Database Systems
(CS 355 / CE 373)

Dr. Umer Tariq
Assistant Professor,
Dhanani School of Science & Engineering,
Habib University

Acknowledgements

- Many slides have been borrowed from the official lecture slides accompanying the textbook:

  Database System Concepts, (2019), Seventh Edition,

  Avi Silberschatz, Henry F. Korth, S. Sudarshan

  McGraw-Hill, ISBN 9780078022159

  The original lecture slides are available at:

  https://www.db-book.com/

- Some of the slides have been borrowed from the lectures by Dr. Immanuel Trummer (Cornell University). Available at: (www.itrummer.org)

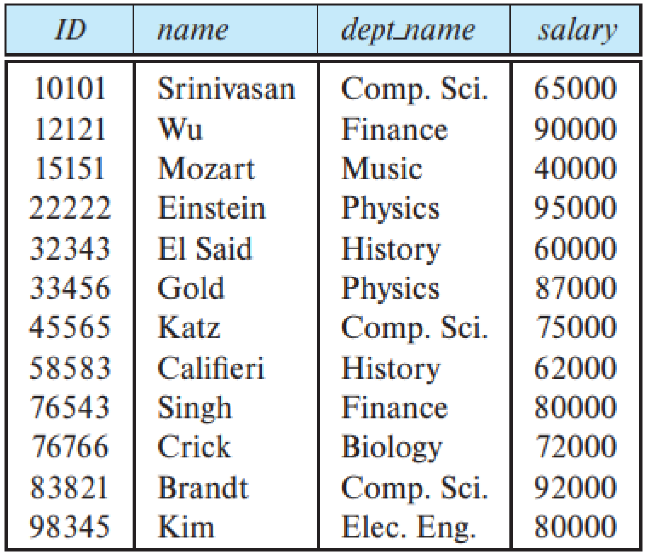
Outline: Week 13

- SQL Views

- SQL Stored Procedures

SQL Views: Motivation

- Suppose we want the administrative staff to work with the University database but we do not want to disclose the salary information to them.



ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

Figure 2.1 The *instructor* relation.

Faculty

ID	name	dept.name
10101	Sri	CS

SQL Views: Motivation

- Suppose we want the administrative staff to work with the University database but we do not want to disclose the salary information to them.

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

Figure 2.1 The *instructor* relation.

SQL Query

create view *faculty as*
select *ID, name, dept_name*
from *instructor;*

Faculty

ID	name	dept_name
10101	Srinivasan	Comp. Sci.
12121	Wu	Finance
15151	Mozart	Music
22222	Einstein	Physics
32343	El Said	History
33456	Gold	Physics
45565	Katz	Comp. Sci.
58583	Califieri	History
76543	Singh	Finance
76766	Crick	Biology
83821	Brandt	Comp. Sci.
98345	Kim	Elec. Eng.

SQL Views

- The ***create view*** command creates a view relation (or virtual table) that contains the results of the query in the *create view* command.
  - The view relation conceptually contains the tuples in the query result, but it is not precomputed and stored.
  - Instead, the database system stores the query expression associated with the view relation.
  - Whenever the view relation is accessed, its tuples are created by computing the query result. Thus, the view relation is created whenever needed, on demand.

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

Figure 2.1 The *instructor* relation.

```
create view faculty as
   select ID, name, dept_name
   from instructor;
```

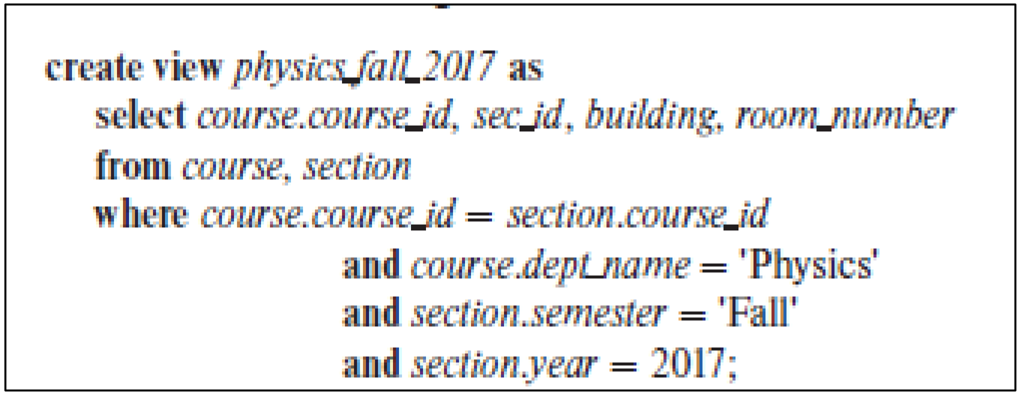
```
select *
from faculty
where dept_name='Physics';
```

Faculty

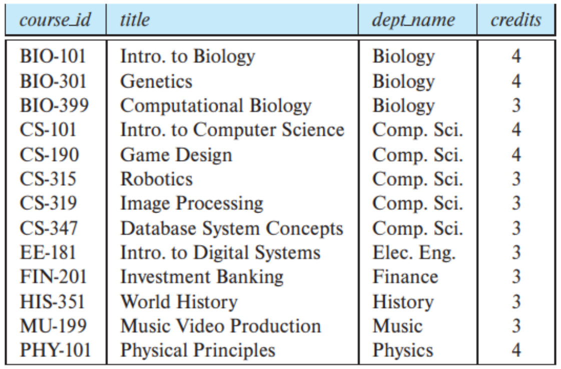
ID	name	dept_name
10101	Srinivasan	Comp. Sci.
12121	Wu	Finance
15151	Mozart	Music
22222	Einstein	Physics
32343	El Said	History
33456	Gold	Physics
45565	Katz	Comp. Sci.
58583	Califieri	History
76543	Singh	Finance
76766	Crick	Biology
83821	Brandt	Comp. Sci.
98345	Kim	Elec. Eng.

SQL Views: Example

- Create a view that lists all course sections offered by the Physics department in the Fall 2017 semester with the building and room number of each section

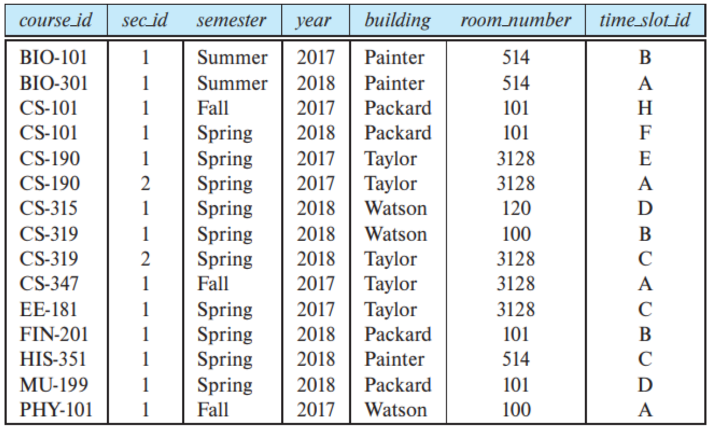


```
create view physics_fall_2017 as
   select course.course_id, sec_id, building, room_number
   from course, section
   where course.course_id = section.course_id
                  and course.dept_name = 'Physics'
                  and section.semester = 'Fall'
                  and section.year = 2017;
```



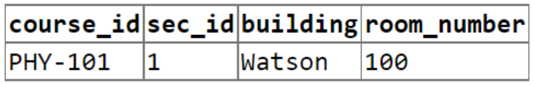
course_id	title	dept_name	credits
BIO-101	Intro. to Biology	Biology	4
BIO-301	Genetics	Biology	4
BIO-399	Computational Biology	Biology	3
CS-101	Intro. to Computer Science	Comp. Sci.	4
CS-190	Game Design	Comp. Sci.	4
CS-315	Robotics	Comp. Sci.	3
CS-319	Image Processing	Comp. Sci.	3
CS-347	Database System Concepts	Comp. Sci.	3
EE-181	Intro. to Digital Systems	Elec. Eng.	3
FIN-201	Investment Banking	Finance	3
HIS-351	World History	History	3
MU-199	Music Video Production	Music	3
PHY-101	Physical Principles	Physics	4

Figure 2.2 The *course* relation.



course_id	sec_id	semester	year	building	room_number	time_slot_id
BIO-101	1	Summer	2017	Painter	514	B
BIO-301	1	Summer	2018	Painter	514	A
CS-101	1	Fall	2017	Packard	101	H
CS-101	1	Spring	2018	Packard	101	F
CS-190	1	Spring	2017	Taylor	3128	E
CS-190	2	Spring	2017	Taylor	3128	A
CS-315	1	Spring	2018	Watson	120	D
CS-319	1	Spring	2018	Watson	100	B
CS-319	2	Spring	2018	Taylor	3128	C
CS-347	1	Fall	2017	Taylor	3128	A
EE-181	1	Spring	2017	Taylor	3128	C
FIN-201	1	Spring	2018	Packard	101	B
HIS-351	1	Spring	2018	Painter	514	C
MU-199	1	Spring	2018	Packard	101	D
PHY-101	1	Fall	2017	Watson	100	A

Figure 2.6 The *section* relation.



course_id	sec_id	building	room_number
PHY-101	1	Watson	100

SQL Views: Using Views

- Using the view *physics_fall_2017*, find all Physics courses offered in the Fall 2017 semester in the Watson building:
  - View names may appear in a query any place where a relation name may appear.

```
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;
```

```
select course_id
from physics_fall_2017
where building = 'Watson';
```

course_id	*title*	*dept_name*	*credits*
BIO-101	Intro. to Biology	Biology	4
BIO-301	Genetics	Biology	4
BIO-399	Computational Biology	Biology	3
CS-101	Intro. to Computer Science	Comp. Sci.	4
CS-190	Game Design	Comp. Sci.	4
CS-315	Robotics	Comp. Sci.	3
CS-319	Image Processing	Comp. Sci.	3
CS-347	Database System Concepts	Comp. Sci.	3
EE-181	Intro. to Digital Systems	Elec. Eng.	3
FIN-201	Investment Banking	Finance	3
HIS-351	World History	History	3
MU-199	Music Video Production	Music	3
PHY-101	Physical Principles	Physics	4

Figure 2.2 The *course* relation.

course_id	*sec_id*	*semester*	*year*	*building*	*room_number*	*time_slot_id*
BIO-101	1	Summer	2017	Painter	514	B
BIO-301	1	Summer	2018	Painter	514	A
CS-101	1	Fall	2017	Packard	101	H
CS-101	1	Spring	2018	Packard	101	F
CS-190	1	Spring	2017	Taylor	3128	E
CS-190	2	Spring	2017	Taylor	3128	A
CS-315	1	Spring	2018	Watson	120	D
CS-319	1	Spring	2018	Watson	100	B
CS-319	2	Spring	2018	Taylor	3128	C
CS-347	1	Fall	2017	Taylor	3128	A
EE-181	1	Spring	2017	Taylor	3128	C
FIN-201	1	Spring	2018	Packard	101	B
HIS-351	1	Spring	2018	Painter	514	C
MU-199	1	Spring	2018	Packard	101	D
PHY-101	1	Fall	2017	Watson	100	A

Figure 2.6 The *section* relation.

course_id	sec_id	building	room_number
PHY-101	1	Watson	100

SQL Standard — MS SQL Server
— PostgreSQL
— MySQL

Materialized Views

- Certain database systems allow view relations to be stored, but they make sure that, if the actual relations used in the view definition change, the view is kept up-to-date. Such views are called "**materialized views**."
  - However, if an instructor tuple is added to or deleted from the instructor relation, the result of the query defining the view would change.
  - View maintenance can be done immediately when any of the relations on which the view is defined is updated. Some database systems, perform view maintenance lazily, when the view is accessed.
  - "SQL does not define a standard way of specifying that a view is materialized."

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

Figure 2.1 The *instructor* relation.

create view *faculty* **as**
  select *ID, name, dept_name*
  from *instructor;*

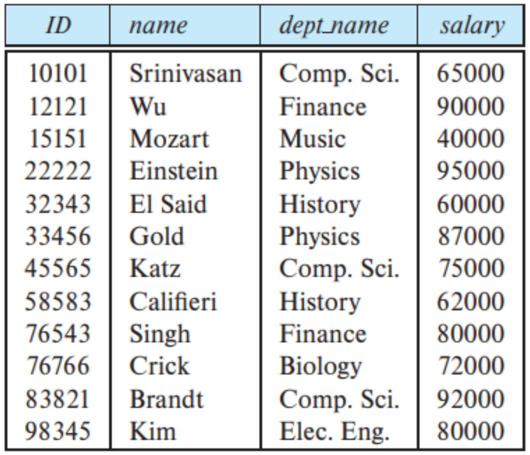
select *
from *faculty*
where *dept_name='Physics';*

ID	name	dept_name
10101	Srinivasan	Comp. Sci.
12121	Wu	Finance
15151	Mozart	Music
22222	Einstein	Physics
32343	El Said	History
33456	Gold	Physics
45565	Katz	Comp. Sci.
58583	Califieri	History
76543	Singh	Finance
76766	Crick	Biology
83821	Brandt	Comp. Sci.
98345	Kim	Elec. Eng.

Update Operations on Views

- Although views are a useful tool for queries, they present serious problems if we express updates, insertions, or deletions with them.
  - The difficulty is that a modification to the database expressed in terms of a view must be translated to a modification to the actual relations in the logical model of the database.
  - Because of problems, modifications are generally not permitted on view relations, except in limited cases.
  - Different database systems specify different conditions under which they permit updates on view relations.

INSERT INTO FACULTY (ID, name, dept-name)
VALUES(—, —, —.)



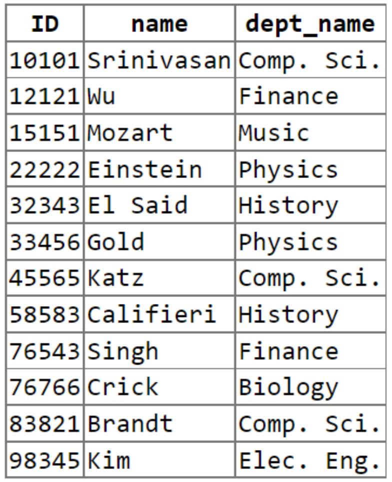
ID	name	dept_name	salary
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

Figure 2.1 The *instructor* relation.

```
create view faculty as
  select ID, name, dept_name
  from instructor;
```

```
select *
from faculty
where dept_name='Physics';
```

Faculty



ID	name	dept_name
10101	Srinivasan	Comp. Sci.
12121	Wu	Finance
15151	Mozart	Music
22222	Einstein	Physics
32343	El Said	History
33456	Gold	Physics
45565	Katz	Comp. Sci.
58583	Califieri	History
76543	Singh	Finance
76766	Crick	Biology
83821	Brandt	Comp. Sci.
98345	Kim	Elec. Eng.

SQL Stored Procedure: Motivation

- For the Northwind Database: Consider the following task :
  - "Delate an Employee"

DB

NO STORED PROCEDURE

for

"Deleting an Employee"

Python

—. connect ('—');

. cursor()

cursor. execute ('SQL Query')

cursor.execute ('SQL Query1: Delete a row from EmployeeTable')

.execute ('SQL Query2: Deal with Not Shipped Order of that Employee)

.execute ('SQL Query3 : Deals with Reporting Hierarchy."

SQL Stored Procedure: Motivation

- For the Northwind Database: Consider the following task
  - "Delate an Employee"

DB

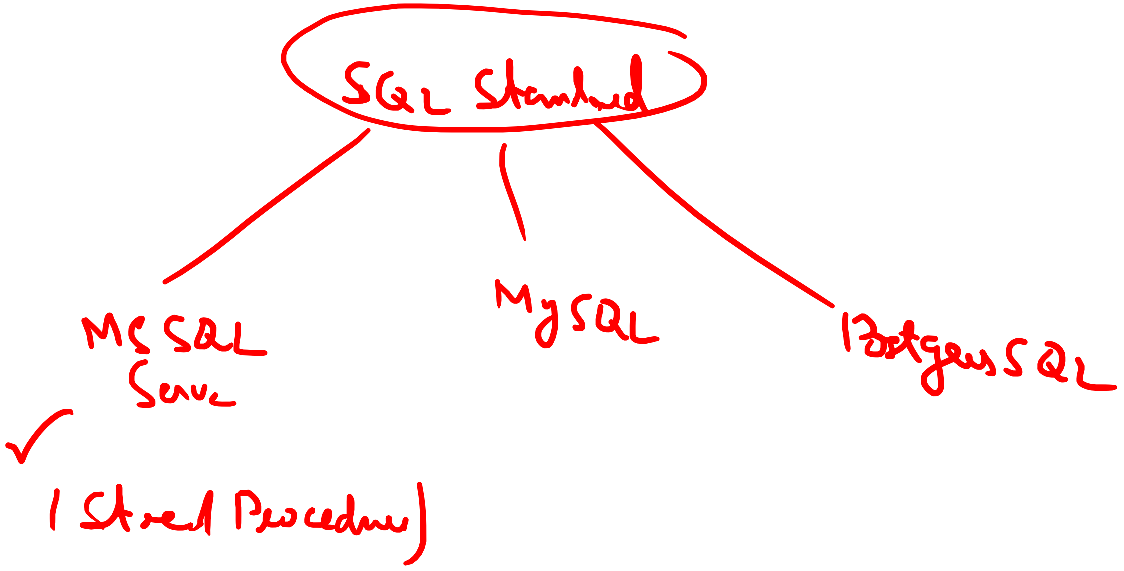
```
CREATE STORED PROCEDURE DeleteEmp
(input parameter)
AS
BEGIN
        SQL Query1 : Delete from EmployeeTable
        SQL Query 2 : Deal with Not Shipped
                      Orders
        SQL Query 3 : Deal with Reporting
                      Hierarchy
END

EXEC DeleteEmp 123
```

Python

```
.commit ( )
.cursor ( )
cursor.execute('EXEC DeleteEmp 123')
```

SQL Stored Procedure: Product-Specific Syntax

- Although the syntax for Stored Procedures is defined by SQL standard, most database products implement non-standard version of this syntax.
  - In the lab, we shall use the syntax of Stored Procedures supported by Microsoft SQL Server (Transact SQL).


SQL Standard
MS SQL Server
MySQL
PostgresSQL
✓
(Stored Procedure)