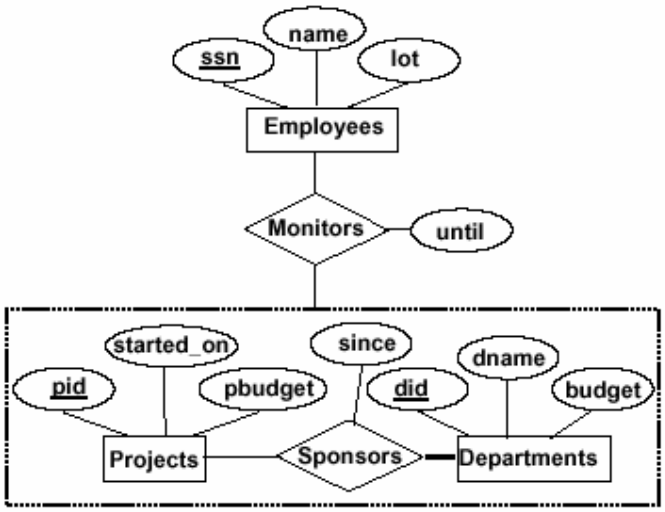
1.2.5. AGGREGASI

Aggregasi digunakan pada saat kita perlu memodelkan apa saja yang terlibat dalam suatu himpunan relasi. Aggregasi membolehkan kita untuk memperlakukan suatu himpunan relasi sebagai himpunan entity untuk tujuan partisipasi dalam relasi yang lain.

Gambar berikut menunjukkan bahwa *Monitors* adalah relasi yang distinct dengan deskripsi atribut. Juga dapat dikatakan bahwa tiap *sponsorship* dimonitor oleh seorang pegawai.



Gambar 1-9: ***Contoh Aggregasi***

1.3. MODEL RELASIONAL

Basis Data Relasional adalah himpunan relasi. Suatu relasi adalah himpunan kolom atau tupel (semua barisnya bersifat distinct/unik).

Sedangkan relasi itu sendiri terdiri dari dua bagian yaitu :

- *Instance :* table dengan baris dan kolom

  #baris = kardinalitas, #kolom/fields = degree/arity
- *Skema :* menentukan nama relasi, plus nama dan tipe kolom

Contoh relasi misal :

Students(sid : string, name : string, login : string, age : integer, gpa : real).

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

Gambar 1-10 : *Contoh Instance dari Relasi Students*

Pada gambar, contoh instance dari relasi *Students* memiliki kardinalitas = 3, degree = 5, semua baris bersifat distinct. (*Pertanyaan :* Apakah semua kolom dalam instance relasi juga harus distinct ?)

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.

Kita dapat melakukan query pada beberapa table yang saling berelasi. Contoh pada table berikut jika 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

Kemudian 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”.

1.3.1. BATASAN INTEGRITAS (INTEGRITY CONSTRAINT)

Batasan Integritas adalah suatu kondisi yang harus bernilai benar untuk suatu instance dalam basis data, misal : batasan domain

- Dispesifikasi saat skema didefinisikan
- Diperiksa pada saat suatu relasi dimodifikasi

Instance dari relasi disebut legal jika bisa memenuhi semua batasan integritas (integrity constraints) yang telah dispesifikasi. Batasan integritas juga digunakan untuk menghindari kesalahan dari entry data

Berikut akan dibahas satu persatu batasan integritas dalam model relasional.

Batasan Kunci Primer (Primary Key Constraints)

Himpunan suatu fields merupakan suatu key dari suatu relasi jika :

- Tidak ada dua tupel yang distinct yang mempunyai nilai yang sama untuk semua key fields, dan
- 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.

Berikut contoh penggunaan batasan kunci primer :

```
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

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*

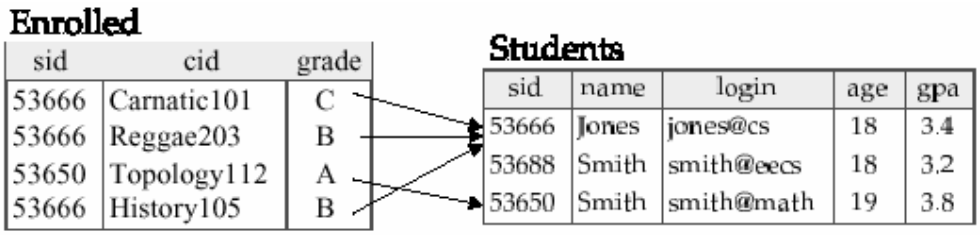
Misal *sid* adalah *foreign key* yang direfer dari relasi *Students* :

- 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)
```



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).

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 pembuatan referential integrity :

```
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)
```

1.4. *STRUCTURED QUERY LANGUAGE*

Structured Query Language (SQL) adalah bahasa database relasional yang dibuat berdasarkan suatu standart. Bentuk dasar dari SQL adalah sebagai berikut :

SELECT [DISTINCT] select-list
FROM from-list
WHERE qualification

Setiap query dalam SQL harus memiliki klausa SELECT, yang menentukan kolom yang akan ditampilkan pada hasil, dan klausa FROM yang menentukan cross product table. Klausa optional WHERE menentukan syarat-syarat seleksi pada table yang ditunjukkan oleh FROM.

Berikut ini akan dibahas sintaksis query SQL dasar dengan lebih mendetail :

- **from list** pada klausa FROM adalah daftar nama table. Nama tabel dapat diikuti oleh nama alias; nama alias berguna ketika nama tabel yang sama muncul lebih dari sekali pada from list
- **select-list** adalah daftar nama kolom (termasuk ekspresinya) dari tabel-tabel yang tercantum pada form list. Nama kolom dapat diawali dengan nama alias dari tabel.
- **Kualifikasi** pada klausa WHERE merupakan kombinasi boolean atau pernyataan kata sambung logika dari kondisi yang menggunakan eksepresi yang melibatkan operator pembanding. Sedangkan ekspresi itu sendiri dapat berupa nama kolom, konstanta atau aritmatika dan string.
- Kata kunci **distinct** bersifat pilihan yang menghapus duplikat dari hasil query.

SQL menyediakan tiga konstruksi set-manipulation yang memperluas query dasar, yaitu UNION, INTERSECT dan EXCEPT. Juga operasi set yang lain seperti : IN (untuk memeriksa apakah elemen telah berada pada set yang ditentukan), ANY dan ALL (untuk membandingkan suatu nilaid engan elemen pada set tertentu), EXISTS (untuk memeriksa apakah suatu set kosong atau isi). Operator IN dan EXISTS dapat diawali dengan NOT.

Fitur SQL yang lain yaitu NESTED QUERY, artinya query yang memiliki query lain di dalamnya, yang disebut dengan subquery. Nested query digunakan jika terdapat suatu nilai yang tidak diketahui (*unknown values).*

SQL mendukung lima operasi aggregat yang diterapkan pada sembarang kolom yaitu :

- COUNT : untuk menghitung cacah
- SUM : menghitung jumlah seluruh nilai
- AVG : menghitung rata-rata nilai
- MAX : mencari nilai paling besar
- MIN : mencari nilai paling kecil.

Kadangkala operasi aggregat diperlukan pada sekeompok grup dari baris pada relasi. Untuk menulis query semacam itu, dibutuhkan klausa GROUP BY. Dan penambahan klausa HAVING jika kita ingin menerapkan suatu kondisi terhadap data yang sudah dikelompokkan dengan GROUP BY.

1.5. NORMALISASI

Normalisasi adalah perbaikan skema database. Latar belakang diperlukannya normalisasi adalah karena adanya penyimpanan informasi yang redundan.

Istilah normalisasi berasal dari E.F. codd, salah seorang perintis teknologi basis data. Normalisasi adalah proses untuk mengubah suatu relasi tertentu ke dalam dua buah relasi atau lebih.

Berikut ini akan dijelaskan proses Normalisasi sampai dengan bentuk normal ketiga.

Bentuk Normal Pertama (1NF)

Suatu relasi dikatakan dalam bentuk normal pertama jika dan hanya jika setiap atribut bernilai tunggal untuk setiap atribut bernilai tunggal untuk setiap baris contoh:

Tabel 1. sebelum bentuk normal pertama

NIP	Nama	Hoby
10113024	Endang C Permana	Olahraga
		Baca Buku
10113025	Samsul	Dengar Musik
		Makan

Table 2. yang sudah dalam bentuk normal pertama