MODEL RELASIONAL

LPT UPI YAI
http://yai.ac.id

MODEL RELATIONAL

3.1 PENDAHULUAN

Model relasi pertama kali dikenalkan oleh **Codd**, pada tahun 1971. Sejak itu model relasi memainkan peranan yang sangat penting dalam berbagai perancangan basis data.

Ada tiga alas an mengapa model relasi mempunyai peranan penting dalam perancangan basis data yaitu :

- mempunyai piranti komunikasi yang baik antara *user* & *designer*

  artinya relasi merepresentasikan struktur data yang dapat dimengerti oleh user maupun *designer*.
- model relasional mendefinisikan salah satu kriteria perancangan basis data yang penting yaitu relasi bentuk normal.
- Struktur data yang direpresentasikan oleh relasi dapat segera dikonversikan & diimplementasikan ke RDBMS.

3.2 APA YANG DISEBUT DENGAN RELASI

Terdiri dari tiga kolom, yaitu **Person_Id**, **Proj_No** dan **Total_Time**. Tabel ini menyimpan waktu yang dihabiskan oleh *person* pada proyek tersebut. Selain tabel Work terdapat juga relasi Persons. Relasi ini menyimpan secara detail tentang person yang bekerja pada proyek dimaksud.

Relasi pertama kali didefinisikan menggunakan teori himpunan. Cara termudah untuk mendefinisikan sebuah relasi adalah
sebagai sebuah tabel dimana data-datanya disimpan dalam baris tabel.
Contoh Table : Tabel Work dibawah ini

WORK		
Person_Id	**Proj_No**	**Total_Time**
P1	PROJ1	20

P3	PROJ1	16
P2	PROJ2	35
P2	PROJ3	42
P3	PROJ2	17
P3	PROJ1	83
P4	PROJ3	41

PERSON

Person_Id	Date_Of_birth	Name
P1	JAN 62	JOE
P4	FEB 65	MARY
P3	AUG 33	ANDREW
P2	JUL 48	JOE

3.3 TERMINOLOGI

- Relasi = Tabel
- Atribut relasi = Kolom tabel
- Tuple = Baris

Domain

	Part_Id	Deskripsi	Warna	Berat
Dimensi	(tipe : alpha)	(tipe : string)	(tipe : numeric)	(tipe : numeric)

PARTS

Part_No	Part_Name	Warna	Berat	Max_Dim
P1	Gergaji	Hitam	12	25
P2	Palu	Coklat	20	19

P3	Bor	Abu-abu	88	45

3.4 STRUKTUR LOJIK & FISIK

Relasi adalah sebuah representasi data lojik bukan fisik. Relasi menggambarkan struktur data tanpa memperhatikan bagaimana data disimpan atau diakses.

Representasi lojik berarti bahwa sebuah relasi harus :

- tidak terdapat duplikasi baris
- urutan baris tidak diperhatikan
- setiap kolom dalam suatu relasi mempunyai sebuah nama yang unik

struktur fisik diperlukan selama implementasi fisik yaitu pada saat menentukan layout data & indeks yang digunakan untuk mengakses data dalam suatu relasi. Hal penting lainnya yang harus diperhatikan dalam merancang relasi adalah bahwa nama-nama atribut relasi harus menggambarkan sumber data berasal.

3.5 REDUNDANSI & DUPLIKASI

3.5.1 Redundansi

Salah satu dari perancangan lojik basis data adalah meniadakan redundansi. Redundansi terjadi jika fakta yang sama disimpan lebih dari sekali. Contoh redundansi adalah relasi Project_Data berikut : dalam relasi Project_Data, Project_Budget dari sebuah proyek disimpan lebih dari sekali. Yaitu sebanyak orang yang bekerja pada proyek tersebut. Hal ini tidak diinginkan karena menyebabkan banyak kerugian sebagai berikut :

- Jika Project_Budget untuk **Proj_No** berubah maka harus dilakukan perubahan lebih dari satu baris dalam relasi tersebut.
- Setiap kali penambahan orang baru untuk suatu proyek maka harus dimasukkan pula budget untuk proyek tersebut.
- Sebuah proyek yang tidak ada orangnya, seperti PROJ4, akan hanya mempunyai sebuah nilai Project_Budget tetapi tidak punya nilai untuk atribut lainnya. Ini berarti bahwa operasi-operasi terhadap relasi tersbut harus disesuaikan dengan keadaan struktur basis data saat itu.

PROJECT_DATA

Person_Id	Proj_No	Project_Budget	Total_Time
P1	PROJ1	20	20
P3	PROJ1	20	16
P2	PROJ2	17	35
P2	PROJ3	84	42
P3	PROJ2	17	17
P3	PROJ1	20	83
P4	PROJ3	84	41
-	PROJ4	90	-

3.5.2 Duplikasi

Duplikasi berbeda dengan redundansi. Kadang-kadang duplikasi diperlukan dalam basis data sementara redundansi harus dihindari.

Contoh duplikasi adalah relasi USE dibawah ini. Relasi USE memiliki lebih dari satu baris yang mempunyai nilai yang sama untuk atribut **Project_Id**, yaitu Proj1. Demikian juga untuk Proj2. Nilai-nilai ini harus disimpan lebih dari sekali karena menggambarkan fakta yang berbeda.

USE

Proj_Id	Part_No	Qty_Used
Proj1	P1	17
Proj2	P2	85
Proj1	P3	73
Proj2	P2	80

ASSIGNMENTS

Person_Id	Dept	Date_Of_Birth	Date_Started	Date_Finished
P1	Sales	1 June 53	2 June 80	5 Aug 83
P2	Sales	3 July 51	5 Aug 81	9 Dec 82
P3	Accounting	8 Aug 60	3 Feb 79	17 Jul 82

P1	Production	1 June 53	11 Mar 82	3 Feb 85

Sedangkan dalam relasi Assigment, satu nilai atribut Date_Of_Birth, yaitu 1 June 1953, dapat juga muncul lebih dari sekali. Berbeda dengan Proj1 & Proj2 di atas, 1 June 53 menunjukkan fakta yang sama yaitu menunjukkan orang yang sama. Hal ini menyebabkan terjadinya redundansi dalam relasi Assignment.

3.6 MENGHILANGKAN REDUNDANSI

Salah satu cara untuk menghilangkan redundansi adalah dengan dekomposisi. Sbuah relasi yang menyimpan sebuah fakta lebih dari sekali dapat didekomposisi ke dalam relasi-relasi yang hanya menyimpan sebuh fakta sekali.

Contoh relasi Project_Data di atas dapat didekomposisi menjadi relasi Projects dan Work di bawah ini .

PROJECT

Proj_No	**Project_Budget**
Proj1	20
Proj2	17
Proj3	84
Proj4	90

WORK

Person_Id	**Proj_No**	**Total_Time_Spent_By_ Person_On_Project**
P1	Proj1	20
P3	Proj1	16

P2	Proj2	35
P2	Proj3	42
P3	Proj2	17
P3	Proj1	83
P4	Proj3	41

3.7 QUERY LANGUAGE

Query language adalah suatu bahasa yang menyediakan fasilitas bagi user untuk mengakses informasi dari basis data. Pada umumnya level bahasa ini lebih tinggi dari bahasa pemrograman standar. Bahasa query dapat dikategorikan sebagai prosedural & non-prosedural.

Dalam bahsa prosedural, user menginstruksikan ke sistem agar membentuk serangkaian operasi dalam basis data untuk mengeluarkan hasil yang diinginkan. Dalam bahasa non-prosedural, *user* mendeskripsikan informasi yang diinginkan tanpa memberikan prosedur detail untuk menghasilkan informasi tersebut.
Sebagian besar system basis data relasional yang beredar dipasaran menawarkan bahasa query dengan pendekatan prosedural & non-prosedural.

Bebrapa bahasa query yang murni adalah : **aljabar relasional** (*relational algebra*) merupakan bahasa *query* prosedural, sedangkan **kalkulus relasional tuple** (*tuple relational calculus*) & **kalkulus relasional domain**(*domain relational calculus*) adalah non-prosedural. Berikut hanya akan dibahas bahasa query bukan bahasa data-manipulation yang lengkap, yaitu bahasa yang tidak hanya terdiri dari bahas *query* tetapi juga bahasa untuk memodifikasi basis data, seperti perintah ***insert*** dan ***delete*** tuple.

3.7.1 Aljabar Relasional (*Relational Algebra*)

Aljabar relasional adalah sebuah bahasa *query* prosedural yang terdiri dari sekumpulan operasi dimana masukkannya adalah satu atau dua relasi dan keluarannya adalah sebuah relasi baru sebagai hasil dari operasi tersebut. Operasi-operasi dasar dalam aljabar relasional adalah : **select**, **project**, **union**, **set difference**, **Cartesian product** dan

rename. Disamping operasi-operasi dasar terdapat beberapa operasi lainnya seperti **set intersection**, **natural join**, **division** dan **assignment**. Operasi-operasi ini akan didefinisikan dalam terminology operasi dasar.

3.7.1.1 Operasi-operasi Dasar

Operasi-operasi select, project dan rename disebut operasi **unary**, karena operasi-operasi tersebut hanya memerlukan satu relasi. Tiga operasi lainnya memerlukan sepasang relasi, disebut operasi **binary**.

a. Operasi Select

Operasi select menyeleksi tuple-tuple yang memenuhi predikat yang diberikan. Digunakan symbol sigma (σ) untuk menunjukkan operasi select. Predikat muncul sebagai subcript dari σ. Argumen relasi diberikan dalam kurung yang mengikuti σ. Jadi untuk menyeleksi tuple-tuple dari relasi *loan* dimana *branch-name*-nya adalah "Perryridge", ditulis :

$$\sigma_{\text{branch-name} = \text{"Perrydge"}}(\text{loan})$$

Jika relasi Loan adalah :

Brach-name	Loan-number	**Ammount**
Downtown	L-17	1000
Redwood	L-23	2000
Perryridge	L-15	1500
Downtown	L-14	1500
Mianus	L-93	500
Round Hill	L-11	900
Perryridge	L-16	1300

Maka hasil dari *query* di atas adalah :

Brach-name	**Loan-number**	**Ammount**
Perryridge	L-15	1500
Perryridge	L-16	1300

Contoh-contoh *query* lain dengan operasi select adalah : $\sigma_{amount>1200}$ **(loan)**, untuk menemukan tuple dengan nilai *amount* lebih dari 1200, σ **brach-name = "Perryridge" and amount >1200** **(loan)**, untuk menemukan tuple dengan branch-name = "Perryridge" dimana nilai amountnya lebih dari 1200.

b. Operasi Project

Seandainya diinginkan semua daftar *loan-number* dan *amount*, tanpa *branch-name*. Dengan operasi project dapat dihasilkan relasi ini. Operasi project disimbolkan dengan symbol phi (π). Nama-nama atribut yang diinginkan tampil ditulias sebagai subcrip π.

Contoh : π **loan-number,amount (loan)**, adalah operasi untuk menampilkan seluruh *loan-number* & *amount* tanpa *branch-name*.

Hasil operasi tersebut adalah :

Loan-number	**Amount**
L-17	1000
L-23	2000
L-15	1500
L-14	1500
L-93	500
L-11	900
L-16	1300

c. Komposisi Operasi-operasi relasional

Berikut contoh *query* yang lebih kompleks : " Temukan kastemer-kastemer yang tinggal di Horison ". Maka *query*-nya ditulis :

$$\pi_{customer\text{-}name}\ (\sigma_{customer\text{-}city="Horison"}(customer))$$

perhatikan bahwa sebagai argumen dari operasi project adalah sebuah ekspresi untuk suatu relasi. Oleh karenanya, operasi-operasi aljabar relasional dapat digabung dengan suatu ekspresi aljabar relasional.

d. Operasi Union

Operasi union dalam aljabar relasional sama halnya dengan operasi union pada aritmatika. Misal user menginginkan nama-nama semua nasabah bank yang memiliki sebuah *account* atau sebuah *loan* atau keduanya. Perhatikan bahwa relasi *customer* tidak mengandung informasi tersebut. Untuk menjawab *query* ini diperlukan informasi dari relasi *Depositor* dan relasi *Borrower*. Dengan operasi union informasi yang diinginkan dapat diperoleh dengan menulis query sebagai berikut :

$$\pi_{\text{customer-name}}(\text{borrower}) \cup \pi_{\text{customer-name}}(\text{depositor})$$

DEPOSITOR

Customer-name	Account-number
Johnson	A-101
Smith	A-215
Hayes	A-102
Turner	A-305
Johnson	A-201
Jones	A-217
Lindsay	A-222

BORROWER

Customer-name	Loan-number
Jones	L-17
Smith	L-23
Hayes	L-15
Jackson	L-14
Curry	L-93
Smith	L-11

Williams	L-17
Adams	L-16

Hasil query yang union di atas adalah :

Customer-name
Johnson
Smith
Hayes
Turner
Jones
Lindsay
Jackson
Curry
Williams
Adams

Operasi union r∪s bernilai benar jika kedua relasi memenuhi dua kondisi berikut :

1. Relasi r & s harus mempunyai jumlah atribut yang sama
2. Domain dati atribut ke-I di relasi r dan domain dari atribut ke-I dari s harus sama untuk semua i.

e. Operasi Pengurangan Himpunan (*Set Difference*)

Disimbolkan dengan tanda "-". Operasi ini untuk menemukan tuple-tuple yang berada pada satu relasi tetapi tidak berada pada relasi yang lainnya. Contoh untuk menemukan semua nasabah bank yang mempunyai account tetapi tidak mempunyai loan, ditulis :

$\pi_{\text{customer-name}}$ **(depositor) -** $\pi_{\text{customer-name}}$**(borrower)**

hasil *query* di atas :

Customer-name
Johnson
Turner
Lindsay

f. Operasi Cartesian -product

Operasi Cartesian-product disimbolkan dengan "**x**". Dengan operasi ini dapat dihasilkan informasi yang merupakan kombinasi dari dua relasi.

Secara garis besar, jika dipunyai relasi **r1(R1)** dan **r2(R2)**, maka **r1 x r2** adalah relasi yang skemanya merupakan gabungan dari **R1 (atribut-atribut relasi r1)** dan **R2 (atribut-atribut r2)**. Relasi R mengandung semua tuple t dimana terdapat sebuah tuple t1 dalam r1 dan t2 dalam r2 sedemikian sehingga t[R1] = t1[R1] dan t[R2]=t2[R2].

Contoh : Jika diinginkan nama-nama dari semua nasabah yang mempunyai pinjaman (*loan*) di bank cabang Perryridge. Maka *query*-nya ditulis :

$\sigma_{\text{branch-name="Perryridge"}}$ **(borrower x loan)**

Relasi yang dihasilkan :

Customer-name	Loan-number	Branch-name	Loan-number	Amount
Jones	L-17	Perryridge	L-15	1500
Jones	L-17	Perryridge	L-16	1300
Smith	L-23	Perryridge	L-15	1500
Smith	L-23	Perryridge	L-16	1300
Hayes	L-15	Perryridge	L-15	1500
Hayes	L-15	Perryridge	L-16	1300

Jackson	L-14	Perryridge	L-15	1500
Jackson	L-14	Perryridge	L-16	1300
Curry	L-93	Perryridge	L-15	1500
Curry	L-93	Perryridge	L-16	1300
Smith	L-11	Perryridge	L-15	1500
Smith	L-11	Perryridge	L-16	1300
Williams	L-17	Perryridge	L-15	1500
Williams	L-17	Perryridge	L-16	1300
Adams	L-16	Perryridge	L-15	1500
Adams	L-16	Perryridge	L-16	1300

g. Operasi Rename

Dilambangkan dengan symbol **rho** (ρ). Sintaks penulisan rename : ρ_{x}**(E).**

Operasi rename mengeluarkan hasil ekspresi E dengan nama x.

Contoh :

$\pi_{account.balance}$ ($\sigma_{account.balance<d.balance}$ (account x ρ_{d}(account)))

3.7.1.2 Definisi Aljabar relasional

Jika E1 & E2 adalah ekspresi aljabar-relasional, maka berikut ini adlah semua ekspresi aljabar-relasional :

- **E1** $\cup$ **E2**
- **E1 - E2**
- **E1 x E2**
- σ_{p}**(E1)**, dimana p adalah sebuah predikat untuk atribut-atribut dalam E1
- π_{s}**(E1)**, dimana s adalah daftar yang terdiri dari bebrapa atribut dalam E1
- ρ_{x}**(E1)**, dimana x adalah nama baru untuk hasil E1

3.7.2 Kalkulus relasional Tuple (*Tuple Relational Calculus*)

Tuple relational calculus adalah *query* yang non-prosedural. *Tuple relational calculus* menggambarkan informasi yang diinginkan tanpa memberi prosedurnya secara detil untuk mendapatkan informasi tersebut.

Sebuah query dalam tuple relational calculus ditulis :

$$\{\mathbf{t} \mid\mid \mathbf{P(t)}\}$$

yaitu : semua tuple t sedemikian sehingga predikat P adalah benar untuk t. dengan mengikuti notasi terdahulu, digunakan t[A] untuk menyatakan nilai tuple t pada atribut A, dan $t \in r$ untuk menyatakan bahwa tuple t berada dalam relasi r.

Contoh Query :

Misal diinginkan informasi *branch-name*, *loan-number* dan *amount* untuk pinjaman di atas $1200. Maka query-nya adalah : $\{\mathbf{t} \mid\mid \mathbf{t} \in \mathbf{loan} \wedge \mathbf{t[amount]>1200}\}$. Andai hanya diinginkan atribut *loan-number* dari relasi loan. Untuk mengekspresikan permintaan ini, diperlukan notasi "terdapatlah (there exists)" dengan notasi :

$$\exists\ \mathbf{t} \in \mathbf{r\ (Q(t))}$$

dibaca **" terdapatlah sebuah tuple t dalam relasi sedemikian sehingga bahwa predikat Q(t) adalah benar"**.

Dengan menggunakan notasi ini maka dapat ditulis *query* "Tampilkan loan number untuk setiap pinjaman yang lebih dari $1200" sebagai

$$\{\mathbf{t} \mid \exists\ \mathbf{s} \in \mathbf{loan\ (t[loan\text{-}number]=s[loan\text{-}number] \wedge s[amount]>1200}\}$$
$$\mid\mid\mid \exists$$

3.7.3 Kalkulus Relasional Domain (*The Domain Relational Calculus*)

Bentuk kalkulus relasional ini menggunakan variabel domain yang mengambil sebuah nilai dari domain atribut, bukan dari nilai seluruh tuple.

Contoh query :

- Tampilkan nama cabang, *loan number* dan jumlah pinjaman yang lebih dari $1200, dengan kalkulus relasional domain, *query*-nya ditulis :

  $\{\langle b,l,a\rangle \mid \exists \langle b,l,a\rangle \in \textbf{loan} \wedge a > 1200\}$

- Tampilkan semua *loan-number* untuk jumlah pinjaman yang lebih dari $1200, dengan kalkulus relasional domain, *query*-nya ditulis :

  $\{\langle l\rangle \mid \exists\, b,a(\langle b,l,a\rangle \in \textbf{loan} \wedge > 1200)\}$

- Tampilkan nama-nama semua customer yang mempunyai pinjaman di cabang Perryridge beserta *loan-number*-nya.

  $\{\langle c,a\rangle \mid\mid\mid \exists\, (\langle c,l\rangle \in \textbf{borrower} \wedge \exists\, b\, (\langle b,l,a\rangle \in \textbf{loan} \wedge b = \text{"Perryridge"}))\}$

- Tampilkan nama-nama semua *customer* yang mempunyai pinjaman, rekening, atau keduanya pada cabang Perryridge :

  $\{\langle c\rangle \mid\mid\mid \exists\, l\, (\langle c,l\rangle \in \textbf{borrower} \wedge \exists\, b,a\, (\langle b,l,a\rangle \in \textbf{loan} \wedge b = \text{"Perryridge"}))$
  $\vee \exists\, b,n\, (\langle b,a,n\rangle \in \textbf{account} \wedge b = \text{"Perryridge"}))\}$

- Tampilkan semua nama-nama *customer* yang mempunyai rekening pada semua cabang-cabang yang berlokasi di Brooklyn :

  $\{\langle c\rangle \mid\mid\mid \forall\, x,y,z\, (\langle x,y,z\rangle \in \textbf{branch}) \wedge y = \text{"Brooklyn"} \Rightarrow$
  $\exists\, a,b(\langle x,a,b\rangle \in \textbf{account} \wedge \langle c,a\rangle \in \textbf{depositor})\}$

3.7.4 Modifikasi Dalam Basis Data

1. Penghapusan (*Deletion*)

Dalam aljabar relasional, operasi deletion diekspresikan dengan :

r ! **r-E**, dimana r adalah relasi dan E adalah sebuah *query* aljabar relasional.

Berikut beberapa contoh dari perintah delete dengan aljabar relasional adalah :

- Hapus semua rekening atas nama Smith

  account ! *account* - $\sigma_{\textbf{customer-name = "Smith"}}$ (*account*)

- Hapus semua pinjaman dengan jumlah diantara 0 s/d 50

loan ! *loan* - $\sigma_{\text{amount > = 0 and amount <=50}}$ (*loan*)

- Hapus semua account pada cabang-cabang yang berlokasi di Needham

r1 ! $\sigma_{\text{branch -city = "Needham"}}$ (*account x branch*)

r2 ! $\pi_{\text{branch-name, account-number, balance}}$ (*r1*)

account ! *account* - *r2*

2. Penyisipan (*Insertion*)

Untuk menyisipkan data ke dalam suatu relasi, dapat dilakukan dengan dua cara yaitu menentukan tuple-tuple yang akan disisipkan. Dalam relasional aljabar, penyisipan diekspresikan dengan : **r** ! **r** $\cup$ **E**, dimana r adalah relasi dan E adalah sebuah ekspresi relasional aljabar.

Contoh : seandainya akan disisipkan fakta bahwa Smith mempunyai $1200 dalam rekening A-973 pada cabang Perryridge. Query-nya ditulis :

account ! *account* $\cup$ {("Perryridge", A-973,1200)}

depositor ! *depositor* $\cup$ {("Smith", A-973)}

Contoh lain :

r1 ! (**σ** $_{\text{branch-name = "Perryridge"}}$ (*borrower* **x** *loan*))

r2 ! $\pi_{\text{branch-name, loan-number}}$ (*r1*)

account ! *account* $\cup$ (*r2* **x** {(200)})

depositor ! *depositor* $\cup$ $\pi_{\text{customer-name, loan-number}}$ (*r1*)

3. Updating

Dapat digunakan operator proyeksi secara umum sebagai berikut :

$r ! \pi_{F1,F2,\ldots Fn} (r)$

Contoh : seandainya akan dibuat bunga tabungan sebesar 5% ditulis query :

account ! $\pi_{\text{branch-name,account-number,balance ! balance*1.05}}$ (*account*)

4. View

View didefinisikan dengan statement *create view*. Untuk mendefinisikan sebuah *view* maka *view* tersebut harus diberi nama dan *query* untuk menghasilkan *view* tersebut.

Format statement *create view* sebagai berikut :

Create view v as <ekspresi query>

Dimana ekspresi *query* adalah sebuah ekspresi *query relation-algebra* yang sah dan nama *view* direpresentasikan dengan v.

Contoh : untuk membuat view yang terdiri dari semua cabang bank dan pelanggannya

Create view *all-customer* **as**

π **branch-name, customer-name** **(***depositor* **x** *account***)**

$\cup$ π **branch-name, customer-name** **(***borrower* **x** *loan***)**