COMP3311 Week 10

By Manhua Lu

Adapted from Kyu-Sang Kim

Announcements

- Please complete the myExperience survey!
  - Would mean a lot to me to help me improve ☺️
- Quiz 6 due Saturday 18 November @11:59pm
- Assignment 2 deadline extended by 12 hours
  - Thursday 16 November 11:59am
- Just final exam left now!! 🙌
  - Monday 4 December
  - In-person lab exam time preference form open now, go fill in!!
  - Make sure to watch live lectures or at least **look at the slides**. Locking (in 22T3 Wk10) was not in the prerecorded vids, but was in the final exam 😳

Learning Objectives

01 → Relational Algebra: Operators

02 → Relational Algebra: Composition

03 → Transaction Processing: Serialisability

04 → Transaction Processing: Precedence Graphs

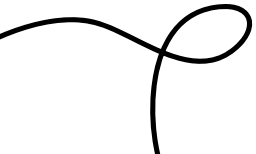
05 → Transaction Processing: View Equivalence

01

Relational Algebra: Operators

01 Relational Algebra: Operators

- Like SQL but more mathematical
- Selection: sel[expr](rel)
  - selects the rows in some relation that satisfies expression
- Projection: proj[a,b,c](rel)
  - filters relation by its columns a, b, c
- Rename: rename[schema](rel)
  - renames the columns of rel with the columns specified in schema



01 Relational Algebra: Operators

Sel[B=1](R) is equivalent to

```
select  *
from    R
where   B = 1;
```

Examples of selection:

R

A	B	C	D
a	1	x	4
b	2	y	5
c	4	z	4
d	8	x	5
e	1	y	4
f	2	x	5

Sel[A=C](R)

A	B	C	D

Sel[B=1](R)

A	B	C	D
a	1	x	4
e	1	y	4

Sel[B>=D](R)

A	B	C	D
c	4	z	4
d	8	x	5

Sel[A=b or A=c](R)

A	B	C	D
b	2	y	5
c	4	z	4

01 → Relational Algebra: Operators

Proj[A, B, C](R) is equivalent to

```
select  distinct A,B,C
from    R;
```

Examples of projection:

R

A	B	C	D
a	1	x	4
b	2	y	5
c	4	z	4
d	8	x	5
e	1	y	4
f	2	x	5

Proj[A,B,C](R)

A	B	C
a	1	x
b	2	y
c	4	z
d	8	x
e	1	y
f	2	x

Proj[B,D](R)

B	D
1	4
2	5
4	4
8	5

← Removes duplicates ‼️

Proj[D](R)

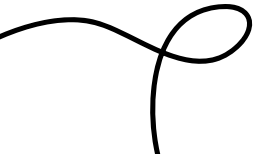
D
4
5

01 → Relational Algebra: Operators

Rename[beerName](Proj[name](Beers))

is equivalent to

```
select  name as beerName
from    Beers;
```



01 → Relational Algebra: Operators

Union: Table1 Union Table2

Intersection: Table1 Intersection Table2

Difference: Table1 - Table2

s1 = Sel [B = 1] (r1)

A	B	C	D
a	1	x	4
e	1	y	4

s2 = Sel [C = x] (r1)

A	B	C	D
a	1	x	4
d	8	x	5

s1 - s2

A	B	C	D
e	1	y	4

s2 - s1

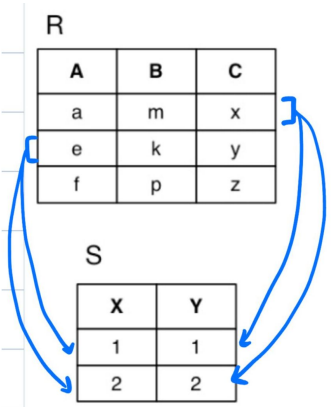
A	B	C	D
d	8	x	5

- Returns tuples that exist in Table1 but DON'T exist in Table2
- NOT symmetric / NOT complement of Intersection
- i.e. Table1 - Table2 != Table2 - Table1

01 → Relational Algebra: Operators

Product: Table1 x Table2

- Pairwise product between two tables
- for every row in Table1, attach every row in Table2
- T1: {a, b, c, d}, T2: {e, f}
- T1 x T2: {T1.a, T1.b, T1.c, T1.d, T2.e, T2.f}



R x S

A	B	C	X	Y
a	m	x	1	1
a	m	x	2	2
e	k	y	1	1
e	k	y	2	2
f	p	z	1	1
f	p	z	2	2

01 Relational Algebra: Operators

Division: Table1 Div Table2

- Returns all rows in T1 which have a match for all the rows in T2
- Captures the notion of “for-all”
  - i.e. returns all R that will satisfy T

R

A	B
4	x
4	y
4	z
5	x
5	y
5	z

S

A	B
4	x
4	y
4	z
5	x
5	z

T

B
x
y

R / T

A
4
5

S / T

A
4

A		B	A/B
X	Y	Y	X
a	1	1	Null
b	2	2	
a	3	3	
d	4	4	

01 Relational Algebra: Operators

Natural join: Table1 join Table2

- Joins two tables on common attributes that are matched

Theta join: Table1 join [condition] Table2

- A natural join with a ‘where’ condition applied

Outer join: Table1 [Full|Left|Right] Outer Join Table 2

- Left outer - all tuples from T1 included (T2 non matches set to null)
- Right outer - all tuples from T2 included (T1 non matches set to null)
- Full outer - both (left and right outer join applies)

02

Relational Algebra: Composition

02 Relational Algebra: Composition

- **Composition**: Relational algebra expressions can be combined by using the result of one expression as the input of another expression
    - This allows us to compose simpler expressions which come together to answer arbitrarily complex queries.

02 Relational Algebra: Composition Questions

4. Consider the following two relations:

R

A	B	C
a1	b1	c1
a1	b2	c2
a2	b1	c1

S

B	C
b1	c1
b2	c2

Give the relation that results from each of the following relational algebra expressions on these relations:

a. *R* Div *S*

b. *R* Div (Sel[B != b1](*S*))

c. *R* Div (Sel[B != b2](*S*))

d. *R* × *S*) - (Sel[R.C=S.C](*R* Join[B=B] *S*)

02 Relational Algebra: Composition Answer

d. $R \times S$) - (Sel[R.C=S.C](R Join[B=B] S)

Answer:

Tmp1 = *R* Join[R.B=S.B] *S* =

R.A	R.B	R.C	S.B	S.C
a1	b1	c1	b1	c1
a1	b2	c2	b2	c2
a2	b1	c1	b1	c1

Tmp2 = Sel[R.C=S.C](*Tmp1*) =

R.A	R.B	R.C	S.B	S.C
a1	b1	c1	b1	c1
a1	b2	c2	b2	c2
a2	b1	c1	b1	c1

Tmp3 = $R \times S$ =

R.A	R.B	R.C	S.B	S.C
a1	b1	c1	b1	c1
a1	b1	c1	b2	c2
a1	b2	c2	b1	c1
a1	b2	c2	b2	c2
a2	b1	c1	b1	c1
a2	b1	c1	b2	c2

Tmp3 - Tmp2 =

R.A	R.B	R.C	S.B	S.C
a1	b1	c1	b2	c2
a1	b2	c2	b1	c1
a2	b1	c1	b2	c2

03

Transaction Processing: Serialisability

03 → Transaction Processing: Serialisability

- **Transaction**:
  - A set of atomic (single operation) actions that a user performs e.g. a read or a write
- **Serial execution**:
  - No overlap between 2 or more transactions →
- **Serialisable schedule**:
  - Given some transactions, I can create a schedule such that it would in effect be similar to a serial execution of them.

```
T1: R(X) W(X) R(Y) W(Y)
T2:                    R(X) W(X)
```

or

```
T1:           R(X) W(X) R(Y) W(Y)
T2: R(X) W(X)
```

03 Transaction Processing: Serialisability

To maintain the integrity of data, transactions must be:

- **Atomic**:
  - Each operation is a single, unsplittable unit
  - Either each operation fully succeeds, or fully fails and rolls back fully
- **Consistent**:
  - Each operations takes a valid database and results in a valid database
- **Isolated**:
  - If several operations are performed together, it should be possible to perform them separately to the same effect
- **Durable**:
  - Resistant to data corruption from events such as power loss

03 Transaction Processing: Serialisability

- **Conflict serialisable schedule**:
  - A schedule that is able to execute read and write operations in the "right" order
  - How to check for conflict serialisable?
    - Conflict serialisable if there are **no cycles in precedence graph**

- A operation is **non-conflicting if swapping the order doesn't affect the result**
  - Basically can't swap anything with a write operation

Consider two transactions T_1 and T_2 acting on data item X.

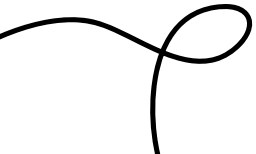
Possible orders for read/write operations by T_1 and T_2:

T_1 first	T_2 first	Equiv?
$R_1(X)\ R_2(X)$	$R_2(X)\ R_1(X)$	yes
$R_1(X)\ W_2(X)$	$W_2(X)\ R_1(X)$	no
$W_1(X)\ R_2(X)$	$R_2(X)\ W_1(X)$	no
$W_1(X)\ W_2(X)$	$W_2(X)\ W_1(X)$	no

If T_1 and T_2 act on different data items, result is always equivalent.

03 → Transaction Processing: Serialisability

- **View serialisable schedule:**
  - A schedule that allows all read operations to “see” the correct version of data
  - See later slides on view equivalence and how we can determine this

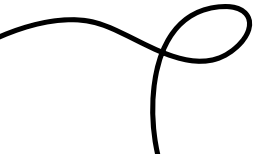


04

Transaction Processing: Precedence Graph

04 Transaction Processing: Precedence Graph

- If something needs to **write** or **read** before something else, then draw an edge
- If there **is a cycle**, then **NOT conflict serialisable**



04 Transaction Processing: Precedence Graph

```
T1:  R(X) W(X)           R(Y)
T2:           R(X) W(Y)
```

T1 -> T2 (draw an edge because T1 has to finish writing X before T2 reads X)

T2 -> T1 (draw an edge because T2 has to finish writing Y before T1 reads Y)

→ There is a cycle → NOT conflict serialisable

04 Transaction Processing: Precedence Graph Question

11. Draw the precedence graph for the following schedule (where C means "commit"):

```
T1:      R(A) W(Z)               C
T2:               R(B) W(Y)        C
T3: W(A)                    W(B)     C
```

12. [based on Ramakrishnan, ex.17.2]
Consider the following incomplete schedule *S*:

```
T1: R(X) R(Y) W(X)           W(X)
T2:               R(Y)           R(Y)
T3:                    W(Y)
```

a. Determine (by using a precedence graph) whether the schedule is conflict-serializable

b. Modify *S* to create a complete schedule that is conflict-serializable

05

Transaction Processing: View Equivalence

05 Transaction Processing: View Equivalence

View equivalence properties checklist:

1. If a transaction performs the first read on the original, it must also perform the first read on the serialised schedule
2. If a transaction performs the final write on the original, it must also perform the final write on the serialised schedule
3. Any **Read after Write** must be the same in both schedules

How to check for view serialisable?

- **Check all possible serial schedules** if they are view equivalent
- If every possibility not view equivalent then not view serialisable
- A conflict serialisable schedule is also view serialisable

05 Transaction Processing: View Equivalence Question

13. [based on Ramakrishnan, ex.17.3]

For each of the following schedules, state whether it is conflict-serializable and/or view-serializable. If you cannot decide whether a schedule belongs to either class, explain briefly. The actions are listed in the order they are scheduled, and prefixed with the transaction name.

a. `T1:R(X) T2:R(X) T1:W(X) T2:W(X)`

b. `T1:W(X) T2:R(Y) T1:R(Y) T2:R(X)`

c. `T1:R(X) T2:R(Y) T3:W(X) T2:R(X) T1:R(Y)`

d. `T1:R(X) T1:R(Y) T1:W(X) T2:R(Y) T3:W(Y) T1:W(X) T2:R(Y)`

e. `T1:R(X) T2:W(X) T1:W(X) T3:W(X)`

AND THAT’S IT!

All the best for your final exam!!!

Hope to see you all around sometime 👋