# BT5110 Data Management and Warehousing

## Tutorial 2: Simple and Algebraïc Queries

Mark Meng Huasong

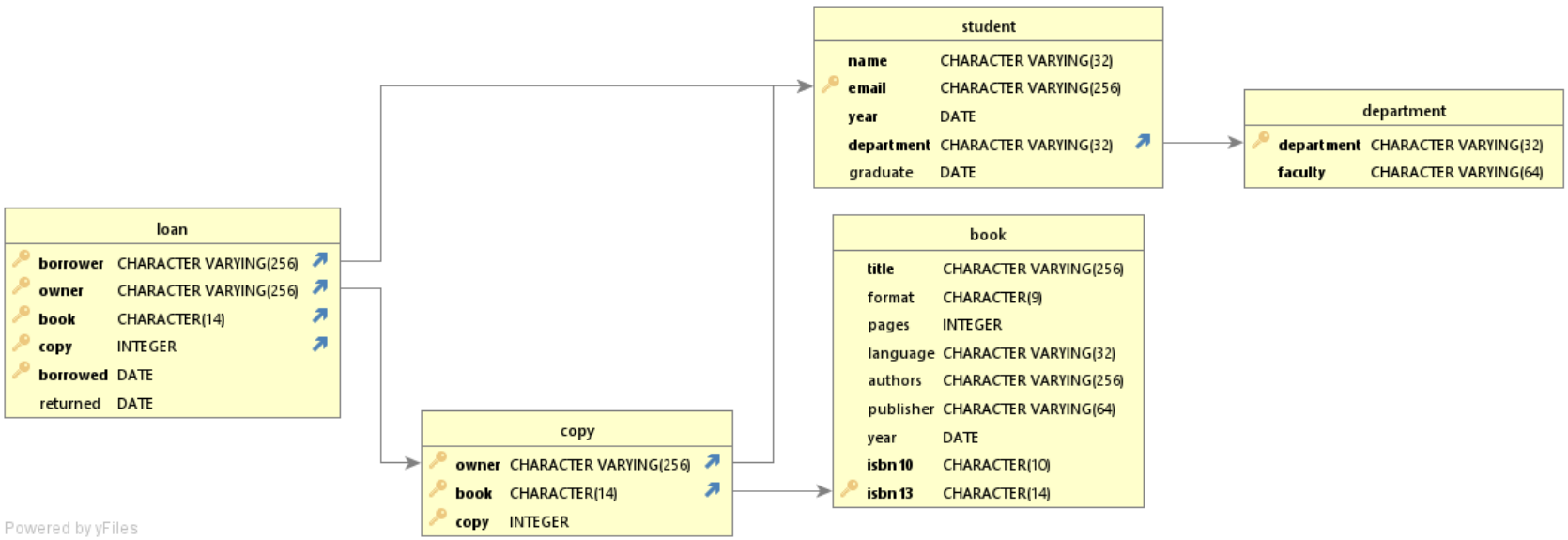
School of Computing
National University of Singapore

30 Aug - 3 Sep, 2021

# Quick Recap: End of Last Tutorial

What we have done in the last week:

(1) Create 4 tables and populate data;
(2) Update all 'CS' with 'Computer Science' in `department` column;
(3) Drop `available` from table `copy`;
(4) Create a separate table `department` and migrate `faculty` from table `student` to table `department`.




(plotted by DbVisualizer)

# Question 1 (a-c)

(a) Print the different departments.

(b) Print the different departments in which students are enrolled.

(c) Let us check the integrity of the data. Print the emails of the students who borrowed or lent a copy of a book before they joined the university. There should not be any. Use a simple query.

# Solution 1 (a, b)

(a) Print the different departments.

```
1   SELECT d.department FROM department d;
```

13 row are returned. Notice that the query does not require `DISTINCT` to eliminate duplicates. Duplicates are guaranteed not to occur because `department` is the `PRIMARY KEY` of the table `department`.

(b) Print the different departments in which students are enrolled.

```
1   SELECT DISTINCT s.department FROM student s;
```

13 row are returned. There could be departments in which no student is enrolled. This is the case of the department of `Undecidable Computations`. We need to look into the `student` table.

**Do these two questions return the exactly SAME outputs?**

# Solution 1 (a, b) Cont.

**Notice**

The outputs of these two queries have the same contents but with different orders.

```
BT5110=# select d.department from department d;
 department
------------
 CS
 History
 Physics
 Geography
 Language
 Economics
 Biology
 EE
 Math
 IS
 CE
 ME
 Chemistry
(13 rows)
```

```
BT5110=# select DISTINCT s.department from student s;
 department
------------
 EE
 CS
 ME
 Language
 Math
 CE
 IS
 Physics
 Economics
 Geography
 Biology
 Chemistry
 History
(13 rows)
```

**Extra**: We can add "`ORDER BY d.department ASC`" in 1(a) and "`ORDER BY s.department ASC`" in 1(b) to make these two outputs same in ordering.

# Solution 1 (c)

(c) Let us check the integrity of the data. Print the emails of the students who borrowed or lent a copy of a book before they joined the university. **There should not be any.** Use a simple query.

**Solution:** Don't forget to consider both *borrowing* and *lending* scenarios.

```
1  SELECT DISTINCT s.email FROM loan l, student s
2  WHERE (s.email = l.borrower AND l.borrowed < s.year)
3    OR (s.email = l.owner AND l.borrowed < s.year);
```

# Question 1 (d)

For each copy that has been borrowed and returned, print the **duration** of the loan. Order the results in **ascending** order of the ISBN13 and **descending** order of duration.

**How can the duration be derived? Can we use "returned - borrowed AS duration"?**

```
1    SELECT book, returned - borrowed AS duration
2    FROM loan
3    ORDER BY book ASC, duration DESC;
```

**Any issue in the code above?**

# Solution 1 (d)

Let's manually exclude NULL values in `returned` column:

```
1    SELECT book, returned - borrowed + 1 AS duration
2    FROM loan
3    WHERE NOT (returned ISNULL)
4    ORDER BY book ASC, duration DESC;
```

**Notice**

ASC is the default, but it is strongly recommended to indicate it for clarity.

**Result**: 4871 rows are returned ($\neq$ number of rows in table `loan`).

# Solution 1 (d) Cont.

**What if the question ask like follows:**

For each copy ~~that has been borrowed and returned~~ record in the `loan` table, print the **duration** of the loan. Order the results in **ascending** order of the ISBN13 and **descending** order of duration.

# Solution 1 (d) Cont.

Notice that the duration can be **null** if the book has not been returned yet. To answer this question, you need to calculate the duration until a specific date (e.g., December 31, 2010) to include the books that have not been returned yet.

```
1  SELECT book, ((CASE WHEN returned ISNULL
2      THEN '2010-12-31'
3      ELSE returned END) - borrowed + 1) AS duration
4    FROM loan
5    ORDER BY book ASC, duration ASC;
```

**Result**: 4976 rows are returned (= number of rows in table `loan`)

You can also use a Postgres reserved command "`current_date`" to obtain the date of today (by doing this, some `duration` values will be huge).

# Question 1 (e)

For each loan of a book published by Wiley that has not been returned, print the title of the book, the name and faculty of the owner and the name and faculty of the borrower. Use **CROSS JOIN**.

Let's do 4 steps to form your query as follows:

Step (1) FIND all output columns;
Step (2) IDENTIFY all value constraints;
Step (3) CONFIRM all tables involved;
Step (4) CONNECT tables involved (w. necessary relation constraints).

*** LIVE DEMO ***

# Solution 1 (e)

We join primary keys and foreign keys to ***stitch*** tables together properly.

```
1   SELECT b.title,
2     s1.name AS ownername,
3     d1.faculty AS ownerFaculty,
4     s2.name AS borrowername,
5     d2.faculty AS  borrowerfaculty
6   FROM loan l, book b,  copy c,
7     student s1, student s2,
8     department d1, department d2
9   WHERE c.book = b.ISBN13                    (1)
10    AND c.book = l.book                      (2)
11    AND c.copy = l.copy                      (3)
12    AND c.owner = l.owner                    (4)
13    AND l.owner = s1.email                   (5)
14    AND l.borrower = s2.email                (6)
15    AND s1.department = d1.department        (7)
16    AND s2.department = d2.department        (8)
17    AND b.publisher ='Wiley'                 (A)
18    AND l.returned ISNULL;                   (B)
```

**10** rows are returned.

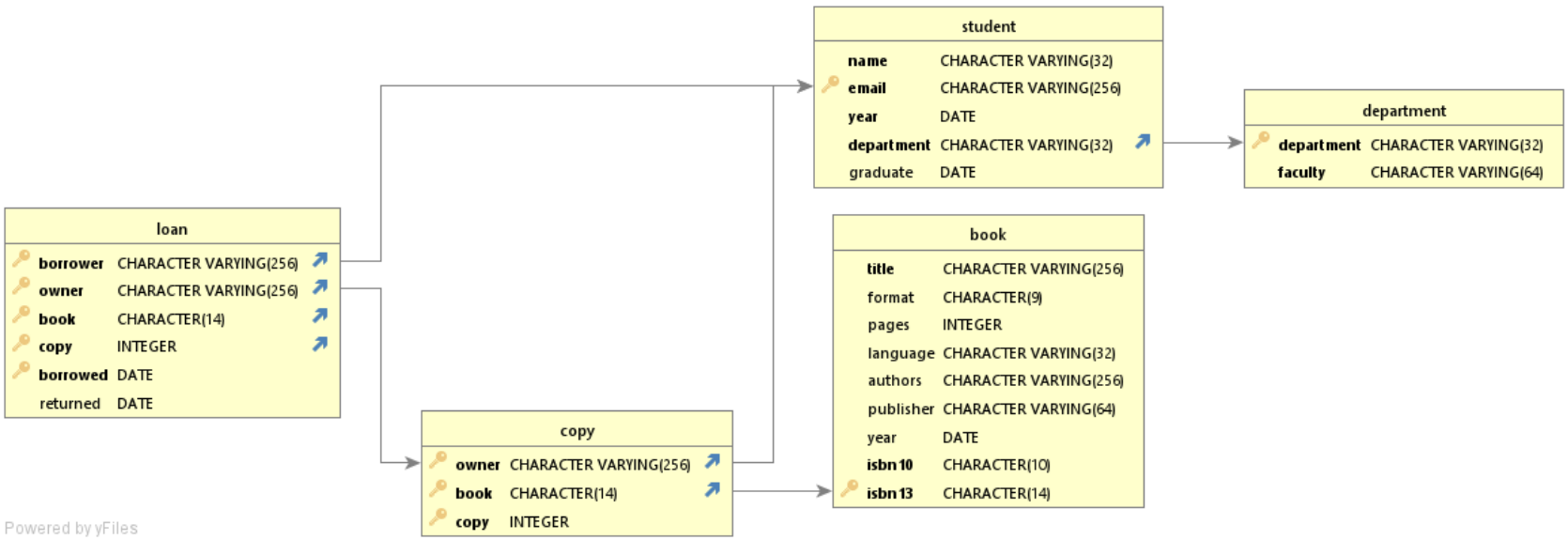
# Solution 1 (e) Cont.

You can omit the table copy and the copy column since the existence of the corresponding rows and values is guaranteed by design and by the foreign and primary key constraints.

```
1   SELECT b.title,
2     s1.name AS ownername,
3     d1.faculty AS ownerFaculty,
4     s2.name AS borrowername,
5     d2.faculty AS  borrowerfaculty
6   FROM loan l, book b,
7     student s1, student s2,
8     department d1, department d2
9   WHERE l.book = b.ISBN13
10    AND l.owner = s1.email
11    AND l.borrower = s2.email
12    AND s1.department = d1.department
13    AND s2.department = d2.department
14    AND b.publisher ='Wiley'
15    AND l.returned ISNULL;
```

# Question 2 (a)

For each loan of a book published by Wiley that has not been returned, print the title of the book, the name and faculty of the owner and the name and faculty of the borrower. Use **INNER JOIN**.




This is the same question with Q1(e), but requires you to use INNER JOIN.

# Solution 2 (a)

Let's convert the solution of Q1(e) now!

```
1   SELECT b.title,
2     s1.name AS ownername,
3     d1.faculty AS ownerFaculty,
4     s2.name AS borrowername,
5     d2.faculty AS  borrowerfaculty
6   FROM loan l, book b,
7     student s1, student s2,
8     department d1, department d2
9   WHERE l.book = b.ISBN13
10    AND l.owner = s1.email
11    AND l.borrower = s2.email
12    AND s1.department = d1.department
13    AND s2.department = d2.department
14    AND b.publisher ='Wiley'
15    AND l.returned ISNULL;
```

```
1   SELECT b.title,
2     s1.name AS ownername,
3     d1.faculty AS ownerFaculty,
4     s2.name AS borrowername,
5     d2.faculty AS
          borrowerfaculty
6   FROM loan l
7     INNER JOIN ____ ON ________
8     INNER JOIN ____ ON ________
9     INNER JOIN ____ ON ________
10    INNER JOIN ____ ON ________
11    INNER JOIN ____ ON ________
12  WHERE b.publisher ='Wiley'
13    AND l.returned ISNULL;
```

# Solution 2 (a) Cont.

You can omit the table copy and the copy column since the existence of the corresponding rows and values is guaranteed by design and by the foreign and primary key constraints.

```
1   SELECT b.title,
2     s1.name AS ownername,
3     d1.faculty AS ownerFaculty,
4     s2.name AS borrowername,
5     d2.faculty AS  borrowerfaculty
6   FROM loan l
7     INNER JOIN book b ON l.book=b.ISBN13
8     INNER JOIN student s1 ON l.owner = s1.email
9     INNER JOIN student s2 ON l.borrower = s2.email
10    INNER JOIN department d1 ON s1.department = d1.department
11    INNER JOIN department d2 ON s2.department = d2.department
12  WHERE  b.publisher ='Wiley'
13    AND l.returned ISNULL;
```

**10** rows are returned.

# Solution 2 (a) Cont.

The code below is **without omitting table copy**. The simplified version is shown on the previous slide.

```
1     SELECT b.title ,
2     s1. name AS ownername ,
3     d1. faculty AS ownerFaculty ,
4     s2. name AS borrowername ,
5     d2. faculty AS borrowerfaculty
6     FROM loan l
7     INNER JOIN book b ON l. book = b. ISBN13
8     INNER JOIN copy c ON c. book = l. book
9     AND c. copy = l. copy AND c. owner = l. owner
10    INNER JOIN student s1 ON l. owner = s1. email
11    INNER JOIN student s2 ON l. borrower = s2. email
12    INNER JOIN department d1 ON s1. department = d1. department
13    INNER JOIN department d2 ON s2. department = d2. department
14    WHERE b. publisher ='Wiley'
15    AND l. returned ISNULL ;
```

# Question 2 (b)

Print the emails of the different students who borrowed or lent a copy of a book on the day that they joined the university. Use an algebraïc query.

**Set operations (UNION, INTERSECT and EXCEPT) could help to address this question.**

# Solution 2 (b)

```
1  SELECT  s.email
2  FROM loan l, student s
3  WHERE s.email = l.borrower AND l.borrowed = s.year
4  UNION
5  SELECT  s.email
6  FROM loan l, student s
7  WHERE s.email = l.owner AND l.borrowed = s.year;
```

**19** rows are returned.

DISTINCT is not needed because UNION eliminates duplicates (so do INTERSECT, EXCEPT and MINUS).

# Question 2 (b) Cont.

There is an alternative (simple) way to write the query, without SET operations.

The corresponding simple query is generally preferable.

```
1  SELECT DISTINCT s.email
2  FROM loan l,  student s
3  WHERE (s.email = l.borrower OR s.email = l.owner)
4    AND l.borrowed = s.year;
```

**Notice**

The simple query requires an explicit `DISTINCT`.

# Question 2 (c)

Print the emails of the different students who borrowed and lent a copy of a book on the day that they joined the university. Use an algebraïc query.

**Which set operation (UNION, INTERSECT and EXCEPT) should we use for this question?**

# Solution 2 (c)

```
1  SELECT  s.email
2  FROM loan l, student s
3  WHERE s.email = l.borrower AND l.borrowed = s.year
4  INTERSECT
5  SELECT  s.email
6  FROM loan l, student s
7  WHERE s.email = l.owner AND l.borrowed = s.year;
```

**4** rows are returned.

Note that the corresponding simple query is more ***complicated***. It needs ***two*** loan tables.

```
1  SELECT DISTINCT s.email
2  FROM loan l1, loan l2, student s
3  WHERE s.email = l1.borrower AND l1.borrowed = s.year
4  AND s.email = l2.owner AND l2.borrowed = s.year;
```

# Question 2 (d)

Print the emails of the students who borrowed but did not lend a copy of a book on the day that they joined the university. Use an algebraïc query.

**Which set operation should we use for this question?**

# Solution 2 (d)

```
1  SELECT  s.email
2  FROM loan l, student s
3  WHERE s.email = l.borrower AND l.borrowed = s.year
4  EXCEPT
5  SELECT  s.email
6  FROM loan l, student s
7  WHERE s.email = l.owner AND l.borrowed = s.year;
```

**9** rows are returned.

There is no corresponding simple query. We would need to use nested or aggregate queries for this type of questions.

# Question 2 (e)

Print the ISBN13 of the books that have never been borrowed. Use an algebraïc query.

# Solution 2 (e)

**Solution**:

```
1  SELECT b.ISBN13
2  FROM book b
3  EXCEPT
4  SELECT l.book
5  FROM loan l;
```

**0** rows are returned.

or, using an *OUTER JOIN*, which introduces NULL values,

```
1  SELECT b.ISBN13
2  FROM book b LEFT OUTER JOIN loan l ON b.isbn13 = l.book
3  WHERE l.book ISNULL;
```

There is no corresponding simple query. We would need to use nested or aggregate queries for this type of questions.

For any further question, please feel free to email me:

huasong.meng@u.nus.edu