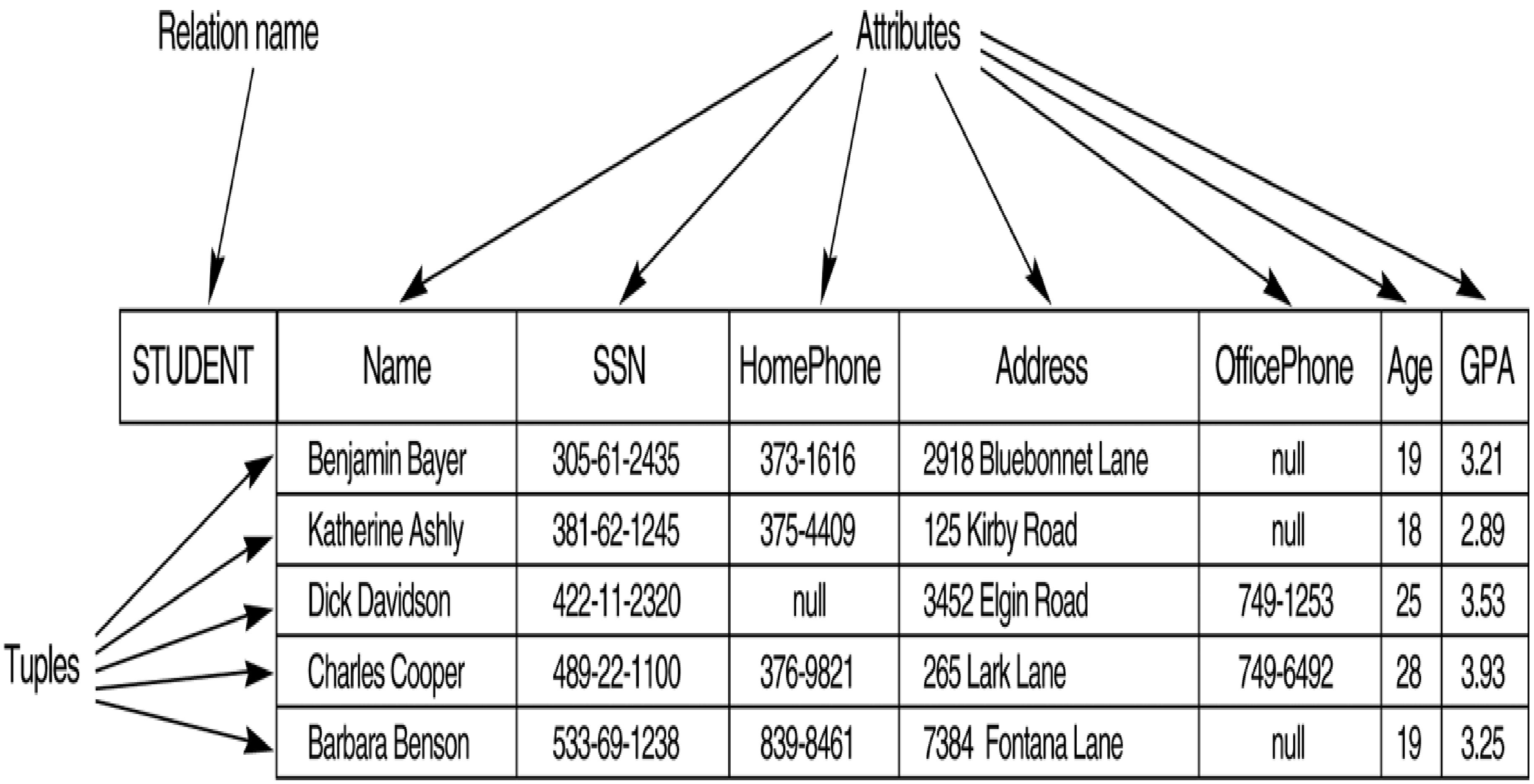
Null values in SQL



STUDENT	Name	SSN	HomePhone	Address	OfficePhone	Age	GPA
	Benjamin Bayer	305-61-2435	373-1616	2918 Bluebonnet Lane	null	19	3.21
	Katherine Ashly	381-62-1245	375-4409	125 Kirby Road	null	18	2.89
	Dick Davidson	422-11-2320	null	3452 Elgin Road	749-1253	25	3.53
	Charles Cooper	489-22-1100	376-9821	265 Lark Lane	749-6492	28	3.93
	Barbara Benson	533-69-1238	839-8461	7384 Fontana Lane	null	19	3.25

Null values in SQL

• testing if an attribute A is null: A **is null,** A **is not null**
Ex: find all employees with unknown phone number

```
select name from employee
where phone is null
```

• arithmetic operations involving any null return null

if Salary is null, then Salary + 1 evaluates to null

• comparisons involving any null return unknown
new truth value

if Salary is null, then Salary = 0 evaluates to unknown

Null values in SQL

- Boolean operations must now handle 3 truth values:
  true, false, unknown
- Boolean expressions involving unknown are evaluated using the following truth tables:

AND			
	true	unknown	unknown
	false	unknown	false
	unknown	unknown	unknown

NOT		
	unknown	unknown

OR			
	true	unknown	true
	false	unknown	unknown
	unknown	unknown	unknown

- WHERE clause conditions evaluating to unknown are treated as false

Examples

Movie

title	director	actor
Tango	Berto	Brando
Psycho	Hitch	Perkins
Bambi	null	null

```
Select title
Where dir = 'Hitch'
```

title
Psycho

```
Select title
Where dir <> 'Hitch'
```

title
Tango
Bambi

A: yes
B: no

Examples

Movie	title	director	actor
	Tango	Berto	Brando
	Psycho	Hitch	Perkins
	Bambi	null	null

Select title
Where dir = ‘Hitch’

title
Psycho

Select title
Where dir <> ‘Hitch’

title
Tango
Bambi

A: yes
B: no

title
Tango

B

Examples

Movie	title	director	actor
	Tango	Berto	Brando
	Psycho	Hitch	Perkins
	Bambi	null	null

Select title
Where dir = 'null'

title
Bambi

A: yes
B: no

Examples

Movie

title	director	actor
Tango	Berto	Brando
Psycho	Hitch	Perkins
Bambi	null	null

Select title
Where dir = 'null'

title
Bambi

A: yes
B: no

Select title
Where dir is null

title
Bambi

Null values in SQL

Anomalies of null semantics

if Salary is null, then:

-- Salary* 0 evaluates to null
-- Salary > 0 evaluates to unknown even if the domain is restricted to positive integers in the schema definition
-- Consider the queries

```
select name from employee
where Salary <= 100 OR Salary > 100
```

and

```
select name from employee
```

Are these equivalent? A: yes B: no

Null values in SQL

Anomalies of null semantics

if Salary is null, then:

-- Salary* 0 evaluates to null
-- Salary > 0 evaluates to unknown even if the domain is restricted to positive integers in the schema definition
-- Consider the queries

```
select name from employee
where Salary <= 100 OR Salary > 100
```

and

```
select name from employee
```

These are not equivalent if some salaries are null

Null Values and Aggregates

- Total all loan amounts

  select sum (*amount*)
  from *loan*

  - Above statement ignores null amounts
  - Result is *null* if there is no non-null amount
- All aggregate operations except **count(*)** ignore tuples with null values on the aggregated attributes.

Suppose R has a single attribute A. Are these equivalent?

```
select  count(*)  from R

select  count(A) from R
```

A: yes   B: no

Null Values and Group-By

- Null group-by attributes are treated like any other value

R	A	B
	2	3
	2	5
	Null	0
	Null	1
	Null	2

```
SELECT A, COUNT(B) AS C
GROUP BY A
```

A	C
2	2
Null	3

SQL: Natural Join

Combines tuples from two tables by matching on common attributes

movie	title	director	actor
	Tango	Berto	Brando
	Sky	Berto	Winger
	Psycho	Hitchcock	Perkins

schedule	theater	title
	Hillcrest	Tango
	Paloma	Tango
	Paloma	Bambi
	Ken	Psycho

movie **natural join** schedule	title	director	actor	theater
	Tango	Berto	Brando	Hillcrest
	Tango	Berto	Brando	Paloma
	Psycho	Hitchcock	Perkins	Ken

SQL: Natural Join

Find the directors of movies showing in Hillcrest:

```
select  director
from movie natural join schedule
where theater = ‘Hillcrest’
```

Many variations of joins are available in SQL

SQL: creating nulls with Outer Joins

Idea: to avoid losing tuples in natural joins, pad with null values

P <outer join> Q

- natural left outer join: keep all tuples from left relation (P)
- natural right outer join: keep all tuples from right relation (Q)
- natural full outer join: keep all tuples from both relations

SQL: creating nulls with Outer Joins

Combines tuples from two tables by matching on common attributes

movie	title	director	actor
	Tango	Berto	Brando
	Sky	Berto	Winger
	Psycho	Hitchcock	Hopkins

schedule	theater	title
	Hillcrest	Tango
	Paloma	Tango
	Paloma	Bambi
	Ken	Psycho

movie **natural left outer join** schedule	title	director	actor	theater
	Tango	Berto	Brando	Hillcrest
	Tango	Berto	Brando	Paloma
	Psycho	Hitchcock	Hopkins	Ken
	Sky	Berto	Winger	*null*

SQL: creating nulls with Outer Joins

Combines tuples from two tables by matching on common attributes

movie

title	director	actor
Tango	Berto	Brando
Sky	Berto	Winger
Psycho	Hitchcock	Hopkins

schedule

theater	title
Hillcrest	Tango
Paloma	Tango
Paloma	Bambi
Ken	Psycho

movie **natural right outer join** schedule

title	director	actor	theater
Tango	Berto	Brando	Hillcrest
Tango	Berto	Brando	Paloma
Psycho	Hitchcock	Hopkins	Ken
Bambi	*null*	*null*	Paloma

SQL: creating nulls with Outer Joins

Combines tuples from two tables by matching on common attributes

movie

title	director	actor
Tango	Berto	Brando
Sky	Berto	Winger
Psycho	Hitchcock	Hopkins

schedule

theater	title
Hillcrest	Tango
Paloma	Tango
Paloma	Bambi
Ken	Psycho

movie **natural full outer join** schedule

title	director	actor	theater
Tango	Berto	Brando	Hillcrest
Tango	Berto	Brando	Paloma
Psycho	Hitchcock	Hopkins	Ken
Bambi	*null*	*null*	Paloma
Sky	Berto	Winger	*null*

Example: Find theaters showing only movies by Berto

```
select theater from schedule
where theater not in
    (select theater
     from schedule natural left outer join
            (select title, director from movie where director = 'Berto')
     where director is null)
```

```
select title, director from movie where director = 'Berto'
```

Movie

title	director	actor
Tango	Berto	Brando
Sky	Berto	Winger
Psycho	Hitchcock	Hopkins

schedule

theater	title
Hillcrest	Tango
Paloma	Tango
Paloma	Psycho

Example: Find theaters showing only movies by Berto

```
select theater from schedule
where theater not in
    (select theater
     from schedule natural left outer join
              (select title, director from movie where director = 'Berto')
     where director is null)
```

```
schedule natural left outer join (select title, director from movie
                                   where director = 'Berto')
```

title	director
Tango	Berto
Sky	Berto

schedule theater	title
Hillcrest	Tango
Paloma	Tango
Paloma	Psycho

theater	title	director
Hillcrest	Tango	Berto
Paloma	Tango	Berto
Paloma	Psycho	null

Summary of basic SQL Queries

- A query in SQL can consist of up to six clauses, but only the first two, SELECT and FROM, are mandatory.
- The clauses are specified in the following order:

```
SELECT      <attribute list>
FROM        <table list>
[WHERE      <condition>]
[GROUP BY <grouping attribute(s)>]
[HAVING     <group condition>]
[ORDER BY <attribute list>]
```

Summary of basic SQL Queries (cont.)

- The SELECT-clause lists the attributes or functions to be retrieved
- The FROM-clause specifies all relations (or aliases) needed in the query but not those needed in nested queries
- The WHERE-clause specifies the conditions for selection of tuples from the relations specified in the FROM-clause
- GROUP BY specifies grouping attributes
- HAVING specifies a condition for selection of groups
- ORDER BY specifies an order for displaying the result of a query
- A query is evaluated by first applying the WHERE-clause, then GROUP BY and HAVING, and finally the SELECT-clause

SQL update language Insertions

- inserting tuples
  - INSERT INTO *R*
    VALUES (*v1,…,vk*);
- some values may be left NULL
- use results of queries for insertion
  - INSERT INTO *R* SELECT … FROM ... WHERE

```
INSERT INTO Movie
VALUES (“Matchpoint”, “Allen”, “Allen”);
```

```
INSERT INTO Movie(Title,Director)
VALUES (“Matchpoint”, “Allen”);
```

```
INSERT INTO BertoMovie
        SELECT * FROM Movie
        WHERE Director = “Berto”
```

SQL update language: Updates and Deletions

- ***Deletion*** basic form: delete every tuple that satisfies <*cond*>
  - DELETE FROM *R* WHERE <*cond*>
- ***Update*** basic form: update every tuple that satisfies <*cond*> in the way specified by the SET clause

  -
    ```
    UPDATE R
     SET A1=<exp1>,
           …
         Ak=<expk>
     WHERE <cond>
    ```

Delete the movies that are not currently playing

```
DELETE FROM Movie
WHERE Title NOT IN SELECT Title
                    FROM Schedule
```

Change all "Berto" entries to "Bertolucci"

```
UPDATE Movie
SET Director="Bertolucci"
WHERE Director="Berto"
```

Increase all salaries in the Toys dept by 10%

```
UPDATE Employee
SET Salary = 1.1 * Salary
WHERE Dept = "Toys"
```

The "rich get richer" exercise:
Increase by 10% the salary of the employee with the highest salary

Example: delete all theaters showing more than one title

```
delete from schedule s
where exists (select * from schedule
                        where theater = s.theater and title <> s.title)
```

Schedule	theater	title
	Hillcrest	Amour
	Hillcrest	0 dark 30
	Paloma	Django

Assume this semantics:
for each s in schedule
if where clause is satisfied
then delete s

$\downarrow$ Result after delete?

A: yes B: no

Schedule	theater	title
	Paloma	Django

Example: delete all theaters showing more than one title

```
delete from schedule s
where exists (select * from schedule
              where theater = s.theater and title <> s.title)
```

Schedule	theater	title
	Hillcrest	Amour
	Hillcrest	0 dark 30
	Paloma	Django

↓ Result after delete?

Schedule	theater	title
	Paloma	Django

Correct semantics:
1. Find all theaters showing more than one title
2. Delete all theaters found in 1.

A: yes B: no