**RANGKUMAN PENGANTAR SISTEM BASIS DATA SEMESTER I**

1. **Arsitektur Basis Data [Pertemuan III - 26 July 2022]**

Terdapat 3 tipe arsitektur basis data (Connoly & Begg 2010, p108-115).

1. Teleprosesing

- tradisional

- beban lebih pada komputer pusat

- banyak terminal, satu komputer pusat

- terminal2 disebut bisu/dumb karena tidak bisa bekerja sendiri

- biasa di mainframe/komputer besar

- time sharing sistem, komputer pusat melayani dengan membagi waktu

- cont: atm mesin bank

1. File Server

- komputer2 yg terhubung ke jaringan biasanya LAN dgn tujuan agar menyimpan dan sharing berbagai file

- tiap proses akan dilakukan di komputer/workstation masing2, jika butuh file baru akses ke server

- client biasanya minta data dan pengkuncian data ke server. dan tanggapan dari client biasanya memberi data dan mengunci data & statusnya.

- rugi karena jaringan jdi padat, dbms di tiap komputer yg terhubung mesti selalu di backup sehingga memakan memory dan proses backup ini menjadi responsibility si programmer, mengontrol currency, recovery dan integrity lebih rumit, komputer klien/workstation harus memiliki kemampuan proses yg tinggi agar response time nya faster

1. Client Server

- klien : mengelola logika, logika pemrosesan data, logika bisnis

- server : penyimpanan, pengaksesan dan pemrosessan db

- server harus memiliki kemampuan proses yg tinggi

- beban jaringan berkurang

- authenticaion,integraation and pemelihrarn data dictionrary dilakukan pd db server

- db server merupakan implementasi dari **two-tier architecture**

Keuntungan

- biaya hardware berkurang. cukup invest di server saja

- biaya komunikasi berkurang. app hanya mengirim request untuk akses db ke seluruh jaringan, hasilnya lebih hemat data yg dikirim ke seluruh jaringan

- konsistensi data meningkat

**Two-tier Architecture**

beberapa client/komputer mengakses ke db server melalui jaringan LAN. klien2 disebut **system interface user** dan db server disebut  **database management service**.

Tugas client sebagai antarmuka pengguna, interaksi db, pengambilan dan modifikasi data, bisnis logic dan penanganan error.

Tugas db server management data, security, query, triggerm prosedur tersimpan, catching error.

**Three-tier Architecture**

Beberapa client/desktop mengakses ke application server dan db server melalui jaringan LAN. client termasuk ke lapisan **client side presentation.** app server disebut **server side business logic**. db server disebut **backend storage**.

Client side presentaiton

- interact with user

- jika basis web dibuat pake html css dll

Server side business logic

- disebut juga middleware, bagian yg responsible atas cara kerja app

Backend Storage

- control the way its store data

- db that can be used is vary. such as rlational databse (RDBMS)

Example

client accessing web server (such as apache, xitami, etc) then server will access middleware(PHP, ASP, dll) that interact with db. after the request complete, it will be send spontatinously back to client

Benefit

- tech flexibility

its easy to change the engine of DBMS nor make it possible for middle tier goes to different platform

- Long terms development with lower cost

changes can be applied only in middle tier than change it to all of the app layers

- competitive

interact fast

Which case that best to applied with three tier arch?

- if the app had a service/ feature more than 50

- if the app had a different version with different language

- if the app had two or more DBMS sources

- if the app will be used for more than 3 years

- if the app should be able to handle 50.000 minimum transactions per day

1. **Arsitektur Basis Data V (Aug 2 2022)**

Arsitektur yg umum dipakai adalah **client server.**

**Client server** memiliki 3 proses:

- bagian penyimpanan data : DML pada database

- bagian proses : business layer

- bagian tampilan : show data

Pemusatan sumber daya dlm **client server** ada 3 :

1. fat client -> berat di client, ringan di server
2. thin client -> ringat di client, berat di server
3. distributed -> berimbang
4. PERTEMUAN 6 (Aug 6 2022)

**Aljabar dan Kalkulus Relasional**

# Query

Manipulasi data dari database

# Tipe bahasa query

- Aljabar Relasional

- prosedural

- bersifat operasional dan detail

- berguna utk representasi eksekusi perencanaan

- Kalkulus Relasional

- non prosedural

- deskripsi tanpa prosedur yg detail

- basis dari bahasa query

# Operasi aljabar relational

- Selection

definisi : memilih himpunan baris / relasi

simbol :

SQL : SELECT \* FROM tb\_mahasiswa WHERE nama = “ani” AND nama = “anita”

ALJABAR :

- Projection (select attribute / phi)

definisi : memilih kolom/menghapus kolom tertentu dari relasi

simbol : (phi)

SQL : SELECT nama, usia FROM tb\_mahasiswa WHERE age = 19

ALJABAR :

- Union

definisi : mendapatkan tupple/record yg terdapat pada relasi satu/dua/dll

simbol :

karakteristik : data duplikat hanya ditampilkan sekali

SQL : SELECT \* FROM tb\_1 UNION SELECT \* FROM tb\_2

ALJABAR :

- Intersection

definisi : irisan data

simbol :

karakteristik : no duplicate

SQL : SELECT \* FROM tb\_1 INTERSECT SELECT \* FROM tb\_2

ALJABAR :

- Set diference

definisi : mendapat tupple/record pada relasi satu yg tidak ada pada relasi dua

simbol : - (minus)

karakteristik : pengambilan unique record berdasar pada relasi pertama yg

disebut. misal student - instruktur, berarti ambil data student yg

tidak ada pada instruktur. sebaliknya jika instruktur - student, berarti

ambil data instruktur yg tidak ada pada student.

SQL : SELECT \* FROM tb\_1 MINUS SELECT \* FROM tb\_2

ALJABAR :

- Cartesian Product/Cross Product

definisi : kombinasi dari dua relasi atau perkalian kartesian antar table

simbol : X

SQL : SELECT \* FROM tb\_1, tb\_2

ALJABAR :

- Division

- Renaming

- Join

definisi : kesamaan data antara table satu dan lainnya

jenis : equi join, self join, inner join, left/ left outer join, right/right outer join,

full join, union

SQL : SELECT tb\_1.col, tb\_2.col FROM tb\_1, tb\_2 WHERE tb\_1.col = tb\_2.col

- Naturan Join

simbol : ⋈

contoh :

mahasiswa (nim, nama, alamat)

nilai (nim, kode\_matkul, nilai\_mid, nilai\_uas)

get nama where student take subject with code “KK21”

- Equi Join

SELECT mhs.nim, mhs.nama, mjr.nama FROM mahasiswa mhs, major mjr WHERE mhs.mjr\_id = mjr.id

- Self Join

definisi : gabungan dari 1 table dgn asumsi table yg digunakan terdiri atas 2 table yg berbeda

SELECT p1.no, p1.nama, p2.nama FROM pegawai p1, pegawai p2 WHERE p1.no = p2.manager\_id

- Inner join

definisi : menampilkan data yg ada di kedua table

SELECT tb\_1.col FROM tb\_1 INNER JOIN tb\_2 ON tb\_1.col = tb\_2.col

- Outer join

definisi : data tetap diikutkan walau tidak memiliki hubungan apapun pada table kedua

tipe :

- Left Join

definisi : mengacu pada table yg kiri/utama/pertama

SELECT tb\_1.col FROM tb\_1 LEFT JOIN tb\_2 ON tb\_1.col = tb\_2.col

- Right Join

definisi : mengacu pada table yg kanan/kedua

SELECT tb\_1.col FROM tb\_1 RIGHT JOIN tb\_2 ON tb\_1.col = tb\_2.col

- Full Join (tidak berlaku di MySQL)

SELECT tb\_1.col FROM tb\_1 FULL JOIN tb\_2 ON tb\_1.col = tb\_2.col

# Operasi Aggregate

- SUM : total nilai

- AVERAGE : rata2

- MIN : terkecil

- MAX : terbesar

- COUNT : jumlah

# Tipe relasi di database

Unary : relasi dari 1 table

Binary : relasi dari 2 atau lebih tabel

Operasi Unary : selection, projection, rename

Operasi Binary : cartesian product, intersection, set difference, dll

# Simbol aljabar relational

R X S -> CARTESIAN PRODUCT (X)

R - S -> SET DIFFERENCE TUPPLE R DAN S (-)

(sigma) -> PREDICATE/WHERE

(irisan) -> INTERSECTION

(phi) -> ATTRIBUTE SELECTION

!= atau <> -> NOT EQUAL

R S -> Tupple R union Tupple

v -> OR

n / -> AND

1. Pertemuan 7 - 6 Aug 2022 (Aljabar Relasional Lanjutan)

# Binary Operations

Join adalah turunan Cartesian Product

# Theta Join (⋈)

definisi : relasi dari dua/ lebih table yg memenuhi predikat/condition (f)

karakteristik : jika predicate/conditionnya hanya operator equality (=), maka theta join dapat disebut juga dengan equi join

rumus : R ⋈ f S

aljabar : tb\_matkul ⋈ tb\_guru.kode\_matkul = tb\_matkul.kode tb\_guru

# Natural Join

definisi : sama dgn equijoin. intinya selalu membandingkan column yg sama pada dua table

# Left Outer Join ()

definisi : berdasarkan table pertama. data yg null pun ikut disertakan

aljabar :

# Semi Join

aljabar :

# Divison

definisi : menghasilkan tupple yg lebih eksak

contoh : tampilkan no client yg melihat properti dgn kamar 3

aljabar : )

# Aggregate Operation (AL atau )

SUM, COUNT, dll

# GROUPING OPERATION (GAAL atau R)

Pertemuan 10 (13 Agustus)

composite key adalah dua atau lebih primary key yg dependent dengan table lain.

# Penulisan SQL

Reserved word : default sql

User defined : dibuat oleh user, merepresentasikan nama

# Domain Constaint

seperti enum

# Referential Integrity

Mereferensikan table

# DDL

create, alter, drop

CREATE SCHEMA tb\_name;

CREATE DATABASE tb\_name;

SHOW DATABASES;

USE tb\_name;

DROP DATABASE tb\_name;

CREATE TABLE produk(

idproduk CHAR(5) NOT NULL,

namaproduk VARCHAR(30) NOT NULL,

satuan ENUM('Pcs', 'Unit', 'Rim', 'Pack') NOT NULL,

harga NUMERIC(10,0) NOT NULL DEFAULT '0', // 10 digit numeric dengan 1 angka dibelakang koma

stok INT(3) NOT NULL DEFAULT '0',

PRIMARY KEY (idproduk),

CONSTRAINT checkidproduk

CHECK(idproduk LIKE 'BR[0-9][0-9][0-9]')) // berarti id produk digit 1 = B, digit 2 = R, digit 3,4,5 itu angka dari 0-9

ENGINE = INNODB DEFAULT CHARSET = latin1;

CREATE TABLE pelanggan(

idpelanggan CHAR(5) NOT NULL,

namapelanggan VARCHAR(30) NOT NULL,

alamat TEXT NOT NULL,

telepon VARCHAR(15) NOT NULL,

email VARCHAR(20) NOT NULL,

PRIMARY KEY(idpelanggan),

CONSTRAINT checkidplg

CHECK(idpelanggan LIKE 'PL[0-9][0-9][0-9]'))

ENGINE = INNODB DEFAULT CHARSET=latin1;

// on delete dan update cascade itu berlaku ke semua table yg berhubungan dengan pesan

CREATE TABLE pesan (

idpesan CHAR(8) NOT NULL,

idpelanggan CHAR(5)NOT NULL,

tglpesan DATE NOT NULL,

PRIMARY KEY(idpesan), KEY idpelanggan(idpelanggan),

CONSTRAINT pesan FOREIGN KEY (idpelanggan)

REFERENCES pelanggan(idpelanggan)

ON DELETE CASCADE

ON UPDATE CASCADE)

ENGINE = INNODB DEFAULT CHARSET=latin1;

CREATE TABLE detail\_pesan (

idpesan CHAR(8) NOT NULL,

idproduk CHAR(5) NOT NULL,

jumlah INT(5) NOT NULL DEFAULT '0',

subtotal NUMERIC(10,0) NOT NULL DEFAULT '0',

PRIMARY KEY(idpesan, idproduk),

KEY FK\_detail\_pesan(idproduk),

KEY FK\_detail\_pesan2(idpesan),

CONSTRAINT FK\_detail\_pesan FOREIGN KEY(idproduk) REFERENCES produk(idproduk),

CONSTRAINT FK\_detail\_pesan2 FOREIGN KEY(idpesan) REFERENCES pesan(idpesan))

ENGINE=INNODB DEFAULT CHARSET=latin1;

CREATE TABLE faktur(

id\_faktur CHAR(9)NOT NULL,

idpesan CHAR(8) NOT NULL,

tgl\_faktur DATE NOT NULL,

PRIMARY KEY (id\_faktur),

KEY fk\_faktur (idpesan),

CONSTRAINT fk\_faktur FOREIGN KEY (idpesan) REFERENCES pesan(idpesan))

ENGINE=INNODB DEFAULT CHARSET=latin1;