select \* from student, instructor where student.dept\_name = instructor.dept\_name

**VIEWS**

**create view** *v* **as** *<* query expression >

view definition causes the saving of an expression

A view of instructors without their salary   
 **create view *faculty*****as   
 select** *ID*, *name*, *dept\_name* **from** *instructor*

Find all instructors in the Biology department   
 **select** *name* **from *faculty*****where** *dept\_name =* 'Biology'

Create a view of department salary totals   
 **create view *departments\_total\_salary(dept\_name, total\_salary)*****as  
 select** *dept\_name*, **sum** (*salary*)  
 **from** *instructor* **group by** *dept\_name*;

Burada dept\_name ve total\_salary view’in sütunları

* **create view *physics\_fall\_2017* as  
   select** *course*.*course\_id*, *sec\_id*, *building*, *room\_number* **from** *course*, *section* **where** *course*.*course\_id* = *section*.*course\_id* **and** *course*.*dept\_name* = 'Physics'  
   **and** *section*.*semester* = 'Fall'  
   **and** *section*.*year* = '2017’;
* **create view *physics\_fall\_2017\_****watson* **as  
   select** *course\_id*, *room\_number* **from *physics\_fall\_2017*****where** *building*= 'Watson';
* **create view *physics\_fall\_2017\_watson* as**

**select** *course\_id*, *room\_number*

**from** (**select** *course*.*course\_id*, *building*, *room\_number*

**from** *course*, *section*

**where** *course*.*course\_id* = *section*.*course\_id*

**and** *course*.*dept\_name* = 'Physics'

**and** *section*.*semester* = 'Fall'

**and** *section*.*year* = '2017')

**where** *building*= 'Watson';

update için:

* from 🡪 1 db relation
* select 🡪 contains only attribute names of the relation, distinct olmamalı
* group by veya having olmamalı

**TRANSACTION**

A computer screen with white text

Description automatically generated



**Constraints on a Single Relation**

* not null
* primary key
* unique
  + **unique** ( *A*1, *A*2, …, *A*m)
  + The unique specification states that the attributes *A*1, *A*2, …, *A*m  form a candidate key.
  + candidate key’ler null olabilir
* check(P) ----> P is predicate
  + P must be satisfied by every tuple in relation
  + ensure that semester is one of fall, winter, spring or summer

**create table** *section*

(*course\_id* **varchar** (8),

*sec\_id* **varchar** (8),

*semester* **varchar** (6),

*year* **numeric** (4,0),

*building* **varchar** (15),

*room\_number* **varchar** (7),

*time slot id* **varchar** (4),

**primary key** (*course\_id*, *sec\_id*, *semester*, *year*),

**check** (*semester* **in** ('Fall', 'Winter', 'Spring', 'Summer')))

**referential integrity**

Ensures that a value that appears in one relation for a given set of attributes also appears for a certain set of attributes in another relation

* + Example: If “Biology” is a department name appearing in one of the tuples in the *instructor* relation, then there exists a tuple in the *department* relation for “Biology”.

Foreign *keys can be* specified as part of the SQL **create** **table**  statement

**foreign key** (*dept\_name*) **references** *department*

SQL allows a list of attributes of the referenced relation to be specified explicitly.

**foreign key** (*dept\_name*) **references** *department* (*dept\_name*)

**Assertion**

**create assertion** <assertion-name> **check** (<predicate>);

A screen shot of a computer

Description automatically generated

**Built-in Data Types**

**date** '2005-7-27'

**time** '09:00:30'  **time** '09:00:30.75'

**timestamp** '2005-7-27 09:00:30.75'

interval '1' day

* + Subtracting a date/time/timestamp value from another gives an interval value
  + Interval values can be added to date/time/timestamp values

Creating type:

**create type** *Dollars* **as numeric (12,2) final**

**create table** *department* (*dept\_name* **varchar** (20),  
 *building* **varchar** (15),  
 *budget Dollars*);

**Domain**

**create domain** *person\_name* **char**(20) **not null**

Types and domains are similar. Domains can have constraints, such as **not null**, specified on them

**create domain** *degree\_level* **varchar**(10)  
 **constraint** *degree\_level\_test* **check** (**value in** ('Bachelors', 'Masters', 'Doctorate'));

**Index**

An **index** on an attribute of a relation is a data structure that allows the database system to find those tuples in the relation that have a specified value for that attribute efficiently, without scanning through all the tuples of the relation.

**create index** <name> **on** <relation-name> (attribute);

* **create table** *student*(*ID* **varchar** (5),  
  *name* **varchar** (20) **not null**,  
  *dept\_name* **varchar** (20),  
  *tot\_cred* **numeric** (3,0) **default** 0,  
  **primary key** (*ID*))
* **create index** *studentID\_index* **on** *student*(*ID*)
* **select \*   
   from**  *student* **where**  *ID =* '12345'
  + This query can be executed by using the index to find the required record, without looking at all records of *student*

**Authorization**

read, insert, update, delete

* Each of these types of authorizations is called a **privilege**. We may authorize the user all, none, or a combination of these types of privileges on specified parts of a database, such as a relation or a view.
* Forms of authorization to modify the database schema
  + **Index** - allows creation and deletion of indices.
  + **Resources** - allows creation of new relations.
  + **Alteration** - allows addition or deletion of attributes in a relation.
  + **Drop** - allows deletion of relations.

**grant** <privilege list> **on** <relation or view > **to** <user list>

* + **grant** **select on** *department* **to** Amit, Satoshi

privileges:

* select
* insert
* update
* delete
* all privileges

Yetkiyi geri alma:

**revoke** <privilege list> **on** <relation or view> **from** <user list>

**Roles**

* + **create role** <role-name>
  + **grant** <role> **to** <users>

privilege to role:

**grant** **select** **on** *takes* **to** *instructor*

role privileges to role

* + **create** **role** *teaching\_assistant*
  + **grant** *teaching\_assistant* **to** *instructor*;
    - *Instructor* inherits all privileges of *teaching\_assistant*
* **create view** *geo\_instructor* **as**(**select** \*  
  **from** *instructor***where** *dept\_name* = 'Geology');
* **grant select on** *geo\_instructor* **to**  *geo\_staff*
  + **select** \*  
    **from** *geo\_instructor*;

Bu durumda eğer geo\_staff’ın instructor table’ına erişim izni yoksa geo\_staff kullanıcısı geo\_instructor view’una erişirken hata alır. Çünkü geo\_instructor görünümü instructor tablosundan veri çekmek için oluşturulmuştur ve bu nedenle instructor tablosuna erişim izni gereklidir.

Aynı şekilde view’in yaratıcısı da instructor tablosuna erişim izni olmalıdır.

* + **grant select on** *department* **to** Amit **with grant option**;
    - **Amit başka kullanıcılara yetki verebilir**
  + **revoke select on** *department* **from** Amit, Satoshi **cascade**;
    - yetkiler geri alınır ve yetki bu kullanıcılara bağlı olarak başka kullanıcılara verilmiş olabilecek tüm bağlantılar da geri alınır.
  + **revoke select on** *department* **from** Amit, Satoshi **restrict**;
    - sadece iki kullanıcıya verilen yetki geri alınır.

**create table** *course* (  
 (…  
 *dept\_name* **varchar**(20),  
 **foreign key** (*dept\_name*) **references** *department* **on delete cascade  
 on update cascade**,  
 . . .)

A black background with white text

Description automatically generated