

Principles of Database Systems (CS307)

Lecture 13: View; Access Control

Ran Cheng

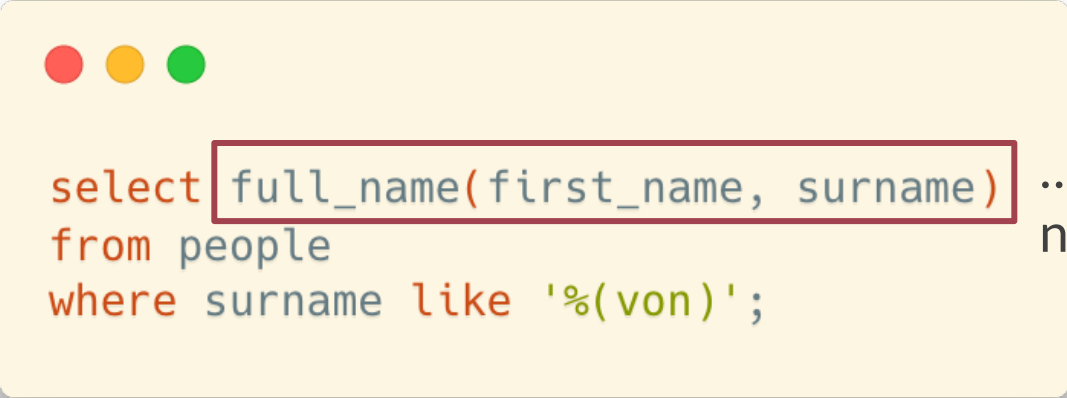
Department of Computer Science and Engineering
Southern University of Science and Technology

- Most contents are from slides made by Stéphane Faroult, Dr Yuxin Ma and the authors of Database System Concepts (7th Edition).
- Their original slides have been modified to adapt to the schedule of CS307 at SUSTech.

View

Recall: Function

- Used for returning numbers, strings, dates, etc.
 - Or, return a single value
- Trigger?



```
select full_name(first_name, surname)
from people
where surname like '%(von)';
```

... returns a string of the full name

View

- Reuse of the relational operations (as we reuse codes in a program)
 - ... which returns relations (tables) instead of simple values



```
create view viewname as  
select ...  
from ...  
where ...
```

View

- Example: Create a view like this:



```
create view vmovies as
select m.movieid,
       m.title,
       m.year_released,
       c.country_name
from movies m join countries c
on c.country_code = m.country;
```

View

- Example: Create a view like this:
 - ... where the result of the inner select query looks like this:



```
create view vmovies as
select m.movieid,
       m.title,
       m.year_released,
       c.country_name
from movies m join countries c
on c.country_code = m.country;
```

	movieid	title	year_released	country_name
1	1	12 stulyev	1971	Russia
2	2	Al-mummia	1969	Egypt
3	3	Ali Zaoua, prince de la rue	2000	Morocco
4	4	Apariencias	2000	Argentina
5	5	Ardh Satya	1983	India
6	6	Armaan	2003	India
7	7	Armaan	1966	Pakistan
8	8	Babettes gæstebud	1987	Denmark
9	9	Banshun	1949	Japan
10	10	Bidaya wa Nihaya	1960	Egypt
11	11	Variety	2008	United States
12	12	Bon Cop, Bad Cop	2006	Canada
13	13	Brilliantovaja ruka	1969	Russia
14	14	C'est arrivé près de chez vous	1992	Belgium
15	15	Carlota Joaquina - Princesa d.	1995	Brazil
16	16	Cicak-man	2006	Malaysia
17	17	Da Nao Tian Gong	1965	China
18	18	Das indische Grabmal	1959	Germany
19	19	Das Leben der Anderen	2006	Germany
20	20	Den store gavtyv	1956	Denmark

View vs. Table

- In practice, there isn't much to a view
 - It's basically a named query
 - And, why not just consider it as a “virtual table”?



```
create view vmovies(v_id, v_title, v_year, v_country) as
select m.movieid,
       m.title,
       m.year_released,
       c.country_name
from movies m
      join countries c
        on c.country_code = m.country;
```

View vs. Table

- In practice, there isn't much to a view
 - It's basically a named query
 - And, why not just consider it as a “virtual table”?



The columns can be renamed

- Otherwise, the original names in the tables will be used

```
create view vmovies(v_id, v_title, v_year, v_country) as
select m.movieid,
       m.title,
       m.year_released,
       c.country_name
from movies m
      join countries c
      on c.country_code = m.country;
```


View vs. Table

- In practice, there isn't much to a view
 - It's basically a named query
 - And, why not just consider it as a “virtual table”?



The columns can be renamed

- Otherwise, the original names in the tables will be used

```
create view vmovies(v_id, v_title, v_year, v_country) as
select m.movieid,
       m.title,
       m.year_released,
       c.country_name
from movies m
      join countries c
      on c.country_code = m.country;
```

Drop the existing view before creating a new one with the same name:




```
drop view vmovies;
```

View vs. Table


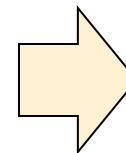
- A view can be as a complicated query as you want
 - It will usually return something that isn't as normalized as your tables, but easier to understand.
- Imagine: a function of a select query
- From a programming language's view:
 - Tables -> variables
 - Rows -> values

View vs. Table

- Once the view is created,
 - ... we can query the view exactly as if it were a table



```
create view vmovies as
select m.movieid,
       m.title,
       m.year_released,
       c.country_name
from movies m join countries c
on c.country_code = m.country;
```

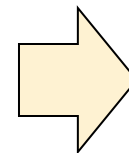


```
select *
from vmovies
where country_name = 'Italy';
```

View vs. Table

- Once the view is created,
 - ... we can query the view exactly as if it were a table

```
create view vmovies as
select m.movieid,
       m.title,
       m.year_released,
       c.country_name
from movies m join countries c
on c.country_code = m.country;
```



```
select *
from vmovies
where country_name = 'Italy';
```

Use the view name in the
from clause

- ... as if it were a “table”

Dynamic Content in Views

- When the rows change in the tables where a view relies on, the result of a view will change as well
- Think it like a table variable, or a query result
- * Or, think it like a relational function which returns a relation

Dynamic Content in Views

- Beware that **columns** are the one in tables when the view was created
 - **Columns added later** to tables in the view **won't be added** even if the view was created with “**select ***”
 - And, “**select ***” is a **bad practice** in view. **We will not know** what exact columns are selected once the columns in the tables are changed.

```
• • •  
  
-- Create the view first  
create view vmovies as  
select * -- IT IS A BAD PRACTICE OF USING * HERE  
from movies m join countries c  
    on c.country_code = m.country;  
  
-- Alter the table then  
alter table movies add column test_col int not null default 0;  
  
-- There won't be a column "test_col" in the query result  
select * from vmovies;
```

View vs. Table (Cont.)

- View looks like Table, tastes like Table – **functionalized (函数化) Table**
 - Sometimes it is used as a table, table

AI is functionalized intelligence ?



回复 @sama

ChatGPT is scary good. We are not far from dangerously strong AI.

[翻译推文](#)

上午3:48 · 2022年12月4日

已知一张桌子的价钱是一把椅子的10倍，又知一张桌子比一把椅子多288元，一张桌子和一把椅子各多少元？

一张桌子的价钱是一把椅子的10倍，即一张桌子的价钱是一把椅子的价钱的10倍+288元，用数学公式表示为：桌子价钱=椅子价钱×10+288元。

已知一张桌子比一把椅子多288元，即一张桌子的价钱是一把椅子的价钱+288元，用数学公式表示为：桌子价钱=椅子价钱+288元。

把两个公式结合起来，得到一张桌子的价钱是：桌子价钱=椅子价钱×10+288元=椅子价钱+288元

是：桌子价钱=椅子价钱×10+288元=椅子价钱+288元。

将这个公式化简一下，得到：椅子价钱×9=288元。

将这个公式再化简一下，得到：椅子价钱=288元/9=32元。

最后，用桌子价钱=椅子价钱×10+288元的公式计算一张桌子的价钱，得到：桌子价钱=32元×10+288元=320元+288元=608元。

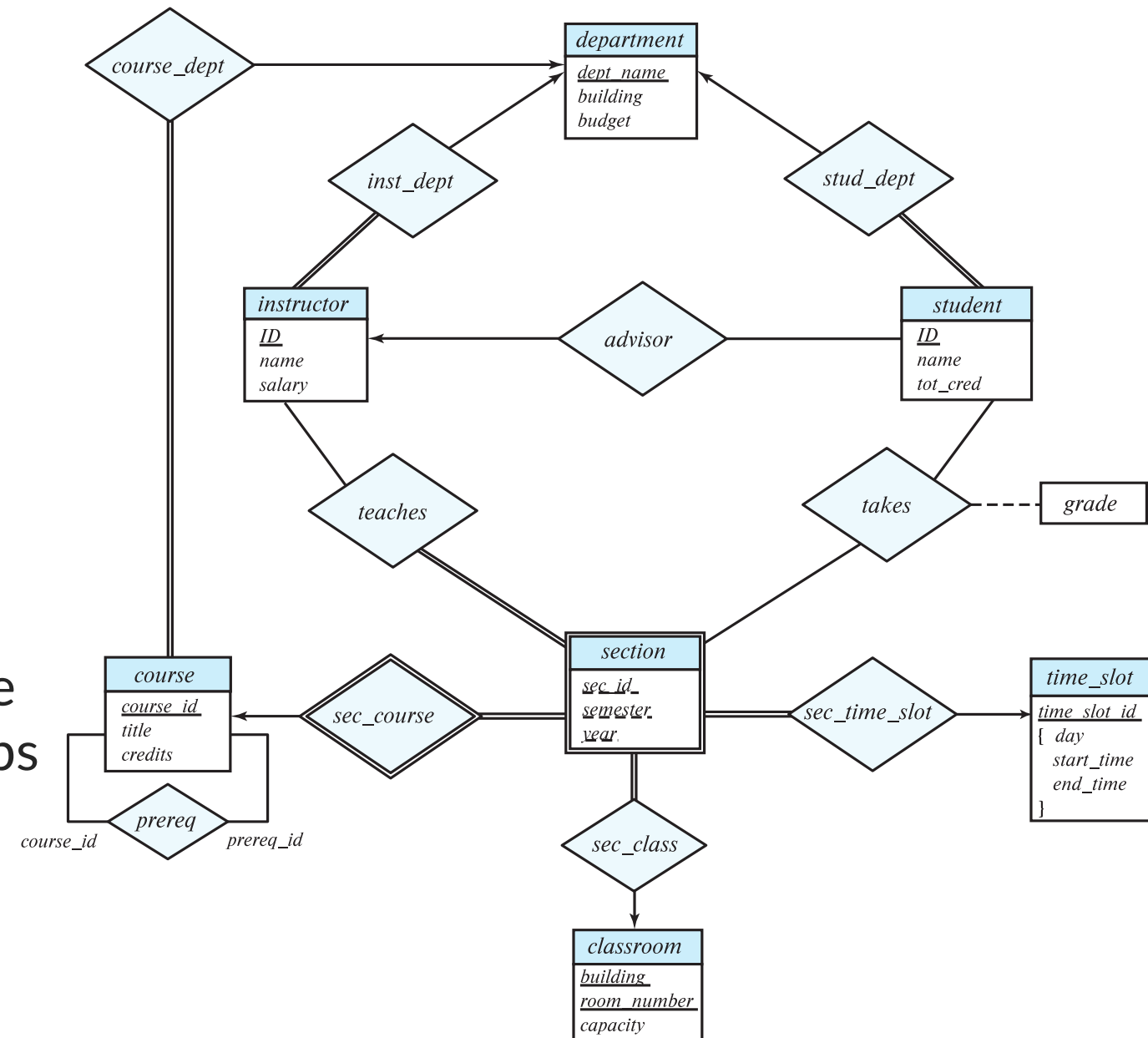
所以，一张桌子和一把椅子各多少元分别是608元和32元。

View vs. Table (Cont.)

- View looks like Table, tastes like Table
 - Sometimes it is used as a table
- Usage Scenario 1: **Simplify Complex Queries**
 - Simplify a complicated query result into a single named query
 - E.g. Many business reports are based on the same set of joins, with just variations on the columns that you aggregate or order by
 - One command, same query (Example?)
 - 饭卡余额

View vs. Table (Cont.)

- View looks like Table, tastes like Table
 - Sometimes it is used as a table
- Usage Scenario 2: An alternative way to implement E-R models
 - Sometimes, we may NOT be able to use tables to model entities and relationships
 - Access control (部分可见)
 - Dirty and messy original data (预处理)
 - Since views look like tables, we can use views to represent entities or relationships
 - ... based on some existing objects (like tables)



There are more than one way to implement E-R models

Drawback of Views

- View looks like Table, tastes like Table
 - But it is still not a table.
 - In some cases, it may cause performance issues or unnecessary operations
 - * ... since the users of the views don't know the details of the views

Drawback of Views

- View looks like Table, tastes like Table
 - Example: A refined way to display the **credit information** with a view



```
create view vmovie_credits
as
select m.title,
       m.year_released release,
       case c.credited_as
         when 'A' then 'Actor'
         when 'D' then 'Director'
         else '?'
       end duty,
       full_name(p.first_name, p.surname) name
from movies m
     inner join credits c
              on c.movieid = m.movieid
     inner join people p
              on p.peopleid = c.peopleid;
```

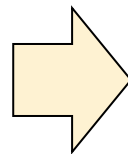
	title	release	duty	name
1	"Erogotoshitachi" yori Jinruigaku nyūmon	1966	Director	Shohei Imamura
2	"Erogotoshitachi" yori Jinruigaku nyūmon	1966	Actor	Shoichi Ozawa
3	"Erogotoshitachi" yori Jinruigaku nyūmon	1966	Actor	Sumiko Sakamoto
4	'76	2016	Actor	Ramsey Nouah
5	'76	2016	Actor	Ibinabo Fiberesima
6	(T)Raumschiff Surprise	2004	Actor	Michael Herbig
7	(T)Raumschiff Surprise	2004	Director	Michael Herbig
8	(T)Raumschiff Surprise	2004	Actor	Til Schweiger
9	(T)Raumschiff Surprise	2004	Actor	Anja Kling
10	(T)Raumschiff Surprise	2004	Actor	Rick Kavanian
11	(T)Raumschiff Surprise	2004	Actor	Christian Tramitz
12	(T)Raumschiff Surprise	2004	Actor	Sky du Mont
13	002 operazione luna	1965	Actor	Linda Sini
14	002 operazione luna	1965	Director	Lucio Fulci

Drawback of Views

- View looks like Table, tastes like Table, **BUT** if you **REALLY** treat it as a Table
 - Example: A refined way to display the credit information with a view
 - However, if we only want to get all distinct movie titles, it will return the correct result, but there are a lot of useless works in the internal query of the view



```
select distinct title
from vmovie_credits
```



```
create view vmovie_credits
as
select m.title,
       m.year_released release,
       case c.credited_as
         when 'A' then 'Actor'
         when 'D' then 'Director'
         else '?'
       end duty,
       full_name(p.first_name, p.surname) name
from movies m
      inner join credits c
              on c.movieid = m.movieid
      inner join people p
              on p.peopleid = c.peopleid;
```

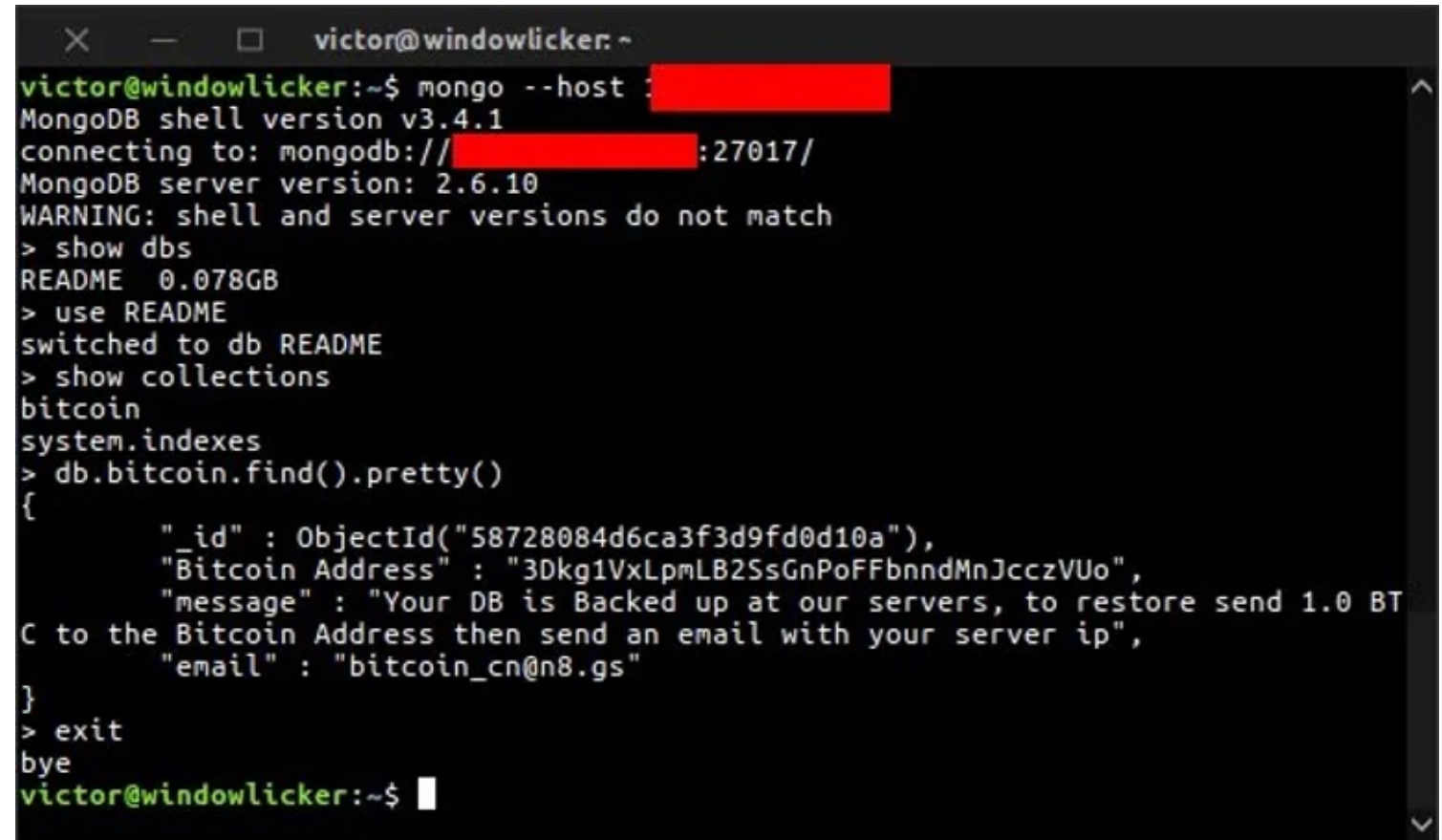


Do we really need the joins?

Access Control

Authentication (身份验证)

- To access a database, you must be authenticated, which often means entering a username and a password.
- If you don't set a proper password, you will put your database system in danger

A terminal window titled 'victor@windowlicker: ~' showing a MongoDB shell session. The user runs 'mongo --host [redacted]', connecting to a MongoDB instance on port 27017. The shell version is v3.4.1 and the server version is 2.6.10. A warning message states 'WARNING: shell and server versions do not match'. The user enters 'show dbs', showing 'README 0.078GB'. Then 'use README' is entered, switching to the 'db README' database. 'show collections' is entered, showing 'bitcoin' and 'system.indexes'. Finally, 'db.bitcoin.find().pretty()' is entered, displaying a JSON document with fields: '_id', 'Bitcoin Address', 'message', and 'email'. The message is a ransom note. The session ends with 'exit' and 'bye'.

<https://www.zdnet.com/article/mongodb-ransacked-now-27000-databases-hit-in-mass-ransom-attacks/>

Privileges

- Besides, DBMS usually provide **another layer of access control** on objects in the system
 - Operations (select, update, insert, delete, etc.)
 - Objects (table, database, views, trigger, etc.)

Grant and Revoke Access

- grant and revoke
 - Can be used on the table or the database level



```
-- grant <right> to <account>  
grant select on movies to test_user;  
  
-- revoke <right> from <account>  
revoke select on movies from test_user;
```

The possible privileges are:

```
SELECT  
INSERT  
UPDATE  
DELETE  
TRUNCATE  
REFERENCES  
TRIGGER  
CREATE  
CONNECT  
TEMPORARY  
EXECUTE  
USAGE
```


How Can Views Help

- User access control
 - By creating a view that only returns rows associated with the user name, we can control the privileges for a specific user in this table



```
create view my_stuff  
as  
select * from stuff  
where username = user
```



```
-- E.g., restrict the users to only access  
-- the movies assigned to their user names  
create view user_view as  
select movieid,  
       title,  
       country,  
       year_released,  
       runtime  
from movies  
where user_name = user;
```

How Can Views Help

- User access control
 - By creating a view that only returns rows associated with the user name, we can control the privileges for a specific user in this table

```
create view my_stuff
as
select * from stuff
where username = user
```

```
-- E.g., restrict the users to only access
-- the movies assigned to their user names
create view user_view as
select movieid,
       title,
       country,
       year_released,
       runtime
from movies
where user_name = user;
```

user: An internal variable that returns the current user name

- Remember **new** in triggers?
- How about **NULL** and **old** in triggers?

Something more about Trigger: NULL

- Trigger functions are always declared as “RETURNS trigger”, but what you actually have to return is
 - for **statement level** triggers, the value always **NULL**
 - for **row level** triggers, a **row** of the table on which the trigger is defined
- The **return value is ignored** for **row level AFTER** triggers, so you may as well return **NULL** in that case.
- For **row level BEFORE triggers**, there are two cases:
 - if the trigger returns **NULL**, the triggering operation is **aborted**, and the row will not be modified
 - for **INSERT** and **UPDATE** triggers, the returned row is the **input** for the triggering DML statement

Something more about Trigger: NEW and OLD

- **NEW** and **OLD** in row level triggers in a row level trigger function contain the **new row version** and the **old row version** respectively. They can be used in the RETURN statement.
- Note that NEW is NULL in ON DELETE triggers and OLD is NULL in ON INSERT triggers.

trigger invocation	NEW is set	OLD is set
ON INSERT	new row version	NULL
ON UPDATE	new row version	old row version
ON DELETE	NULL	old row version