BAB 03

DESAIN BASIS DATA

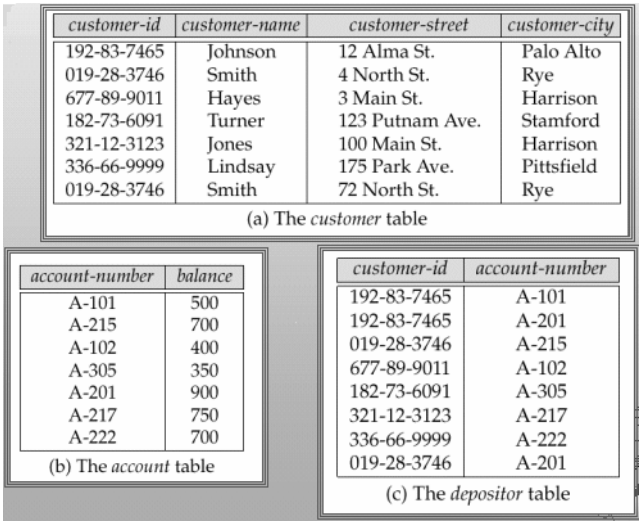
Bab tiga ini akan membahas antara lain istilah-istilah pada model relasional dan tahapan-tahapan perancangan basis data dari perencanaan basis data hingga pemeliharaan operasional, di mana setiap tahapan akan dikupas secara detail dan disertai beberapa contoh kasus.

Sebelum kita membahas desain basis data, ada baiknya kita mengetahui sekilas mengenai model relasional terlebih dahulu. Kemudian masuk pada pembahasan desain basis data yang dilengkapi dengan teknik pencarian fakta.

3.1 Istilah-Istilah Model Relasional

Ada beberapa istilah model relasional yang perlu diperhatikan, yaitu:

- *Relation* atau *table* atau *file* adalah representasikan tabel yang terdiri atas sejumlah baris dan sejumlah kolom.
- *Attribute* atau *column* atau *field* adalah kolom pada tabel.
- *Tuple* atau *row* atau *record* adalah baris pada tabel.
- *Domain* adalah himpunan nilai dari satu atau lebih *atributte.*
- *Degree* adalah banyaknya *atributte* atau kolom pada tabel.
- *Cardinality* adalah banyaknya *tuple* atau baris pada tabel.
- *Relational Basis data* adalah kumpulan relasi ternormalisasi dengan nama relasi yang jelas.



customer-id	customer-name	customer-street	customer-city
192-83-7465	Johnson	12 Alma St.	Palo Alto
019-28-3746	Smith	4 North St.	Rye
677-89-9011	Hayes	3 Main St.	Harrison
182-73-6091	Turner	123 Putnam Ave.	Stamford
321-12-3123	Jones	100 Main St.	Harrison
336-66-9999	Lindsay	175 Park Ave.	Pittsfield
019-28-3746	Smith	72 North St.	Rye

(a) The *customer* table

account-number	balance
A-101	500
A-215	700
A-102	400
A-305	350
A-201	900
A-217	750
A-222	700

(b) The *account* table

customer-id	account-number
192-83-7465	A-101
192-83-7465	A-201
019-28-3746	A-215
677-89-9011	A-102
182-73-6091	A-305
321-12-3123	A-217
336-66-9999	A-222
019-28-3746	A-201

(c) The *depositor* table

Gambar 3.1 Bagian Model Relasional

Tabel 3.1 Contoh Bagian Model Relasional

Relation	***Attribute***	***Degree***	***Tuple***	***Cardinality***
The Customer Table	• Customer-Id • Customer-Name • Customer-Street • Customer-City	4	• Baris 1 • Baris 2 • : • : • Baris 7	7
The Depositor Table	• Customer-Id • Account-Number	2	• Baris 1 • Baris 2 • : • : • Baris 8	8
The Account Table	• Account-Number • Balance	2	• Baris 1 • Baris 2 • : • : • Baris 7	7

Attribute	**Nama *Domain***	**Keterangan**	**Definisi *Domain***
Customer-Id	Customer-Id	Kumpulan seluruh Customer-Id	Character Size 4 Range: A001 – Z999
Customer-Name	Customer-Name	Kumpulan seluruh Customer-Name	Character Size 50

Customer-Street	Customer-Street	Kumpulan seluruh Customer-Street	Character Size 50
Customer-City	Customer-City	Kumpulan seluruh Customer-City	Character Size 50

Tabel 3.2 Istilah Model Relasional

Istilah Formal	Istilah Lain 1	Istilah Lain 2
Relation	*Table*	*File*
Tuple	*Row*	*Record*
Attribute	*Column*	*Field*

Relasi Basis Data

- Skema Relasi

  Nama relasi didefinisikan oleh himpunan pasangan atribut dan nama domain
- Skema Basis Data Relasional

  Himpunan skema relasi dengan nama yang berbeda

Sifat-Sifat Relasi

- Nama relasi berbeda satu sama lain dalam skema relasional
- Setiap sel (baris,kolom) dari relasi berisi satu nilai atomik atau nilai tunggal
- Setiap atribut memiliki nama yang berbeda
- Nilai suatu atribut berasal dari domain yang sama
- Setiap *tuple* berbeda dan tidak ada duplikasi *tuple*

Relational Key

- *Superkey*

  Merupakan atribut atau himpunan atribut yang mengidentifikasi secara unik *tuple-tuple* yang ada dalam relasi.
- *Candidate key*

  Memiliki ketentuan berikut:
  - *Superkey* (K) dalam relasi
  - Untuk setiap relasi R, nilai K akan mengidentifikasi secara unik *tuplenya*.
- *Composite key*

  Saat suatu *Candidate key* memiliki lebih dari 1 atribut maka akan disebut *Composite key*.

- *Primary key*

  Candidate key yang dipilih untuk identifikasi *tuple* secara unik dalam suatu relasi.

- *Foreign key*

  Atribut atau himpunan atribute dalam relasi yang dibandingkan dengan *candidate key* pada beberapa relasi.

- *Alternate key*

  Candidate key yang tidak terpilih sebagai *primary key.*

Tabel 3.3 Contoh Relational Key

Relasi	*Superkey*	*Candidate Key*	*Primary Key*	*Foreign Key*	*Alternate key*
The Customer Table	• (Customer-Id) • (Customer-Id, Customer-Name) • (Customer-Id, Customer-Name, Customer-City) dan seterusnya	• **Customer-Id** • **Customer-Name** • **Customer-Birth**	• **Customer-Id**	• Customer-City Customer-City ini termasuk contoh FK dengan syarat ketentuan: terdapat tabel City yang di dalamnya berisi *field* City-Id dengan tipe dan panjang data yang sama degnan Customer-City tentunya.	• **Customer-Name** • **Customer-Birth**
The Depositor Table	• Customer-Id • Account-Number	.....	.....	• **Customer-Id**	.....
The Account Table	• Account-Number • Balance	.....	.....	.....	.....
The City Table	• City-Id • City-Name	.....	.....	.....	.....

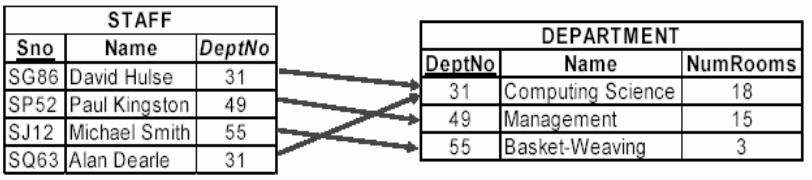
Jangan lupa lengkapi Tabel 3.3 ini sebagai latihan sampai di mana Anda memahami *Relational Key*.

Terlihat pada contoh *Superkey* dalam kondisi apa pun Customer-Id termasuk dalam *Superkey,* baik berupa atribut ataupun gabungan atribut.

- (Customer-Id)
- (Customer-Id, Customer-Name)
- (Customer-Id, Customer-Name, Customer-City)
- (Customer-Id, Customer Street, Customer-City)

Dengan melihat contoh di atas, *Superkey* memiliki kelemahan atau masalah sebagai berikut.

- Memiliki atribut yang tidak begitu dibutuhkan untuk mengidentifikasikan secara unik.
- Contohnya:
    - (Customer-Id)
    - (Customer-Id, Customer-Name) ⇒ Customer-Name tidak unik
    - (Customer-Id, Customer-Name, Customer-City) ⇒ Customer-Name, Customer-City tidak unik
    - (Customer-Id, Customer Street, Customer-City) ⇒ Customer-Street, Customer-City tidak unik dan tidak dibutuhkan



STAFF		
Sno	Name	*DeptNo*
SG86	David Hulse	31
SP52	Paul Kingston	49
SJ12	Michael Smith	55
SQ63	Alan Dearle	31

DEPARTMENT		
DeptNo	Name	NumRooms
31	Computing Science	18
49	Management	15
55	Basket-Weaving	3

Gambar 3.2 Contoh Primary Key dengan Foreign Key

Relational Integrity

- *Null*

  Karekteristik *Null* adalah:
    - Representasi nilai atribut yang tidak diketahui atau tidak digunakan dalam *tuple*
    - Berkaitan dengan ketidaklengkapan atau pengecualian data
    - Representasi tidak adanya suatu nilai dan tidak sama dengan nol atau spasi
- *Entity Entegrity*

  Pada relasi dasar, tidak ada atribut ataupun *primary key* yang bernilai *NULL*.
- *Referential Entegrity*

  Jika terdapat *foreign key* dalam suatu relasi, maka nilai *foreign key* tersebut akan dibandingkan dengan nilai *candidate key* dari beberapa *tuple* pada relasi itu sendiri atau nilai foreign key harus *NULL* seluruhnya.
- *Enterprise Constraint*

  Aturan tambahan yang dispesifikasikan oleh *user* atau DBA.

Tabel 3.4 Contoh Relational Integrity

Null	*Entity Entegrity*	*Referential Entegrity*	*Enterprise Constraint*
• Field Customer-Description pada baris ke dua (lihat Tabel 3.5)	• *Field* Customer-Id pada tabel The Customer Table (lihat Gambar 3.1)	• Field Customer-Id pada tabel The Depositor Table (lihat Gambar 3.1)	• Jika jumlah Account-number untuk setiap Customer-Id diberi batas. Misalnya: 1 Customer-Id = 5 Account-Number, maka jika lebih dari 5 tidak diper-bolehkan.

Tabel 3.5 Contoh Null

Customer-Id	Customer-Name	Customer-Street	Customer-City	Customer-Description
123-12-123	Martin	Jl. S. Parman	Jakarta	Ada ATM
123-12-124	Indrajani	Jl.Kav. DKI	Jakarta	
123-12-125	Lusi	Jl. Palmerah	Jakarta	Belum ada ATM

Penjelasan Tabel 3.4 adalah sebagai berikut.

Contoh Entity Entegrity

- Tidak diperkenankan kita meng-INSERT data pada The Customer Table dengan *null* untuk *field* Customer-Id-nya.

Contoh *Referential Entegrity*

- Tidak diperkenankan kita meng-INSERT data pada The Depositor Table dengan Customer-Id yang tidak terdapat pada The Customer Table. Contohnya: menambah data pada The Depositor Table dengan Customer-Id = ’123-12-678’ di mana Customer-Id tersebut tidak ada pada The Customer Table.

Relational Aljabar Versus Relational Calculus Versus Basis Data

Relasi Aljabar adalah suatu bahasa penulisan dengan operasi-operasi di dalamnya yang digunakan untuk melakukan satu atau lebih relasi tanpa harus mengubah relasi awalnya dan menghasilkan relasi yang baru sehingga nantinya digunakan untuk mendapatkan informasi yang diper-lukan.

Operasi Dasar Relasi Aljabar:

a) Operasi *unary* → operasi yang memakai satu relasi

- *Selection* (σ), digunakan untuk memilih baris (row) suatu relasi

  $\sigma_{predicate}(R)$ operasi seleksi bekerja pada satu relasi R dan mendefinisikan relasi yang berisi hanya tuple R yang memenuhi kondisi (*predicate*).

  Untuk *predicate* yang lebih rumit dapat dibuat menggunakan operator logikal $\wedge$ (AND), $\vee$ (OR) dan $\sim$ (NOT).

- *Projection,* (π) digunakan untuk merinci kolom

  $\pi_{a1, \ldots, an}(R)$ operasi projeksi bekerja pada satu relasi R dan mendefinisikan relasi yang berisi subset R secara vertikal, menampilkan nilai untuk atribut tertentu dan menghilangkan nilai atribut yang ganda.

b) Operasi *binary* → operasi yang memakai dua atau sepasang relasi

- *Union*

  R $\cup$ S menyatukan dua relasi R dan S, mendefinisikan relasi yang berisi seluruh tuple R atau S atau keduanya R dan S serta menghilangkan nilai atribut yang ganda.

  Atribut (R) = Atribut (S).

- *Intersection*

  R $\cap$ S, operasi yang mendefinisikan suatu relasi yang berisi sekumpulan tuple yang ada di R dan S.

  Atribut (R) = Atribut (S).

- *Set Difference*

  R $-$ S, operasi yang mendefinisikan suatu relasi yang berisi tuple yang ada di relasi R tapi tidak terdapat di relasi S.

  Atribut (R) = Atribut (S).

- *Cartesian Product*

  Cross-product atau *cartesian product*: R $\times$ S, operasi yang menghasilkan relasi yang merupakan gabungan setiap *tuple* pada relasi R dengan setiap *tuple* pada relasi S.

Join Operation

Memiliki karekteristik sebagai berikut:

- Join merupakan turunan dari *Cartesian product.*
- Equivalen dengan fungsi *selection*, menggunakan predikat join sebagai fungsi selection pada *Cartesian Product* dari dua buah relasi.

Jenis-jenis operasi join:

- *Theta join (θ join)*

Notasi *Theta join* adalah sebagai berikut.

$$R \bowtie_F S$$

Join ini mendefinisikan relasi yang terdiri atas *tuple-tuple* predikat F dari *Cartesian product* relasi R dan S. Predikat F dapat berupa operator pembanding seperti <, ≤, >, ≥, ≠, **dan** =. *Theta join* ini dapat dituliskan menggunakan operasi dasar *selection* dan operasi *Cartesian product* menjadi bentuk:

$$R \bowtie_F S = \sigma_F(R \times S)$$

- *Equijoin* (bagian dari *Theta join*)

Theta join yang memiliki predikat F berisi sama dengan atau =.

- *Natural join*

Notasi *Natural join* adalah sebagai berikut:

$$R \bowtie S$$

Merupakan bentuk lanjutan dari *equijoin* dua relasi R dan S dengan seluruh atributnya, di mana tiap atribut hanya muncul satu kali sebagai relasi hasil *join*.

- *Outer join*

Notasi *Outer join* adalah sebagai berikut:

$$R \bowtie S$$

Dalam menghubungkan dua relasi, dapat terjadi suatu *tuple* pada relasi R tidak memiliki *tuple* yang *match* pada relasi S, sehingga *tuple* yang tidak *match* tersebut akan dieliminir.

- *Semi join*

Notasi *Semi join* adalah sebagai berikut:

$$R \rhd_F S$$

Mendefinisikan relasi yang berisikan *tuple-tuple* dari relasi R yang ada pada *join* R dan S.

Juga dapat dinotasikan dengan *Projection.*

$$R \rhd_F S \quad = \Pi_A(R \bowtie_F S)$$

- *Division*

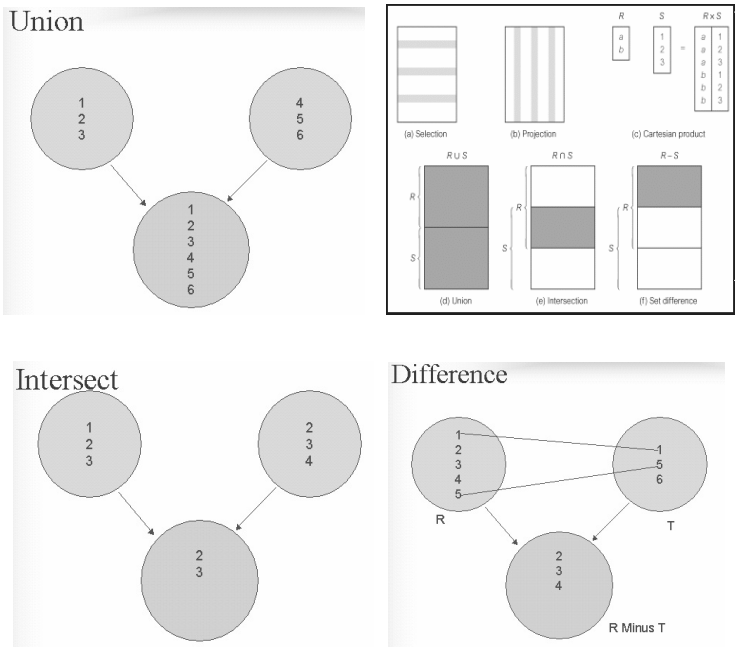
Notasi *Division* adalah sebagai berikut:

$$R \div S$$

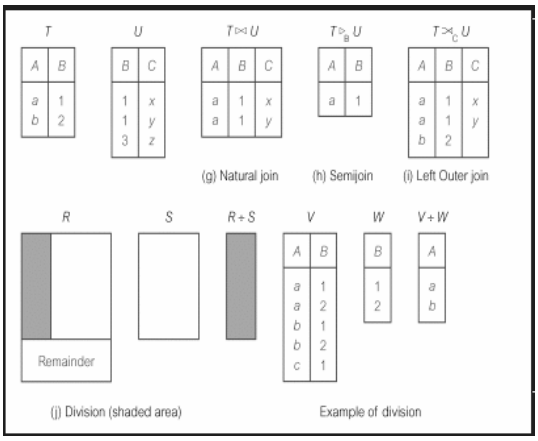
Operasi *division* R/S didefinisikan sebagai relasi pada atribut C yang berisi sekumpulan *tuple* dari relasi R yang bersesuaian dengan kombinasi tiap *tuple* yang ada di relasi S.

Tabel 3.6 Operasi Relasi Aljabar

Operation	Notation	Function
Selection	$\sigma_{predicate}(R)$	untuk memilih satu sub-set record dalam suatu relasi yang memenuhi kondisi pemilihan
Projection	$\prod a_1 \ldots\ldots a_n (R)$	Untuk memilih attribut vertikal (kolom) tertentu dari himpunan / subhimpunan dan membuang anggota yang duplikat
Union	$R \cup S$	untuk menggabungkan semua anggota relasi (R) saja atau (S) saja atau kedua-duanya dan membuang anggota yang duplikat.
Set Difference	$R - S$	untuk memilih anggota R yang bukan anggota S
Intersection	$R \cap S$	untuk menggabungkan semua anggota relasi (R) dan (S) yang sama
Cartesian Product	$R \times S$	untuk menghubungkan setiap record (R) dengan setiap record yang ada di (S)
Natural Join	$R \bowtie S$	Yakni operasi equijoin yang mana yang mana pasangan attribut-attribut yang di'join'kan **memiliki nama yang sama**
Divison	$R \div S$	untuk menggabungkan attribut C (himpunan A - B) yang terdiri dari himpunan yang berada di R yang menghubungkan kombinasi dari setiap anggota yang ada di S



Gambar 3.3 Relasi Aljabar



Gambar 3.4 Join

Untuk pembahasan contoh kasus digunakan skema basis data di bawah ini:

Siswa (IDSiswa: string, NamaSiswa: string, IPK: real)

Ruang(IDRuang: string, NamaRuang: string)

Belajar (IDSiswa: string, IDRuang: string, TanggalBelajar: date)

Tabel 3.7 Instance S1 Siswa 3 rows dan S2 Siswa 2 rows

Siswa

IDSiswa	NamaSiswa	IPK
108960	Indrajani	3,50
108100	Martin	3,85
108115	Lusi	2,75

Siswa

IDSiswa	NamaSiswa	IPK
108210	Gerry	3,60
108211	Kenny	2,10

Tabel 3.8 Instance R1 Ruang dan B1 Belajar

Ruang

IDRuang	NamaRuang
301	Kelas 301
302	Kelas 302
601	Lab Komputer
602	Lab Bahasa

Belajar

IDSiswa	IDRuang	TanggalBelajar
108960	301	10-Feb-08
108100	302	5-Feb-08
108115	301	7-Feb-08
108960	601	10-Feb-08
108115	601	5-Feb-08

Contoh beberapa kasus Relasi Aljabar:

1. Tampilkan siswa yang IPK-nya lebih besar daripada 3 dari S1 Siswa.
   Rumus: $\sigma_{IPK>3}$**(S1)**
   Hasil:

Tabel 3.9 Hasil $\sigma_{IPK>3}$(S1)

IDSiswa	NamaSiswa	IPK
108960	Indrajani	3,50
108100	Martin	3,85

Dalam bahasa basis data dapat ditulis sebagai berikut:

```
SELECT *
FROM S1
WHERE IPK > 3;
```

2. Tampilkan NamaSiswa dan IPK dari S1 Siswa.

   Rumus: $\pi_{NamaSiswa, IPK}$**(S1)**

   Hasil:

Tabel 3.10 Hasil $\pi_{NamaSiswa, IPK}$(S1)

NamaSiswa	IPK
Indrajani	3,50
Martin	3,85
Lusi	2,75

Dalam bahasa basis data dapat ditulis sebagai berikut:

```
SELECT NamaSiswa, IPK
FROM S1;
```

3. Tampilkan NamaSiswa dan IPK yang IPK nya lebih besar daripada 3 dari S1 Siswa.

   Rumus: $\pi_{IDSiswa, NamaSiswa}$ ($\sigma_{IPK>3}$ (S1))

   Hasil:

Tabel 3.11 Hasil $\pi_{IDSiswa, NamaSiswa}$ ($\sigma_{IPK>3}$ (S1))

IDSiswa	NamaSiswa
108960	Indrajani
108100	Martin

Dalam bahasa basis data dapat ditulis sebagai berikut:

```
SELECT NamaSiswa, IPK
FROM S1
WHERE IPK > 3;
```

4. Tampilkan seluruh IDSiswa yang terdapat pada *instance* Siswa S1 dan S2.

   Rumus: $\pi_{IDSiswa}(S1) \cup \pi_{IDSiswa}(S2)$

   Hasil:

Tabel 3.12 Hasil $\pi_{IDSiswa}(S1) \cup \pi_{IDSiswa}(S2)$

IDSiswa
108960
108100
108115
108210
108211

Dalam bahasa basis data dapat ditulis sebagai berikut:

```
SELECT .....
FROM .....
WHERE .....
```

5. Tampilkan seluruh IDRuang yang memiliki NamaRuang dan paling sedikitnya terdapat 1 siswa yang menggunakan ruang tersebut.

   Rumus: $\pi_{IDRuang}(B1) \cap \pi_{IDRuang}(R1)$

   Hasil:

Tabel 3.13 Hasil $\pi_{IDRuang}(B1) \cap \pi_{IDRuang}(R1)$

IDRuang
301
302
601

Dalam bahasa basis data dapat ditulis sebagai berikut:

```
SELECT .....
FROM .....
WHERE .....
```

6. Tampilkan seluruh IDRuang di mana terdapat pada Ruang dan tidak terdapat pada Belajar.

   Rumus: $\pi_{IDRuang}(R1) - \pi_{IDRuang}(B1)$

   Hasil:

Tabel 3.14 Hasil $\pi_{IDRuang}(R1) - \pi_{IDRuang}(B1)$

IDRuang
602

Dalam bahasa basis data dapat ditulis sebagai berikut:

SELECT .....
FROM .....
WHERE .....

7. Tampilkan IDSiswa, NamaSiswa, dan IPK dari S1 yang aktif belajar.
   Rumus: **S1 × B1**
   Hasil:

Tabel 3.15 Hasil S1 × B1

(IDSiswa)	NamaSiswa	IPK	(IDSiswa)	IDRuang	TanggalBelajar
108960	Indrajani	3,50	108960	301	10-Feb-08
108960	Indrajani	3,50	108100	302	5-Feb-08
108960	Indrajani	3,50	108115	301	7-Feb-08
108960	Indrajani	3,50	108960	601	10-Feb-08
108960	Indrajani	3,50	108115	601	5-Feb-08
108100	Martin	3,85	108960	301	10-Feb-08
108100	Martin	3,85	108100	302	5-Feb-08
108100	Martin	3,85	108115	301	7-Feb-08
108100	Martin	3,85	108960	601	10-Feb-08
108100	Martin	3,85	108115	601	5-Feb-08
108115	Lusi	2,75	108960	301	10-Feb-08
108115	Lusi	2,75	108100	302	5-Feb-08
108115	Lusi	2,75	108115	301	7-Feb-08
108115	Lusi	2,75	108960	601	10-Feb-08
108115	Lusi	2,75	108115	601	5-Feb-08

Dalam bahasa basis data dapat ditulis sebagai berikut:

```
SELECT  S.IdSiswa, S.NamaSiswa, S.IPK, B.IdSiswa, B.IdRuang,
        B.TanggalBelajar
FROM S1 S, B1 B
```

8. Pada Tabel 3.16 terlihat kerangkapan nilai pada beberapa *tuple*. *Cross-product* tersebut diperbaiki dengan *join condition* yang memiliki kondisi khusus, yaitu R.nama1 = S.nama2, di mana nama1 dan nama2 adalah atribut yang ada di relasi R dan S, seperti pada Tabel 3.17.
   Rumus: $S1 \bowtie_{B.IdSiswa = S.IdSiswa} B1$
   Hasil:

Tabel 3.16 Contoh *Equijoin*

(IDSiswa)	NamaSiswa	IPK	(IDSiswa)	IDRuang	TanggalBelajar
108960	Indrajani	3,50	108960	301	10-Feb-08
108960	Indrajani	3,50	108960	601	10-Feb-08
108100	Martin	3,85	108100	302	5-Feb-08
108115	Lusi	2,75	108115	301	7-Feb-08
108115	Lusi	2,75	108115	601	5-Feb-08

Dalam bahasa basis data dapat ditulis sebagai berikut:

SELECT .....
FROM .....
WHERE .....

9. Tampak perbedaan dengan contoh kasus no. 8. Contoh ini tidak menampilkan (IDSiswa) sebanyak 2 kali.

   Rumus: $S1 \bowtie B1$

   Hasil:

Tabel 3.17 Contoh *Natural Join*

IDSiswa	NamaSiswa	IPK	IDRuang	TanggalBelajar
108960	Indrajani	3,50	301	10-Feb-08
108960	Indrajani	3,50	601	10-Feb-08
108100	Martin	3,85	302	5-Feb-08
108115	Lusi	2,75	301	7-Feb-08
108115	Lusi	2,75	601	5-Feb-08

Dalam bahasa basis data dapat ditulis sebagai berikut:

SELECT .....
FROM .....
WHERE .....

10. Relasi A adalah relasi yang berisi IDRuang yang ditempati siswa. A/B1 memperhitungkan IDSiswa yang menempati semua IDRuang yang ada pada instance relasi B1.

    Rumus: **R:S**

    Hasil:

Tabel 3.18 Contoh *Division*

A

IDSiswa	IDRuang
108960	301
108960	302
108960	601
108960	602
108100	301
108100	302
108115	302
108115	602

B1

IDRuang
302

B2

IDRuang
302
602

B3

IDRuang
301
302
602

A/B1

IDSiswa
108960
108100
108115

A/B2

IDSiswa
108960
108115

A/B3

IDSiswa
108960

Dalam bahasa basis data dapat ditulis sebagai berikut:

SELECT .....
FROM .....
WHERE .....

Relasi *Calculus* adalah relasi yang lebih menspesifikasikan apa yang harus ditampilkan daripada bagaimana menampilkannya.

Contoh notasi relasi *Calculus*:

Tampilkan semua tuple S dengan formula F(S) dan nilai kebenarannya true, maka ditulis: {S| F(S)}.

Contoh kasus relasi *Calculus*:

Tampilkan IDSiswa, NamaSiswa untuk semua siswa dengan IPK lebih besar dari 3", ditulis {S|Siswa(S) $\wedge$ S.IPK > 3}.

S.IPK berarti nilai atribut IPK pada tuple variabel S. Untuk mendapatkan atribut tertentu seperti IPK dapat ditulis: {S.IPK|Siswa(S) $\wedge$ S.IPK > 3}.

Terdapat dua *quatifier* yang dapat digunakan untuk menulis formula:

- *Existential quntifier* **($\exists$)**

  Digunakan pada formula yang harus bernilai benar untuk paling sedikit satu instance.

  Sebagai contoh:

  Siswa(S) $\wedge$ ($\exists$B) (Belajar(B) $\wedge$ (B.IDSiswa = S.IDSiswa) $\wedge$ B.IDRuang = '301')

  Artinya "Terdapat tupel Belajar B yang memiliki nilai IDSiswa yang sama dengan dengan IDSiswa pada tuple Siswa S, dan dengan IDRuang 301".

  Dalam bahasa basis data dapat ditulis sebagai berikut:

  SELECT .....
  FROM .....
  WHERE .....

- *Universal quantifier* **($\forall$)**

  Digunakan untuk formula yang harus bernilai benar untuk semua instance.

  Sebagai contoh:

  ($\forall$B)(B.IDRuang $\neq$ '301')

Artinya “Untuk semua tuple Belajar yang IDRuang-nya bukan 301”.

Dalam bahasa basis data dapat ditulis sebagai berikut:

SELECT .....
FROM .....
WHERE .....

Kita dapat menggunakan aturan De Morgan untuk existential dan universal quantifier.

$(\exists X)(F(X)) \equiv \sim(\forall X)(\sim(F(X)))$

$(\forall X)(F(X)) \equiv \sim(\exists X)(\sim(F(X)))$

$(\exists X)(F_1(X) \wedge F_2(X)) \equiv \sim(\forall X)(\sim F_1(X)) \vee \sim(F_2(X)))$

$(\forall X)(F_1(X) \wedge F_2(X)) \equiv \sim(\exists X)(\sim F_1(X)) \vee \sim(F_2(X)))$

Dengan aturan De Morgan formula $(\forall B)(B.IDRuang \neq \text{‘301’})$ dapat ditulis sebagai berikut:

$\sim(\exists B)(B.IDRuang = \text{‘301’})$, yang artinya ”Tidak terdapat IDRuang dengan IDRuang 301”.

Dalam bahasa basis data dapat ditulis sebagai berikut:

SELECT .....
FROM .....
WHERE .....

Tuple variabel yang dibatasi oleh $\forall$ atau $\exists$ disebut variabel terikat, sebaliknya tuple variabel yang tidak dibatasi oleh $\forall$ atau $\exists$ disebut variabel bebas, yang pada relational *Calculus* berada di sisi kiri tanda |. Sebagai contoh adalah:

{S.NamaSiswa, S.IPK | Siswa(S) $\wedge$ ($\exists$B) (Belajar(B) $\wedge$ (B.IDSiswa = S.IDSiswa) $\wedge$ (B.IDRuang = ’301’)}

S adalah variabel bebas dan S terikat secara berurutan dengan tiap tuple Siswa.

Contoh tuple relational *Calculus* lainnya:

Tampilkan ruang yang tidak digunakan untuk belajar.

{R.IDRuang, R.NamaRuang | Ruang(R) $\wedge$ (~($\exists$B)(Belajar(B) $\wedge$ (R.IDRuang = B.IDRuang)))}

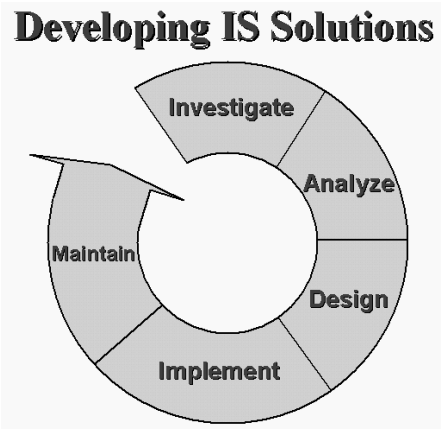
Dalam bahasa basis data dapat ditulis sebagai berikut:

SELECT .....
FROM .....
WHERE .....

Jika Anda telah membaca Bab 7 tentang SQL DML, sebaiknya Anda lengkapi beberapa isian seperti no. 4, 5, 6, 8, 9, 10 pada relasi aljabar dan juga relasi kalkulus sebagai latihan. Gunakan sintaks INNER JOIN, LEFT JOIN, dan sebagainya.

3.2 Tahapan Perancangan Basis Data

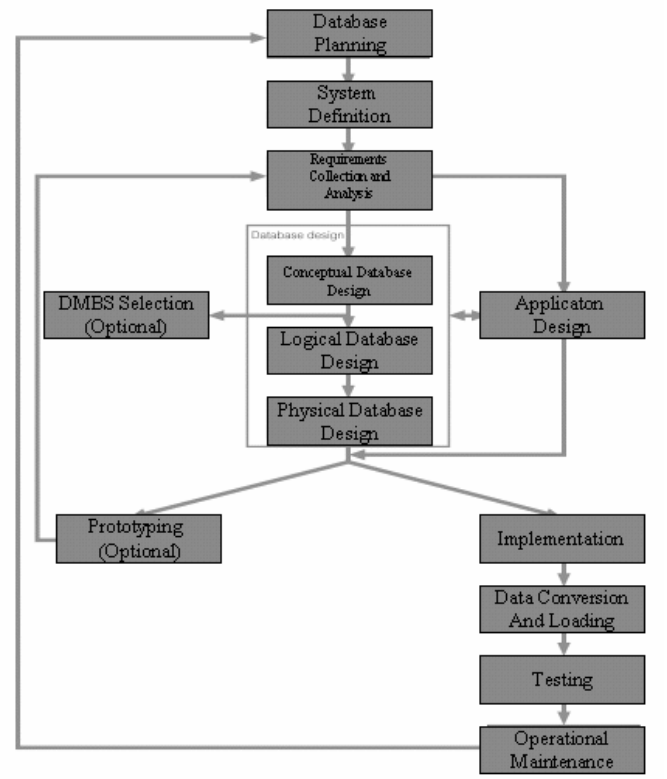
Tak dapat dipungkiri lagi bahwa ada hubungan yang erat antara sistem informasi dengan basis data. Basis data merupakan komponen mendasar suatu sistem informasi, di mana pengembangan atau penggunaannya harus dilihat dari perspektif yang lebih luas berdasarkan kebutuhan organisasi. Dengan sistem informasi, memungkinkan terjadi proses pengumpulan data, pengaturan, pengawasan, dan penyebaran informasi ke seluruh organisasi. Berikut tahapan pengembangan sistem informasi.



Gambar 3.5 Tahapan Pengembangan Sistem Informasi

Anda dapat bandingkan dengan tahapan perancangan basis data yang akan dijelaskan pada halaman berikut ini.

Secara garis besar tahapan perancangan basis data dapat Anda lihat pada gambar berikut.



Gambar 3.6 Tahapan Basis Data

***Database Planning* (Perencanaan Basis Data)**

Merupakan aktivitas manajemen untuk merealisasikan tahapan *Database Application Lifecycle* secara efektif dan efisien. Perencanaan basis data mencakup cara pengumpulan data, format data, dokumentasi yang diperlukan, cara membuat desain, dan implementasi. Perencanaan Basis data terintegrasi dengan keseluruhan strategi sistem informasi organisasi.

Terdapat 3 hal yang berkaitan dengan strategi sistem informasi, yaitu:

- Identifikasi rencana dan sasaran organisasi termasuk mengenai sistem informasi yang dibutuhkan.
- Evaluasi sistem informasi yang ada untuk menetapkan kelebihan dan kekurangan yang dimiliki oleh sistem tersebut.
- Penaksiran kesempatan teknik informatika yang mungkin memberikan keuntungan kompetitif.

Metodologi untuk mengatasi hal tersebut terbagi atas:

- Mendefinisikan *mission statement* untuk sistem basis data.

  Dalam *mission statement* didefinisikan tujuan utama pembuatan basis data. *Mission statement* membantu menjelaskan tujuan proyek basis

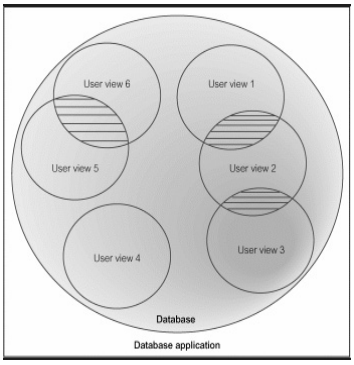
data dan memberikan tahapan yang jelas, efektif, dan efisien dari aplikasi basis data.

- Mendefinisikan *mission objectives*.

  Tiap objek mengidentifikasikan kembali tugas-tugas tertentu yang harus didukung basis data. Dapat juga disertai dengan beberapa informasi tambahan yang menjelaskan pekerjaan yang harus diselesaikan, sumber daya yang digunakan, dan biaya untuk membiayai hal tersebut.

***System Definition* (Definisi Sistem)**

Definisi sistem bertujuan untuk mendeskripsikan batasan dan ruang lingkup aplikasi basis data serta sudut pandang *user* yang utama. Aplikasi basis data seharusnya memiliki satu atau lebih *user views*. *User view* mendefinisikan apa yang diharapkan dari aplikasi basis data berdasarkan peranan pekerjaan seperti manajer dan *supervisor* atau berdasarkan area aplikasi perusahaan seperti pemasaran, personalia dan pengendalian persediaan. Mengidentifikasi *user view* membantu untuk memastikan agar tidak ada pengguna basis data yang terlupakan dan mengetahui apa yang diinginkan pengguna saat aplikasi baru akan dibuat. Selain itu, *user view* juga membantu dalam mengembangkan aplikasi basis data yang rumit dan dapat menguraikannya menjadi sub-sub bagian yang lebih sederhana.



Gambar 3.7 Basis Data dengan User Views

Requirement Collection and Analysis
(Analisis dan Pengumpulan Kebutuhan)

Merupakan proses mengumpulkan dan menganalisa informasi tentang organisasi yang akan didukung oleh aplikasi basis data dan menggunakan informasi tersebut untuk mengindentifikasikan kebutuhan user terhadap sistem yang baru.

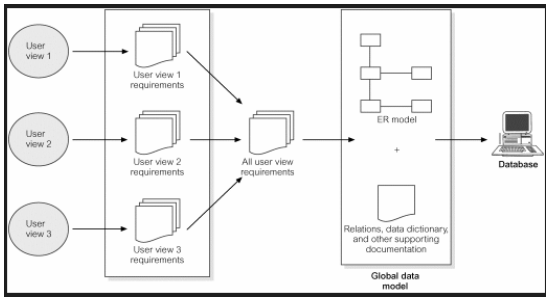
Informasi yang dikumpulkan dapat berupa deskripsi data yang digunakan atau dihasilkan, detail bagaimana data digunakan atau dihasilkan, dan beberapa kebutuhan tambahan untuk aplikasi basis data yang baru.

Informasi tersebut dianalisa untuk mengidentifikasikan kebutuhan *user* dan diharapkan tersedia pada aplikasi basis data yang baru.

Aktivitas penting lainnya dalam tahap ini adalah memastikan bagaimana menangani aplikasi basis data dengan *multiple user views*. Ada tiga macam pendekatan yang bisa digunakan dalam hal ini, yaitu:

1. *Centralized approach*

   Kebutuhan untuk tiap pengguna dibuat ke dalam satu *Set of Requirement* dan model data global dibuat berdasarkan hal itu.

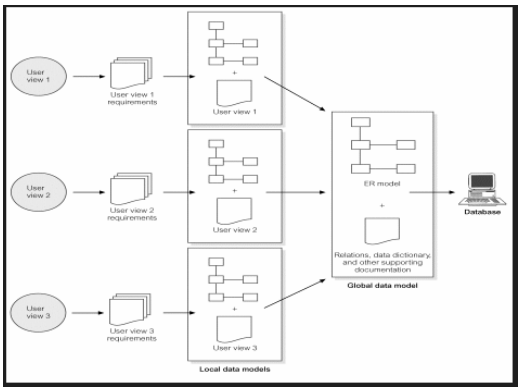


Gambar 3.8 Pendekatan Sentralisasi

Setiap *user view* memiliki kebutuhan-kebutuhan yang berbeda di mana seluruh kebutuhan tersebut akan dikumpulkan dan dibuatkan menjadi suatu *global data model* yang nantinya diperlukan dalam pembuatan basis data.

2. *View integration approach*

   Kebutuhan tiap *user view* dibuat dalam model data yang terpisah. Model data yang menggambarkan *single user view* disebut model data lokal, disusun dalam bentuk diagram dan dokumentasi yang mendeskripsikan kebutuhan *user view* basis data. Model data lokal ini kemudian digabungkan untuk menghasilkan model data global, yang menggambarkan seluruh *user view* untuk basis data.



Gambar 3.9 Local Data Model dan Global Data Model

3. Gabungan antara kedua pendekatan tersebut.

***Database Design* (Desain Basis Data)**

Desain basis data adalah proses membuat desain yang akan mendukung operasional dan tujuan perusahaan. Tujuan desain basis data adalah:

- Menggambarkan relasi data antara data yang dibutuhkan oleh aplikasi dan *user view.*
- Menyediakan model data yang mendukung seluruh transaksi yang diperlukan.
- Menspesifikasikan desain dengan struktur yang sesuai dengan kebutuhan sistem.

Ada beberapa pendekatan yang dapat digunakan dalam mendesain basis data, yaitu:

- *Top-down*

  Diawali dengan membuat data model. Pendekatan *top-down* dapat diilustrasikan menggunakan *entity-relationship* (ER) model yang *high level*, kemudian mengidentifikasikan *entity,* dan *relationship* antar *entity* organisasi. Pendekatan ini sesuai bagi basis data yang kompleks.

- *Bottom-up*

  Dimulai dari level dasar *attribute* (*properti entity* dan *relationship*), menganalisa hubungan antar *attribute*, mengelompokkannya dalam suatu relasi yang menggambarkan tipe *entity* dan relasi antara *entity*. Pendekatan ini sesuai bagi basis data dengan jumlah *attribute* yang sedikit.

- *Inside-out*

  Mirip seperti pendekatan *bottom-up*, perbedaannya adalah pada tahap awal mengidentifikasi *major entity* lalu menguraikannya menjadi *entity-entity* relasi dan *attribute-attribute* yang berhubungan dengan *major entity.*

- *Mixed*

  Menggunakan pendekatan *Bottom-up* dan *Top-down*.

Data Modelling

- Untuk memahami arti atau semantik data.
- Untuk memudahkan komunikasi mengenai informasi yang dibutuhkan.

Membuat data model membutuhkan jawaban dan pertanyaan tentang *entities, relationships,* dan *attributes.* Model data memastikan kita memahami:

- Setiap *user* perspektif terhadap data.

- Sifat data itu sendiri, tidak tergantung terhadap representasi fisiknya.
- Kegunaan data melalui *user view*.

Kriteria untuk menghasilkan model data yang optimal adalah sebagai berikut:

- Validitas struktural

  Kondisi di mana harus konsisten dengan definisi *enterprise* dan informasi organisasi.
- Kesederhanaan

  Mudah dimengerti baik oleh profesional sistem informasi maupun pengguna non teknik.
- Ketepatan

  Kemampuan untuk membedakan antara data yang berlainan dan hubungan antara data dengan batasan-batasan.
- Tidak rangkap

  Pengeluaran informasi yang tidak berhubungan dengan data lain dan representasi setiap bagian informasi hanya satu kali.
- Digunakan bersama

  Tidak ditentukan untuk aplikasi atau teknologi tertentu dan dapat digunakan oleh banyak pengguna.
- Perluasan penggunaan

  Kemampuan untuk menyusun dan mendukung kebutuhan baru dengan akibat sampingan yang minimal terhadap pengguna yang telah ada.
- Integritas

  Konsistensi dengan cara yang digunakan oleh *enterprise* dan pengaturan informasi.
- Representasi Diagram

  Kemampuan untuk merepresentasikan model menggunakan notasi diagram yang mudah dimengerti.

Ada tiga fase dalam membuat desain basis data, yaitu:

1. *Conseptual Database Design*

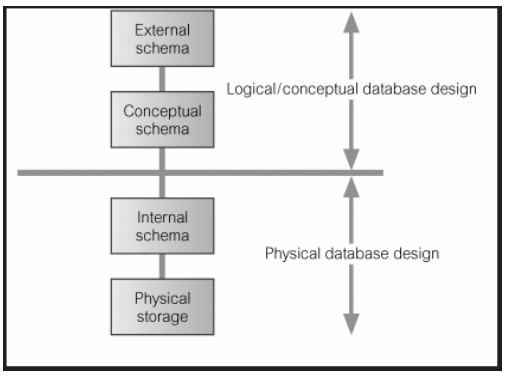
   Merupakan suatu proses pembentukan model yang berasal dari informasi yang digunakan dalam perusahaan yang bersifat inde-penden dari keseluruhan aspek fisik. Model data tersebut dibangun menggunakan informasi dalam spesifikasi kebutuhan *user* dan merupakan sumber informasi untuk fase desain logikal.
2. *Logical Database design*

   Merupakan suatu proses pembentukan model yang berasal dari informasi yang digunakan dalam perusahaan yang berdasarkan model

data tertentu namun independen terhadap DBMS tertentu dan aspek fisik lainnya. Misalnya relasional. Model data konseptual yang telah dibuat sebelumnya, diperbaiki dan dipetakan kembali ke dalam model data logikal.

3. *Physical Database design*

   Merupakan proses yang menghasilkan deskripsi implementasi basis data pada penyimpanan sekunder. Menggambarkan struktur penyimpanan dan metode akses yang digunakan untuk mencapai akses yang efisien terhadap data. Dapat dikatakan juga, desain fisikal merupakan cara pembuatan menuju DBMS tertentu.



Gambar 3.10 Model Data dan Arsitektur ANSI-SPARC

***DBMS Selection* (Seleksi DBMS)**

Seleksi DBMS adalah kegiatan memilih DBMS yang akan digunakan dalam pembuatan basis data. Pemilihan DBMS yang tepat sangat mendukung aplikasi basis data.

Langkah utama dalam pemilihan DBMS:

- Definisikan waktu untuk melakukan studi referensi
- Catat dua atau tiga produk yang akan dievaluasi untuk digunakan
- Evaluasi produk tersebut
- Rekomendasikan produk yang dipilih dan buat laporan yang mendukungnya

***Application Design* (Desain Aplikasi)**

Desain aplikasi merupakan perancangan *user interface* dan program aplikasi yang menggunakan dan melakukan proses terhadap basis data. Desain basis data dan aplikasi dilakukan secara paralel.

Ada dua aktivitas penting yang ada di dalamnya, yaitu:

- *Transaction design*
- *User interface design*

Transaction Design

Merupakan tindakan atau serangkaian tindakan yang dilakukan oleh *single user* atau program aplikasi, yang mengakses atau mengubah isi basis data. *Transaction design* bertujuan untuk mendefinisikan dan mendokumentasikan karekteristik transaksi berlevel tinggi yang dibutuhkan dalam basis data, meliputi:

- Data yang digunakan oleh transaksi
- Karekteristik fungsional dari transaksi
- *Output* transaksi
- Kepentingan bagi pengguna
- Tingkat penggunaan yang diharapkan

Ada tiga tipe utama transaksi, yaitu:

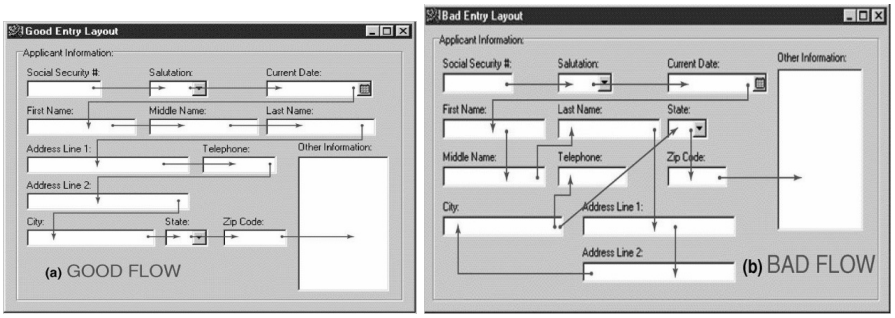
- *Retrieval*, untuk mendapatkan data guna ditampilkan pada layar atau laporan
- *Update*, untuk memasukkan *record* baru, menghapus *record* yang lama, atau memodifikasi *record* yang ada dalam basis data
- *Mixed*, gabungan antara transaksi *retrieval* dan *update*

User Interface Design

Beberapa aturan pokok dalam pembuatan *user interface* antara lain:

- Pemberian nama suatu *form* atau *report* cukup jelas dan menerangkan fungsi suatu *form* atau *report*.
- Penggunaan istilah yang sering digunakan untuk menyampaikan instruksi bagi *user* dan jika terdapat informasi tambahan yang dibutuhkan maka sebaiknya tersedia dalam *helpscreen*.
- Field yang saling berhubungan ditempatkan pada *form* atau *report* yang sama dengan urutan yang logis dan konsisten.
- Tampilan *form* atau *report* harus menarik dan sesuai dengan rencana kerja yang telah disepakati.
- Penggunaan label yang sering digunakan.
- Istilah dan singkatan yang digunakan harus konsisten.
- Penggunaan warna harus konsisten untuk setiap *form* atau *report*.
- Jumlah digit yang disediakan untuk *data entry* harus diketahui oleh *user.*
- *User* dapat dengan mudah menjalankan operasi yang diinginkan dengan menggerakan *cursor* pada *form* atau *report*.
- User dapat dengan mudah memasukkan data dan melakukan perubahan terhadap nilai *field*.
- Pesan kesalahan jika memasukkan data yang salah.

- *Field* yang tidak bersifat *mandatory* dapat terlihat jelas oleh pengguna.Misalnya tidak diberi tanda *.
- Saat user meletakkan *cursor* pada suatu *field*, keterangan mengenai *field* tersebut sebaiknya dapat terlihat.
- Terdapat indikator yang menjelaskan bahwa suatu proses telah selesai dilakukan.



Gambar 3.11 Contoh User Interface Design yang Baik dan yang Tidak Baik

***Prototyping (*Prototipe)**

Fungsinya membuat model kerja suatu aplikasi basis data. Tahapan ini bersifat opsional. Tujuan utama tahapan ini adalah:

- Untuk mengidentifkasi fitur sistem yang sedang berjalan
- Untuk memberikan perbaikan-perbaikan atau penambahan fitur baru
- Untuk klarifikasi kebutuhan *user*
- Untuk evaluasi kelayakan dan kemungkinan apa yang terjadi dari desain sistem

Terdapat dua macam prototipe yang digunakan saat ini, yaitu:

- *Requirements prototyping*

  Menggunakan prototipe untuk menentukan kebutuhan aplikasi basis data yang diinginkan dan ketika kebutuhan tersebut terpenuhi maka prototipe akan dibuang.
- *Evolutionary prototyping*

  Digunakan untuk tujuan yang sama. Perbedaannya adalah prototipe ini tidak dibuang tetapi dikembangkan untuk pengembangan selanjutnya menjadi aplikasi basis data yang digunakan.

***Implementation* (Implementasi)**

Merupakan realisasi fisik dari basis data dan desain aplikasi. Implementasi basis data dicapai dengan:

- DDL untuk membuat skema basis data dan *database files* yang kosong.
- DDL untuk membuat *user view* yang diinginkan.

- 3GL atau 4GL untuk membuat program aplikasi. Termasuk transaksi basis data yang menggunakan DML atau ditambahkan pada bahasa pemrograman.

***Data Conversion and Loading* (Konversi Data)**

Pemindahan data yang ada ke dalam basis data yang baru dan mengonversikan aplikasi yang ada agar dapat menggunakan basis data yang baru. Tahapan ini dibutuhkan ketika sistem basis data baru menggantikan yang lama. DBMS biasanya memiliki fitur untuk memanggil ulang *file* yang telah ada ke dalam basis data baru. Dapat juga mengonversi dan menggunakan program aplikasi dari sistem yang lama untuk digunakan oleh sistem yang baru.

***Testing* (Pengujian)**

Suatu proses eksekusi program aplikasi dengan tujuan untuk menemukan kesalahan dengan skenario tes yang direncanakan dan data yang sesungguhnya. Pengujian hanya akan terlihat jika terjadi kesalahan pada software.

***Operational Maintenance* (Operasional Pemeliharaan)**

Suatu proses pengawasan dan pemeliharaan sistem setelah instalasi, yang mencakup:

- Pengawasan kinerja sistem, jika kinerja menurun maka memerlukan perbaikan atau pengaturan ulang basis data.
- Pemeliharaan dan pembaharuan aplikasi basis data (jika dibutuhkan).
- Penggabungan kebutuhan baru ke dalam aplikasi basis data.

3.3 Teknik Pencarian Fakta (Fact Finding Techniques)

Fact Finding adalah proses formal menggunakan teknik seperti wawancara dan daftar pertanyaan untuk mengumpulkan fakta tentang sistem, kebutuhan, dan pilihan.

Setiap tahapan dalam siklus hidup basis data membutuhkan teknik pencarian fakta.

Tabel 3.19 Contoh Hubungan Tahapan Siklus Hidup Basis Data dengan Teknik Pencarian Fakta

Tahapan Siklus Hidup Basis Data	Contoh Data	Contoh Dokumentasi yang Dihasilkan
Perencanaan Basis Data	Tujuan dan sasaran projek basis data	Statemen misi dan sasaran sistem basis data
Definisi Sistem	Deskripsi pandangan *user* yang meliputi peranan pekerjaan atau area bisnis aplikasi	Definisi lingkup dan batasan aplikasi basis data. Definisi *user* mengenai *view* yang mendukung mereka.

Kebutuhan-kebutuhan pengumpulan dan analisa	Kebutuhan *user* akan *view* dan spesifikasi sistem yang mencakup kebutuhan kinerja dan keamanan	Spesifikasi berbagai *user* dan kebutuhan sistem
Perancangan basis data	*User* memberikan respon untuk memeriksa kembali desain logika basis data dan fungsi-fungsi yang berhubungan dengan target DBMS	Konseptual dan Logikal desain basis data seperti ER, kamus data, dan skema relasional. Rancangan fisikal basis data.
Perancangan aplikasi	*User* memberikan respon untuk memeriksa desain *user interface*	Meliputi deskripsi program dan *user interface*
Pemilihan DBMS	Fungsi-fungsi yang berhubungan dengan target DBMS	Berbagai evaluasi dan rekomendasi DBMS
Prototipe	*User* memberikan respon mengenai prototipe	Perubahan berbagai kebutuhan user dan spesifikasi sistem
Implementasi	Fungsi-fungsi yang berhubungan dengan target DBMS	
Konversi Data dan *Loading*	Format data sekarang. Kemampuan impor data.	
Pengujian	Hasil pengujian	Strategi pengujian yang digunakan dan analisa hasil pengujian
Pemeliharaan (*Optional*)	Pengujian hasil kinerja. Perubahan atau penambahan berbagai kebutuhan *user* dan sistem	Manual *user.* Analisa hasil kinerja. Perubahan berbagai kebutuhan *user* dan spesifikasi sistem.

Ada lima teknik pencarian fakta yang digunakan:

- Menguji dokumentasi

  Uji dokumentasi bermanfaat jika kita sedang berusaha mendalami kebutuhan basis data yang akan datang.

Tabel 3.20 Contoh Menguji Dokumentasi

Tujuan Dokumentasi	Contoh Dokumentasi
Deskripsi masalah dan kebutuhan basis data	Memo internal, email, dan keluhan karyawan atau pelanggan saat rapat, serta dokumen yang menjelaskan berbagai masalah. Pemeriksaan kembali kinerja dan berbagai laporan
Deskripsi bagian perusahaan yang dapat menimbulkan masalah	Struktur organisasi, statemen misi, dan rencana strategi perusahaan. Tugas dan deskripsi pekerjaan Contoh-contoh formulir dan laporan yang masih manual Contoh-contoh formulir dan laporan yang telah terkomputerisasi
Deskripsi sistem sekarang	Diagram aliran data dan bentuk-bentuk diagram lainnya. Kamus data

	Perancangan sistem basis data Dokumentasi program. Manual *user* atau pelatihan.

- Wawancara

  Teknik ini paling sering digunakan dan sangat berguna dibandingkan teknik-teknik pencarian data lainnya. Terdapat 2 jenis wawancara:

  1. Wawancara tidak terstruktur

     Wawancara tidak terstruktur dilakukan jika tujuan wawancara bersifat umum dan memiliki sedikit pertanyaan yang bersifat spesifik. Pewancara mengharapkan orang yang sedang diwawancarai itu untuk menyediakan suatu kerangka dan arah kepada pewawancara. Wawancara jenis ini banyak menimbulkan kehilangan fokus dan karena alasan itulah hasil wawancara ini tidak baik bagi analisa dan perancangan basis data.

  2. Wawancara terstruktur

     Pewawancara mempunyai banyak pertanyaan yang spesifik. Keberhasilannya tergantung pada tanggapan orang yang sedang diwawancarai dan apakah pewawancara dapat mengarahkan pertanyaan tambahan secara langsung untuk memperoleh klarifikasi atau perluasan.

     Ada dua jenis pertanyaan yang dapat diajukan yaitu pertanyaan terbuka dan pertanyaan tertutup. Perbedaannya adalah pertanyaan terbuka memperbolehkan orang yang sedang diwawancarai untuk memberikan respon pada berbagai pertanyaan yang sesuai dengan apa yang terdapat pada pikiran orang yang sedang diwawancarai. Sedangkan pertanyaan tertutup membatasi jawaban pada pilihan tertentu, singkat, dan tanggapan secara langsung.

Tabel 3.21 Keuntungan dan Kerugian Wawancara

Keuntungan	Kerugian
Memperbolehkan orang yang sedang diwawancarai untuk menjawab dengan bebas dan secara terbuka ke pertanyaan	Mahal dan sangat memakan waktu. Tidak praktis.
Memperbolehkan orang yang sedang diwawancarai untuk merasakan bagian dari proyek	Kesuksesan tergantung pada keterampilan komunikasi pewawancara.
Memperbolehkan pewawancara untuk mengikuti komentar menarik yang dibuat oleh orang yang sedang diwawancarai	Kesuksesan dapat bergantung pada kesediaan orang yang sedang diwawancarai untuk mengambil bagian dari wawancara.
Memperbolehkan pewawancara menyesuaikan atau mengulang kata yang dipertanyakan ke pewawancara	
Memperbolehkan pewawancara untuk mengamati bahasa tubuh orang yang sedang diwawancarai	

- Observasi

  Pengamatan adalah salah satu teknik pencarian data yang paling efektif untuk pemahaman suatu sistem.

Tabel 3.22 Keuntungan dan Kerugian Observasi

Keuntungan	Kerugian
Memperbolehkan kebenaran fakta dan data untuk diperiksa.	Orang-orang dapat mengetahui atau tidak mengetahui ketika mereka sedang diamati. Jika mereka mengetahui sedang diamati, maka mereka dapat melakukan suatu hal yang berbeda.
Pengamat dapat melihat secara nyata, apa yang dilaksanakan	Dapat ketinggalan pengamatan ketika terdapat perbedaan tingkat kesulitan dan jumlah pekerjaan dalam satu periode.
Pengamat dapat memperoleh gambaran data suatu lingkungan fisik suatu tugas.	Beberapa tugas tidak dapat selalu dilakukan cara yang sama di mana mereka diamati
Relatif murah	Dapat menjadi tidak praktis.
Pengamat dapat melakukan pengukuran kerja.	

- Riset

  Riset aplikasi dan masalah, jurnal komputer, buku petunjuk, dan *internet* seperti *bulletin board* adalah sumber informati yang baik dan dapat menyediakan informasi mengenai bagaimana orang lain telah memecahkan masalah.

Tabel 3.23 Keuntungan dan Kerugian Riset

Keuntungan	Kerugian
Dapat menghemat waktu jika solusi telah ada.	Memakan waktu lama.
Dapat melihat bagaimana orang lain telah memecahkan permasalahan serupa atau menjumpai kebutuhan serupa	Memerlukan akses ke sumber informasi yang sesuai dengan masalah.
Menyimpan hasil riset terbaru	Tidak dapat memecahkan masalah karena masalah tidak didokumentasikan pada tempat dengan baik.

- Kuesioner

  Adalah teknik pencarian data dengan melakukan survei melalui daftar pertanyaan.

  Terdapat 2 jenis pertanyaan dalam kuesioner:

  1. *Free format*

     Free format memberikan kebebasan responden untuk menjawab pertanyaan.

2. *Fix format*.

   Fix format memerlukan tanggapan spesifik dari individu, dengan apa pun pertanyaan, responden harus memilih jawaban yang tersedia.

Tabel 3.24 Keuntungan dan Kerugian Kuesioner

Keuntungan	Kerugian
Orang-orang dapat melengkapi dan mengembalikan daftar pertanyaan dengan nyaman.	Jumlah responden rendah, mungkin hanya 5% sampai 10%.
Relatif murah untuk mengumpulkan data dari sejumlah besar orang-orang.	Daftar pertanyaan dapat dikembalikan tidak lengkap.
Orang-orang lebih mungkin untuk menyediakan fakta nyata sebagai tanggapan yang dijaga kerahasiaannya	Tidak ada kesempatan untuk menyesuaikan atau mengulang apa yang telah ditafsirkan keliru.
Tangapan dapat dibuat tabel dan dianalisa secara cepat.	Tidak dapat mengamati dan meneliti bahasa tubuh responden.
	Perlu waktu untuk membuat daftar pertanyaan.

3.4 Contoh Kasus Pertama

Toko komputer **"Anugrah"** adalah sebuah toko yang melayani pembelian komputer dan komponen-komponennya dengan menyediakan berbagai merk dan jenis. Setiap komponen PC memiliki harganya tersendiri tergantung jenis komponennya. Karena kegiatan operasional toko ini terlalu banyak, maka pihak toko merasa sangat perlu membuat suatu basis data yang berguna untuk menyimpan seluruh transaksi yang terjadi pada toko tersebut, sehingga dapat meningkatkan efektivitas kinerja toko tersebut. Berikut ini basis data dari toko komputer **"Anugrah"**.

Nama Tabel: MsCustomer

Primary Key: **CustomerID**

Keterangan: Tabel ini berisi data pelanggan

Tabel 3.25 Tabel MsCustomer

Nama Field	Tipe Data	Length	Keterangan
CustomerID	Char	5	Harus isi dan panjang = 5, harus diawali dengan 'C'
CustomerName	Varchar	20	-
CustomerAddress	Varchar	50	-
CustomerPhone	Char	11	Harus angka dan panjangnya max 11

Nama Tabel: MsMotherboard

Primary Key: **MotherboardID**

Keterangan: Tabel ini berisi data produk Motherboard

Tabel 3.26 Tabel MsMotherboard

Nama Field	Tipe Data	Length	Keterangan
MotherboardID	Char	5	Harus isi dan panjang = 5, harus diawali 'MB'
CpuSocket	Int	-	Harus isi
MemorySocket	Char	4	Harus isi dan panjang = 4, harus diawali dengan 'DDR'
VgaSlot	Varchar	4	Harus isi
Stock	Int	-	-

Nama Tabel: MsCPU

Primary Key: **CPUID**

Keterangan: Tabel ini berisi data produk CPU

Tabel 3.27 Tabel MsCPU

Nama Field	Tipe Data	Length	Keterangan
CPUID	Char	5	Harus isi dan panjang = 5, harus diawali dengan 'CP'
Core	Varchar	10	-
Dual-Core	Varchar	3	Hanya boleh berisi 'YES' atau 'NO'
Architecture(nm)	Int	-	-
Clock(GHz)	Int	-	-
FSB(MHz)	Int	-	-
Multiplier	Varchar	5	-
Cache(MB)	Int	-	-
Stock	Int	-	-

Nama Tabel: MsMemory

Primary Key: **MemoryID**

Keterangan: Tabel ini berisi data produk Memory

Tabel 3.28 Tabel MsMemory

Nama Field	Tipe Data	Length	Keterangan
MemoryID	Char	5	Harus isi dan panjang = 5, harus diawali dengan 'M'
Clock(MHz)	Int	-	-
Timing	Varchar	8	-
Capacity(MB)	Int	-	-
Stock	Int	-	-

Nama Tabel: MsVGA

Primary Key: **VGAID**

Keterangan: Tabel ini berisi data produk VGA

Tabel 3.29 Tabel MsVGA

Nama Field	Tipe Data	Length	Keterangan
VGAID	Char	5	Harus isi dan panjang = 5, harus diawali dengan 'VG'
Core	Char	4	-
CoreClock(MHz)	Int	3	-
MemoryType	Char	4	Harus diawali dengan 'DDR'
MemoryClock(MHz)	Int	-	-
MemoryCapacity(MB)	Int	-	-
MemoryBus(bit)	Int	-	-
DirectX	Char	4	-
Stock	Int	-	-

Nama Tabel: MsBrand

Primary Key: **BrandID**

Foreign Key: **MotherboardID, CPUID, MemoryID, VGAID**

Keterangan: Tabel ini berisi data merek produk-produk yang dijual

Tabel 3.30 Tabel MsBrand

Nama Field	Tipe Data	Length	Keterangan
BrandID	Char	5	Harus isi dan panjang = 5, harus diawali dengan 'B'
MotherboardID	Char	10	Harus isi dan panjang = 5, harus diawali 'MB'
CpuID	Char	5	Harus isi dan panjang = 5, harus diawali dengan 'CP'
MemoryID	Char	5	Harus isi dan panjang = 5, harus diawali dengan 'M'
VgaID	Char	5	Harus isi dan panjang = 5, harus diawali dengan 'VG'
Brand	Varchar	20	-

Nama Tabel: Trans

Primary Key: **TransID**

Foreign Key: **CustomerID**

Keterangan: Tabel ini berisi data transaksi yang terjadi

Tabel 3.31 Tabel Trans

Nama Field	Tipe Data	Length	Keterangan
TransID	Char	5	Harus isi dan panjang = 5, harus diawali dengan 'T'
CustomerID	Char	5	Harus isi dan panjang = 5, harus diawali dengan 'C'
TransDate	Datetime	-	-

Nama Tabel: TrDetailPC

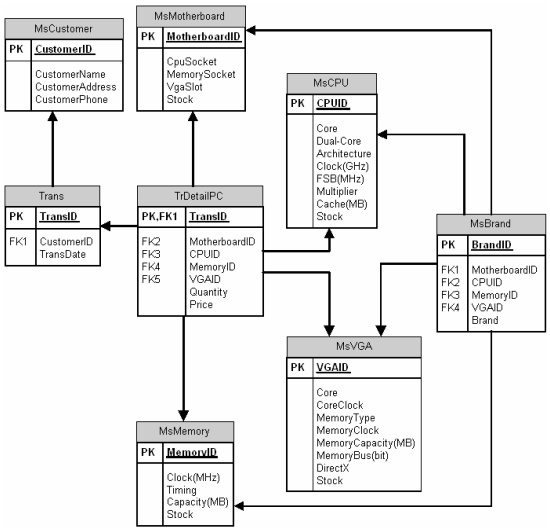
Primary Key: **TransID**

Foreign Key: **TransID, MotherboardID, CPUID, MemoryID, VGAID**

Keterangan: Tabel ini berisi data detail transaksi yang terjadi

Tabel 3.32 Tabel TrDetailPC

Nama Field	Tipe Data	Length	Keterangan
TransID	Char	5	Harus isi dan panjang = 5, harus diawali dengan 'T'
MotherboardID	Char	5	Harus isi dan panjang = 5, harus diawali 'MB'
CpuID	Char	5	Harus isi dan panjang = 5, harus diawali dengan 'CP'
MemoryID	Char	5	Harus isi dan panjang = 5, harus diawali dengan 'M'
VgaID	Char	5	Harus isi dan panjang = 5, harus diawali dengan 'VG'
Quantity	Int	-	-
Price	Decimal	10	-

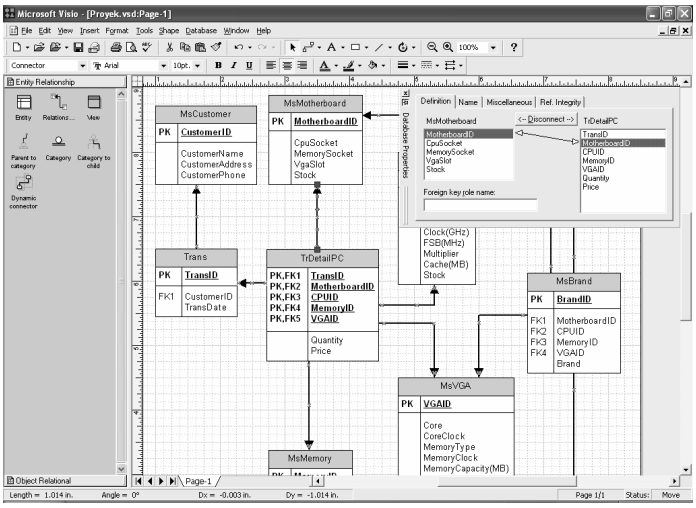


Gambar 3.12 Basis Data Relasional Toko Anugrah

Microsoft Visio 2000

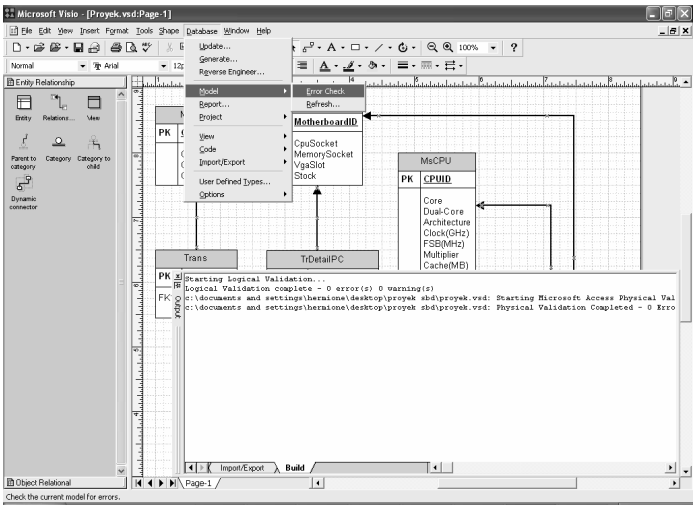
Tahapan merancang basis data dengan Microsoft Visio 2000.

1. Gunakan Microsoft Visio untuk merancang basis data relasional Anda.



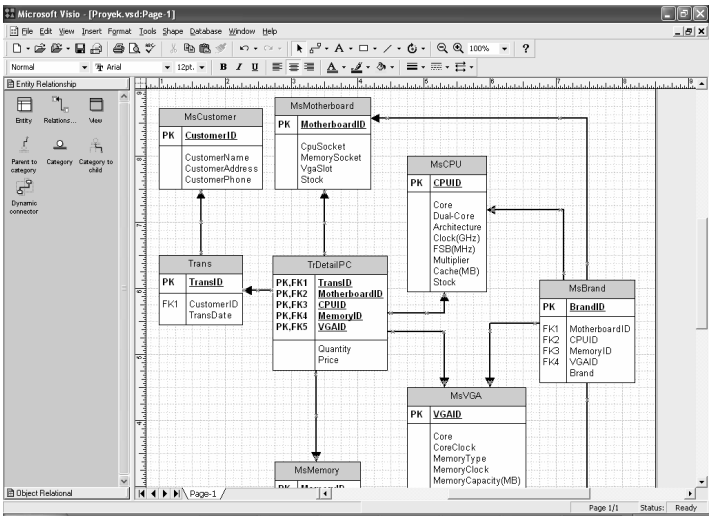
Gambar 3.13 Rancangan Basis Data Relasional

2. Validasikan Basis Data melalui menu *Database* lalu pilih *Model* kemudian *Error Check*. Akan terlihat laporan apakah rancangan basis data Anda memiliki kesalahan *logical* atau *physical.*



Gambar 3.14 Validasi Rancangan Basis Data

3. Hasil akhir desain basis data.



Gambar 3.15 Validasi Rancangan Basis Data

3.5 Contoh Kasus Kedua

Amazing Course adalah sebuah tempat kursus komputer. Mereka yang ingin kursus komputer harus melakukan registrasi terlebih dahulu. Awalnya tempat kursus ini menggunakan sistem manual, namun karena banyaknya orang yang mendaftar maka dirancanglah sebuah *database* agar mempermudah kerja para pegawainya.

Nama Tabel: MsAnggota

Primary key: **KdAnggota**

Tabel 3.33 Tabel MsAnggota

Nama Field	Tipe Data	Length	Keterangan
KdAnggota	char	5	Harus isi dan panjang=5,harus diawali dengan 'NT' dan 3 digit terakhirnya angka.
Nama	varchar	30	Harus diisi
Alamat	varchar	50	-
Telp	numeric	-	Panjangnya maximal 10

Nama Tabel: MsKasir

Primary key: **KdKasir**

Tabel 3.34 Tabel MsKasir

Nama Field	Tipe Data	Length	Keterangan
KdKasir	char	5	Harus isi dan panjang=5,harus diawali dengan 'KY' dan 3 digit terakhirnya angka
Nama	varchar	30	Harus diisi
Alamat	varchar	50	-
Telp	numeric	-	Panjangnya maximal 10

Nama Tabel: MsKursus

Primary Key: **KdKursus**

Tabel 3.35 Tabel MsKursus

Nama Field	Tipe Data	Length	Keterangan
KdKursus	char	5	Harus isi dan panjang=5,harus diawali dengan 'MK' dan 3 digit terakhirnya angka
NamaKursus	varchar	30	Harus diisi
Biaya	numeric	-	-

Nama Tabel: TrHeaderPendaftaran

Primary Key: **KdPendaftaran**

Foreign Key: **KdAnggota, KdKasir**

Tabel 3.36 Tabel TrHeaderPendaftaran

Nama Field	Tipe Data	Length	Keterangan
KdPendaftaran	char	5	Harus isi dan panjang=5,harus diawali dengan 'KP' dan 3 digit terakhirnya angka
KdKasir	char	5	-
KdAnggota	char	5	-
TanggalPendaftaran	datetime	-	-

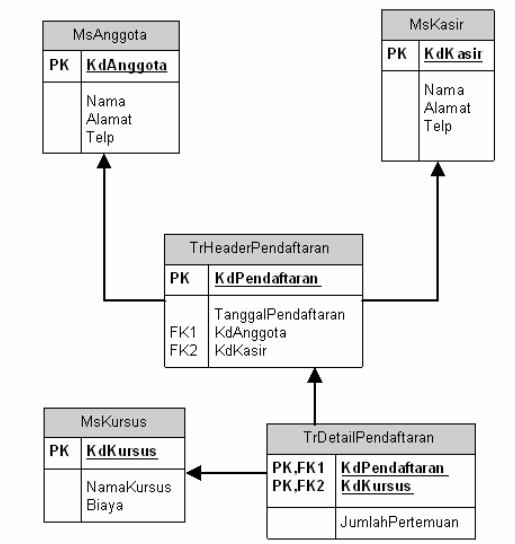
Nama Tabel: TrDetailPendaftaran

Primary Key: **KdKursus, KdPendaftaran**

Foreign Key: **KdKursus, KdPendaftaran**

Tabel 3.37 Tabel TrHeaderPendaftaran

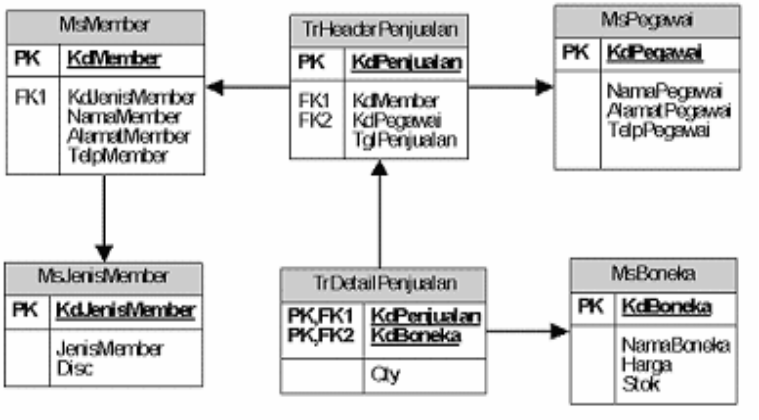
Nama Field	Tipe Data	Length	Keterangan
KdKursus	char	5	-
KdPendaftaran	char	5	-
JumlahPertemuan	int	-	Harus diisi



Gambar 3.16 Basis Data Relasional Amazing Course

3.6 Latihan

1. Sebutkan perbedaan antara *Candidate Key* dengan *Alternate Key*!
2. Dengan menggunakan tabel-tabel di bawah ini, manakah yang *primary key, foreign key, candidate key,* dan *alternate key?* Berikan alasan Anda!



Gambar 3.17 Basis Data Relasional Toko Boneka

3. Sebutkan tahapan basis data dan jelaskan secara singkat!

***