UCLA

CS143
Advanced SQL

Professor Junghoo “John” Cho

What to Learn

- Aggregate functions
- Window function
- Case expression
- Order by and Fetch first
- Data modification
- NULL and three-valued logic
- Outer join
- Multiset semantic for set operators
- SQL expressive power and recursion

Q1: Average GPA of all students

Student(sid, name, addr, age, GPA)

sid	name	addr	age	GPA
301	Andy	183 Westwood	19	2.1
303	Elaine	301 Wilshire	17	3.9
401	James	183 Westwood	17	3.5
208	Esther	421 Wilshire	20	3.1

Class(dept, cnum, sec, unit, title, instructor)

dept	cnum	sec	unit	title	instructor
CS	112	01	03	Modeling	Dick Muntz
CS	143	01	04	DB Systems	John Cho
EE	143	01	03	Signal	Dick Muntz
ME	183	02	05	Mechanics	Susan Tracey

Enroll(sid, dept, cnum, sec)

sid	dept	cnum	sec
301	CS	112	01
301	CS	143	01
303	EE	143	01
303	CS	112	01
401	CS	112	01

Key Challenge of Q1

- What we learned: Information from *one* input tuple per output

sid	name	addr	age	GPA
~~301~~	~~Andy~~	~~183 Westwood~~	~~19~~	~~2.1~~
303	Elaine	301 Wilshire	17	3.9
401	James	183 Westwood	17	3.5
~~208~~	~~Esther~~	~~421 Wilshire~~	~~20~~	~~3.1~~

$\xrightarrow{\text{GPA > 3.2}}$

3.9
3.5

- What we need to do: Combine information from *multiple input tuples* into a *single* output tuple

sid	name	addr	age	GPA
301	Andy	183 Westwood	19	2.1
303	Elaine	301 Wilshire	17	3.9
401	James	183 Westwood	17	3.5
208	Esther	421 Wilshire	20	3.1

$\xrightarrow{\text{AVG(GPA)}}$

3.15

Q1: Average GPA of all students

Student(sid, name, addr, age, GPA)

sid	name	addr	age	GPA
301	Andy	183 Westwood	19	2.1
303	Elaine	301 Wilshire	17	3.9
401	James	183 Westwood	17	3.5
208	Esther	421 Wilshire	20	3.1

Class(dept, cnum, sec, unit, title, instructor)

dept	cnum	sec	unit	title	instructor
CS	112	01	03	Modeling	Dick Muntz
CS	143	01	04	DB Systems	John Cho
EE	143	01	03	Signal	Dick Muntz
ME	183	02	05	Mechanics	Susan Tracey

Enroll(sid, dept, cnum, sec)

sid	dept	cnum	sec
301	CS	112	01
301	CS	143	01
303	EE	143	01
303	CS	112	01
401	CS	112	01

Aggregate Functions

- Allows “aggregating” results from multiple tuples to produce a single output tuple
- AVG, SUM, COUNT, MIN, MAX on single attribute
  - COUNT(*): counts the number of matching tuples

Q2: Number of students taking CS classes

Student(sid, name, addr, age, GPA)

sid	name	addr	age	GPA
301	Andy	183 Westwood	19	2.1
303	Elaine	301 Wilshire	17	3.9
401	James	183 Westwood	17	3.5
208	Esther	421 Wilshire	20	3.1

Class(dept, cnum, sec, unit, title, instructor)

dept	cnum	sec	unit	title	instructor
CS	112	01	03	Modeling	Dick Muntz
CS	143	01	04	DB Systems	John Cho
EE	143	01	03	Signal	Dick Muntz
ME	183	02	05	Mechanics	Susan Tracey

Enroll(sid, dept, cnum, sec)

sid	dept	cnum	sec
301	CS	112	01
301	CS	143	01
303	EE	143	01
303	CS	112	01
401	CS	112	01

Q3: Average GPA of students who take CS classes

Student(sid, name, addr, age, GPA)

sid	name	addr	age	GPA
301	Andy	183 Westwood	19	2.1
303	Elaine	301 Wilshire	17	3.9
401	James	183 Westwood	17	3.5
208	Esther	421 Wilshire	20	3.1

Class(dept, cnum, sec, unit, title, instructor)

dept	cnum	sec	unit	title	instructor
CS	112	01	03	Modeling	Dick Muntz
CS	143	01	04	DB Systems	John Cho
EE	143	01	03	Signal	Dick Muntz
ME	183	02	05	Mechanics	Susan Tracey

Enroll(sid, dept, cnum, sec)

sid	dept	cnum	sec
301	CS	112	01
301	CS	143	01
303	EE	143	01
303	CS	112	01
401	CS	112	01

Q4: Average GPA for each age group

Student(sid, name, addr, age, GPA)

sid	name	addr	age	GPA
301	Andy	183 Westwood	19	2.1
303	Elaine	301 Wilshire	17	3.9
401	James	183 Westwood	17	3.5
208	Esther	421 Wilshire	20	3.1

Age	AVG(GPA)
17	3.7
19	2.1
20	3.1

GROUP BY and SELECT attributes

- Q: Is the following query meaningful?

```
SELECT sid, age, AVG(GPA)
FROM Student
GROUP BY age;
```

- With GROUP BY, SELECT can have only aggregate functions or attributes that have a single value in each group

Q5: Number of classes each student takes

Student(sid, name, addr, age, GPA)

sid	name	addr	age	GPA
301	Andy	183 Westwood	19	2.1
303	Elaine	301 Wilshire	17	3.9
401	James	183 Westwood	17	3.5
208	Esther	421 Wilshire	20	3.1

Class(dept, cnum, sec, unit, title, instructor)

dept	cnum	sec	unit	title	instructor
CS	112	01	03	Modeling	Dick Muntz
CS	143	01	04	DB Systems	John Cho
EE	143	01	03	Signal	Dick Muntz
ME	183	02	05	Mechanics	Susan Tracey

Enroll(sid, dept, cnum, sec)

sid	dept	cnum	sec
301	CS	112	01
301	CS	143	01
303	EE	143	01
303	CS	112	01
401	CS	112	01

Q: What about students who take no classes

Q6: Students who take two or more classes

Student(sid, name, addr, age, GPA)

sid	name	addr	age	GPA
301	Andy	183 Westwood	19	2.1
303	Elaine	301 Wilshire	17	3.9
401	James	183 Westwood	17	3.5
208	Esther	421 Wilshire	20	3.1

Class(dept, cnum, sec, unit, title, instructor)

dept	cnum	sec	unit	title	instructor
CS	112	01	03	Modeling	Dick Muntz
CS	143	01	04	DB Systems	John Cho
EE	143	01	03	Signal	Dick Muntz
ME	183	02	05	Mechanics	Susan Tracey

Enroll(sid, dept, cnum, sec)

sid	dept	cnum	sec
301	CS	112	01
301	CS	143	01
303	EE	143	01
303	CS	112	01
401	CS	112	01

HAVING Clause

- Check aggregate conditions
  - Example: Students who take two classes or more
- Appear after GROUP BY

Next Topic

- Aggregate functions
- Window function
- Case expression
- Order by and Fetch first
- Data modification
- NULL and three-valued logic
- Outer join
- Multiset semantic for set operators
- SQL expressive power and recursion

Q7: Per each student, return their name, GPA and the overall GPA average

sid	name	addr	age	GPA
301	Andy	183 Westwood	19	2.1
303	Elaine	301 Wilshire	17	3.9
401	James	183 Westwood	17	3.5
208	Esther	421 Wilshire	20	3.1

→

name	GPA	AVG(GPA)
Andy	2.1	3.15
Elaine	3.9	3.15
James	3.5	3.15
Esther	3.1	3.15

- Q: Will this work?

```
SELECT name, GPA, AVG(GPA)
FROM Student;
```

- Correct answer: Use window function!

```
SELECT name, GPA, AVG(GPA) OVER()
FROM Student;
```

Window Function

- Introduced in SQL 2003
- Syntax: FUNCTION(*attr*) OVER()
  - Use the same aggregate FUNCTION(*attr*), but append OVER()
  - Example: MAX(GPA) OVER()
- Interpretation
  - Generate *one output tuple per input tuple*, but FUNCTION(*attr*) is computed over *all* input tuples

Q8: Per each student, return their name, GPA and the average of GPA their age group

sid	name	addr	age	GPA
301	Andy	183 Westwood	19	2.1
303	Elaine	301 Wilshire	17	3.9
401	James	183 Westwood	17	3.5
208	Esther	421 Wilshire	20	3.1

name	GPA	AVG(GPA)
Andy	2.1	2.1
Elaine	3.9	3.7
James	3.5	3.7
Esther	3.1	3.1

- Apply AVG(GPA) only within their “group” or “partition”, not over the entire input tuples
- PARTITION BY
  - SELECT name, GPA, AVG(GPA) OVER(PARTITION BY age)
    FROM Student;
  - PARTITION BY for window function ≅ GROUP BY for aggregate function
- Read textbook Sec 5.5 to learn more on Window function
  - ORDER BY, RANK(), NTILE(), window range…

CASE Expression

- Limited version of “If then else”
  - Returns different values depending on conditions
- Syntax:

```
CASE
    WHEN <condition> THEN <expr>
    WHEN <condition> THEN <expr>
    ELSE <expr>
END
```

- Can be used anywhere a column name can be referenced
  - SELECT, WHERE, GROUP BY, …

Q9: Average GPA within child/adult group

sid	name	addr	age	GPA
301	Andy	183 Westwood	19	2.1
303	Elaine	301 Wilshire	17	3.9
401	James	183 Westwood	17	3.5
208	Esther	421 Wilshire	20	3.1

→

AVG(GPA)	
3.7	(child group)
2.6	(adult group)

Q9: Average GPA within child/adult group

sid	name	addr	age	GPA
301	Andy	183 Westwood	19	2.1
303	Elaine	301 Wilshire	17	3.9
401	James	183 Westwood	17	3.5
208	Esther	421 Wilshire	20	3.1

→

age_group	AVG(GPA)
child	3.7
adult	2.6

- What if we want to show “child” and “adult” as part of output, not just the average?

Q9: Average GPA within child/adult group

sid	name	addr	age	GPA
301	Andy	183 Westwood	19	2.1
303	Elaine	301 Wilshire	17	3.9
401	James	183 Westwood	17	3.5
208	Esther	421 Wilshire	20	3.1

→

childGPA	adultGPA
3.7	2.6

- What about this output?

ORDER BY

- SQL is based on multiset semantics
  - Duplicates are allowed
  - Tuple order is ignored
- Still, for presentation purposes, it may be useful to order the result tuples by certain attribute(s)
  - Example: Order student tuples by GPA
- SELECT sid, GPA
  FROM Student
  ORDER BY GPA DESC, sid ASC
  - Default is ASC if omitted
  - ORDER BY does not change SQL semantics. It is purely for presentation

Q10: Top-3 students ordered by their GPA

- Sometimes we just want a few rows from the result. Is there a way to limit the result size?
- A:
  ```
  SELECT * FROM Students
  ORDER BY GPA DESC
  FETCH FIRST 3 ROWS ONLY
  ```
- FETCH FIRST Clause in SQL 2008
  - [OFFSET <num> ROWS] FETCH FIRST <count> ROWS ONLY
  - Skip the first <num> tuples and return the subsequent <count> rows
  - Unfortunately, this was standardized too late. Many variations are being used
    - MySQL: LIMIT <count> OFFSET <num>

General SQL SELECT

- SELECT attributes, aggregates
  FROM relations
  WHERE conditions
  GROUP BY attributes
  HAVING aggregate condition
  ORDER BY attributes
  FETCH FIRST n ROWS ONLY
- SELECT appears first, but is the last clause to be "interpreted"

Data Modification in SQL

- Insert tuple (301, ‘CS’, 201, 01) to Enroll table

- Populate Honors table with students of GPA > 3.7

- Syntax: INSERT INTO *relation tuples*;

Data Modification in SQL

- Delete all students who are not taking classes

- Syntax: DELETE FROM *relation* WHERE *condition*;
- Increase all CS course numbers by 100

- Syntax: UPDATE *relation*
  SET *A1 = V1, …, An= Vn*
  WHERE *condition*;

SQL: More Tricky Details

- NULL values
- Outer join
- Bag semantics for set operators
- Expressive power of SQL and recursion

Dealing with NULL

- Q: What will be returned from the following query if GPA is NULL?

```
SELECT name
FROM Student
WHERE GPA * 100/4 > 90
```

- Q: What should be the result from GPA * 100?
  - If input to an *arithmetic operator* is NULL, its output is NULL
- Q: What should be the result from NULL > 90?
  - Arithmetic comparison with NULL returns ***Unknown***

3-Valued Logic

- SQL is based on ***three-valued logic.***
  - All conditions are evaluated to be: **True**, **False** or **Unknown**
  - SQL returns a tuple if the result from condition is **True**
    - **False** or **Unknown** tuples will not be returned

-
  ```
  SELECT name
  FROM Student
  WHERE GPA * 100/4 > 90
  ```

Truth Table of Three-valued Logic

Assume GPA is NULL and age is 17

- Q: GPA > 3.7 AND age > 18. What is the result of this condition?
- Q: GPA > 3.7 OR age > 18. What is the result of this condition?

AND	True	False	Unknown
True	True	False	
False	False	False	

OR	True	False	Unknown
True	True	True	
False	True	False	

NULL and Aggregates

- Q: What should be the result for the following queries?

```
SELECT SUM(GPA) FROM Student

SELECT AVG(GPA) FROM Student

SELECT COUNT(GPA) FROM Student

SELECT COUNT(*) FROM Student
```

sid	GPA
1	3.0
2	3.6
3	2.4
4	NULL

NULL and Aggregates

- Aggregate functions ignore NULL values
  - Except COUNT(*), which counts a NULL valued tuple as a “valid” tuple
  - Note that COUNT(attr) does ignore a NULL valued attr

- When an input to an aggregate function is empty (= no input tuples):
  - COUNT() returns 0
  - All others return NULL

NULL and Set Operators

- Q: What should be {2.4, 3.0, NULL} ∪ {3.6, NULL}?

- NULL is treated like other regular values for set operators

Checking NULL

- In case we need to explicitly check whether an attribute value is NULL, we can use “IS NULL” or “IS NOT NULL” operator
  - Note that “= NULL” or “<> NULL” does ***not*** work!
- COALESCE() function
  - Return first non-NULL value in the list
  - Example: COALESCE(phone, email, addr)

Q: number of classes each student takes. return 0-class students as well

Student(<u>sid</u>, name, addr, age, GPA)

sid	name	addr	age	GPA
301	Andy	183 Westwood	19	2.1
303	Elaine	301 Wilshire	17	3.9
401	James	183 Westwood	17	3.5
208	Esther	421 Wilshire	20	3.1

Class(<u>dept</u>, <u>cnum</u>, <u>sec</u>, unit, title, instructor)

dept	cnum	sec	unit	title	instructor
CS	112	01	03	Modeling	Dick Muntz
CS	143	01	04	DB Systems	John Cho
EE	143	01	03	Signal	Dick Muntz
ME	183	02	05	Mechanics	Susan Tracey

Enroll(sid, dept, cnum, sec)

sid	dept	cnum	sec
301	CS	112	01
301	CS	143	01
303	EE	143	01
303	CS	112	01
401	CS	112	01

Outer join preserves dangling tuples

Outer Join

Student

sid	name
301	John
303	Elaine

Enroll

sid	cid
301	CS143
401	CS112

- Student LEFT OUTER JOIN Enroll ON Student.sid = Enroll.sid
- Student RIGHT OUTER JOIN Enroll ON Student.sid = Enroll.sid
- Student FULL OUTER JOIN Enroll ON Student.sid = Enroll.sid

SQL and Multiset Semantics

- Multiset (= Bag)
    - A set with duplicate elements
    - Order of elements does not matter
    - {a, a, b, c} = {a, b, c, a} $\neq$ {a, b, c}
- SQL is based on multiset semantics
    - We already learned how duplicates are generated and kept in SQL
    - Use DISTINCT to eliminate duplicates in the result
    - Exception: ***set operators are based on set semantics***

Multiset Semantics for Set Operators

- To use bag semantics for set operators, use **ALL** keyword
  - UNION ALL, INTERSECT ALL, EXCEPT ALL
- Q: {a, a, b} ∪ {a, b, c}?
- Q: {a, a, a, b, c} ∩ {a, a, b}?
- Q: {a, a, b, b} – {a, b, b, c}?

Multiset Semantics and Equivalence Rules

Under multiset semantics:

- $R \cup S = S \cup R$?

- $R \cap S = S \cap R$?

- $R \cap (S \cup T) = (R \cap S) \cup (R \cap T)$?

- Not all set equivalence rules hold under multiset. Be careful!!!

Expressive Power of SQL

- Q: Find all ancestors of Susan

Parent

child	parent
Susan	John
John	James
James	Elaine
…	…

- Q: Find all cities reachable from A?

Reachable

City 1	Citi 2
A	B
B	D
B	E
…	…

Expressive Power of SQL

- SQL is a very expressive language, but its expressive power is limited
  - SQL is not a “Turing-complete” language
- For example, the closure of a set cannot be computed using SQL92
  - Example: all ancestors, all reachable nodes
  - Support for recursion is needed to compute a closure
- SQL99 added support for recursion

  -
    ```
    WITH RECURSIVE Ancestor(child, ancestor) AS (
                 (SELECT * FROM Parent)
       UNION
                 (SELECT P.child, A.ancestor
                  FROM Parent P, Ancestor A
                  WHERE P.parent = A.child) )
    SELECT ancestor FROM Ancestor WHERE child=‘Susan’;
    ```
  - Read textbook for more details on SQL99 Recursion

What We Learned

- Aggregate function
- Window function
- Case expression
- Order by and Fetch first
- Data Modification
- NULL and three-valued logic
- Outer join
- Multiset semantic for set operators
- SQL expressive power and recursion