HALMSTAD UNIVERSITY

DS4001 Databases (7.5 credits)

Lecture 8 – Intermediate SQL & Exercises

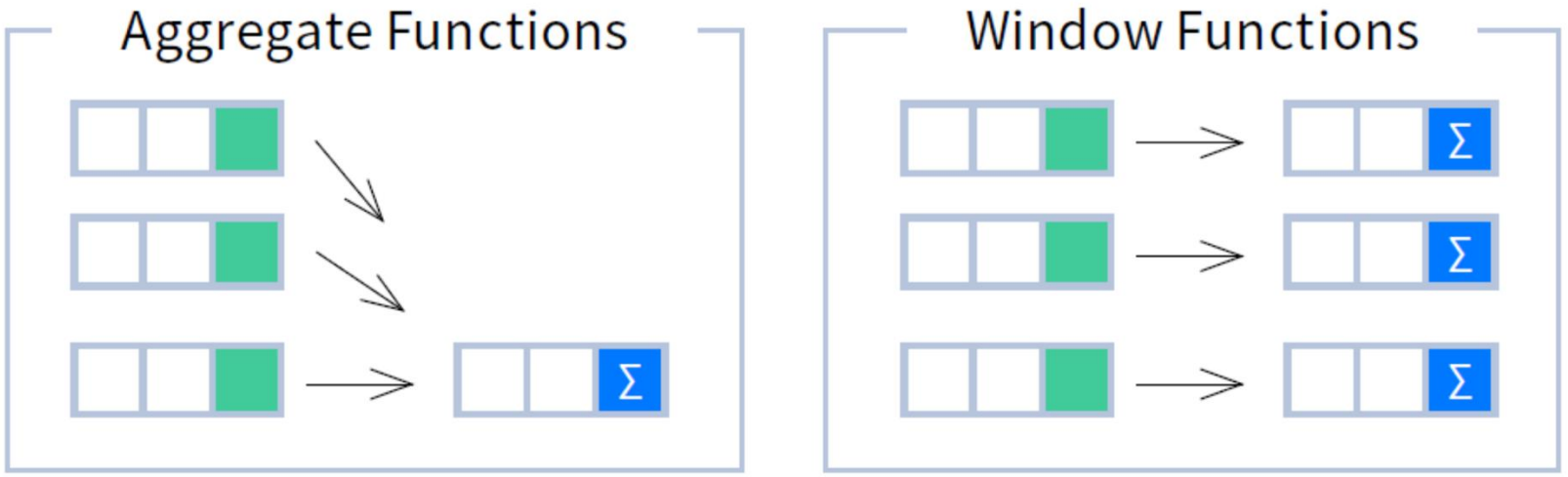
Yuantao Fan

yuantao.fan@hh.se

Halmstad University

Window Functions

- “A window function performs a calculation across a set of table rows (tuples) that are somehow related to the current row (tuple).”
- Which is comparable to the aggregate function, but it does not collapse the rows to become grouped into a single output row



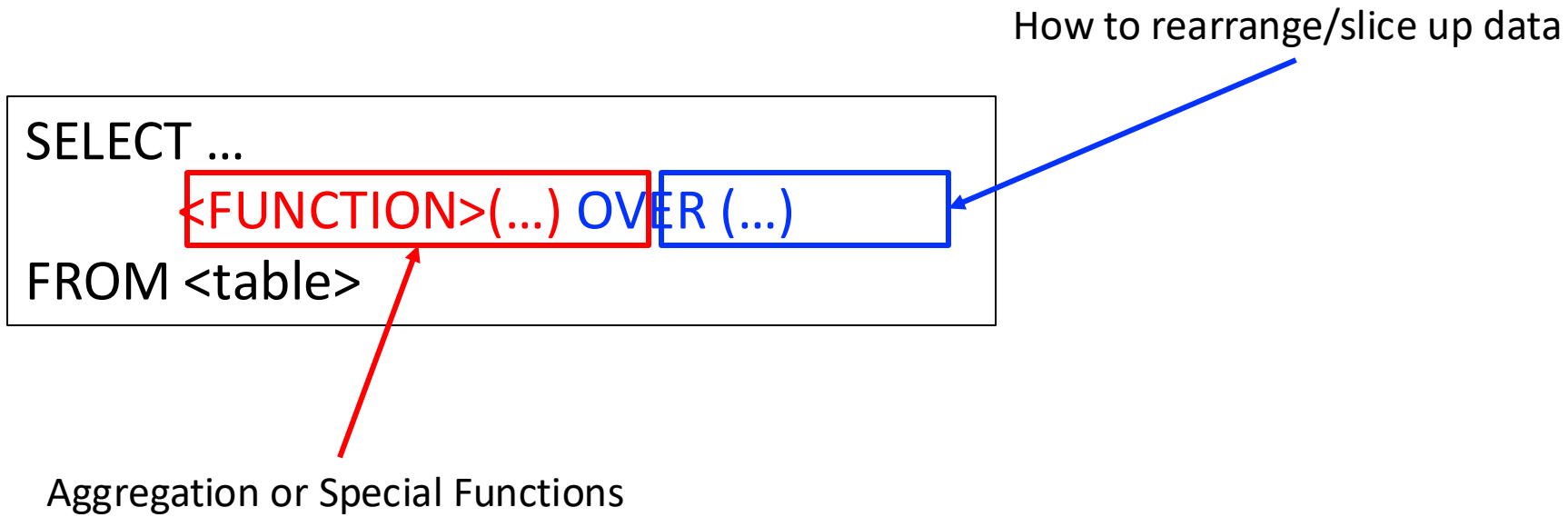
Window Functions

- “A window function performs a calculation across a set of table rows (tuples) that are somehow related to the current row (tuple).”
- Which is comparable to the aggregate function, but it does not collapse the rows to become grouped into a single output row

```
SELECT …
        <FUNCTION>(…) OVER (…)
FROM <table>
```

Window Functions

- “A window function performs a calculation across a set of table rows (tuples) that are somehow related to the current row (tuple).”
- Which is comparable to the aggregate function, but it does not collapse the rows to become grouped into a single output row



Window Functions

- Aggregation functions
  - MAX, MIN, AVG, …
- Special window functions:
  - ROW_NUMBER() – gives the number of the current row
  - RANK() – order position of the current row

```
SELECT …
        <FUNCTION>(…) OVER (…)
FROM <table>
```

```
SELECT …
        ROW_NUMBER() OVER() AS row_num
FROM Enrolled
```

sid	cid	grade
c1	11	A
c1	33	A
p2	44	A
p5	44	B
m4	11	A
p2	11	B
m4	22	B
p5	33	C
c1	22	A

sid	cid	grade	row_num
c1	11	A	1
c1	33	A	2
p2	44	A	3
p5	44	B	4
m4	11	A	5
p2	11	B	6
m4	22	B	7
p5	33	C	8
c1	22	A	9

Window Functions

- Aggregation functions
  - MAX, MIN, AVG, …
- Special window functions:
  - ROW_NUMBER() – gives the number of the current row
  - RANK() – order position of the current row

```
SELECT …
        <FUNCTION>(…) OVER (…)
FROM <table>
```

```
SELECT cid, sid,
        ROW_NUMBER() OVER(PARTITION BY cid) FROM
Enrolled
```

sid	cid	grade
c1	11	A
c1	33	A
p2	44	A
p5	44	B
m4	11	A
p2	11	B
m4	22	B
p5	33	C
c1	22	A

Window Functions

- Aggregation functions
  - MAX, MIN, AVG, …
- Special window functions:
  - ROW_NUMBER() – gives the number of the current row
  - RANK() – order position of the current row

```
SELECT …
        <FUNCTION>(…) OVER (…)
FROM <table>
```

```
SELECT cid, sid,
        ROW_NUMBER() OVER(PARTITION BY cid) FROM
Enrolled
```

sid	cid	grade
c1	11	A
c1	33	A
p2	44	A
p5	44	B
m4	11	A
p2	11	B
m4	22	B
p5	33	C
c1	22	A

cid	sid	ROW_NUMBER() OVER(PARTITION BY cid)
11	c1	1
11	m4	2
11	p2	3
22	m4	1
22	c1	2
33	c1	1
33	p5	2
44	p2	1
44	p5	2

Window Functions

- Aggregation functions
  - MAX, MIN, AVG, …
- Special window functions:
  - ROW_NUMBER() – gives the number of the current row
  - RANK() – order position of the current row

```
SELECT …
        <FUNCTION>(…) OVER (…)
FROM <table>
```

```
SELECT cid, sid,
        ROW_NUMBER() OVER(ORDER BY cid) FROM
Enrolled
```

sid	cid	grade
c1	11	A
c1	33	A
p2	44	A
p5	44	B
m4	11	A
p2	11	B
m4	22	B
p5	33	C
c1	22	A

cid	sid	ROW_NUMBER() OVER(ORDER BY cid)
11	c1	1
11	m4	2
11	p2	3
22	m4	4
22	c1	5
33	c1	6
33	p5	7
44	p2	8
44	p5	9

Window Functions

- Aggregation functions
  - MAX, MIN, AVG, …
- Special window functions:
  - ROW_NUMBER() – gives the number of the current row
  - RANK() – order position of the current row

```
SELECT …
        <FUNCTION>(…) OVER (…)
FROM <table>
```

```
SELECT cid, sid,
        ROW_NUMBER() OVER(ORDER BY cid DESC)
FROM Enrolled
```

sid	cid	grade
c1	11	A
c1	33	A
p2	44	A
p5	44	B
m4	11	A
p2	11	B
m4	22	B
p5	33	C
c1	22	A

cid	sid	ROW_NUMBER() OVER(ORDER BY cid DESC)
44	p2	1
44	p5	2
33	c1	3
33	p5	4
22	m4	5
22	c1	6
11	c1	7
11	m4	8
11	p2	9

Window Functions

- Aggregation functions
  - MAX, MIN, AVG, …
- Special window functions:
  - ROW_NUMBER() – gives the number of the current row
  - RANK() – order position of the current row

```
SELECT cid, sid,
       RANK() OVER(ORDER BY cid ASC) FROM Enrolled
```

```
SELECT cid, sid,
       RANK() OVER(ORDER BY cid DESC) FROM Enrolled
```

sid	cid	grade
c1	11	A
c1	33	A
p2	44	A
p5	44	B
m4	11	A
p2	11	B
m4	22	B
p5	33	C
c1	22	A

cid	sid	RANK() OVER(ORDER BY cid ASC)
11	c1	1
11	m4	1
11	p2	1
22	m4	4
22	c1	4
33	c1	6
33	p5	6
44	p2	8
44	p5	8

cid	sid	RANK() OVER(ORDER BY cid DESC)
44	p2	1
44	p5	1
33	c1	3
33	p5	3
22	m4	5
22	c1	5
11	c1	7
11	m4	7
11	p2	7

Window Functions

- Find the student with the second highest grade for each course

sid	cid	grade
c1	11	A
c1	33	A
p2	44	A
p5	44	B
m4	11	A
p2	11	B
m4	22	B
p5	33	C
c1	22	A

Window Functions

- Find the student with the first highest grade for each course

sid	cid	grade
c1	11	A
c1	33	A
p2	44	A
p5	44	B
m4	11	A
p2	11	B
m4	22	B
p5	33	C
c1	22	A

```
SELECT * FROM (
                    SELECT *, RANK() OVER(PARTITION BY cid
                                                ORDER BY grade ASC) AS _rank
          FROM Enrolled) AS _ranking;
```

sid	cid	grade	_rank
c1	11	A	1
m4	11	A	1
p2	11	B	3
c1	22	A	1
m4	22	B	2
c1	33	A	1
p5	33	C	2
p2	44	A	1
p5	44	B	2

Window Functions

- Find the student with the second highest grade for each course

sid	cid	grade
c1	11	A
c1	33	A
p2	44	A
p5	44	B
m4	11	A
p2	11	B
m4	22	B
p5	33	C
c1	22	A

```
SELECT * FROM (
                    SELECT *, RANK() OVER(PARTITION BY cid
                                          ORDER BY grade ASC) AS _rank
          FROM Enrolled) AS _ranking
           WHERE _ranking._rank=2;
```

sid	cid	grade	_rank
m4	22	B	2
p5	33	C	2
p5	44	B	2

Window Functions

- Find the student with the first highest grade for each course

sid	cid	grade
c1	11	A
c1	33	A
p2	44	A
p5	44	B
m4	11	A
p2	11	B
m4	22	B
p5	33	C
c1	22	A

```
SELECT * FROM (
                    SELECT *, RANK() OVER(PARTITION BY cid
                                          ORDER BY grade ASC) AS _rank
          FROM Enrolled) AS _ranking
           WHERE _ranking._rank=1;
```

sid	cid	grade	_rank
c1	11	A	1
m4	11	A	1
c1	22	A	1
c1	33	A	1
p2	44	A	1

Common Table Expressions

- Provides a way to write auxiliary statements for use in a larger query
  - Just like a temporary table for one query
- An alternative to views and nested queries
- WITH ... AS

```
WITH <temp_table_name> AS (
    SELECT 1
)
SELECT * FROM <temp_table_name>
```

Common Table Expressions

- Provides a way to write auxiliary statements for use in a larger query
  - Just like a temporary table for one query
- An alternative to views and nested queries
- WITH ... AS

```
WITH cte_tab AS (
    SELECT 1
)
SELECT * FROM cte_tab;
```

Common Table Expressions

- Provides a way to write auxiliary statements for use in a larger query
  - Just like a temporary table for one query
- An alternative to views and nested queries
- WITH ... AS
- Bind output columns to names before the keyword AS

```
WITH cte_tab (col1, col2) AS (
      SELECT 1, 2
)
SELECT col1 + col2 FROM cte_tab;
```

col1 + col2
3

Common Table Expressions

- Find student record with the highest id that is enrolled in at least one course

sid	cid	grade
1	11	A
1	33	A
2	44	A
5	44	B
4	11	A
2	11	B
4	22	B
5	33	C
1	22	A

sid	full_name	major	age	GPA
1	Alice	CS	21	4
2	Albert	PHY	22	3.9
3	Tim	EE	20	3.9
4	Kayle	MATH	19	3.8
5	Yasuo	PHY	19	3.7

Common Table Expressions

- Find student record with the highest id that is enrolled in at least one course

```
WITH tempSource (maxId) AS (
    SELECT MAX(sid) FROM enrolled
)
SELECT *
FROM student, tempSource
WHERE student.sid = tempSource.maxID
```

sid	cid	grade
1	11	A
1	33	A
2	44	A
5	44	B
4	11	A
2	11	B
4	22	B
5	33	C
1	22	A

sid	full_name	major	age	GPA
1	Alice	CS	21	4
2	Albert	PHY	22	3.9
3	Tim	EE	20	3.9
4	Kayle	MATH	19	3.8
5	Yasuo	PHY	19	3.7

sid	full_name	major	age	GPA	maxId
5	Yasuo	PHY	19	3.7	5

Common Table Expressions - Recursion

- Provides a way to write auxiliary statements for use in a larger query
  - Just like a temporary table for one query
- An alternative to views and nested queries
- WITH ... AS
- Bind output columns to names before the keyword AS
- Recursion

```
WITH RECURSIVE cte_source (ctr) AS (
      (SELECT 1)
      UNION
      (SELECT ctr + 1 FROM cte_source
      WHERE ctr < 5)
)
SELECT * FROM cte_source;
```

Print numbers from 1 to 5

ctr
1
2
3
4
5

Triggers

- A trigger is a procedure which is executed implicitly whenever the triggering event
  - usually a named database object that is associated with a table
  - activates when a particular event occurs for the table
- Triggering event
  - Insert, Update, Delete statement
- Purposes
  - Maintain integrity constraints
  - Record or tracking auditing information of the changes
  - Send a signal to a program that processing needs to be performed when the triggering event occurred

Triggers

- DML statements
  - INSERT, UPDATE, DELETE
- Timming of triggers
  - Before
  - After
  - Instead of
- Level
  - Row
    - Keywords – FOR EACH ROW
  - Statment

Triggers

- DML statements
  - INSERT, UPDATE, DELETE
- Timming of triggers
  - Before
  - After
  - Instead of
- Level
  - Row
    - Keywords – FOR EACH ROW
  - Statment

```
CREATE TABLE account (acc_num INT, amount FLOAT);

CREATE TRIGGER ins_sum BEFORE INSERT ON account
FOR EACH ROW SET @sum = @sum + NEW.amount;
```

```
SET @sum = 0;
INSERT INTO account
VALUES(10,20),(11,100),(12,-30),(13,50);
SELECT @sum AS 'Total amount inserted';
```

Total amount inserted
140

Triggers

- DML statements
  - INSERT, UPDATE, DELETE
- Timming of triggers
  - Before
  - After
  - Instead of
- Level
  - Row
    - Keywords – FOR EACH ROW
  - Statment

```
CREATE TABLE account (acc_num INT, amount FLOAT);

CREATE TRIGGER ins_sum BEFORE INSERT ON account
FOR EACH ROW SET @sum = @sum + NEW.amount;
```

```
SET @sum = 0;
INSERT INTO account
VALUES(10,20),(11,100),(12,-30),(13,50);
SELECT @sum AS 'Total amount inserted';
```

```
DROP TRIGGER [db_name].ins_sum;
```