BAB 3
MODEL RELASIONAL

Mengapa perlu mempelajari Model Relasional ?

- Model basis data yang paling banyak digunakan
  - Vendors : IBM, Informix, Microsoft, Oracle, Sybase, dll.
  - Yang menjadi saingan berat akhir2 ini adalah model berorientasi obyek
    - ObjectStore, Versant, Ontos
    - Informix Unviersal Server, UniSQL, O2, Oracle, DB2

Definisi : Basis Data Relasional

- *Basis Data Relasional :* himpunan relasi
- *Relasi :* terdiri dari dua bagian :
  - *Instance :* table dengan baris dan kolom
    #baris = kardinalitas, #kolom/fields = degree/arity
  - *Skema :* menentukan nama relasi, plus nama dan tipe kolom
    - *Misal Students(sid :* string, name : string, login : string, age : integer, gpa : real).
- Suatu relasi adalah himpunan kolom atau tupel (semua barisnya bersifat distinct/unik).

Contoh : Instance dari Relasi *Students*

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

- Kardinalitas = 3, degree = 5, semua baris bersifat distinct
- Apakah semua kolom dalam instance relasi juga harus distinct ?

Bahasa Query Relasional (Relational Query Language)

- Kekuatan utama dari model relasional adalah kesederhanaannya, dan kelebihannya adalah dalam melakukan query atas data.
- Query dapat ditulis secara intuitif, dan DBMS bertanggungjawab untuk mengevaluasinya secara efisien.
  - Kunci : semantic yang tepat untuk relational query.
  - Mengijinkan pengoptimasi untuk memperluas atau mengatur kembali operasinya, dan memastikan bahwa hasil yang diperoleh tidak berubah.

SQL (Structured Query Language)

- Dikembangkan oleh IBM (system R) pada tahun 70-an
- Perlu dilakukan standardisasi sejak banyak dipakai oleh banyak vendor
- Standart SQL :
  - SQL-86
  - SQL-89
  - SQL-92
  - SQL-99

- Untuk menemukan pelajar berusia 18 tahun dari table, dapat ditulis bahasa query :

```
SELECT *
FROM Students S
WHERE S.age=18
```

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	18	3.2

SELECT *
FROM Students S
WHERE S.age=18

- Untuk menemukan field *names* dan *logins*, ganti baris pertama query tersebut dengan :
  SELECT S.name, S.login

Melakukan Query pada beberapa Tabel yang Berelasi

- Bila terdapat table *Enrolled* yang berelasi dengan table *Students* sebelumnya dengan key field *sid* :

sid	cid	grade
53831	Carnatic101	C
53831	Reggae203	B
53650	Topology112	A
53666	History105	B

- Dan diberikan query :
  SELECT S.name, E.cid
  FROM Students S, Enrolled E
  WHERE S.sid=E.sid and E.grade="A"
- Maka table yang dihasilkan dari query tersebut adalah :

S.name	E.cid
Smith	Topology112

Yaitu mencari data *Students* (nama *Students* dan mata kuliah yang diikutinya) yang mendapat nilai "A".

Membuat Relasi dalam SQL

- Tabel Students dibuat model relasionalnya dengan tipe domain pada tiap field yang terspesifikasi :

```
CREATE TABLE Students
    (sid : CHAR(20),
    name : CHAR(20),
    login : CHAR(10),
    age : INTEGER,
    gpa : REAL)
```

- Contoh lain adalah table *Enrolled* yang berisi tentang informasi mata kuliah yang diambil oleh *Students :*

```
CREATE TABLE Enrolled
(sid : char(20),
cid : char(20),
grade : char(2))
```

Menghapus suatu Relasi

- Perintah berikut :

```
DROP TABLE Students
```

  Menghapus relasi Students.

- Perintah Query berikut :

```
ADD COLUMN firstYear : integer
```

  Menambahkan satu field baru yaitu *firstYear;* tiap tuple dalam instance diperluas dengan penambahan nilai *null* pada field yang baru.

Menambah dan Menghapus Tuple

- Satu tupel tunggal dapat disisipkan dengan cara :

```
INSERT INTO Students(sid,name,login,age,gpa)
VALUES (53688,'Smith','smith@ee',18,3.2)
```

- Semua tupel yang ada dapat di-delete dengan menggunakan beberapa kondisi tertentu (missal : menghapus tupel dengan field name=Smith)

```
DELETE
FROM Students S
WHERE S.name='Smith'
```

Batasan Integritas (Integrity Constraints)

- Batasan Integritas adalah suatu kondisi yang harus bernilai benar untuk suatu instance dalam basis data, missal : batasan domain
  - Dispesifikasi saat skema didefinisikan
  - Diperiksa pada saat suatu relasi dimodifikasi
- Instance dari relasi disebut legal jika bias memenuhi semua batasan integritas (integrity constraints) yang telah dispesifikasi
  - DBMS tidak mengiinkan suatu instance yang tidak legal.
- Jika DBMS memeriksa *Integrity Constraints* , maka data yang disimpan lebih dapat mencerminkan dunia nyata (real-world meaning).
  - Juga sangat perlu dihindari kesalahan dari entry data

Batasan Kunci Primer (Primary Key Constraints)

- Himpunan suatu fields merupakan suatu key dari suatu relasi jika :

1. Tidak ada dua tupel yang distinct yang mempunyai nilai yang sama untuk semua key fields, dan
2. Key tersebut tidak memiliki subset.
   - Pernyataan 2 salah ? bagaimana dengan *superkey*
   - Jika terdapat lebih dari satu key untuk suatu relasi, maka salah satu dari key tersebut akan dipilih oleh DBA untuk menjadi primary key.

* Misal : *sid* adalah key untuk relasi *Students. (*Bagaimana dengan *name),* himpunan key (*sid,gpa*) adalah merupakan superkey.

Primary dan Candidate Key dalam SQL

- Dari kemungkinan banyak candidate keys (dispesifikasi dengan menggunakan UNIQUE), salah satunya dapat dipilih menjadi *primary key.*
- Seorang *Students* dapat mengambil suatu course dan hanya menerima satu nilai untuk *grade* dari course yang diikutinya.

```
CREATE TABLE Enrolled
( sid CHAR(20),
cid CHAR(20),
grade CHAR(2),
PRIMARY KEY (sid,cid)

CREATE TABLE Enrolled
(sid CHAR(20),
```

```
cid CHAR(20),
grade CHAR(2),
PRIMARY KEY (sid)
UNIQUE(cid,grade))
```

Foreign Keys, Referential Integrity

- Foreign key adalah himpunan fields dalam satu relasi yang digunakan untuk melakukan referensi ke tupel pada relasi yang lain (Harus berkorespondensi dengan *primary key* pada relasi yang kedua). Berlaku seperti *logical pointer*
- Missal *sid* adalah *foreign key* yang direfer dari relasi *Students* :
  - Enrolled(sid : string, cid : string, grade : string)

Foreign Keys dalam SQL

- Hanya *Students* yang terdaftar dalam relasi *Students* yang diperbolehkan untuk mengikuti suatu perkuliahan (course).

```
CREATE TABLE Enrolled
(sid CHAR(20), cid CHAR(20), grade CHAR(2),
 PRIMARY KEY(sid,cid),
 FOREIGN KEY(sid) REFERENCES Students)
```

Enrolled

sid	cid	grade
53666	Carnatic101	C
53666	Reggae203	B
53650	Topology112	A
53666	History105	B

Students

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

Mengusahakan suatu Referential Integrity

- Misal pada relasi *Students* dan *Enrolled; sid* dalam *Enrolled* adalah foreign key yang mereferensi relasi *Students*
- Apa yang harus dilakukan jika tupel *Enrolled* dengan suatu data *Students* yang tidak terdaftar dalam relasi *Students* disisipkan ? (Hindari hal ini).
- Apa yang harus dilakukan jika tupel *Students* di-hapus ?
  - Hapus juga semua tupel *Enrolled* yang merefer ke tupel *Students* yang dihapus tersebut
  - Tidak mengijinkan dilakukan penghapusan jika tupel tersebut merefer ke tupel pada relasi yang lain (alternatif lain dari yang pertama)
  - Ubah *sid* dalam tupel *Enrolled* menjadi *default sid (*alternatif yang lain lagi).
  - (Dalam SQL, juga dapat dilakukan setting pada tupel *Enrolled* yang direfer oleh tupel *Students* yang dihapus tersebut dengan memberikan nilai khusus yaitu *null,* yang artinya 'tidak diketahui' (*unknown* atau *inapplicable).*

- Sama halnya jika primary key dari tupel Students dilakukan perubahan (update).

Referential Integrity dalam SQL/92

- SQL/92 mendukung pilihan berikut untuk perintah delete dan update :
  - Default-nya adalah tidak dilakukan apa-apa (pembatalan perintah delete/update).
  - CASCADE (juga men-delete semua tupel yang merefer ke tupel yang di-delete).
  - Set nilai NULL/DEFAULT (Set nilai foreign key dari tupel yang direferensi).
- Contoh :

```
CREATE TABLE Enrolled
(sid : CHAR(20),
cid : CHAR(20),
grade : CHAR(2),
PRIMARY KEY(sid,cid),
FOREIGN KEY(sid)
REFERENCES Students
ON DELETE CASCADE
ON UPDATE SET DEFAULT)
```

View

- View hanyalah sebuah relasi, yang juga bisa menyimpan definisi dari himpunan tuple.

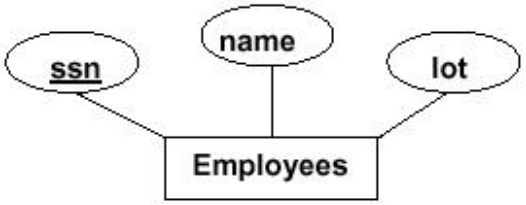
```
CREATE VIEW YoungActiveStudents(name,grade)
        AS SELECT S.name, E.grade
        FROM Students S, Enrolled E
        WHERE S.sid=E.sid and S.age<21
```

- View dapat dihapus dengan menggunakan perintah DROP VIEW

Perintah DROP TABLE dapat juga digunakan jika terdapat view pada table

Desain Logika Basis Data : Dari ER ke Relasional

- Himpunan entitas suatu table basis data :

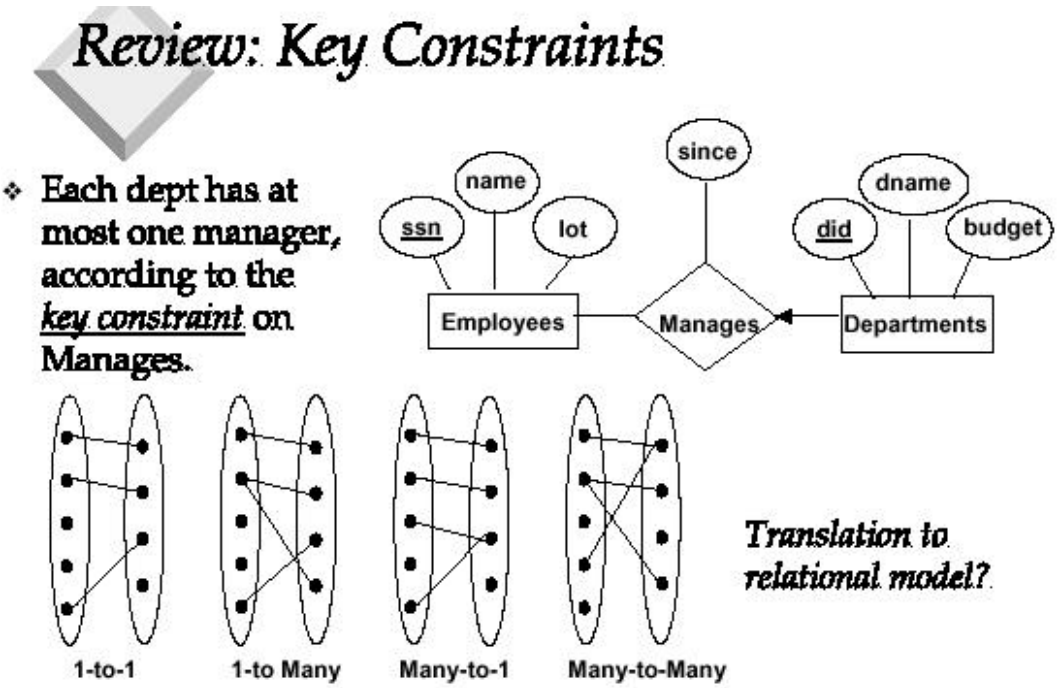



```
CREATE TABLE Employees
(ssn CHAR(11),
name CHAR(20),
lot INTEGER,
PRIMARY KEY (ssn))
```

Himpunan Relasi ke Tabel

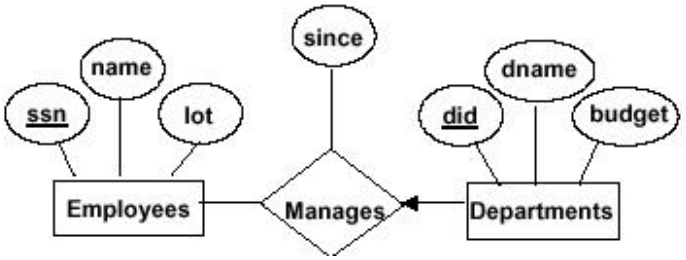
- Untuk melakukan translasi dari himpunan relasi ke relasi, atribut relasi harus termasuk :
  - Key untuk tiap himpunan entity yang berpartisipasi (sebagai foreign key).
    - Himpunan bentuk atribut ini merupakan superkey bagi relasi
  - Semua atribut deskriptif

```
CREATE TABLE Works_In)
Ssn CHAR(1)
Did INTEGER,
Since DATE,
PRIMARY KEY(ssn,did),
FOREIGN KEY (ssn)
        REFERENCES Employees,
FOREIGN KEY (did)
        REFERENCES Departements)
```

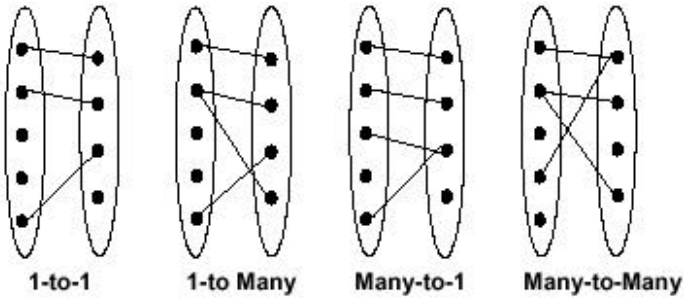


Review : Key Constraints

- Tiap departement kebanyakan mempunyai seorang manager yang berhubungan dengan key constraints pada *Manages*



- Translasi ke model relasional ?



Translasi ER Diagram dengan *Key Constraints*

- Memetakan relasi ke dalam table :

- Memisahkan table *Employee* dan *Departments*

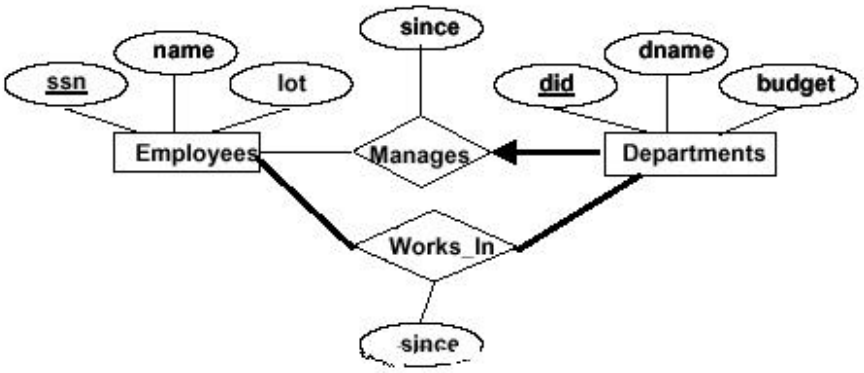
```
CREATE TABLE Manages(
  ssn CHAR(11),
  did INTEGER,
  since DATE,
  PRIMARY KEY (did),
  FOREIGN KEY (ssn) REFERENCES Employees,
  FOREIGN KEY (did) REFERENCES Departments)
```

- Jika tiap department memiliki seorang manager tertentu, maka kita kombinasikan *Manages* dan *Departments.*

```
CREATE TABLE Dept_Mgr(
  did INTEGER,
  dname CHAR(20),
  budget REAL,
  ssn CHAR(11),
  since DATE,
  PRIMARY KEY (did),
  FOREIGN KEY (ssn) REFERENCES Employees)
```

Review : *Participation Constraints*

- Apakah tiap departement memiliki seorang manager ?
  - Jika ya, maka ini disebut dengan participation constraint : yaitu partisipasi dari *Departments* dalam *Manages* yang bisa bersifat total (atau sebagian).
  - Apakah nilai dari table *Departments* harus tampak dalam kolom pada table *Manages* (yang memiliki nilai *ssn* bukan null)



Participation Constraint dalam SQL

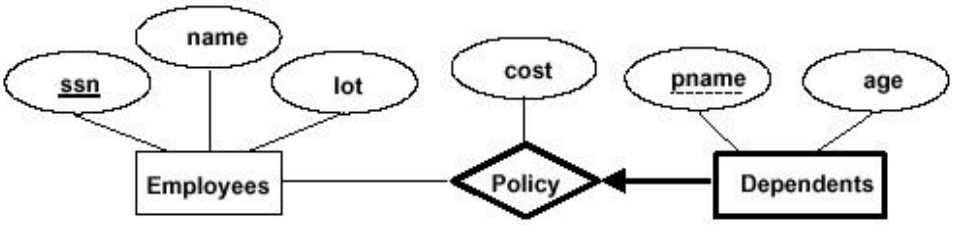
- Pada saat *participation constraint* yang ditangkap melibatkan satu himpunan entity dalam relasi binary, tapi tanpa melakukan pengurutan ulang untuk menguji constraint yang bersangkutan.

CREATE TABLE Dept_Mgr(

did INTEGER,
dname CHAR(20),
budget REAL,
ssn CHAR(11) NOT NULL,
since DATE,
PRIMARY KEY (did),
FOREIGN KEY (ssn) REFERENCES Employees,
ON DELETE NO ACTION)

Review : Entiti Lemah

- Entiti lemah (weak entity) dapat diidentifikasi secara unik hanya dengan memperhatikan primary key dari entity yang lain (owner).
  - Himpunan entity owner dan himpunan entity lemah harus berpartisipasi dalam relasi one-to-many (1 owner, dengan banyak entity lemah)
  - Himpunan entity lemah harus mempunyai partisipasi total dalam mengidentifikasi himpunan relasi ini.

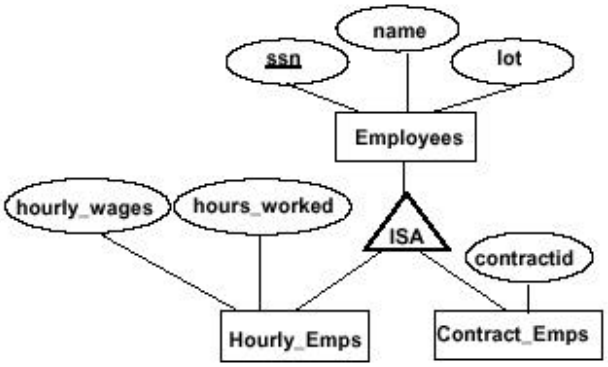


Translasi Himpunan Entiti Lemah

- Himpunan entity lemah dan himpunan relasi yang diidentifikasi dapat ditranslasikan ke dalam table tunggal.
  - Pada saat entity owner dihapus, semua entity lemah harus juga dihapus.

CREATE TABLE Dep_Policy (
Pname CHAR(20),
Age INTEGER,
Cost REAL,
Ssn CHAR(11) NOT NULL,
PRIMARY KEY (pname,ssn),
FOREIGN KEY (ssn) REFERENCES Employees,
ON DELETE CASCADE)

Review : Hirarki ISA

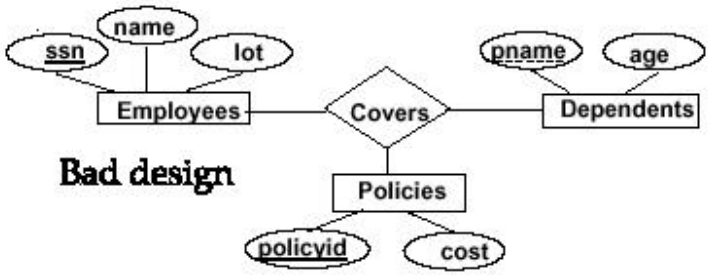


- Sebagaimana dalam C++ maupun bahasa pemrograman yang lain, suatu atribut dapat diturunkan.
- Jika kita deklarasikan A ISA B, setiap entity A juga merupakan entity B.
  - *Overlap constraints :* Bolehkan seorang pegawai berstatus pegawai perjam (*Hourly_Emps)* dan pegawai kontrak *(Contract_Emps) ?*
  - *Covering constraints :* Apakah setiap entity Employees juga merupakan entity *Hourly_Emps* dan *Contract_Emps ?*

Translasi Hirarki ISA ke Relasi :

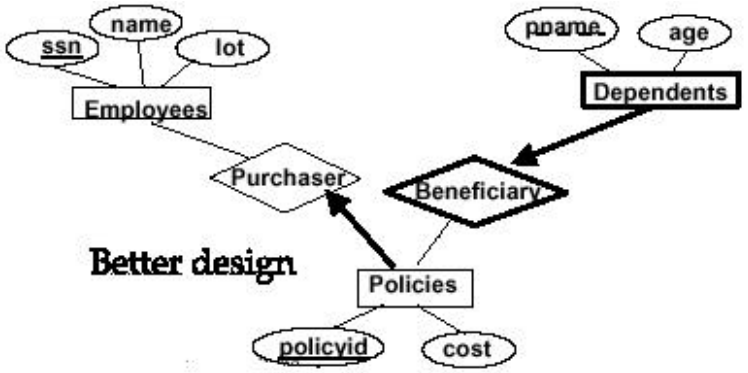
- Pendekatan umum :
  - Contoh : terdapat tiga relasi yang terlibat : Employees, Hourly_Emps dan Contract_Emps.
    - Hourly_Emps : Tiap data employee disimpan dalam *Employees.* Untuk pegawai dengan hitungan gaji perjam, informasi tambahan disimpan dalam *Hourly_Emps (hourly_wages, hours_worked,ssn); Tupel Hourly_Emps harus dihapus jika tupel Employees* yang berelasi dengannya dihapus.
- Alternatif lain : hanya *Hourly_Emps* dan *Contract_Emps*
  - *Hourly_Emps : ssn, name, lot, hourly_wages, hours_worked*
  - Tiap employee harus menjadi anggota salah satu dari sub class ini.

Review : Relasi Binary vs. Ternary



- Jika tiap *policy* hanya dimiliki oleh satu orang *employee :*

- *Key constraint* pada *policies* mengartikan *policy* hanya dapat mengkover satu dependent



- *Constraint* apa yang ditambahkan dalam diagram yang kedua ?

Relasi Binary vs. Ternary

- *Key Constraints* membolehkan kita untuk mengkobinasikan *Purchaser* dengan *Policies* dan *Beneficiary* dengan *Dependents*
- *Participation Constraint* merupakan *NOT NULL Constraints.*
- Apa yang terjadi jika *Policies* merupakan himpunan entity lemah ?

```
CREATE TABLE Policies
( PolicyID INTEGER,
  cost REAL,
  ssn CHAR(11) NOT NULL,
  PRIMARY KEY (PolicyID),
  FOREIGN KEY (ssn) REFERENCES Employees,
        ON DELETE CASCADE)
CREATE TABLE Dependents
( pname CHAR(20),
  age INTEGER,
  PolicyID INTEGER,
  PRIMARY KEY (pname, PolicyID),
  FOREIGN KEY (PolicyID) REFERENCES Policies,
  ON DELETE CASCADE)
```

Ringkasan : Model Relasional

- Tabel yang merepresentasikan data
- Model yang paling banyak digunakan
- *Integrity Constraints* dapat dispesifikasi oleh DBA, berdasarkan aplikasi semantic. DBMS memeriksa kebenarannya.
  - Dua hal penting pada *integrity constraint : primary* dan *foreign key*.
  - Sebagai tambahan, selalu ada *domain constraint*

- Memiliki banyak kelebihan dan sifatnya alami dalam menyatakan suatu query
- Aturan untuk mentranslasikan ER kedalam model relasional.