************************************************
** Perhatikan penggunaan fungsi layout di bawah ini  **

   EMPNO|ENAME     |JOB       |HIREDATE |     SAL|    COMM|  DEPTNO
========|==========|==========|=========|========|========|========
   7,369|SMITH     |CLERK     |17-DEC-80|     800|kosong  |      20
   7,499|ALLEN     |SALESMAN  |20-FEB-81|   1,600|     300|      30
   7,521|WARD      |SALESMAN  |22-FEB-81|   1,250|     500|      30
   7,566|JONES     |MANAGER   |02-APR-81|   2,975|kosong  |      20
   7,654|MARTIN    |SALESMAN  |28-SEP-81|   1,250|   1,400|      30
   7,698|BLAKE     |MANAGER   |01-MAY-81|   2,850|kosong  |      30
   7,782|CLARK     |MANAGER   |09-JUN-81|   2,450|kosong  |      10
   7,788|SCOTT     |ANALYST   |19-APR-87|   3,000|kosong  |      20
   7,839|KING      |PRESIDENT|17-NOV-81|   5,000|kosong  |      10
   7,844|TURNER    |SALESMAN  |08-SEP-81|   1,500|       0|      30
   7,876|ADAMS     |CLERK     |23-MAY-87|   1,100|kosong  |      20
   7,900|JAMES     |CLERK     |03-DEC-81|     950|kosong  |      30
   7,902|FORD      |ANALYST   |03-DEC-81|   3,000|kosong  |      20
   7,934|MILLER    |CLERK     |23-JAN-82|   1,300|kosong  |      10

14 rows selected.

Input truncated to 22 characters
*** SELESAI ***
```

- **Soal Lat 06 :**

```
set termout on
set echo off
set verify off
prompt **********************************************************
prompt *** LATIHAN KE 6 MENGGUNAKAN FUNGSI TAMPILAN FORMAT ***
prompt ******************** Fungsi break **********************
prompt **********************************************************
break on deptno skip 1
select deptno, ename, job, mgr, hiredate, sal, comm from emp order by deptno;

prompt *** SELESAI ***
```

- **Program dan Tampilan :**

```
set termout on
set echo off
set verify off
prompt **********************************************************
```

```
prompt *** LATIHAN KE 6 MENGGUNAKAN FUNGSI TAMPILAN FORMAT ***
prompt ********************* Fungsi break *********************
prompt ******************************************************
break on deptno skip 1
select deptno, ename, job, mgr, hiredate, sal, comm from emp order by deptno;

prompt *** SELESAI ***
```

```
******************************************************
*** LATIHAN KE 6 MENGGUNAKAN FUNGSI TAMPILAN FORMAT ***
********************* Fungsi break *********************
******************************************************

  DEPTNO|ENAME     |JOB       |     MGR|HIREDATE |     SAL|    COMM
========|==========|==========|========|=========|========|========
      10|CLARK     |MANAGER   |   7,839|09-JUN-81|   2,450|kosong
        |KING      |PRESIDENT|kosong   |17-NOV-81|   5,000|kosong
        |MILLER    |CLERK     |   7,782|23-JAN-82|   1,300|kosong

      20|SMITH     |CLERK     |   7,902|17-DEC-80|     800|kosong
        |ADAMS     |CLERK     |   7,788|23-MAY-87|   1,100|kosong
        |FORD      |ANALYST   |   7,566|03-DEC-81|   3,000|kosong
        |SCOTT     |ANALYST   |   7,566|19-APR-87|   3,000|kosong
        |JONES     |MANAGER   |   7,839|02-APR-81|   2,975|kosong

      30|ALLEN     |SALESMAN  |   7,698|20-FEB-81|   1,600|     300
        |BLAKE     |MANAGER   |   7,839|01-MAY-81|   2,850|kosong
        |MARTIN    |SALESMAN  |   7,698|28-SEP-81|   1,250|   1,400
        |JAMES     |CLERK     |   7,698|03-DEC-81|     950|kosong
        |TURNER    |SALESMAN  |   7,698|08-SEP-81|   1,500|       0
        |WARD      |SALESMAN  |   7,698|22-FEB-81|   1,250|     500

14 rows selected.

Input truncated to 22 characters
*** SELESAI ***
```

- **Soal Lat 07 :**

```
set termout on
set echo off
set verify off
prompt ******************************************************
prompt *** LATIHAN KE 7 MENGGUNAKAN FUNGSI TAMPILAN FORMAT ***
prompt ********************* Fungsi Compute *******************
prompt ******************************************************
break on deptno skip 1
compute max of sal on deptno
select deptno, ename, job, mgr, hiredate, sal, comm from emp order by deptno;
```

```
prompt *** SELESAI ***
```

- **Program dan Tampilan :**

```
set termout on
set echo off
set verify off
prompt ******************************************************
prompt *** LATIHAN KE 7 MENGGUNAKAN FUNGSI TAMPILAN FORMAT ***
prompt ******************** Fungsi Compute ********************
prompt ******************************************************
break on deptno skip 1
compute max of sal on deptno
select deptno, ename, job, mgr, hiredate, sal, comm from emp order by deptno;

prompt *** SELESAI ***
```

```
*********************************************************
*** LATIHAN KE 7 MENGGUNAKAN FUNGSI TAMPILAN FORMAT ***
******************** Fungsi Compute *********************
*********************************************************

  DEPTNO|ENAME     |JOB       |     MGR|HIREDATE |     SAL|    COMM
========|==========|==========|========|=========|========|========
      10|CLARK     |MANAGER   |   7,839|09-JUN-81|   2,450|kosong
        |KING      |PRESIDENT|kosong   |17-NOV-81|   5,000|kosong
        |MILLER    |CLERK     |   7,782|23-JAN-82|   1,300|kosong
*********|          |          |        |         |--------|
maximum |          |          |        |         |   5,000|

      20|SMITH     |CLERK     |   7,902|17-DEC-80|     800|kosong
        |ADAMS     |CLERK     |   7,788|23-MAY-87|   1,100|kosong
        |FORD      |ANALYST   |   7,566|03-DEC-81|   3,000|kosong
        |SCOTT     |ANALYST   |   7,566|19-APR-87|   3,000|kosong
        |JONES     |MANAGER   |   7,839|02-APR-81|   2,975|kosong
*********|          |          |        |         |--------|
maximum |          |          |        |         |   3,000|

      30|ALLEN     |SALESMAN  |   7,698|20-FEB-81|   1,600|     300
        |BLAKE     |MANAGER   |   7,839|01-MAY-81|   2,850|kosong
        |MARTIN    |SALESMAN  |   7,698|28-SEP-81|   1,250|   1,400
        |JAMES     |CLERK     |   7,698|03-DEC-81|     950|kosong
        |TURNER    |SALESMAN  |   7,698|08-SEP-81|   1,500|       0
        |WARD      |SALESMAN  |   7,698|22-FEB-81|   1,250|     500
*********|          |          |        |         |--------|
maximum |          |          |        |         |   2,850|


14 rows selected.

Input truncated to 22 characters
*** SELESAI ***
```

- **Soal Lat 08 :**

```
set termout on
set echo off
set verify off

prompt*************************************************************
prompt***LATIHAN KE 8 MENGGUNAKAN FUNGSI TAMPILKAN FORMAT***
prompt***FUNGSI COLUMN
prompt*************************************************************

set pagesize 50
```

```
set linesize 100
set underline"="

clear breaks
clear compute
clear sql

column dname format a20 heading 'Departemen'
column ename format a30 heading 'Nama Karyawan'
column sal format 888G888D88 heading 'Gaji'

select a.dname,b.ename,b.sal from dept a,emp b where a.deptno=b.deptno order
by a.deptno;

prompt***Selesai***
```

- **Program dan Tampilan :**

```
set termout on
set echo off
set verify off

prompt*************************************************************
prompt***LATIHAN KE 8 MENGGUNAKAN FUNGSI TAMPILKAN
FORMAT***
prompt***FUNGSI COLUMN
prompt*************************************************************

set pagesize 50
set linesize 100
set underline"="

clear breaks
clear compute
clear sql

column dname format a20 heading 'Departemen'
column ename format a30 heading 'Nama Karyawan'
column sal format 888G888D88 heading 'Gaji'

select a.dname,b.ename,b.sal from dept a,emp b where a.deptno=b.deptno order
by a.deptno;

prompt***Selesai***
```

```
************************************************
*** LATIHAN KE 8 MENGGUNAKAN TAMPILAN FORMAT ***
****************** FUNGSI COLUMN ***************

Gaji            ENAME               SAL
=============== ========== ==========
ACCOUNTING      CLARK              2450
ACCOUNTING      KING               5000
ACCOUNTING      MILLER             1300
RESEARCH        SMITH               800
RESEARCH        ADAMS              1100
RESEARCH        FORD               3000
RESEARCH        SCOTT              3000
RESEARCH        JONES              2975
SALES           ALLEN              1600
SALES           BLAKE              2850
SALES           MARTIN             1250
SALES           JAMES               950
SALES           TURNER             1500
SALES           WARD               1250

14 rows selected.

*** Selesai ***
```

- ***<u>Mengatur tampilan layer</u>***

  - **Soal Lat 09 :**

```
set termout on
set echo off
set verify off
Prompt ******************************************************
Prompt *** LATIHAN KE 9 MENGGUNAKAN FUNGSI TAMPILAN FORMAT ***
Prompt ***************** FUNGSI TTITLE DAN BTITLE ************
Prompt ******************************************************

/* Setting Format Layar */
Set pagesize 30
set linesize 70
set underline "="

/* membersihkan atribute dari memory */
clear breaks
clear compute
clear sql
clear buff

/* pemberian judul atas dan bawah dengan menggunakan ttitle dan btitle */
```

```
ttitle center "*** LAPORAN KARYAWAN BERDASARKAN DEPARTEMEN
***" skip 1-
        right "Hal :"format 999 sql.pno -
        left "user :"sql.user skip""

btitle center "*** akhir halaman ***"

/* memberi format pada kolom */
column dname format a20 heading 'Departemen'
column ename format a30 heading 'Nama Karyawan'
column sal format 999G999D99 heading 'Gaji'

/* perintah sql */
select a.dname, b.ename,b.sal from dept a, emp b where a.deptno=b.deptno;
```

- **Prgram dan Tampilan :**

```
set termout on
set echo off
set verify off
Prompt ******************************************************
Prompt *** LATIHAN KE 9 MENGGUNAKAN FUNGSI TAMPILAN
FORMAT ***
Prompt ***************** FUNGSI TTITLE DAN BTITLE ************
Prompt ******************************************************

/* Setting Format Layar */
Set pagesize 30
set linesize 70
set underline "="

/* membersihkan atribute dari memory */
clear breaks
clear compute
clear sql
clear buff

/* pemberian judul atas dan bawah dengan menggunakan ttitle dan btitle */
ttitle center "*** LAPORAN KARYAWAN BERDASARKAN DEPARTEMEN
***" skip 1-
        right "Hal :"format 999 sql.pno -
        left "user :"sql.user skip""

btitle center "*** akhir halaman ***"
```

```
/* memberi format pada kolom */
column dname format a20 heading 'Departemen'
column ename format a30 heading 'Nama Karyawan'
column sal format 999G999D99 heading 'Gaji'

/* perintah sql */
select a.dname, b.ename,b.sal from dept a, emp b where a.deptno=b.deptno;
```

```
************************************************************
*** LATIHAN KE 9 MENGGUNAKAN FUNGSI TAMPILAN FORMAT ***
***************** FUNGSI TTITLE DAN BTITLE *************
************************************************************

            *** LAPORAN KARYAWAN BERDASARKAN DEPARTEMEN ***
user :SCOTT                                                 Hal :   1

Departemen           |Nama Karyawan                  |       Gaji
=====================|===============================|===========
RESEARCH             |SMITH                          |      800.00
SALES                |ALLEN                          |    1,600.00
SALES                |WARD                           |    1,250.00
RESEARCH             |JONES                          |    2,975.00
SALES                |MARTIN                         |    1,250.00
SALES                |BLAKE                          |    2,850.00
ACCOUNTING           |CLARK                          |    2,450.00
RESEARCH             |SCOTT                          |    3,000.00
ACCOUNTING           |KING                           |    5,000.00
SALES                |TURNER                         |    1,500.00
RESEARCH             |ADAMS                          |    1,100.00
SALES                |JAMES                          |      950.00
RESEARCH             |FORD                           |    3,000.00
ACCOUNTING           |MILLER                         |    1,300.00


                              *** akhir halaman ***

14 rows selected.
```

- **Soal Lat 10 :**

```
set termout on
set echo off
set verify off
Prompt ************************************************************
```

```
Prompt *** LATIHAN KE 10 MENGGUNAKAN FUNGSI TAMPILAN FORMAT ***
Prompt ********** FUNGSI REPHEADER DAN REPFOOTER **********
Prompt ************************************************************

/* Setting Format Layar */
Set pagesize 24
set linesize 70
set underline "="

/* membersihkan atribute dari memory */
clear breaks
clear compute
clear sql
clear buff
clear column

/* pemberian judul pada awal laporan dan akhir laporan */

repheader center "*** Awal Laporan ***"
repfooter center "*** Akhir Laporan ***"

/* pemberian judul atas dan bawah dengan menggunakan ttitle dan btitle pada setiap halaman

*/
ttitle center "*** LAPORAN KARYAWAN BERDASARKAN DEPARTEMEN ***" skip 1-
        right "Hal :"format 999 sql.pno -
        left "user :"sql.user skip""

btitle center "*** akhir halaman ***"

/* memberi format pada kolom */
column dname format a20 heading 'Departemen'
column ename format a30 heading 'Nama Karyawan'
column sal format 999G999D99 heading 'Gaji'

/* perintah sql */
select a.dname, b.ename,b.sal from dept a, emp b where a.deptno=b.deptno order by a.dname;
```

- **Program dan Tampilan :**

```
set termout on
set echo off
set verify off
Prompt ************************************************************
Prompt *** LATIHAN KE 10 MENGGUNAKAN FUNGSI TAMPILAN FORMAT ***
Prompt ************** FUNGSI REPHEADER DAN REPFOOTER **********
Prompt ************************************************************

/* Setting Format Layar */
Set pagesize 24
set linesize 70
set underline "="

/* membersihkan atribute dari memory */
clear breaks
clear compute
clear sql
clear buff
clear column

/* pemberian judul pada awal laporan dan akhir laporan */

repheader center "*** Awal Laporan ***"
repfooter center "*** Akhir Laporan ***"

/* pemberian judul atas dan bawah dengan menggunakan ttitle dan btitle pada setiap halaman

*/
ttitle center "*** LAPORAN KARYAWAN BERDASARKAN DEPARTEMEN ***" skip 1-
        right "Hal :"format 999 sql.pno -
        left "user :"sql.user skip""

btitle center "*** akhir halaman ***"

/* memberi format pada kolom */
column dname format a20 heading 'Departemen'
column ename format a30 heading 'Nama Karyawan'
```

```
column sal format 999G999D99 heading 'Gaji'

/* perintah sql */
select a.dname, b.ename,b.sal from dept a, emp b where a.deptno=b.deptno order
by a.dname;
```

```
************************************************************
*** LATIHAN KE 10 MENGGUNAKAN FUNGSI TAMPILAN FORMAT ***
************** FUNGSI REPHEADER DAN REPFOOTER ***********
************************************************************

            *** LAPORAN KARYAWAN BERDASARKAN DEPARTEMEN ***
user :SCOTT                                                 Hal :   1

                        *** Awal Laporan ***
Departemen          |Nama Karyawan                  |       Gaji
====================|===============================|===========
ACCOUNTING          |CLARK                          |   2,450.00
ACCOUNTING          |KING                           |   5,000.00
ACCOUNTING          |MILLER                         |   1,300.00
RESEARCH            |SMITH                          |     800.00
RESEARCH            |ADAMS                          |   1,100.00
RESEARCH            |FORD                           |   3,000.00
RESEARCH            |SCOTT                          |   3,000.00
RESEARCH            |JONES                          |   2,975.00
SALES               |ALLEN                          |   1,600.00
SALES               |BLAKE                          |   2,850.00
SALES               |MARTIN                         |   1,250.00
SALES               |JAMES                          |     950.00
SALES               |TURNER                         |   1,500.00
SALES               |WARD                           |   1,250.00
                        *** Akhir Laporan ***
                        *** akhir halaman ***

14 rows selected.
```

Daftar Pustaka

Alan Denis, Barbara Haley Wixon, David Tagerden, *System Analys and Design : an Object – Oriented Approach with UML 2.0*, John Willey and Sons, 2005

Henry C. Lucas Jr., *The Analysis, Design and Implementation of Information Systems 4th*, McGraw Hill, 1992

Satzinger, Jackson, Burd, *Object-Oriented Analysis and Design with the Unified Process*, Course Technology, 2005

Simon Bennet, Steve McRobb and Ray Farmer, *Object-Oriented System Analysis and Design Using UML*, McGraw Hill, 2006

Wendy and Michael Boggs, *UML with Rational Rose 2002*, Sibex Inc., 2002