CS5300
Database Systems

Views

A.R. Hurson
323 CS Building
hurson@mst.edu

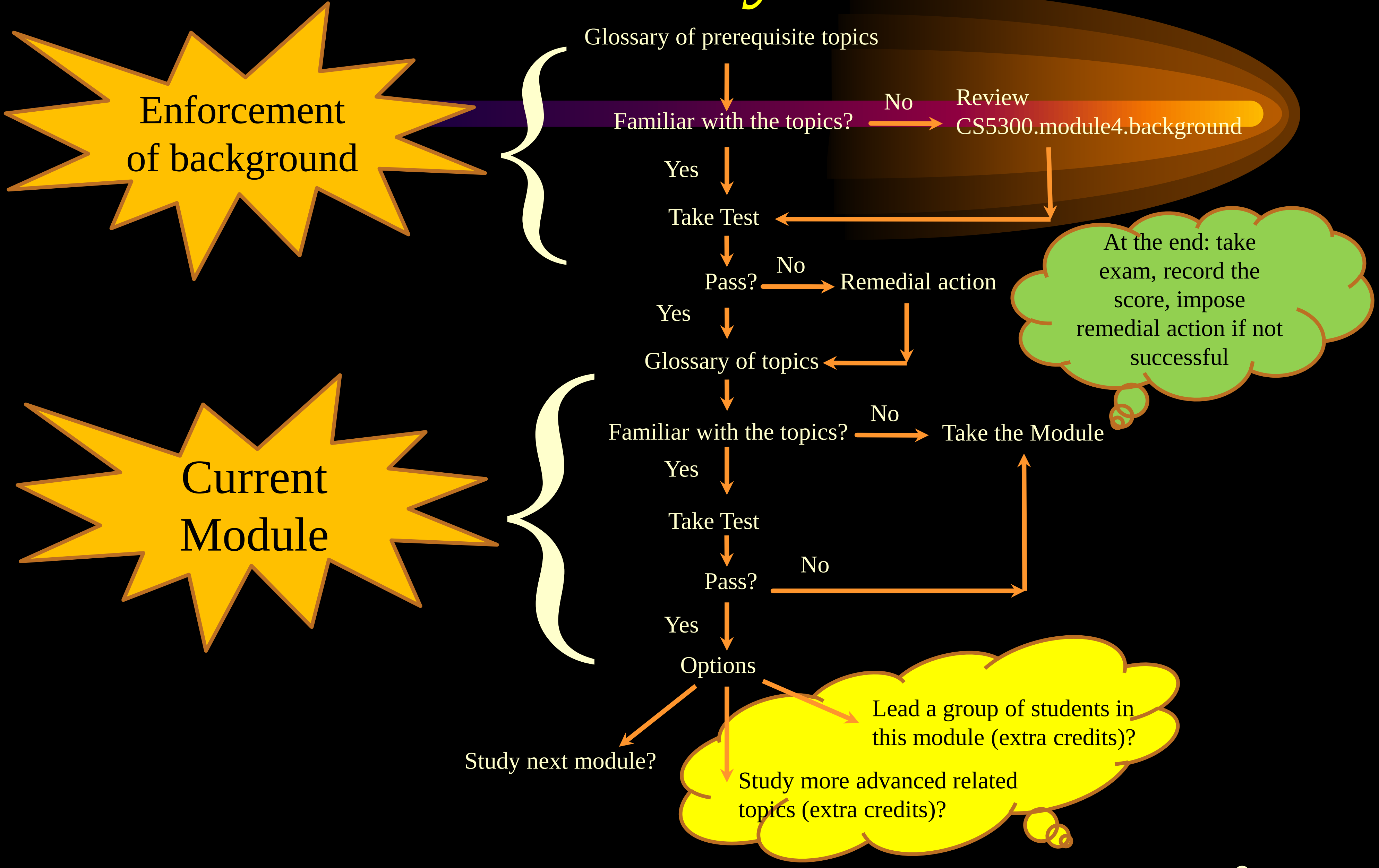
Database Systems

Note, this unit will be covered in two lectures. In case you finish it earlier, then you have the following options:

1) Take the early test and start CS5300.module5
2) Study the supplement module (supplement CS5300.module4)
3) Act as a helper to help other students in studying CS5300.module4

Note, options 2 and 3 have extra credits as noted in course outline.

Database Systems



Enforcement of background

Glossary of prerequisite topics

Familiar with the topics?

No

Review CS5300.module4.background

Yes

Take Test

Pass?

No

Remedial action

Yes

Glossary of topics

At the end: take exam, record the score, impose remedial action if not successful

Current Module

Familiar with the topics?

No

Take the Module

Yes

Take Test

Pass?

No

Yes

Options

Study next module?

Lead a group of students in this module (extra credits)?

Study more advanced related topics (extra credits)?

Extra Curricular activities

Database Systems

- You are expected to be familiar with:
  - Relational database model,
  - SQL
- If not, you need to study CS5300.module4.background

Database Systems

- Views
  - A view is a virtual table — a table that does not exist in its own right but looks to the user as it did. A view is not supported by its own, physically, distinguishable stored data. Instead its definition in terms of other table is stored in the system.

Database Systems

- Views
  - For example, when

```
CREATE VIEW               GOOD_SUPPLIERS
        AS    SELECT      S#, Status, City
              FROM        S
              WHERE       Status > 15;
```

is executed, the sub-query following *AS* is not executed. It is simply saved in the system catalog.

Database Systems

S#	Sname	Status	City
S_1	Smith	20	London
S_2	Jones	10	Paris
S_3	Blake	30	Paris
S_4	Clark	20	London
S_5	Adams	30	Athens

Database Systems

- Views
  - After creation, *GOOD_SUPPLIERS* is a window into the real table *S*. It is also dynamics — changes to *S* is automatically reflected in *GOOD_SUPPLIERS* and likewise changes in *GOOD_SUPPLIERS* is reflected in *S*. Finally, user can write queries against *GOOD_SUPPLIERS*.
  - In case user issues a query against *GOOD_SUPPLIERS*, the system automatically converts it into proper query.

Database Systems

- Views
  - For example,

```
SELECT    *
FROM      GOOD_SUPPLIERS
WHERE     City <> ‘London’;
```

  - is translated into

```
SELECT    S#, Status, City
FROM      S
WHERE     City <> ‘London’
     AND  Status > 15;
```

Database Systems

S#	Sname	Status	City
S_1	Smith	20	London
S_2	Jones	10	Paris
S_3	Blake	30	Paris
S_4	Clark	20	London
S_5	Adams	30	Athens

Result

S#	Status	City
S_3	30	Paris
S_5	30	Athens

Database Systems

- Views
  - UPDATE operation is done in the same way

```
UPDATE      GOOD_SUPPLIERS
SET         Status = Status + 10
WHERE       City = ‘Paris’;
```

will be converted to;

```
UPDATE      S
SET         Status = Status + 10
WHERE       City = ‘Paris’
      AND   Status > 15 ;
```

Database Systems

- View Definition
  - The general syntax is;

```
CREATE VIEW      view [ (column [, column ] … )]
        AS       sub-query
        [ WITH CHECK OPTION ];
```

Database Systems

- Views

```
CREATE VIEW    REDPARTS ( P#, Pname, WT, City)
AS  SELECT     P#, Pname, Weight, City
     FROM      P
               WHERE    Color = ‘Red’;
```

P#	Pname	Color	Weight	City
P_1	Nut	Red	12	London
P_2	Bolt	Green	17	Paris
P_3	Screw	Blue	17	Rome
P_4	Screw	Red	14	London
P_5	Cam	Blue	12	Paris
P_6	Cog	Red	19	London

Database Systems

- Views
  - Column names must be specified, if
    - any column of the view is derived from a function, an operational expression, or a literal,
    - two or more columns of view would otherwise have the same name.

```
CREATE VIEW          PQ ( P#, TOTQTY)
      AS SELECT      P#, SUM (QTY)
          FROM        SP
           GROUP BY    P#;
```

Database Systems

- Running Example

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date

DEPT_Location

Dnumber	Dlocation

PROJECT

Pname	Pnumber	Plocation	Dnum

WORKS_ON

Essn	Pno	Hours

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship

Database Systems

- Running Example

```
CREATE VIEW     WORKS_ON1
    AS SELECT   FNAME, LNAME, PNAME, HOURS
       FROM     EMPLOYEE, PROJECT, WORKS_ON
       WHERE    SSN=ESSN AND PNO=PNUMBER
```

WORKS_ON1

FNAME	LNAME	PNAME	HOURS

Database Systems

- Running Example

```
SELECT     FNAME, LNAME
  FROM     WORKS_ON1
 WHERE     PNAME=‘ProjectX’;
```

Will be translated into

```
SELECT     FNAME, LNAME
FROM       EMPLOYEE, PROJECT, WORKS_ON
WHERE      SSN=ESSN  AND  PNO=PNUMBER  AND
           PNAME=‘ProjectX’;
```

Database Systems

- Running Example

```
CREATE VIEW      DEPT_INFO(DEPT_NAME, NO_EMPLOYEES,
                 TOTAL_SALARY)
      AS SELECT  DNAME, COUNT(*), SUM (SALARY)
      FROM       DEPARTMENT, EMPLOYEE
      WHERE      DNUMBER=DNO
      GROUP BY   DNAME;
```

DEPT_INFO

DEPT_NAME	NO_EMPLOYEES	TOTAL_SALARY

Database Systems

- Running Example
  - DROP VIEW command can be used to dispose a view.

```
DROP VIEW     WORKS_ON1
```

Database Systems

Updating a View

- All views are not updatable. Consider the following examples:

```
CREATE VIEW        S#_City
      AS SELECT    S#, City
           FROM    S

CREATE VIEW       Status_City
      AS SELECT   Status, City
          FROM    S
```

Database Systems

S#_City

S#	Sname	Status	City
S_1	Smith	20	London
S_2	Jones	10	Paris
S_3	Blake	30	Paris
S_4	Clark	20	London
S_5	Adams	30	Athens

Status_City

S#	Sname	Status	City
S_1	Smith	20	London
S_2	Jones	10	Paris
S_3	Blake	30	Paris
S_4	Clark	20	London
S_5	Adams	30	Athens

Database Systems

Updating a View

- S#_City is theoretically updatable, since:
    - We can insert a new record into S#_City view, say (S_6, Rome), by actually inserting (S_6, NULL, NULL, Rome).
    - We can delete an existing record from it, say the record (S_1, London), by actually deleting (S_1, Smith, 20, London).
    - We can update an existing field in the view, say to change the *City* for supplier S_1 from ‘London’ to ‘Rome’.

Database Systems

Updating a View

- In case of Status_City:
  - We cannot insert a new record into the view, say the record (40, Rome).
  - We cannot delete a record from the view, say the record (20, London). The system will try to delete the corresponding record from the relation but which one?
  - Similarly, we cannot update a record in the view, say to change (20, London) to (20, Rome), the system will try to change the corresponding record in the base relation, but which one?

Database Systems

- Updating a View
  - A column-subset view is updatable if it preserves the primary key of the underlying base relation.

Database Systems

◆Updating a View

```
CREATE VIEW        London_Suppliers
   AS SELECT       S#, Sname, Status, City
        FROM       S
       WHERE       City = ‘London’;
```

- This is a row-subset view, it contains the primary key and hence, it is updatable.

Database Systems

Updating a View: Join View

- Consider the following;

```
CREATE VIEW    COLOCATED (S#, Sname, Status, SCity,
               P#, Pname, Color, Weight, PCity)
   AS SELECT    S#, Sname, Status, S.City,
                P#, Pname, Color, Weight, P.City
      FROM      S, P
     WHERE      S.City = P.City;
```

Database Systems

- Join View
  - From the standpoint of updatability, the COLOCATED view suffers from all kinds of problems — even though it does include the primary keys of the two base relations:
  - Assume we try to update the row;

    (S_1, Smith, 20, London, P_1, Nut, Red, 12, London)

    to

    (S_1, Smith, 20, Athens, P_1, Nut, Red, 12, London)

Database Systems

- Statistical Summary

```
CREATE VIEW          PQ ( P#, TOTQTY)
    AS SELECT        P#, SUM (QTY)
       FROM          SP
        GROUP BY      P#;
```

  - It is obvious that this view cannot support INSERT, or UPDATE operations against *TOTQTY*.
  - DELETE and UPDATE operations against P# theoretically could be defined to Delete or UPDATE all possible corresponding records in SP.

Database Systems

- Updating a View
  - In short some views are inherently updatable and some needs the help of human user for interpretation.

Database Systems

```
CREATE VIEW          GOOD_SUPPLIERS
     AS    SELECT     S#, Status, City
           FROM       S
           WHERE      Status > 15;
```

- This view is both column and row subset of the base relation, so it is updatable. However,
  - Supplier S_2 is not visible through the *GOOD_SUPPLIERS* view. This does not mean that the user can INSERT a record into the view with Supplier number value S_2, or UPDATE one of the records such that the supplier becomes S_2. Such operations must be rejected.

Database Systems

- Consider the following query

```
UPDATE      GOOD_SUPPLIERS
SET         Status = 5
WHERE       S# = ‘S1’;
```

- Should this update be acceptable? If so, it will remove S_1 from the view.
- Likewise, consider the following;

```
INSERT
INTO         GOOD_SUPPLIERS (S#, Status, City)
VALUE       ( ‘S8’, 5, ‘Stockholm’ ) ;
```

- If accepted, will create a new supplier but that will vanish instantly from the view.

Database Systems

- WITH CHECK OPTION

```
CREATE VIEW          GOOD_SUPPLIERS
        AS   SELECT   S#, Status, City
             FROM     S
             WHERE    Status > 15
             WITH CHECK OPTION;
```

- As a result of ‘WITH CHECK OPTION’ UPDATE and INSERT operations against the view will be checked to ensure that the updated and inserted rows satisfy the view-defining conditions.

Database Systems

- Views
  - A system provides Physical data independence, if users and user programs are independent of the physical structure of the stored data.
  - A system provides logical data independence, if user and user programs are independent of the logical structure of the database. Logical data independence has two aspects:
    - Growth
    - Restructuring

Database Systems

- Growth: As database grows to incorporate new kinds of information, so does the definition of the database. There are two possible types of growth that can occur;
  - The expansion of an existing base table.
  - The inclusion of a new base table.

- Neither of these two kinds of change should have any effect on existing users.

Database Systems

- **Restructuring:** Occasionally, it might become necessary to restructure the database in such a way that, although the overall information content remains the same, the placement of information within the database changes. For example assume we decided to split *S* relation into two relations;

  SX (S#, Sname, City)

  SY (S#, Status)

  - Note that the original *S* relation is the result of equijoin between *SX* and *SY*.

Database Systems

S#	Sname	Status	City
S_1	Smith	20	London
S_2	Jones	10	Paris
S_3	Blake	30	Paris
S_4	Clark	20	London
S_5	Adams	30	Athens

S#	Sname	City
S_1	Smith	London
S_2	Jones	Paris
S_3	Blake	Paris
S_4	Clark	London
S_5	Adams	Athens

S#	Status
S_1	20
S_2	10
S_3	30
S_4	20
S_5	30

Database Systems

- **Restructuring:** To accommodate this change, now we can define a view such as:

```
CREATE VIEW     S (S#, Sname, Status, City )
   AS SELECT    SX.S#, SX.Sname, SY.Status, SX.City
        FROM    SX, SY
       WHERE    SX.S# = SY.S#;
```

  - Any program that previously referred to base table *S* will now refer to view *S*.

Database Systems

- Advantages of view
  - A view provides a certain amount of logical data independence,
  - Views allow the same data to be seen by different users differently,
  - The user’s perception is simplified,
  - Automatic security is provided for hidden data.

Database Systems

- Putting things together
  - A relational system maintains a list of information about the relations and its contents in a set of tables (relations). Collection of these type of relations is called the system catalog or descriptor.
  - The catalog includes:
    - SYSTABLES
    - SYSCOLUMNS

⋮

Database Systems

- SYSTABLES
  - This table contains a row for every named table — base table or view in the entire system.
  - For each such table, it gives the table name, the owner, the number of columns, ….

SYSTABLES

S#	Creator	Colcount	...
S	Janice	4	...
P	Janise	5	...
SP	Janice	3	...

Database Systems

- SYSCOLUMNS
  - This table contains a row for every column of every table in the system. For each such column, it gives the column name, the name of the relation this column belong to, the data type of the column, ….

Database Systems

SYSCOLUMNS

Name	TBname	COLtype	…
S#	S	CHAR	…
Sname	S	CHAR	…
Status	S	SMALLINT	…
City	S	CHAR	…
P#	P	CHAR	…
Pname	P	CHAR	…
Color	P	CHAR	…
Weight	P	SMALLINT	…
City	P	CHAR	…
S#	SP	CHAR	…
P#	SP	CHAR	…
QTY	SP	INTEGER	…

Database Systems

- Querying The catalog
  - Since the catalog consists of tables, just like any other tables, it can be queried by means of SQL SELECT statements. For example;

```
SELECT    TBname
FROM      SYSCOLUMNS
WHERE     Name = ‘S#’;
```

will result:

TBname
S
SP

Database Systems

Querying The Catalog

- As another example, we can question, What columns does table *S* have?

```
SELECT   Name
FROM     SYSCOLUMNS
WHERE    TBName = ‘S’;
```

will result:

Name
S#
Sname
Status
City

Database Systems

- As final example, we may want to know, How many tables does user Janice own?

```
SELECT    Count (*)
FROM       SYSTABLES
WHERE     Creator = ‘Janice’;
```

Database Systems

- Updating the Catalog
  - The catalog cannot be updated using ordinary UPDATE, DELETE, and INSERT statements. This is just for protection of information. It would be very easy to destroy information in the catalog so that the system would not be able to function correctly.
  - Instead it is the data definition statements that perform such updates.

Database Systems

- Updating the Catalog
  - For example the CREATE TABLE statement for table *S* causes:
    - An entry to be made for *S* in the SYSTABLES table, and
    - Four entries, one for each of the four columns of *S*, to be made in the SYSCOLUMNS table.