## Database Management Systems

Vijaya Saradhi

IIT Guwahati

Mon, 17<sup>th</sup> Feb 2020

### DDL

#### Overview

- Used in creating tables that is entities, relations
- Defining domains for each column that is attribute
- Express constraints on tables
- Modify tables
- Modify constraints
- Delete tables, columns within tables and constraints
- User need to have privileges for performing these operations
- Typically database administrator perform these operations
- Database users perform DML

### DDL - Create

#### Permanent table

```
CREATE TABLE student (
    roll_number CHAR(20),
    name CHAR(30),
    login CHAR(20),
    age INT);
```

### DDL - Create temporary table

### Temporary table

```
CREATE TEMPORARY TABLE student(
  roll_number CHAR(20),
  name CHAR(30),
  login CHAR(20),
  age INT);
```

## DDL - Expressing - Keys

### Expressing Keys

```
CREATE TABLE student(
    roll_number CHAR(20),
    name CHAR(30),
    login CHAR(20),
    age INT,
    UNIQUE(login),
    PRIMARY KEY(roll_number));
```

## Referential Integrity Constraint

#### Deletion

- Deletion of student with id 53666 not only affects student table but also grades table
- Possible scenarios of deletion
  - Delete all from grade which references 53666
  - Disallow the deletion of the student row 53666
  - Set the student id to some default value
  - Set the student\_id to null value.
  - However primary keys cannot assume null values and hence this is not a feasible option

```
CREATE TABLE student(
    roll_number CHAR(20),
    name CHAR(30) NOT NULL,
    login CHAR(20),
    age INT NOT NULL,
    UNIQUE(login),
    PRIMARY KEY(roll_number));
```

```
CREATE TABLE course(
    cid CHAR(6),
    title CHAR(20) NOT NULL,
    credits INT NOT NULL,
    PRIMARY KEY(cid));
```

```
CREATE TABLE registers(
    rn CHAR(20),
    course_id CHAR(6),
    PRIMARY KEY(rn, course_id),
    FOREIGN KEY(rn) REFERENCES student(roll_number),
    FOREIGN KEY(course_id) REFERENCES course(cid)
);
```

#### Specify actions

- What happens when a student row gets deleted?
- What happens when a student row gets updated?
- What happens when a course row gets deleted?
- What happens when a course row gets updated?

```
CREATE TABLE registers (
    rn CHAR(20),
    course_id CHAR(6),
    PRIMARY KEY(rn, course_id),
    FOREIGN KEY(rn) REFERENCES student(roll_number)
    ON DELETE CASCADE ON UPDATE NO ACTION,
    FOREIGN KEY(course_id) REFERENCES course(cid)
    ON DELETE CASCADE ON UPDATE CASCADE
);
```

```
CREATE TABLE manager(
    supervisor_id CHAR(20),
    supervisee_id CHAR(20),
    PRIMARY KEY(supervisor_id, supervisee_id),
    FOREIGN KEY(supervisor_id) REFERENCES employee(eid)
    FOREIGN KEY(supervisee_id) REFERENCES employee(eid)
);
```

### Delete all from grade which registers 53666

```
CREATE TABLE registers(
    rn CHAR(20),
    course_id CHAR(6),
    PRIMARY KEY(rn, course_id),
    FOREIGN KEY(rn) REFERENCES student(roll_number)
    ON DELETE CASCADE
);
```

#### Disallow the deletion of the student row

```
CREATE TABLE registers(
    rn CHAR(20),
    course_id CHAR(6),
    PRIMARY KEY(rn, course_id),
    FOREIGN KEY(rn) REFERENCES student(roll_number)
    ON DELETE RESTRICT
);
```

#### Set the student\_id to some default value

```
CREATE TABLE registers (
    rn CHAR(20),
    course_id CHAR(6),
    PRIMARY KEY(rn, course_id),
    FOREIGN KEY(rn) REFERENCES student(roll_number)
   ON DELETE SET DEFAULT rn='1234'
```

#### Set the student\_id to NULL value

```
CREATE TABLE registers(
    rn CHAR(20),
    course_id CHAR(6),
    PRIMARY KEY(rn, course_id),
    FOREIGN KEY(rn) REFERENCES student(roll_number)
    ON DELETE SET DEFAULT rn=NULL
);
```

- A constraint is checked at the end of every SQL statement
- Checks for constraint violations
- SQL statements gets rejected in the case of constraint violations
- Some times this causes inflexibility

#### Table 1

```
CREATE TABLE student(
    roll_number CHAR(20),
    name CHAR(30),
    login CHAR(20),
    age INT,
    honors CHAR(10) NOT NULL,
    UNIQUE(login),
    PRIMARY KEY(roll_number),
    FOREIGN KEY (honors) REFERENCES courses(cid)
);
```

#### Table 2

```
CREATE TABLE course(
   cid CHAR(6),
   title CHAR(20) NOT NULL,
   credits INT NOT NULL,
   grader CHAR(20) NOT NULL,
   PRIMARY KEY(cid),
   FOREIGN KEY(grader) REFERENCES student(roll_number)
);
```

#### Deffer constraint

Disable foreign key checks

```
SET foreign_key_checks = 0;
```

#### Enable constraint check

Enable foreign key checks

```
SET foreign_key_checks = 1;
```

#### DDL - Default Constraint

#### Setting Default Values

```
CREATE TABLE student(
  sid char(9) PRIMARY KEY,
  name varchar(30),
    phone char(10) DEFAULT '1234567890'
);
```

### DDL - Create Domain

#### **Domain Constraints**

- The DOMAIN is a new schema element
- You can think of this as providing alias to the an SQL data type statement
- Allows to declare in in-line macro
- ullet Syntax: CREATE DOMAIN <domain name> AS < data type >

```
CREATE DOMAIN CPI_DATA AS REAL CHECK (value >= 0 AND value <= 10);

CREATE TABLE student(
sid char(9) PRIMARY KEY,
name varchar(30),
cpi CPI_DATA
);
```

## DDL - Naming Constraint

#### Naming constraints

- Every constraint can be given a name
- Names are useful in creating, modifying and deleting constraints on tables
- Every constraint is prefixed with syntax CONSTRAINT [symbol] followed by the actual constraint
- In the following example, c1, c2 and c3 are the names given to each
  of the constraint

```
CONSTRAINT c1 UNIQUE(login)
CONSTRAINT c2 PRIMARY KEY(roll_number)
CONSTRAINT c3 FOREIGN KEY (honors) REFERENCES courses(cid)
```

## DDL - Creating Indexes

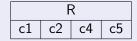
#### Indexes

- Provide handle on adding indexes to existing tables
- Following example creates an index using the first 10 characters of the name column

```
CREATE INDEX part_of_student_name ON student(name(10));
```

### DDL - Adding a column

### Altering Table



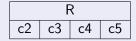
Adding a column between c2 and c4

ALTER TABLE R ADD COLUMN c3 INT AFTER c2; 2

> R c1 c2 с3 c4

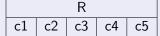
## DDL - Adding a column at the beginning

### Altering Table



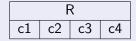
Adding a column c1 at the beginning

ALTER TABLE R ADD COLUMN c1 INT FIRST: 2



### DDL - Adding a column at the end

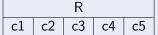
### Altering Table



Adding a column c1 at the end

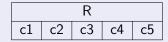
ALTER TABLE R ADD COLUMN c5 INT:

2



## DDL - Dropping a column

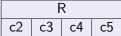
### Altering Table



• Dropping the column c1

```
1 ALTER TABLE R DROP COLUMN c1;
```

2



## DDL - Adding Constraints

```
Primary Key
CREATE TABLE R(c1 INT, c2 INT, c3 INT, c4 INT);
                              c2 | c3 |
                                       c4

    Adding a primary key c1

    ALTER TABLE R ADD CONSTRAINT my_c1 PRIMARY KEY(c1);
2
```

## DDL - Adding Constraints

```
Foreign Key

CREATE TABLE R(c1 INT, c2 INT, c3 INT, c4 INT, PRIMARY KEY(c1 ));

CREATE TABLE S(s1 INT, s2 INT, PRIMARY KEY(s1);
```

Adding a primary key c2 to R

```
ALTER TABLE R ADD CONSTRAINT my_c2_fkey FOREIGN KEY(c2) REFERENCES S(s1);
```

2

## DDL - Dropping Constraints

### Primary Key

ALATER TABLE R DROP CONSTRAINT my\_c1;

#### Foreign Key

ALTER TABLE R DROP CONSTRAINT my\_c2\_fkey;

### DDL - Changing Domains

### Altering Attribute Domains

ALTER TABLE R CHANGE c3 c3 CHAR(20);

ALTER TABLE R CHANGE c3 new\_c3 CHAR(20);

One has to be carful while changing the domains when with columns are either primary key or foreign key constraints.

#### DDL - Default Constraint

### **Expressing Default Constraint**

CREATE TABLE R(c1 INT, c2 INT DEFAULT 441, PRIMARY KEY(c1))

## Primary key vs temporal key

#### Example Schema

- eid and pcn stand for primary key
- Only in the absence of timed attributes
- start\_date and end\_date are included in the relation
- No employee can have a particular position twice at the same time.
- eid, pcn, start\_date, end\_date not a primary key

eid	pcn	start_date	end_date
123	900225	01-Jan-1996	01-June-1996
123	900225	01-Apr-1996	01-Oct-1996

# Primary key vs temporal key

```
CREATE TABLE Incumbents (eid INT, pcn INT, start_date date,
    end_date date.
   CHECK (
       NOT EXISTS (
           SFLFCT *
           FROM Incumbents as 11
           WHERE 1 <
            (SELECT COUNT(eid)
            FROM Incumbents as 12
            WHERE 11.eid = 12.eid
            AND I1.pcn = I2.pcn
            AND I1.start_date < I2.end_date
            AND | 12.start_date < | 11.end_date )
       AND NOT EXISTS (
           SFLFCT *
           FROM Incumbents AS I1
           WHERE II.eid is null OR II.pcn is null
```

### SELECT

#### Overview

- Consists of SIX clauses
- Combines selection and projection operators
- Optionally the following are specified
  - Extended operations
  - Groupy
  - sort (order by)

SELECT list of attributes

FROM list of tables

WHERE Condition

**GROUP BY** list of attributes

HAVING CONDITION

ORDER BY list of attributes

36 / 72

### Algebraic Operators and SQL

#### Overview

```
\sigma, \pi SELECT, FROM, WHERE
```

× comma separated table list after FROM clause

× table\_1 CROSS JOIN table\_2

★ table\_1 JOIN table\_2

Theta Join table\_1 JOIN table\_2 ON Condition

Re-naming AS: SELECT bname AS boat\_name FROM Boats

### Algebraic Operators and SQL

#### **Operators**

- **UNION**
- INTERSECTION (not available in all DBs)
- EXCEPT (not available in all DBs)

### Selection

```
\sigma_{attr3>6}(table1)
                                        table1
                                attr1 attr2
                                                 attr3
                                                  5
                                3
                                                  6
```

```
SELECT attr1, attr2, attr3
FROM table1
WHERE attr3 >= 6:
```

### Selection

# $\sigma_{attr3 \geq 6}(table1)$

table1		
attr1	attr2	attr3
1	2	5
3	4	6
1	2	7
1	2	7

```
SELECT
FROM
       table1
WHERE
       attr3 >= 6:
```

40 / 72

### Projection

### $\pi_{attr1,attr2}(table1)$

table1			
attr1 attr2 attr3			
1	2	5	
3	4	6	
1	2	7	
1	2	7	

```
SELECT
        attr1, attr2
FROM
        table1;
```

# Projection

### $\pi_{attr3}(table1)$

table1			
attr1	attr2	attr3	
1	2	5	
3	4	6	
1	2	7	
1	2	7	

```
SELECT
        attr3
FROM
        table1;
```

### Selection AND Projection

```
\pi_{attr2}(\sigma_{attr3 \geq 6}(table1))
```

table1		
attr1	attr2	attr3
1	2	5
3	4	6
1	2	7
1	2	7

```
 \begin{array}{ll} \text{SELECT} & \text{attr2} \\ \text{FROM} & \text{table1} \\ \text{WHERE} & \text{attr3} >= 6; \end{array}
```

### Cross Product

#### $table1 \times table2$

table1		tab	ole2
Α	В	В	D
1	2	2	3
1	2	4	5
		4	5

t	able1 >	< tabl	e2
Α	В	В	D
1	2	2	3
1	2	4	5
1	2	4	5
1	2	2	3
1	2	4	5
1	2	4	5

```
SELECT
FROM table1
CROSS JOIN table2;
```

### **Cross Product**

table1		tab	ole2
Α	В	В	D
1	2	2	3
1	2	4	5
		4	5

ta	ble1 >	< tabl	e2
Α	В	В	D
1	2	2	3
1	2	4	5
1	2	4	5
1	2	2	3
1	2	4	5
1	2	4	5

```
SELECT
FROM
        table1, table2;
```

### Cross Product - Projecting out duplicate columns

### $\pi_{A,B,D}(table1 \times table2)$

table1		tab	ole2
Α	В	В	D
1	2	2	3
1	2	4	5
		4	5

tabl	e $1  imes$	table2	
Α	В	D	
1	2	3	
1	2	5	
1	2	5	
1	2	3	
1	2	5	
1	2	5	
1	2	-	

```
SELECT A, table 1.B, D
FROM table1
CROSS JOIN table2;
```

#### Natural Join

tal	table1		ole2
Α	В	В	D
1	2	2	3
1	2	4	5
		4	5

	table1	⋈ table2	2
Α	В	В	D
1	2	2	3
1	2	2	3

- In the relational operator, the duplicate column gets projected out
- In SQL, SELECT clause decides for the columns to be retrieved
- The \* specifies retrieving all the columns

```
SELECT *
FROM table1
JOIN table2
ON table1.B = table2.B;
```

### Natural Join - Projecting out Duplicate Columns

### $\pi_{A,table1.B,D}(table1 \bowtie table2)$

table		tab	le2
Α	В	В	D
1	2	2	3
1	2	4	5
		4	5

```
SELECT
       A, table1.B, C
FROM
       table1
JOIN
       table2
       table1.B = table2.B;
ON
```

### Natural Join - Projecting out Duplicate Columns

table1		tab	le2
Α	В	В	D
1	2	2	3
1	2	4	5
		4	5

tabi	e1 ⋈	table2
Α	В	D
1	2	3
1	2	3

```
SELECT A, table2.B, C
FROM table1
JOIN table2
ON table1.B = table2.B;
```

### Theta Join

table1		tab	ole2
Α	В	В	D
1	2	2	3
1	2	4	5
		4	5

ta	ble1	⊲ tabl	e2
Α	В	В	D
1	2	4	5
1	2	4	5
1	2	4	5
1	2	4	5

```
SELECT
       A, table1.B, C
FROM
       table1
JOIN
       table2
WHERE
       table1.B < table2.B;
```

#### Natural Join AND Theta Join

$$\substack{table 1 \\ table 1.B = table 2.B \& table 1.A$$

table		tab	le2
Α	В	В	D
1	2	2	3
1	2	4	5
		4	5

```
A, table1.B, C
SELECT
FROM
       table1
JOIN table 2
ON table 1.B = table 2.B
WHERE table1.A < table2.D;
```

### Re-naming

```
\rho(\textit{RESULT}(A1, B1, B2, D1), \textit{table}1 \underset{\textit{table}1.B = \textit{table}2.B \& \textit{table}1.A < \textit{table}2.D}{\bowtie} table2)
```

table		tab	le2
Α	В	В	D
1	2	2	3
1	2	4	5
		4	5

```
        SELECT
        A AS A1, table1.B AS B1, table2.B AS B2, D AS D1

        FROM
        table1

        JOIN
        table2

        ON
        table1.B = table2.B

        WHERE
        table1.A < table2.D;</th>
```

### Re-naming

```
\rho(\textit{Result}(A1, B1, B2, D1), \textit{table}1 \underset{\textit{table}1.B = \textit{table}2.B \& \textit{table}1.A < \textit{table}2.D}{\bowtie} table2)
```

table		tab	le2
Α	В	В	D
1	2	2	3
1	2	4	5
		4	5

Result			
A1	B1	B2	D1
1	2	2	3
1	2	2	3

```
CREATE TABLE Result(

SELECT A AS A1, table1.B AS B1,

table2.B AS B2, D AS D1

FROM table1

JOIN table2

ON table1.B = table2.B

WHERE table1.A < table2.D;

);
```

### Distinct

### $\pi_{attr1,attr2}(table1)$

table1		
attr1	attr2	attr3
1	2	5
3	4	6
1	2	7
1	2	7

table1		
attr1 attr2		
1	2	
3	4	

54 / 72

SELECT DISTINCT attr1, attr2 FROM table1

## Aggregation Operations - SUM

#### Example

table1		
Α	В	
1	2	
3	4	
1	2	
1	2	

SUM(B) 10

```
SELECT SUM(B)
FROM table1;
```

### Aggregation Operations - Average

#### Example

table1		
Α	В	
1	2	
3	4	
1	2	
1	2	

AVG(A) 1.5

```
SELECT AVG(A)
FROM table1;
```

# Aggregation Operations - MIN

#### Example

table1		
Α	В	
1	2	
3	4	
1	2	
1	2	

MIN(A)

```
SELECT MIN(A)
FROM table1;
```

# Aggregation Operations - MAX

### Example

table1		
Α	В	
1	2	
3	4	
1	2	
1	2	

MAX(A)
3

```
SELECT MAX(A)
FROM table1;
```

### Aggregation Operations - COUNT

#### Example

table1		
Α	В	
1	2	
3	4	
1	2	
1	2	

COUNT(A)

SELECT COUNT(A)
FROM table1;

### **Extended Projection**

$$\pi_{A,B+C\to X}(table1)$$

table1				
A B C				
0	1	2		
0	1	2		
3	4	5		

```
SELECT A, (B + C) AS X FROM table 1;
```

### **Extended Projection**

$$\pi_{B-A\to X,C-B\to Y}(table1)$$

table1			
A B C			
0	1	2	
0	1	2	
3	4	5	

. . . .

```
SELECT (B - A) AS X, (C - B) AS Y FROM table 1;
```

# Sorting

### $\tau_A(table1)$

table1				
A B C				
3 4 5				
1	2			
7	1	2		

table1				
A B C				
1	1	2		
3	4	5		
7	1	2		

62 / 72

SELECT A, B, C FROM table1; ORDER BY A;

## Right Outer Join

### Right Outer Join

U			
Α	В	С	
1	2 5	3	
4		6	
7	8	9	
В	С	D	
2	3	10	
2 2 6	3	11	
6	7	12	

		U	√ V		
Α	В	С	В	С	D
1	2	3	2	3	10
1	2	3	2	3	11
T	1	1	6	7	12

```
SELECT *
FROM U
RIGHT OUTER JOIN V
ON U.B = V.B
AND U.C = V.C;
```

#### Left Outer Join

#### Left Outer Join



	$U\overset{\circ}{\bowtie}V$				
Α	В	С	В	С	D
1	2	3	2	3	10
1	2	3	2	3	11
4	5	6	$\perp$	$\perp$	$\perp$
7	8	9	T	T	T

**SELECT** FROM U LEFT OUTER JOIN V ON U.B = V.B AND U.C = V.C;

### $\gamma_{rating}(Sailors)$

Sailors				
sid	sname	rating	age	
22	Dustin	7	45.0	
29	Brutus	1	33.0	
31	Lubber	8	55.5	
32	Andy	8	25.5	
58	Rusty	10	35.0	
64	Horatio	7	35.0	
71	Zorba	10	16.0	
74	Horatio	9	35.0	
85	Art	3	25.5	
95	Bob	3	63.5	

	$\gamma_{rating}(\cdot)$	Sailors)	
sid	sname	rating	age
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
58	Rusty	10	35.0
74	Horatio	9	35.0
85	Art		25.5

### Group by rating

**SELECT** FROM Sailors **GROUP BY** rating;

#### Output

Janois			
sid	sname	rating	age
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

Sailors

	$\gamma_{rating}(Sailors)$				
sid	sname	rating	age		
22	Dustin	7	45.0		
29	Brutus	1	33.0		
31	Lubber	8	55.5		
58	Rusty	10	35.0		
74	Horatio	9	35.0		
85	Art	3	25.5		

```
Group by rating such that each group has at least two sailors
```

```
SELECT
FROM
        Sailors
GROUP BY rating
HAVING COUNT(rating) > 1;
```

#### Output

Sailors				
sic	sname	rating	age	
22	Dustin	7	45.0	
29	Brutus	1	33.0	
31	Lubber	8	55.5	
32	Andy	8	25.5	
58	Rusty	10	35.0	
64	Horatio	7	35.0	
71	Zorba	10	16.0	
74	Horatio	9	35.0	
85	Art	3	25.5	

$\gamma_{rating}(Sa)$	ilors)	
ame r	ating a	ige
ıstin 7	' 4	5.0
ıbber 8	5 5	5.5
ısty 1	.0 3	35.0
t 3	2	25.5
	ame r ustin 7 ubber 8	ustin 7 4 ubber 8 5 usty 10 3

```
Group by rating such that each group has at least two sailors where sailor age \geq 30
```

```
SELECT
FROM Sailors
WHERE age >= 30
GROUP BY rating
HAVING COUNT(rating) > 1;
```

#### Output

Sailors				
sid	sname	rating	age	
22	Dustin	7	45.0	
29	Brutus	1	33.0	
31	Lubber	8	55.5	
32	Andy	8	25.5	
58	Rusty	10	35.0	
64	Horatio	7	35.0	
71	Zorba	10	16.0	
74	Horatio	9	35.0	

	$\gamma_{rating}$	(Sailors)	
sid	sname	rating	age
22	Dustin	7	45.0
31	Lubber	8	55.5
58	Rusty	10	35.0

#### All six clauses of SELECT

```
Group by rating such that each group has at least two sailors where sailor age \geq 20 sort by sailor names
```

```
SELECT
FROM Sailors
WHERE age >= 20
GROUP BY rating
HAVING COUNT(rating) > 1
ORDER BY
           sname;
```

#### Output

Sailors				
sid	sname	rating	age	
22	Dustin	7	45.0	
29	Brutus	1	33.0	
31	Lubber	8	55.5	
32	Andy	8	25.5	
58	Rusty	10	35.0	
64	Horatio	7	35.0	
71	Zorba	10	16.0	

$\gamma_{rating}(Sailors)$					
sid	sname	rating	age		
85	Art	3	25.5		
22	Dustin	7	45.0		
31	Lubber	8	55.5		
58	Rusty	10	35.0		

### Set Operator - Union

#### $\mathsf{table1} \cup \mathsf{table2}$

table1		tab	ole2
Α	В	В	D
1	2	2	3
1	2	4	5
		4	5

tab	le1	$\bigcup$	table2
Α	В		
1	2		
2	3		
4	5		

```
(SELECT *
FROM table1)
UNION
(SELECT *
FROM table2);
```

## Set Operator - Intersection

#### $table1 \cap table2$

table1		tab	le2
Α	В	В	D
1	2	2	3
1	2	4	5
		4	5

 $\begin{array}{c|c} \mathsf{table1} \cap \mathsf{table2} \\ \hline \mathsf{A} & \mathsf{B} \end{array}$ 

```
SELECT *
FROM table1
WHERE (a, b)

IN

(SELECT *
FROM table2);
```

### Set Operator - Difference

#### table1 - table2

tal	table1		table2	
Α	В	В	D	
1	2	2	3	
1	2	4	5	
		4	5	

table1		_	table2
Α	В		
1	2		
2	3		
4	5		

```
SELECT *
FROM table1
WHERE (a, b)
NOT IN
(SELECT
FROM
        table2);
```