Databases and Information Systems CS303

SQL

23-08-2023

VIEW

View definition

- CREATE VIEW faculty AS
  SELECT ID, name, dept_name
  FROM instructor;

- Views are not precomputed and stored

- Evaluated 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

instructor

View definition

- Create a view that lists all course sections offered by the Physics department in the Fall 2009 semester with the building and room number of each section

- CREATE VIEW physics_fall_2009 AS
  SELECT course.course_id, sec_id, building, room_number
  FROM course NATURAL JOIN section
  WHERE
  course.dept name = ‘Physics’
  AND section.semester = ‘Fall’
  AND section.year = ‘2009’;

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

course

course_id	*sec_id*	*semester*	*year*	*building*	*room_number*	*time_slot_id*
BIO-101	1	Summer	2009	Painter	514	B
BIO-301	1	Summer	2010	Painter	514	A
CS-101	1	Fall	2009	Packard	101	H
CS-101	1	Spring	2010	Packard	101	F
CS-190	1	Spring	2009	Taylor	3128	E
CS-190	2	Spring	2009	Taylor	3128	A
CS-315	1	Spring	2010	Watson	120	D
CS-319	1	Spring	2010	Watson	100	B
CS-319	2	Spring	2010	Taylor	3128	C
CS-347	1	Fall	2009	Taylor	3128	A
EE-181	1	Spring	2009	Taylor	3128	C
FIN-201	1	Spring	2010	Packard	101	B
HIS-351	1	Spring	2010	Painter	514	C
MU-199	1	Spring	2010	Packard	101	D
PHY-101	1	Fall	2009	Watson	100	A

section

View definition

- CREATE VIEW physics_fall_2009 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 = ‘2009’;

- SELECT course_id
  FROM physics_fall_2009
  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

course

course_id	*sec_id*	*semester*	*year*	*building*	*room_number*	*time_slot_id*
BIO-101	1	Summer	2009	Painter	514	B
BIO-301	1	Summer	2010	Painter	514	A
CS-101	1	Fall	2009	Packard	101	H
CS-101	1	Spring	2010	Packard	101	F
CS-190	1	Spring	2009	Taylor	3128	E
CS-190	2	Spring	2009	Taylor	3128	A
CS-315	1	Spring	2010	Watson	120	D
CS-319	1	Spring	2010	Watson	100	B
CS-319	2	Spring	2010	Taylor	3128	C
CS-347	1	Fall	2009	Taylor	3128	A
EE-181	1	Spring	2009	Taylor	3128	C
FIN-201	1	Spring	2010	Packard	101	B
HIS-351	1	Spring	2010	Painter	514	C
MU-199	1	Spring	2010	Packard	101	D
PHY-101	1	Fall	2009	Watson	100	A

section

Materialized Views

- Views that are explicitly stored in the database
- Should be kept up-to-date with respect to actual relations of the database (Materialized view maintenance)
- Updated either immediately or lazily or periodically
- No standard way to specify materialized views.

Updating a View

- Can you insert new tuples to views?
  - Add the appropriate rows to source relations
  - Throw error message saying it is not possible

- View is updatable if the following conditions are satisfied:
  - The FROM clause has only one database relation.
  - The SELECT clause contains only attribute names of the relation, and does not have any expressions, aggregates, or distinct specification.
  - Any attribute not listed in the SELECT clause does not have a not null constraint and is not part of a primary key.
  - The view query does not have a group by or having clause.

Updating a View

- CREATE VIEW history_instructors as
  SELECT *
  FROM instructor
  WHERE dept name= ‘History’;

- What if we insert
  (‘25566’, ‘Brown’, ‘Biology’, 100000)
  Into history_instructors

- Allowed generally
- But we can have check to stop such insertions

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

instructor

Transactions

Transactions

- SQL allows Commit / Roll back of a sequence of statements
- Allows definition of atomic sequence of statements
- More on this when we discuss Transactions

Integrity Constraints

Integrity Constraints

- NOT NULL
- PRIMARY KEY
- FOREIGN KEY (REFERENCES)
- UNIQUE (A1, A2...An)
- CHECK <condition>
- DEFAULT

Foreign Key Constraint

- CREATE TABLE course
  (…
  FOREIGN KEY (dept name) REFERENCES department
  ON DELETE CASCADE
  ON UPDATE SET NULL,
  . . .);

UNIQUE constraint

- 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),
  UNIQUE (course_id, sec_id, room_number)
) ;

CHECK

- 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’))
  );

- CHECK (time_slot_id IN (SELECT time_slot_id FROM time slot))

ASSERTION

- Complex CHECK conditions
- For each tuple in the student relation, the value of the attribute tot_cred must equal the sum of credits of courses that the student has completed.

- CREATE ASSERTION credits_earned_constraint CHECK
  (NOT EXISTS (
      SELECT ID
      FROM student
      WHERE tot_cred <>
        (SELECT sum(credits)
        FROM takes NATURAL JOIN course
        WHERE student.ID= takes.ID
        AND grade IS NOT NULL AND grade<> 'F'
)
)

ID	*name*	*dept_name*	*tot_cred*
00128	Zhang	Comp. Sci.	102
12345	Shankar	Comp. Sci.	32
19991	Brandt	History	80
23121	Chavez	Finance	110
44553	Peltier	Physics	56
45678	Levy	Physics	46
54321	Williams	Comp. Sci.	54
55739	Sanchez	Music	38
70557	Snow	Physics	0
76543	Brown	Comp. Sci.	58
76653	Aoi	Elec. Eng.	60
98765	Bourikas	Elec. Eng.	98
98988	Tanaka	Biology	120

student

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

course

ID	*course_id*	*sec_id*	*semester*	*year*	*grade*
00128	CS-101	1	Fall	2009	A
00128	CS-347	1	Fall	2009	A-
12345	CS-101	1	Fall	2009	C
12345	CS-190	2	Spring	2009	A
12345	CS-315	1	Spring	2010	A
12345	CS-347	1	Fall	2009	A
19991	HIS-351	1	Spring	2010	B
23121	FIN-201	1	Spring	2010	C+
44553	PHY-101	1	Fall	2009	B-
45678	CS-101	1	Fall	2009	F
45678	CS-101	1	Spring	2010	B+
45678	CS-319	1	Spring	2010	B
54321	CS-101	1	Fall	2009	A-
54321	CS-190	2	Spring	2009	B+
55739	MU-199	1	Spring	2010	A-
76543	CS-101	1	Fall	2009	A
76543	CS-319	2	Spring	2010	A
76653	EE-181	1	Spring	2009	C
98765	CS-101	1	Fall	2009	C-
98765	CS-315	1	Spring	2010	B
98988	BIO-101	1	Summer	2009	A
98988	BIO-301	1	Summer	2010	*null*

takes

DEFAULT

- 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));

- INSERT INTO student(ID, name, dept name) VALUES
  (12789, ‘Newman’, ‘Comp. Sci.’);

Index

- Find all instructors in the Physics department
- Find all courses of the student with ID 22201

- An index on an attribute allows to find those tuples in the relation that have a specified value for that attribute without scanning through all the tuples of the relation.

- CREATE INDEX InstDept_index ON instructor(dept_name);

Data Types

LARGE DATA TYPES

- clob() : large character data
- blob() : large binary data

- book_review clob(10KB)
- image blob(10MB)
- movie blob(2GB)

USER DEFINED DATA TYPES

- CREATE TYPE Rupees AS numeric(12,2);

- CREATE TABLE department
  (dept_name varchar (20),
  building varchar (15),
  budget Rupees);

USER DEFINED DATA TYPES

- CREATE TYPE Rupees AS numeric(12,2);
- CREATE TYPE Dollars AS numeric(12,2);

- Rupees and Dollars are of different data types.
- DROP TYPE Rupees

USER DEFINED DOMAIN TYPES

- CREATE DOMAIN NumericRupees AS numeric(12,2) NOT NULL;

- CREATE DOMAIN yearly_salary numeric(8,2)
  CONSTRAINT salary_minimum CHECK(value >= 29000.00);

- CREATE DOMAIN degree_level varchar(10)
  CONSTRAINT degree_values
  CHECK (degree_level IN (‘Bachelors’, ‘Masters’, ‘Doctorate’));

TABLE EXTENSIONS

- CREATE TABLE temp_instructor LIKE instructor;
  (new table temp_instructor will have same schema as instructor)

- CREATE TABLE t1 AS
  (SELECT ID, name
  FROM instructor
  WHERE dept_name= 'Music')
  WITH DATA;

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

instructor

- If WITH DATA is not present then we get an empty table with corresponding attributes

Authorization

Authorization: Grant

- Various privileges:
  - To read data.
  - To insert new data.
  - To update data.
  - To delete data.

- GRANT <privilege list>
  ON <relation name or view name>
  TO <user/role list>;

- GRANT select ON department TO Alice, Charles;

- GRANT update(budget) ON department TO Alice;

- select, insert, update, delete, all privileges

Authorization : Revoke

- Various privileges:
  - To read data.
  - To insert new data.
  - To update data.
  - To delete data.

- REVOKE <privilege list>
  ON <relation name or view name>
  FROM <user/role list>;

- REVOKE select ON department FROM Alice, Charles;

Roles and USER NAMES

- Instead of giving authorization to each individual user, we can give authorization to users based on their roles.

- CREATE ROLE admin;
- GRANT select ON instructor TO admin;
- CREATE USER Alice;
- GRANT admin TO Alice;
- CREATE ROLE dean;
- GRANT admin TO dean;

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

instructor

Roles and USER NAMES

- Admin incharge of Computer Science department can handle only instructors from Computer Science department.

    - CREATE VIEW comp_instructor AS
      (SELECT *
        FROM instructor
        WHERE dept_name = ‘Comp. Sci.’);

    - Creator of this view should have SELECT authorization on instructor

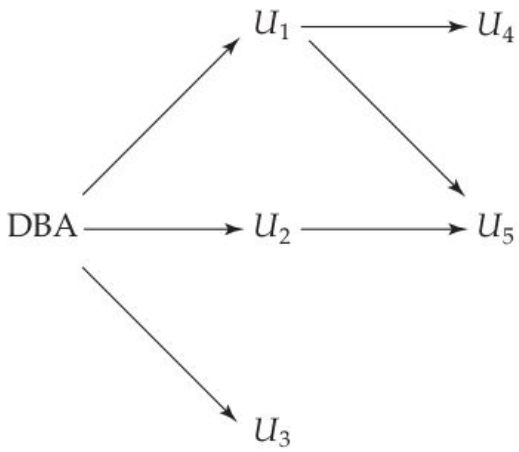
- SELECT * FROM comp_instructor;
    - Translates to a query on instructor relation

Authorization on Schema

- Only the owner of the schema can carry out any modification to the schema.
- Owner can give REFERENCES privilege to other users
    - GRANT REFERENCES (dept_name) ON department TO Alice;
- If Alice creates a new table which references dept_name in department table, then other users will be restricted on updating the department table.

Transfer of privileges

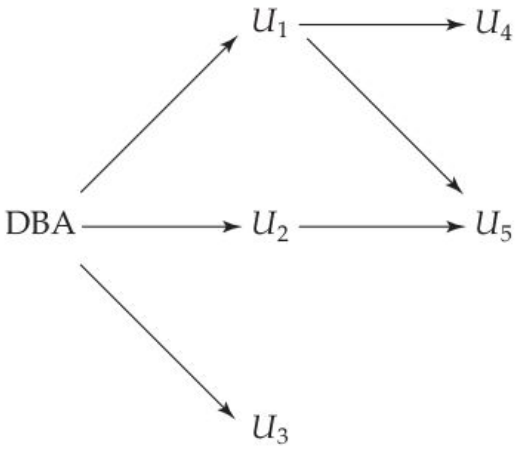
- GRANT SELECT ON department TO Alice WITH GRANT OPTION;



Revoking privileges

- REVOKE select ON department FROM U1 RESTRICT;
- REVOKE select ON department FROM U1 CASCADE;

- REVOKE GRANT OPTION FOR select ON department FROM U1;

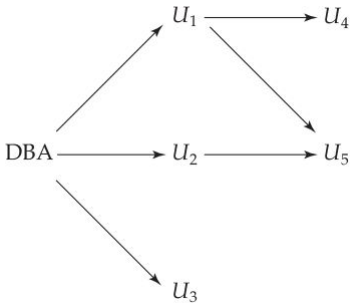


Revoking privileges

- CASCADING / RESTRICTING might not always be desired
  (What to revoke U1 but retain U4)

- Done by allowing a role to grant permission

- SET ROLE role_name;
  (Allowed only if the user is authorized for the role_name)



- GRANT SELECT ON department TO Alice GRANTED BY CURRENT_ROLE;

Reference:

Database System Concepts by Silberschatz, Korth and Sudarshan
Chapter 3,4