CS245: Databases

SQL

Vijaya saradhi

Department of Computer Science and Engineering
Indian Institute of Technology Guwahati

Creation

Elements to be created

- Database
- Table
- Constraint
- Function
- Procedure
- Trigger
- Event
- And other elements

All these elements use **CREATE** statement

Database Creation

Database

- Database is a collection of tables (and programs that manipulate tables)
- Tables cannot exist independently. They must be under a specified database
- In order to create tables, we needs to create a database

Database Creation - SQL statement

Invoking MySQL client

```
mysql -uroot -p
```

```
CREATE DATABASE cs245;
```

```
CREATE DATABASE IF NOT EXISTS cs245;
```

Database Creation - SQL statement

Using a specific database

- DBMS system may hold several databases. We need to specify which database we intend to use
- Only after this step, tables can be created

Specifying Database to Use

```
USE cs245;
```

Creating a table

Syntax

```
CREATE TABLE table_name(
        column_1 data_type,
        column_2 data_type,
    ... column_n data_type);
```

Creating a table

```
CREATE TABLE student(
               sid int,
               name char(50),
               login char(10),
               age int, spi float);
```

student				
sid	name	login	age	spi

CREATE statement

- `sid` is column name `int` is its data type
- `name` is column name `char(50)` is its data type
- `login` is column name `char(10)` is its data type
- `age` is column name `int` is its data type
- `spi` is column name `float` is its data type
- There are no constraints on this table

Creating a table

student				
sid	name	login	age	spi
190101000	Atul Kumar	atul	18	8.0
190101000	Atul Gupta	atul	18	8.2
190101000	Atul M	atul	18	8.2
190101000	Atul Gupta	atul	19	7.2

- Same roll number is assigned to several students
- Same login is assigned to several students
- It is not possible to distinguish between two Atul Gupta's (row 2 & 4)
- In case you have to update the spi of Atul Gupta which row will you update? 2 or 4?

Creating another table

```
CREATE TABLE register(sid int, grade char(2), cid char(6));
```

register		
sid	grade	cid
190101000	AB	CS101
190101000	BB	CS101
190109001	AA	CS101
190109001	BB	CS101

Creating a table with primary key

```
CREATE TABLE student(
                sid int primary key,
                name char(50),
                login char(10),
                age int,
                spi float);
```

Creating a table with primary key

```
CREATE TABLE student(
                sid int primary key,
                name char(50),
                login char(10),
                age int,
                spi float);
```

student				
sid	name	login	age	spi
190101000	Atul Kumar	atul	18	8.0
190101001	Atul Gupta	atul	18	8.2

Creating a table with primary key

```
CREATE TABLE student(
                sid int primary key,
                name char(50),
                login char(10),
                age int,
                spi float);
```

Inserting two identical values of primary key is not allowed

student

sid	name	login	age	spi
190101000	Atul Kumar	atul	18	8.0
190101001	Atul Gupta	atul	18	8.2
190101000	Atul Kumar	atul	18	8.0

Creating a table with two constraints

```
CREATE TABLE student(sid int primary key,
                     name char(50),
                     login char(10) unique,
                     age int,
                     spi float);
```

student				
sid	name	login	age	spi
190101000	Atul Kumar	atul	18	8.0
190101000	Atul Gupta	atul01	18	7.2
190101001	Atul Gupta	atul	18	6.2
190101001	Atul Gupta	atul02	18	8.6

Creating a table with unique key

One constraint alone

- Table with NO primary key constraint
- Having UNIQUE constraint on `login` column
- Note login can take NULL values.

```
CREATE TABLE student(sid int,
                       name char(50),
                       login char(10) unique,
                       age int,
                       spi float);
```

student

sid	name	login	age	spi
190101001	Atul Kumar	atul	18	8.0
190101002	Atul Gupta	atul	18	8.2
190101003	Atul M	atul01	18	7.2
190101004	Atul K	⊥	18	6.2
190101005	Atul H	⊥	18	8.6

Creating a table with primary key

One constraint alone

- Table with NO primary key constraint
- Having UNIQUE constraint on `login` column
- Having NOT NULL constraint on `login` column
- This is identical to specifying `login` column as primary key implicitly

```
CREATE TABLE student(sid int,
                     name char(50),
                     login char(10) unique not null,
                     age int,
                     spi float);
```

student

sid	name	login	age	spi
190101001	Atul Kumar	atul	18	8.0
190101002	Atul Gupta	atul	18	8.2
190101003	Atul M	atul01	18	7.2
190101004	Atul K	⊥	18	6.2
190101005	Atul H	atul02	18	8.6

Creating a table with not null columns

```
CREATE TABLE student(sid int,
                     name char(50),
                     login char(10),
                     age int not null,
                     spi float);
```

student

sid	name	login	age	spi
⊥	Atul Kumar	atul	18	8.0
190101000	Atul Gupta	atul	⊥	8.2
190101000	Atul Gupta	atul01	18	⊥
190101000	Atul Gupta	⊥	18	8.2
190101000	⊥	atul02	18	8.2

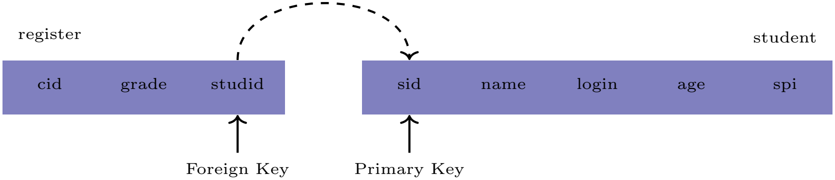
Foreign Key

Cases

- Needs two or more tables (Need not be distinct tables!)
- Each (refering) table must have primary key constraint
- Each (refering) table: primary key is expressed using only one column
- One (refering) table has primary key expressed on only one column. Other (refering) table(s) express primary key using two or more columns
- All (refering) tables express primary key using two or more columns

Draw figure for this explanation;

Foreign Key - Example 1



register

cid grade studid

student

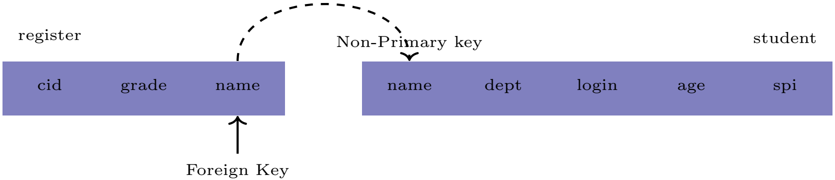
sid name login age spi

Foreign Key

Primary Key

```
CREATE TABLE student(sid int primary key,
                     name char(50),
                     login char(20) unique,
                     age int,
                     spi float);

CREATE TABLE register(cid int,
                      grade char(2),
                      studid int,
                      primary key(cid, studid),
                      foreign key(studid) references student(sid));
```

Foreign Key - Example 2 (Error)



register

cid grade name

Non-Primary key

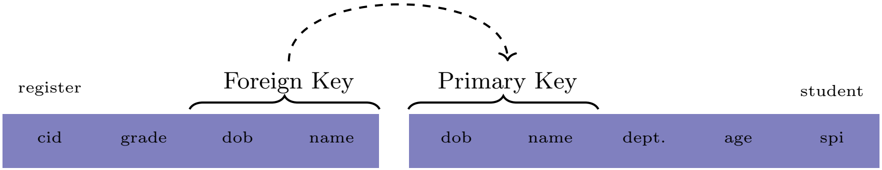
student

name dept login age spi

Foreign Key

```
CREATE TABLE student(name char(50),
                     dept char(10),
                     login char(20) unique,
                     age int,
                     spi float);

CREATE TABLE register(cid int,
                      grade char(2),
                      name char(50),
                      primary key(cid, name)
                      foreign key(name) references student(name));
-- Error name is not primary key in student table
```

Foreign Key - Example 3



register

Foreign Key

Primary Key

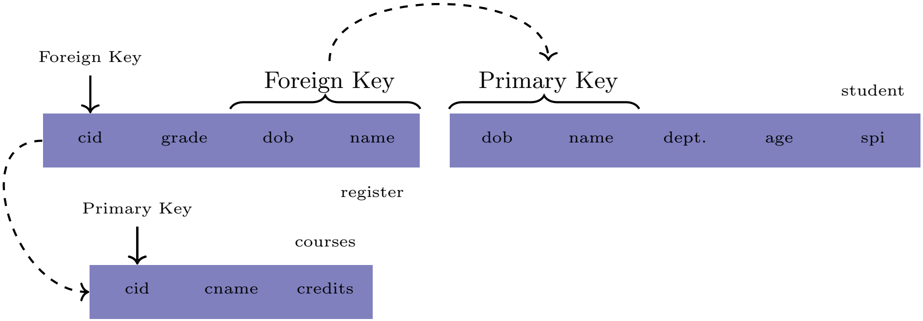
student

cid	grade	dob	name

dob	name	dept.	age	spi

```
CREATE TABLE student(dob date,
                     name char(50),
                     dept char(10),
                     age int,
                     spi float,
                     primary key(dob, name));

CREATE TABLE register(cid int,
                      grade char(2),
                      dob date,
                      name char(50),
                      primary key(cid, dob, name),
                      foreign key(dob, name) references student(dob, name));
```

Foreign Key - Example 4



Foreign Key

Foreign Key

Primary Key

student

cid	grade	dob	name

dob	name	dept.	age	spi

register

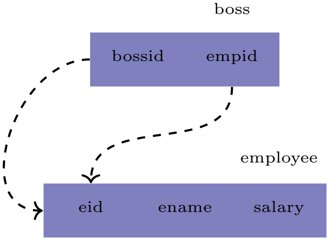
Primary Key

courses

cid	cname	credits

```
CREATE TABLE courses(cid char(2),
                     cname char(50),
                     credits char(6),
                     primary key(cid));

CREATE TABLE register(cid int,
                      grade char(2),
                      dob date,
                      name char(50),
                      primary key(cid, dob, name),
                      foreign key(dob, name) references student(dob, name),
                      foreign key(cid) references courses(cid));
```

Foreign Key - Example 5



boss

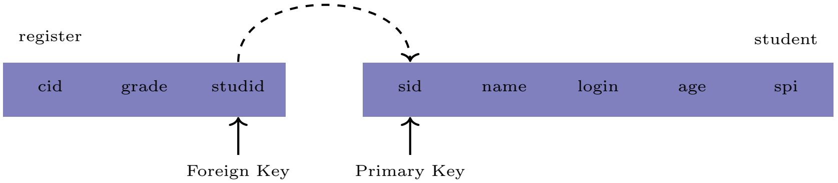
bossid empid

employee

eid ename salary

```
CREATE TABLE employee(eid char(10),
                      ename char(50),
                      salary bigint,
                      primary key(eid));

CREATE TABLE boss(bossid char(10),
                  empid char(10),
                  primary key(bossid, empid),
                  foreign key(bossid) references employee(eid),
                  foreign key(empid) references employee(eid));
```

Foreign Key - Example 1 (Deleting paraent table row) - Action 01 (Delete)



register

cid	grade	studid

student

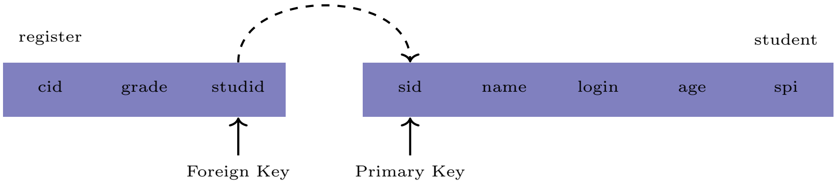
sid	name	login	age	spi

Foreign Key

Primary Key

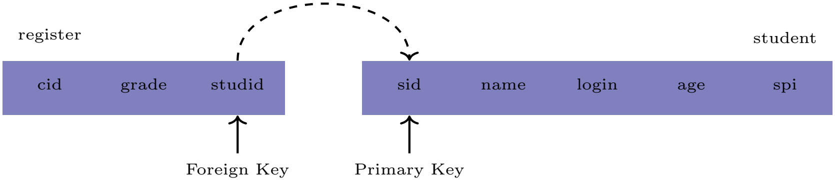
```
CREATE TABLE student(sid int primary key,
                     name char(50),
                     login char(20) unique,
                     age int,
                     spi float);

CREATE TABLE register(cid int,
                      grade char(2),
                      studid int,
                      primary key(cid, studid),
                      foreign key(studid) references student(sid) ON DELETE CASCADE);
```

Foreign Key - Example 1 (Deleting paraent table row) - Action 02 (SET Default value)



- **SET DEFAULT:** This action is recognized by the MySQL parser, but both InnoDB and NDB reject table definitions containing ON DELETE SET DEFAULT or ON UPDATE SET DEFAULT clauses.

Foreign Key - Example 1 (Deleting paraent table row) - Action 03 (set NULL)



register

cid grade studid

student

sid name login age spi

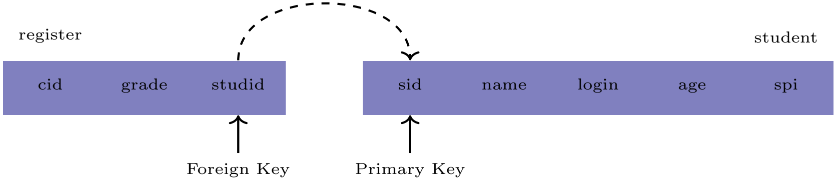
Foreign Key

Primary Key

- **SET NULL:** NOT NULL constraint should not be placed on **studid**

```
CREATE TABLE student(sid int primary key,
                     name char(50),
                     login char(20) unique,
                     age int,
                     spi float);

CREATE TABLE register(cid int,
                      grade char(2),
                      studid int,
                      primary key(cid, studid),
                      foreign key(studid) references student(sid) ON DELETE SET NULL);
```

Foreign Key - Example 1 (Deleting paraent table row) - Action 04 (Disallow)



register

cid grade studid

student

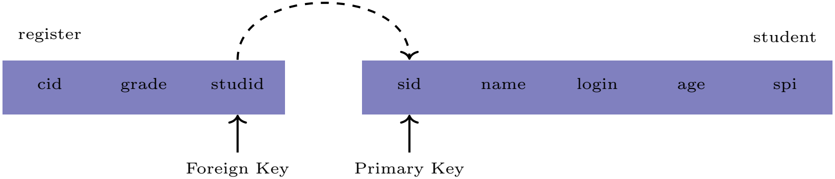
sid name login age spi

Foreign Key

Primary Key

```
CREATE TABLE student(sid int primary key,
                     name char(50),
                     login char(20) unique,
                     age int,
                     spi float);

CREATE TABLE register(cid int,
                      grade char(2),
                      studid int,
                      primary key(cid, studid),
                      foreign key(studid) references student(sid) ON DELETE RESTRICT);
```

Foreign Key - Example 1 (Updating paraent table row) - Action 01 (Update)



register

cid	grade	studid

student

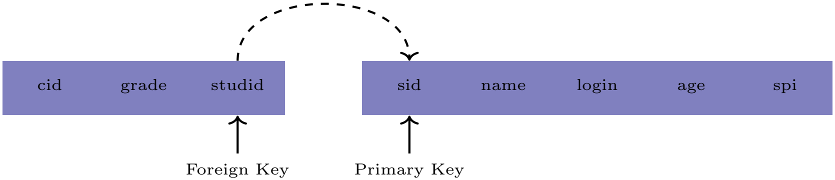
sid	name	login	age	spi

Foreign Key

Primary Key

```
CREATE TABLE student(sid int primary key,
                     name char(50),
                     login char(20) unique,
                     age int,
                     spi float);

CREATE TABLE register(cid int,
                      grade char(2),
                      studid int,
                      primary key(cid, studid),
                      foreign key(studid) references student(sid) ON UPDATE CASCADE);
```

Foreign Key - Example 1 (Updating paraent table row) - Action 02 (SET Default value)



cid	grade	studid

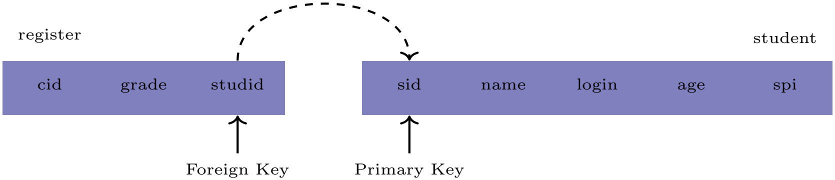
sid	name	login	age	spi

Foreign Key

Primary Key

- **SET DEFAULT:** This action is recognized by the MySQL parser, but both InnoDB and NDB reject table definitions containing ON DELETE SET DEFAULT or ON UPDATE SET DEFAULT clauses.

Foreign Key - Example 1 (Updating paraent table row) - Action 03 (set NULL)



register

cid | grade | studid

student

sid | name | login | age | spi

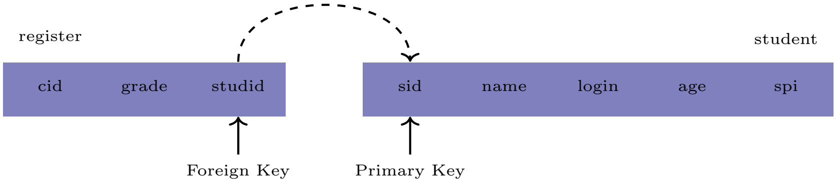
Foreign Key

Primary Key

- **SET NULL:** NOT NULL constraint should not be placed on `studid`

```
CREATE TABLE student(sid int primary key,
                     name char(50),
                     login char(20) unique,
                     age int,
                     spi float);

CREATE TABLE register(cid int,
                      grade char(2),
                      studid int,
                      primary key(cid, studid),
                      foreign key(studid) references student(sid) ON UPDATE SET NULL);
```

Foreign Key - Example 1 (Updating paraent table row) - Action 04 (Disallow)



register

cid	grade	studid

student

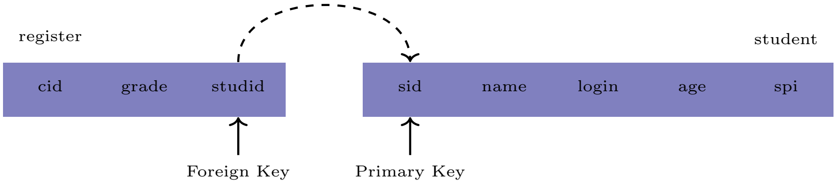
sid	name	login	age	spi

Foreign Key

Primary Key

```
CREATE TABLE student(sid int primary key,
                     name char(50),
                     login char(20) unique,
                     age int,
                     spi float);

CREATE TABLE register(cid int,
                      grade char(2),
                      studid int,
                      primary key(cid, studid),
                      foreign key(studid) references student(sid) ON UPDATE RESTRICT);
```

Foreign Key - Example 1 (Deleting/Updating) - Action 01 (Delete/Update)



register

cid	grade	studid

student

sid	name	login	age	spi

Foreign Key

Primary Key

```
CREATE TABLE student(sid int primary key,
                     name char(50),
                     login char(20) unique,
                     age int,
                     spi float);

CREATE TABLE register(cid int,
                      grade char(2),
                      studid int,
                      primary key(cid, studid),
                      foreign key(studid) references student(sid)
                      ON DELETE CASCADE
                      ON UPDATE CASCADE);
```

DDL - Table manipulations

Possible manipulations

- Add column at beginning
- Add column at the middle
- Add column at the end
- Delete column
- Specify data type
- Modify data type
- Add constraints
- Delete constraints

DDL - Adding a column at the beginning

Altering Table

R			
c2	c3	c4	c5

DDL - Adding a column at the beginning

Altering Table

R			
c2	c3	c4	c5

- Adding a column c1 at the beginning

```
ALTER TABLE R ADD COLUMN c1 INT FIRST;
```

DDL - Adding a column at the beginning

Altering Table

R			
c2	c3	c4	c5

- Adding a column c1 at the beginning

```
ALTER TABLE R ADD COLUMN c1 INT FIRST;
```

R				
c1	c2	c3	c4	c5

DDL - Adding a column at the beginning

R: before adding c1

c2	c3	c4	c5
1	2	3	4
1	2	3	5
1	2	4	6

DDL - Adding a column at the beginning

R: before adding c1

c2	c3	c4	c5
1	2	3	4
1	2	3	5
1	2	4	6

R: after adding c1

c1	c2	c3	c4	c5
⊥	1	2	3	4
⊥	1	2	3	5
⊥	1	2	4	6

DDL - Adding a column at the beginning

- Existing rows will be unaltered
- Values for the new column for each existing rows is not specified
- $\perp$ by default is added to the existing rows

DDL - Adding a column

Altering Table

R			
c1	c2	c4	c5

DDL - Adding a column

Altering Table

R			
c1	c2	c4	c5

- Adding a column between c2 and c4

```
ALTER TABLE R ADD COLUMN c3 INT  AFTER c2 ;
```

DDL - Adding a column

Altering Table

R			
c1	c2	c4	c5

- Adding a column between c2 and c4

```
ALTER TABLE R ADD COLUMN c3 INT AFTER c2;
```

R				
c1	c2	c3	c4	c5

DDL - Adding a column at the end

Altering Table

R			
c1	c2	c3	c4

DDL - Adding a column at the end

Altering Table

R			
c1	c2	c3	c4

- Adding a column c1 at the end

```
ALTER TABLE R ADD COLUMN c5 INT;
```

DDL - Adding a column at the end

Altering Table

R			
c1	c2	c3	c4

- Adding a column c1 at the end

```
ALTER TABLE R ADD COLUMN c5 INT;
```

R				
c1	c2	c3	c4	c5

DDL - Dropping a column (with no constraints)

Altering Table

R				
c1	c2	c3	c4	c5

DDL - Dropping a column (with no constraints)

Altering Table

R				
c1	c2	c3	c4	c5

- Dropping the column c1

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

DDL - Dropping a column (with no constraints)

Altering Table

R				
c1	c2	c3	c4	c5

- Dropping the column c1

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

R			
c2	c3	c4	c5

DDL - Adding Constraints

Primary Key

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

R			
c1	c2	c3	c4

DDL - Adding Constraints

Primary Key

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

R			
c1	c2	c3	c4

- Adding a primary key c1

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

DDL - Adding Constraints

Foreign Key

```sql
CREATE TABLE R(c1 INT, c2 INT, c3 INT, c4 INT, PRIMARY KEY(c1));
CREATE TABLE S(s1 INT, s2 INT, PRIMARY KEY(s1));
```

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));
```

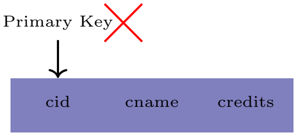
- Adding a foreign key c2 to R

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

DDL - Dropping constraints

- Primary key - simple case
- Primary key - complex case (includes dropping foreign key)
- NULL
- DEFAULT

Primary key deletion - simple case

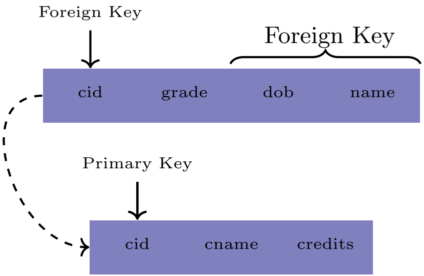


Primary Key

cid cname credits

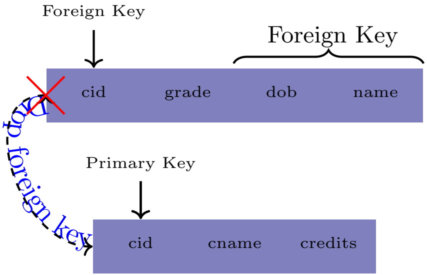
ALTER TABLE R **DROP PRIMARY KEY**;

Primary key deletion - complex case



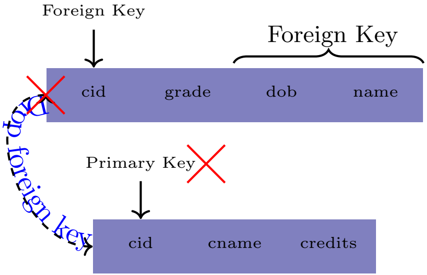
- Remove all foreign keys
- Delete the primary key

Primary key deletion - complex case



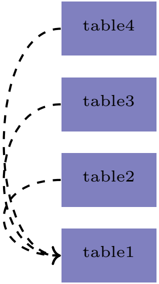
ALTER TABLE R **DROP FOREIGN KEY** my_cid_fkey;

Primary key deletion - complex case



```
ALTER TABLE R DROP FOREIGN KEY my_cid_fkey;
ALTER TABLE R DROP PRIMARY KEY;
```

Primary key deletion - complex case



- Drop foreign key from table4
- Drop foreign key from table3
- Drop foreign key from table2
- Delete the primary key from table1

DDL - Dropping default constraint

DEFAULT value

R			
c2	c3	c4	c5

DDL - Dropping default constraint

DEFAULT value

R			
c2	c3	c4	c5

ALTER TABLE R **ADD COLUMN** C1 **INT DEFAULT** 10 **FIRST**;

R				
c1	c2	c3	c4	c5

DDL - Dropping default constraint

DEFAULT value

R			
c2	c3	c4	c5

ALTER TABLE R **ADD COLUMN** C1 **INT DEFAULT** 10 **FIRST**;

R				
c1	c2	c3	c4	c5

ALTER TABLE R **DROP COLUMN** C1;

DDL - Dropping NOT NULL constraint

NOT NULL column

R			
c2	c3	c4	c5

DDL - Dropping NOT NULL constraint

NOT NULL column

R			
c2	c3	c4	c5

```
ALTER TABLE R ADD COLUMN C1 INT NOT NULL 10 FIRST;
```

R				
c1	c2	c3	c4	c5

DDL - Dropping NOT NULL constraint

NOT NULL column

R			
c2	c3	c4	c5

```
ALTER TABLE R ADD COLUMN C1 INT NOT NULL 10 FIRST;
```

R				
c1	c2	c3	c4	c5

```
ALTER TABLE R DROP COLUMN C1;
```

DDL - Changing Domains

Altering Attribute Domains

```
ALTER TABLE R CHANGE c3 c3 CHAR(20);

ALTER TABLE R CHANGE c3 new_c3 CHAR(30);
```

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

DDL - changing domains pit falls

Altering Attribute Domains

c1 (int)
129
130
131
132

```
ALTER TABLE R CHANGE c1 c1 tinyint;
```

will result in an error and the operation is not permitted due to `Out of range value for column 'c1'`

DDL - Changing Domains

Altering Attribute Domains

c1 (int)
1
2
3
4

```
ALTER TABLE R CHANGE c1 c1 tinyint;
```

No issues. c1 is made `tinyint`.

DDL - Default Constraint

Expressing Default Constraint

```
CREATE TABLE R(c1 INT, c2 INT DEFAULT 245, PRIMARY KEY(c1))
```

Data inserting/updating/deletion

- Inserting rows into table
  - One row
    - Insert all the columns of the row
    - Inserting fewer columns of the row
  - DEFAULT columns cases
  - Two rows
  - Loading a local file
- Updating rows in the table
  - One row
  - Multiple rows
- Deleting rows from the table
  - One row
  - Multiple rows

Insert one row

Inserting one row

- Insert all the columns of the row
- Inserting fewer columns of the row
- Specify table into which the row will be inserted
- Is the row added at the beginning? in the middle? or at the end?

R				
c1	c2	c3	c4	c5
1	2	3	4	5

```
INSERT INTO R(c1 , c2 , c3 , c4 , c5) VALUES (1, 2, 3, 4, 5);
```

Insert one row

Inserting one row

R				
c1	c2	c3	c4	c5
1 10	2 20	3 30	4 40	5 50

```
INSERT INTO R(c1, c2, c3, c4, c5) VALUES (10, 20, 30, 40, 50);
```

Insert one row - specify few columns

All columns having no constraints

R				
c1	c2	c3	c4	c5
1	2	3	4	5
10	20	30	40	50
100	⊥	300	400	⊥

```
INSERT INTO R(c1, c3, c4) VALUES (100, 300, 400);
```

Insert one row - specify few columns

c2 cannot take NULL values

say c2 has NOT NULL constraint
constraint violation: INSERT statement is rejected by DBMS

R				
c1	c2	c3	c4	c5
1	2	3	4	5
10	20	30	40	50
100	⊥	300	400	⊥

```
INSERT INTO R(c1 , c3 , c4) VALUES (15 , 35 , 45);
```

Insert one row - specify few columns

DEFAULT value constraint

say c2 has DEFAULT value constraint as 250
while inserting, only c1, c3 & c4 values are being inserted, due to default constraint on column c2, 250 also insert along with c1 = 150, c3 = 350, c4 = 450

R				
c1	c2	c3	c4	c5
1	2	3	4	5
10	20	30	40	50
100	⊥	300	400	⊥
150	250	350	450	⊥

```
INSERT INTO R(c1, c3, c4) VALUES (150, 350, 450);
```

Insert one row - specify few columns

FOREIGN KEY constraint

say c2 is a foreign key pointing to `cid` of table S
Table S do not have cid=22 (c2)
INSERT statement will be rejected by DBMS

R				
c1	c2	c3	c4	c5
1	2	3	4	5
10	20	30	40	50
100	200	300	400	⊥
150	250	350	450	⊥

S		
cid	cname	cedits
2	SQL	3
20	C++	6
200	R	4
250	Python	8

```
INSERT INTO R(c1, c2, c3, c4, c5) VALUES (11, 22, 33, 44, 55);
```

Insert two rows

Inserting two rows

R				
c1	c2	c3	c4	c5
1	2	3	4	5
10	20	30	40	50
100	⊥	300	400	⊥
150	250	350	450	⊥
170	270	370	470	570
180	280	380	480	580

```
INSERT INTO R(c1, c2, c3, c4, c5) VALUES (170, 270, 370, 470, 570), (180, 280, 380, 480, 580);
```

Insert a local file into a table

File must meet all table constraints

Invoke mysql as: `mysql -uroot -p --local-infile`
to read data from local files

R				
c1	c2	c3	c4	c5

```
LOAD DATA LOCAL INFILE '/home/saradhi/tableR-data.csv'
INTO TABLE R
FILEDS TERMINATED BY ','
LINES TERMINATED BY '\n';
```

Insert a local file into a table

File must meet all table constraints

First line of the file contains header; ignore header

R				
c1	c2	c3	c4	c5

```
LOAD DATA LOCAL INFILE '/home/saradhi/tableR-data.csv'
INTO TABLE R
 FILEDS TERMINATED BY ','
 LINES TERMINATED BY '\n'
IGNORE 1 LINES;
```

Insert a local file into a table

File must meet all table constraints

First 10 lines of the file contains header and comments; ignore them

R				
c1	c2	c3	c4	c5

```
LOAD DATA LOCAL INFILE '/home/saradhi/tableR-data.csv'
INTO TABLE R
FILEDS TERMINATED BY ','
LINES TERMINATED BY '\n'
IGNORE 10 LINES;
```

Insert a local file into a table

File must meet all table constraints

Columns are separated by space

R				
c1	c2	c3	c4	c5

```
LOAD DATA LOCAL INFILE '/home/saradhi/tableR-data.csv'
INTO TABLE R
 FILEDS TERMINATED BY '␣'
 LINES TERMINATED BY '\n'
IGNORE 10 LINES;
```

Insert a local file into a table

File must meet all table constraints

Columns are separated by '#'

R				
c1	c2	c3	c4	c5

```
LOAD DATA LOCAL INFILE '/home/saradhi/tableR-data.csv'
INTO TABLE R
 FILEDS TERMINATED BY '#'
 LINES TERMINATED BY '\n'
IGNORE 10 LINES;
```

Specifying order of row insertion?

Can we instruct DBMS?

- Row storage is internal to the DBMS
- This burden of storage is decoupled from users
- A table with primary key constraint, records are stored in the sorted order of the primary key
- Detailed discussion of storage will be covered when discussing DBMS internals

Updating one row

Updating one row

Assume c1 is a primary key column

R				
c1	c2	c3	c4	c5
1	2	3	4	5
10	20	30	40	50
100	⊥	300	400	⊥
150	250	350	450	⊥

This update statement will be allowed

```
UPDATE R SET c1 = 5 where c1 = 1;
```

Updating one row

Updating one row

Assume c1 is a primary key column

		R		
c1	c2	c3	c4	c5
1	2	3	4	5
10	20	30	40	50
100	⊥	300	400	⊥
150	250	350	450	⊥

This update statement will be allowed

UPDATE R **SET** c1 = 5 **where** c1 = 1;

This update statement will be rejected

UPDATE R **SET** c1 = 10 **where** c1 = 1;

Updating multiple rows

Updating multiple rows

R				
c1	c2	c3	c4	c5
1	2	3	4	5
10	20	30	40	50
100	⊥	300	400	⊥
150	250	350	450	⊥

```
UPDATE R SET c1 = 101 where c1 >= 100;
```

Deleting one row

Deleting one row

Assume c1 is a primary key column

R				
c1	c2	c3	c4	c5
1	2	3	4	5
10	20	30	40	50
100	⊥	300	400	⊥
150	250	350	450	⊥

```
DELETE FROM R WHERE c1 = 1;
```

Deleting multiple rows

Deleting multiple rows

R				
c1	c2	c3	c4	c5
1	2	3	4	5
10	20	30	40	50
100	⊥	300	400	⊥
150	250	350	450	⊥

```
DELETE FROM R WHERE c1 >= 100;
```

Reading Data From Tables

- Selecting columns
- Selecting rows
- Select rows and columns
- Table operations
  - Union of two tables
  - Intersection between two tables
  - Difference of two tables
  - Cross product of two tables
  - Joining two tables
    - Natural join
    - Inner join (theta join)
    - Left outer join
    - Right outer join
    - Full outer join
  - Group by
  - Distinct rows/columns
  - Sort rows
  - Extended selection

Selecting columns of a table

SELECT statement

- Is the most frequently used statement
- Is at the heart of the querying database tables
- Important as SELECT statement combines more than 9 relational algebraic operators
- We build from basics to advanced query structures

SELECT statement structure

SELECT list the **column names**
FROM list the **table names**
WHERE specify the condition
GROUP BY list the **column names**
HAVING specify the condition
ORDER BY specify the **column names**;

SELECT statement structure

SELECT

- SELECT statement result in a table
- The result table will not be explictly stored in the database
- Compose several SELECT statements to perform a required query
- Needed privileges to perform the select statemet!

Operations on tables using SELECT

A quick list

- Select - columns
- Select - rows
- Select - rows & columns
- Select - remove duplicates
- Select - perform column sum, minimum, maximum, average, count
- Select - sort
- Select - group by specified column
- Select - create new columns by using expressoins/functions

SELECT - columns

Select specified list of columns

Select rating from 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

=

New Table
7
1
8
8
10
7
10
9
3
3

Select specified list of columns

```sql
SELECT   rating
FROM     Sailors;
```

SELECT - columns

Select specified list of columns

Select sid and rating

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

=

New Table	
sid	rating
22	7
29	1
31	8
32	8
58	10
64	7
71	10
74	9
85	3
95	3

Select specified list of columns

```
SELECT   sid, rating
FROM     Sailors;
```

SELECT - columns: order of selection

order of list of columns

Order of columns need not be identical to the table stored in the database. Select rating, sid

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

=

New Table

rating	sid
7	22
1	29
8	31
8	32
10	58
7	64
10	71
9	74
3	85
3	95

Select specified list of columns

```
SELECT   rating, sid
FROM     Sailors;
```

SELECT - columns: order of selection

list all columns

Order of columns need not be identical to the table stored in the database. Select rating, sid

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

=

New Table

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

Select specified list of columns

```
SELECT  sid, sname, rating, age
FROM    Sailors;
```

SELECT - columns: Wild character

list all columns

Order of columns need not be identical to the table stored in the database. Select rating, sid

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

=

New Table

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

Select specified list of columns

```
SELECT  * -- Specify regular expression; will construct list of column names
FROM    Sailors;
```

SELECT - rows: one specific row

select all rows that meet specific condition

Select one specific row; Example: sid = 58

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

=

New Table

sid	sname	rating	age
58	Rusty	10	35.0

Select specified list of columns

```
SELECT   sid, sname, rating, age
FROM     Sailors
WHERE    sid = 58;
```

SELECT - rows: one specific row

select all rows that meet specific condition

Select one specific row; Example: sid = 58

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

=

New Table			
sid	sname	rating	age
58	Rusty	10	35.0

Select specified list of columns

```
SELECT   * -- wild character; list all columns of row containing sid=58
FROM     Sailors
WHERE    sid = 58;
```

SELECT - rows: several rows

select all rows that meet specific condition

Select one specific row; Example: sname = Horatio

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

=

New Table			
sid	sname	rating	age
64	Horatio	7	35.0
74	Horatio	9	35.0

Select specified list of columns

```
SELECT  sid, sname, rating, age
FROM    Sailors
WHERE   sname = 'Horatio';
␣␣␣␣␣␣␣␣␣␣ ␣␣␣␣␣␣␣␣␣␣ ␣␣␣␣␣␣␣␣␣␣ ␣␣␣␣␣␣␣␣␣␣ ␣␣␣␣␣␣␣␣␣␣
```

SELECT - rows: several rows

select all rows that meet specific condition

Selecting rows with complex Example: sailors whose rating more than 6 and name should not be Horatio

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

=

New Table

sid	sname	rating	age
22	Dustin	7	45.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
71	Zorba	10	16.0

Select specified list of columns

```
SELECT   sid, sname, rating, age
FROM     Sailors
WHERE    (rating > 6 AND sname <> 'Horatio');
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
```

SELECT - rows & columns

select specified rows and columns of a given condition

Selecting sname and rating columns of saiolrs whose age is greater than or equal to 30 and name should not be Horatio

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

=

New Table	
sname	rating
Dustin	7
Brutus	1
Lubber	8
Rusty	10
Bob	3

Select specified list of columns

```
SELECT  sname, rating
FROM    Sailors
WHERE   (age >= 30 AND sname <> ‘Horatio’);
```

SELECT - remove duplicates

Definition

$r_i[a_k] = r_j[a_k] \;\; \forall\, i \neq j; \;\; \forall\, k = 1, 2, \cdots,$ number of columns

Removing duplicates

a1	a2	a3
1	2	3
1	2	3
1	2	3
1	2	4

Rows 1, 2 & 3 are duplicate; fourth row is not a duplicate

SELECT - remove duplicates

Remove Duplicates - SQL

```
SELECT DISTINCT a1, a2, a3
FROM    tableA;
```

SELECT - remove duplicates

Remove Duplicates - SQL

```
SELECT DISTINCT a1, a2, a3
FROM     tableA;
```

Remove Duplicates - SQL

a1	a2	a3
1	2	3
1	2	4

SELECT - perform column SUM - 01

Aggregate operations - SQL

```
SELECT SUM( a1 ) , SUM( a2 ) , SUM( a3 )
FROM      tableA ;
```

SELECT - perform column SUM - 01

Aggregate operations - SQL

```
SELECT SUM( a1 ) ,  SUM( a2 ) ,  SUM( a3 )
FROM      tableA ;
```

Aggregation operation - SUM

SUM(a1)	SUM(a2)	SUM(a3)
4	8	13

SELECT - perform column SUM - 01

Aggregate operations - SQL

```
SELECT SUM( a1 ) , SUM( a2 ) , SUM( a3 )
FROM      tableA ;
```

Aggregation operation - SUM

SUM(a1)	SUM(a2)	SUM(a3)
4	8	13

new result table; columns SUM(a1), ... created! data type same as column a1

SELECT - perform column SUM - 02

a1	a2	a3
1	2	3
1	2	3
1	2	3
1	2	4
$\perp$	$\perp$	$\perp$

SELECT - perform column SUM - 02

a1	a2	a3
1	2	3
1	2	3
1	2	3
1	2	4
⊥	⊥	⊥

```
SELECT SUM( a1 ) , SUM( a2 ) , SUM( a3 )
FROM      tableA ;
```

SELECT - perform column SUM - 02

a1	a2	a3
1	2	3
1	2	3
1	2	3
1	2	4
⊥	⊥	⊥

```
SELECT SUM( a1 ) , SUM( a2 ) , SUM( a3 )
FROM      tableA ;
```

SUM(a1)	SUM(a2)	SUM(a3)
4	8	13

new result table; columns SUM(a1), ... created! data type same as column a1

SELECT - perform column MIN, MAX, AVG

a1	a2	a3
1	2	3
1	2	3
1	2	3
4	5	6
⊥	⊥	⊥

SELECT - perform column MIN, MAX, AVG

a1	a2	a3
1	2	3
1	2	3
1	2	3
4	5	6
⊥	⊥	⊥

```
SELECT MIN(a1), MAX(a2), AVG(a3)
FROM     tableA;
```

SELECT - perform column MIN, MAX, AVG

a1	a2	a3
1	2	3
1	2	3
1	2	3
4	5	6
$\perp$	$\perp$	$\perp$

```
SELECT MIN( a1 ) , MAX( a2 ) , AVG( a3 )
FROM     tableA ;
```

MIN(a1)	MAX(a2)	AVG(a3)
1	5	3.75

SELECT - perform column - row COUNT - 01

a1	a2	a3
1	2	3
1	2	3
1	2	3
4	5	6

SELECT - perform column - row COUNT - 01

a1	a2	a3
1	2	3
1	2	3
1	2	3
4	5	6

```
SELECT  count(a1)
FROM      tableA;
```

SELECT - perform column - row COUNT - 01

a1	a2	a3
1	2	3
1	2	3
1	2	3
4	5	6

```
SELECT  count(a1)
FROM      tableA;
```

COUNT(a1)
4

SELECT - perform column - row COUNT - 02

a1	a2	a3
1	2	3
1	2	3
1	2	3
4	5	6
$\perp$	$\perp$	$\perp$

SELECT - perform column - row COUNT - 02

a1	a2	a3
1	2	3
1	2	3
1	2	3
4	5	6
⊥	⊥	⊥

```
SELECT count(a1)
FROM   tableA;
```

SELECT - perform column - row COUNT - 02

a1	a2	a3
1	2	3
1	2	3
1	2	3
4	5	6
⊥	⊥	⊥

```
SELECT  count(a1)
FROM        tableA;
```

COUNT(a1)
4

SELECT - perform row COUNT

a1	a2	a3
1	2	3
1	2	3
1	2	3
4	5	6
⊥	⊥	⊥

SELECT - perform row COUNT

a1	a2	a3
1	2	3
1	2	3
1	2	3
4	5	6
⊥	⊥	⊥

```
SELECT * -- wild character counts rows
FROM     tableA;
```

SELECT - perform row COUNT

a1	a2	a3
1	2	3
1	2	3
1	2	3
4	5	6
⊥	⊥	⊥

```
SELECT * -- wild character counts rows
FROM      tableA;
```

COUNT(a1)
4

SELECT - sort - 01

sort specified columns in ascending order (by default)

Example Relation

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

SELECT - sort - 01

sort specified columns in ascending order (by default)

Example Relation

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

Sorting

```
SELECT * FROM Sailors ORDER BY rating
```

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

SELECT - sort - 02

sort specified columns in descending order

Example Relation

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

SELECT - sort - 02

sort specified columns in descending order

Example Relation

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

Sorting

```
SELECT * FROM Sailors ORDER BY rating DESC
```

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

SELECT - sort - 03

sort multiple columns

Example Relation

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

SELECT - sort - 03

sort multiple columns

Example Relation

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

Sorting

SELECT * FROM Sailors ORDER BY rating, age

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

Grouping - 01

Grouping on Department attribtue

	Department	
	EEE	
	CSE	
	EEE	
	CSE	
	…	
	…	
	CSE	
	…	

Grouping - 01

On grouping on
Department attribtue

	Department	
	CSE	
	CSE	
	CSE	
	EEE	
	EEE	
	...	
	...	
	...	

Grouping - 01

On grouping on Department attribtue

	Department	
	CSE	
	~~CSE~~	
	~~CSE~~	
	EEE	
	~~EEE~~	
	...	
	...	
	...	

Grouping - 02

- Partitions rows of table into groups on the given column (cid)
- Each group (cid) consists of all rows having one particular assignment of values
- If there are no grouping attributes, entire relation is one group
- For each group (cid) produce one row consisting of
  - The grouping attributes' values for that group and
  - The aggregations over all row of that group for the aggregated colum on column list (cid)

Grouping Example

Example Relation

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

Grouping Example

Example Relation

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

Grouping

SELECT * FROM Sailors GROUP BY rating

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

Grouping Example

Example Relation

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

Grouping

SELECT * FROM Sailors GROUP BY rating

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

Grouping Example

Example Relation

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

Grouping

SELECT * FROM Sailors GROUP BY rating

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

Grouping Example

Example Relation

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

Grouping

SELECT * FROM Sailors GROUP BY rating

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

AS - re-nameing columns - 01

Create new column X using B, C

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

A	X
0	3
0	3
3	9

SQL statement

```
SELECT A, (B + C) AS X
FROM    table1;
```

AS - re-nameing columns - 02

Create new columns X, Y using B, A and C, B

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

X	Y
1	1
1	1
1	1

SQL statement

```
SELECT (B – A) AS X, (C – B) AS Y
FROM    table1;
```

AS - re-nameing table - 03

Create new table using SELECT

table1

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

table2

A	X
0	3
0	3
3	9

SQL statement

```
CREATE TABLE table2 AS (SELECT A, (B + C) AS X FROM table1);
```

Table Operators

Binary Operator - Union

a1	a2	a3
1	2	3
4	5	6

$\cup$

b1	b2	b3
1	2	3
7	8	9

$=$

a1	a2	a3
1	2	3
4	5	6
7	8	9

Union Compatibility

- Two tables should have identical number of columns
- Every column must have identical data type

Union - SQL Statement

a1	a2	a3
1	2	3
4	5	6

∪

b1	b2	b3
1	2	3
7	8	9

=

a1	a2	a3
1	2	3
4	5	6
7	8	9

```
(SELECT a1 , a2 , a3 FROM TableA )
          UNION
(SELECT b1 , b2 , b3 FROM TableB ) ;
```

Union Computation - 01

Binary Operator - Union

$A \cup B = \{ e \mid e \in A \textbf{ OR } e \in B \}$

Binary Operator - Union

a1	a2	a3
1	2	3
4	5	6

$\cup$

b1	b2	b3
1	2	3
7	8	9

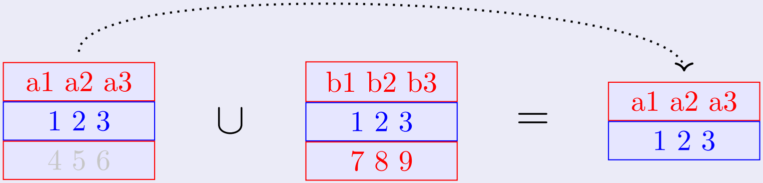
$=$

Union Computation - 02

Binary Operator - Union

$A \cup B = \{ e \mid e \in A \text{ OR } e \in B\}$

Binary Operator - Union



a1	a2	a3
1	2	3
4	5	6

$\cup$

b1	b2	b3
1	2	3
7	8	9

$=$

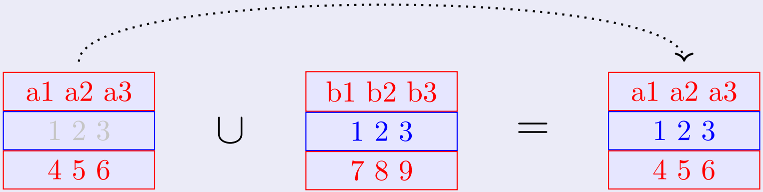
a1	a2	a3
1	2	3

Union Computation - 03

Binary Operator - Union

$A \cup B = \{\, e \mid e \in A \text{ OR } e \in B \}$

Binary Operator - Union



a1	a2	a3
1	2	3
4	5	6

$\cup$

b1	b2	b3
1	2	3
7	8	9

$=$

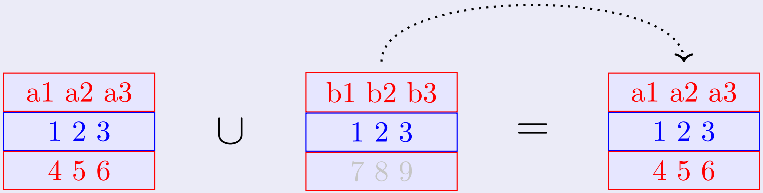
a1	a2	a3
1	2	3
4	5	6

Union Computation - 04

Binary Operator - Union

$A \cup B = \{ e \mid e \in A \textbf{ OR } e \in B \}$

Binary Operator - Union



a1 a2 a3
1 2 3
4 5 6

$\cup$

b1 b2 b3
1 2 3
7 8 9

$=$

a1 a2 a3
1 2 3
4 5 6

Binary Operator - Union

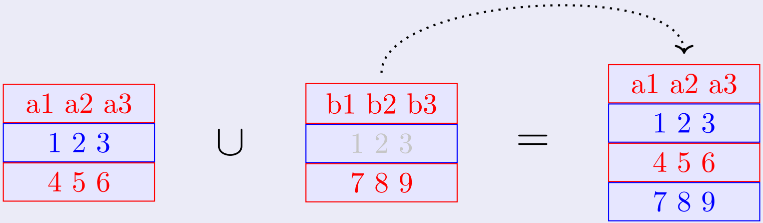
Include in the result table when explictly stated to retain duplicates!

Union Computation - 05

Binary Operator - Union

$A \cup B = \{ e \mid e \in A \textbf{ OR } e \in B \}$

Binary Operator - Union



a1	a2	a3
1	2	3
4	5	6

$\cup$

b1	b2	b3
1	2	3
7	8	9

$=$

a1	a2	a3
1	2	3
4	5	6
7	8	9

Table Operators

Binary Operator - Union

b1	b2	b3
1	2	3
7	8	9

$\cup$

a1	a2	a3
1	2	3
4	5	6

$=$

b1	b2	b3
1	2	3
4	5	6
7	8	9

Union Compatibility

- Two tables should have identical number of columns
- Every column must have identical data type

Union - SQL Statement

b1	b2	b3
1	2	3
7	8	9

$\cup$

a1	a2	a3
1	2	3
4	5	6

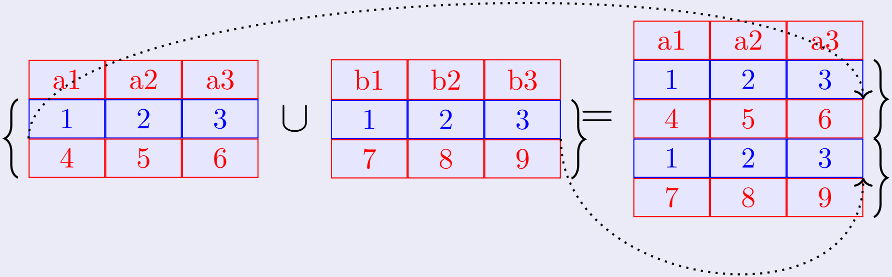
$=$

b1	b2	b3
1	2	3
4	5	6
7	8	9

```
(SELECT b1, b2, b3 FROM TableB)
        UNION
(SELECT a1, a2, a3 FROM TableA);
```

Table Operators

Binary Operator - Union Duplicates?



a1	a2	a3
1	2	3
4	5	6

∪

b1	b2	b3
1	2	3
7	8	9

=

a1	a2	a3
1	2	3
4	5	6
1	2	3
7	8	9

```
(SELECT  a1 ,  a2 ,  a3 FROM TableA)
          UNION ALL
(SELECT  b1 ,  b2 ,  b3 FROM TableB);
```

Table Operators

Binary Operator - Union Duplicates?

a1	a2	a3
1	2	3
4	5	6
7	8	9

A

∪

b1	b2	b3
1	2	3
4	5	6
7	8	9

B

=

a1	a2	a3
1	2	3
4	5	6
7	8	9
1	2	3
4	5	6
7	8	9

A

B

Table Operators

Binary Operator - Union

a1	a2	a3
1	2	3
4	5	6

$\cup$

b1	b2	b3
AA	AB	AC
BB	BC	BD

= Error!

Union Compatibility

- Two tables should have identical number of columns
- Incompatible data types

Table Operators

Binary Operator - Union

a1	a2	a3
1	2	3
4	5	6

$\cup$

b1	b2	b3	b4
1	2	3	4
10	20	30	40

= Error!

Union Compatibility

- Two tables have different number of columns
- Every column must have identical data type

Table Operators

Binary Operator - Intersect

$A \cap B = \{ e \mid e \in A \textbf{ AND } e \in B \}$

Binary Operator - Intersection

a1	a2	a3
1	2	3
4	5	6

$\cap$

b1	b2	b3
1	2	3
7	8	9

$=$

a1	a2	a3
1	2	3

Union Compatibility

- Two tables should have identical number of columns
- Every column must have identical data type

Table Operators

Binary Operator - Intersect

$$A \cap B = \{\, e \mid e \in A \ \textbf{AND}\ e \in B \}$$

Binary Operator - Intersection

b1	b2	b3
1	2	3
7	8	9

$\cap$

a1	a2	a3
1	2	3
4	5	6

$=$

b1	b2	b3
1	2	3

Union Compatibility

- Two tables should have identical number of columns
- Every column must have identical data type

Table Operators

Binary Operator - Intersect

$A \cap B = \{ e \mid e \in A \textbf{ AND } e \in B\}$

Binary Operator - Intersect

a1	a2	a3
1	2	3
4	5	6

$\cap$

b1	b2	b3
1	2	3
7	8	9

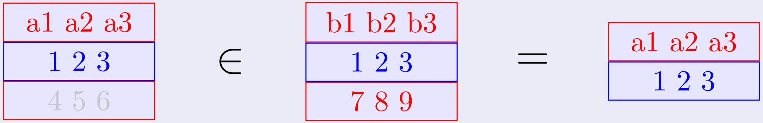
$=$

Table Operators

Binary Operator - Intersect

$$A \cap B = \{ e \mid e \in A \textbf{ AND } e \in B \}$$

Binary Operator - Intersect



a1	a2	a3
1	2	3
4	5	6

$\in$

b1	b2	b3
1	2	3
7	8	9

$=$

a1	a2	a3
1	2	3

Binary Operator - Intersect testing

- (a1 == b1) AND (a2 == b2) AND (a3 = b3)?
- That is: (1 == 1) AND (2 == 2) AND (3 == 3)? Yes;
- Include first row of tableA in result table

Table Operators

Binary Operator - Intersect

$A \cap B = \{ e \mid e \in A \textbf{ AND } e \in B \}$

Binary Operator - Intersect

a1	a2	a3
1	2	3
4	5	6

$\in$

b1	b2	b3
1	2	3
7	8	9

$=$

a1	a2	a3
1	2	3

Binary Operator - Intersect testing

- (a1 == b1) AND (a2 == b2) AND (a3 = b3)?
- That is: (4 == 1) AND (5 == 2) AND (6 == 3)? No;

Table Operators

Binary Operator - Intersect

$$A \cap B = \{ e \mid e \in A \textbf{ AND } e \in B\}$$

Binary Operator - Intersect

a1	a2	a3
1	2	3
4	5	6

$\in$

b1	b2	b3
1	2	3
7	8	9

$=$

a1	a2	a3
1	2	3

Binary Operator - Intersect testing

- (a1 == b1) AND (a2 == b2) AND (a3 = b3)?
- That is: (4 == 1) AND (5 == 2) AND (6 == 3)? No;
- Test next row of tableB
- That is: (4 == 7) AND (5 == 8) AND (6 == 9)? No;
- There are no row in tableB; Do not include (4, 5, 6) in result table

Table Operators

Binary Operator - Intersection Duplicates?

a1	a2	a3
1	2	3
4	5	6

$\cap$

b1	b2	b3
1	2	3
7	8	9

$=$

a1	a2	a3
1	2	3

Table Operators

Binary Operator - Intersect

$A \cap B = \{ e \mid e \in A \textbf{ AND } e \in B \}$

Binary Operator - Intersection Duplicates?

a1	a2	a3
1	2	3
1	2	3
1	2	3
4	5	6

$\cap$

b1	b2	b3
1	2	3
4	5	6
7	8	9

$=$

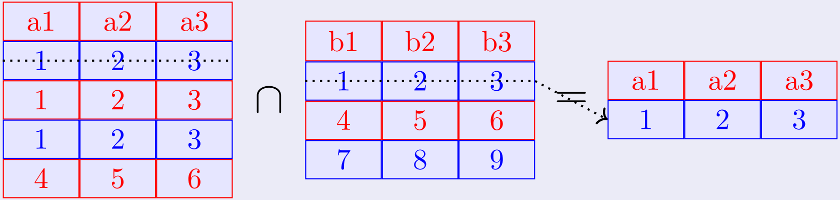
a1	a2	a3
1	2	3

Table Operators

Binary Operator - Intersect

$A \cap B = \{ e \mid e \in A \textbf{ AND } e \in B \}$

Binary Operator - Intersection Duplicates?



a1	a2	a3
1	2	3
1	2	3
1	2	3
4	5	6

$\cap$

b1	b2	b3
1	2	3
4	5	6
7	8	9

$=$

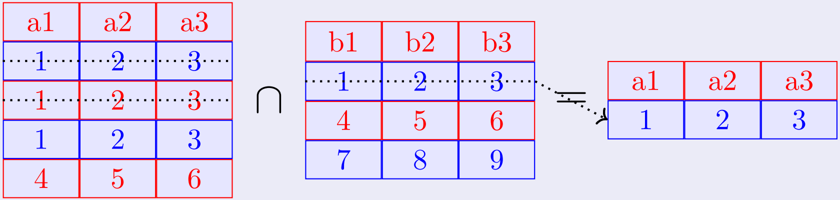
a1	a2	a3
1	2	3

Table Operators

Binary Operator - Intersect

A ∩ B = { e | e ∈ A **AND** e ∈ B}

Binary Operator - Intersection Duplicates?



a1	a2	a3
1	2	3
1	2	3
1	2	3
4	5	6

∩

b1	b2	b3
1	2	3
4	5	6
7	8	9

=

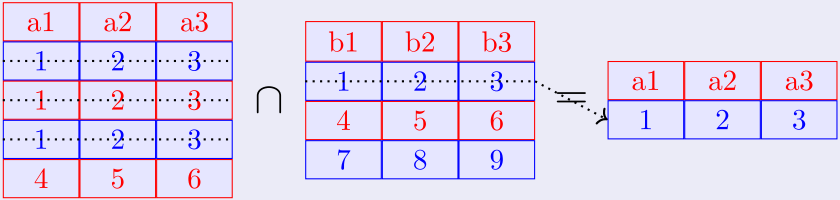
a1	a2	a3
1	2	3

Table Operators

Binary Operator - Intersect

$A \cap B = \{ e \mid e \in A$ **AND** $e \in B \}$

Binary Operator - Intersection Duplicates?



a1	a2	a3
1	2	3
1	2	3
1	2	3
4	5	6

$\cap$

b1	b2	b3
1	2	3
4	5	6
7	8	9

$=$

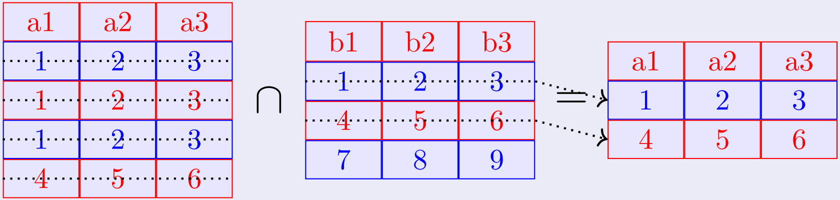
a1	a2	a3
1	2	3

Table Operators

Binary Operator - Intersect

$A \cap B = \{ e \mid e \in A \ \textbf{AND} \ e \in B \}$

Binary Operator - Intersection Duplicates?



a1	a2	a3
1	2	3
1	2	3
1	2	3
4	5	6

$\cap$

b1	b2	b3
1	2	3
4	5	6
7	8	9

$=$

a1	a2	a3
1	2	3
4	5	6

Table Operators

Binary Operator - Intersect

$A \cap B = \{\, e \mid e \in A \textbf{ AND } e \in B \}$

Binary Operator - Intersection

a1	a2	a3
1	2	3
4	5	6

$\cap$

b1	b2	b3
AA	AB	AC
BB	BC	BD

= Error!

Union Compatibility

- Two tables should have identical number of columns
- Incompatible data types

Table Operators

Binary Operator - Intersect

$A \cap B = \{\ e \mid e \in A$ **AND** $e \in B\}$

Binary Operator - Intersection

a1	a2	a3
1	2	3
4	5	6

$\cap$

b1	b2	b3	b4
1	2	3	4
10	20	30	40

= Error!

Union Compatibility

- Two tables have different number of columns
- Every column must have identical data type

Table Operators

Binary Operator - Difference

$$A - B = \{ e \mid e \in A \textbf{ AND } e \notin B \}$$

Binary Operator - Difference

a1	a2	a3
1	2	3
4	5	6

−

b1	b2	b3
1	2	3
7	8	9

=

a1	a2	a3
4	5	6

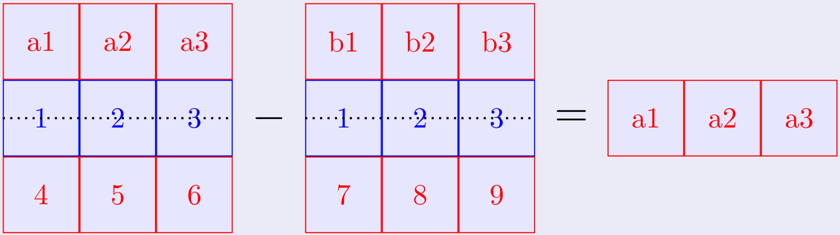
```
SELECT a1, a2, a3
FROM TableA
WHERE (a1, a2, a3)
IN
(SELECT b1, b2, b3 FROM TableB);
```

Table Operators

Binary Operator - Difference

$A - B = \{ e \mid e \in A \textbf{ AND } e \notin B \}$

Binary Operator - Difference



a1	a2	a3
1	2	3
4	5	6

−

b1	b2	b3
1	2	3
7	8	9

=

a1	a2	a3

Union Compatibility

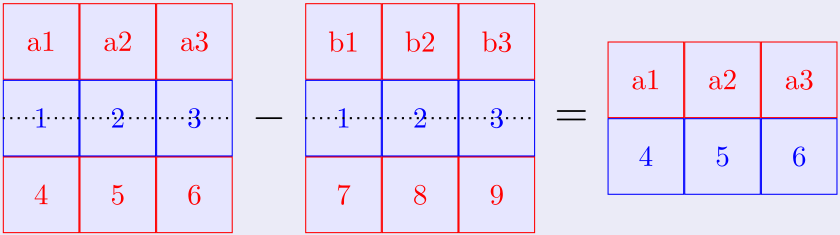
- Two tables should have identical number of columns
- Every column must have identical data type

Table Operators

Binary Operator - Difference

$$A - B = \{\, e \mid e \in A \ \textbf{AND}\ e \notin B \}$$

Binary Operator - Difference



a1	a2	a3
~~1~~	~~2~~	~~3~~
4	5	6

$-$

b1	b2	b3
~~1~~	~~2~~	~~3~~
7	8	9

$=$

a1	a2	a3
4	5	6

Union Compatibility

- Two tables should have identical number of columns
- Every column must have identical data type

Table Operators

Binary Operator - Difference

b1	b2	b3
~~1~~	~~2~~	~~3~~
7	8	9

−

a1	a2	a3
~~1~~	~~2~~	~~3~~
4	5	6

=

b1	b2	b3
7	8	9

Union Compatibility

- Two tables should have identical number of columns
- Every column must have identical data type

Table Operators

Binary Operator - Difference Duplicates?

a1	a2	a3
~~1~~	~~2~~	~~3~~
4	5	6

$-$

b1	b2	b3
~~1~~	~~2~~	~~3~~
7	8	9

$=$

a1	a2	a3
4	5	6

Table Operators

Binary Operator - Difference Duplicates?

a1	a2	a3
~~1~~	~~2~~	~~3~~
1	2	3
1	2	3
4	5	6

−

b1	b2	b3
~~1~~	~~2~~	~~3~~
1	2	3
4	5	6
7	8	9

=

a1	a2	a3
1	2	3
4	5	6

Table Operators

Binary Operator - Difference Duplicates?

a1	a2	a3
~~1~~	~~2~~	~~3~~
~~1~~	~~2~~	~~3~~
1	2	3
4	5	6

−

b1	b2	b3
~~1~~	~~2~~	~~3~~
~~1~~	~~2~~	~~3~~
4	5	6
7	8	9

=

a1	a2	a3
1	2	3
4	5	6

Table Operators

Binary Operator - Difference

a1	a2	a3
1	2	3
4	5	6

−

b1	b2	b3
AA	AB	AC
BB	BC	BD

= Error!

Union Compatibility

- Two tables should have identical number of columns
- Incompatible data types

Table Operators

Binary Operator - Difference

a1	a2	a3
1	2	3
4	5	6

−

b1	b2	b3	b4
1	2	3	4
10	20	30	40

= Error!

Union Compatibility

- Two tables have different number of columns
- Every column must have identical data type

Table operators summary of definitions

Union of involving duplicates

- Let a row $t \in R$ appears n times
- Let $t \in S$ appears m times
- $t \in (R \cup S)$ appears $(n + m)$ times

Intersection involving duplicates

- Let a row $t \in R$ appears n times
- Let $t \in S$ appears m times
- $t \in (R \cap S)$ appears $\min(n, m)$ times

Difference involving duplicates

- Let a row $t \in R$ appears n times
- Let $t \in S$ appears m times
- $t \in (R - S)$ appears $\max(0, (n - m))$ times

SQL Statements Summary - Union

- Union (Distinct rows)

```
(SELECT a1, a2, a3 FROM TableA)
        UNION
(SELECT b1, b2, b3 FROM TableB);
```

- Union (Retain Duplicates)

```
(SELECT a1, a2, a3 FROM TableA)
        UNION ALL
(SELECT b1, b2, b3 FROM TableB);
```

SQL Statements Summary - Intersection

- Intersection (Distinct rows)

```
SELECT  DISTINCT  a1 ,  a2 ,  a3
FROM    TableA
WHERE   (a1 ,  a2 ,  a3)
IN
(SELECT  b1 ,  b2 ,  b3  FROM  TableB);
```

- Intersection (Retain Duplicates)

```
SELECT  a1 ,  a2 ,  a3
FROM    TableA
WHERE   (a1 ,  a2 ,  a3)
IN
(SELECT  b1 ,  b2 ,  b3  FROM  TableB);
```

SQL Statements Summary - Difference

- Difference (Distinct)

```
SELECT DISTINCT a1 , a2 , a3
FROM   TableA
WHERE  (a1 , a2 , a3)
NOT IN
 (SELECT b1 , b2 , b3 FROM TableB);
```

- Difference (Retain Duplicates)

```
SELECT a1 , a2 , a3
FROM   TableA
WHERE  (a1 , a2 , a3)
NOT IN
 (SELECT b1 , b2 , b3 FROM TableB);
```