

# JULY 17

## DATABASES:

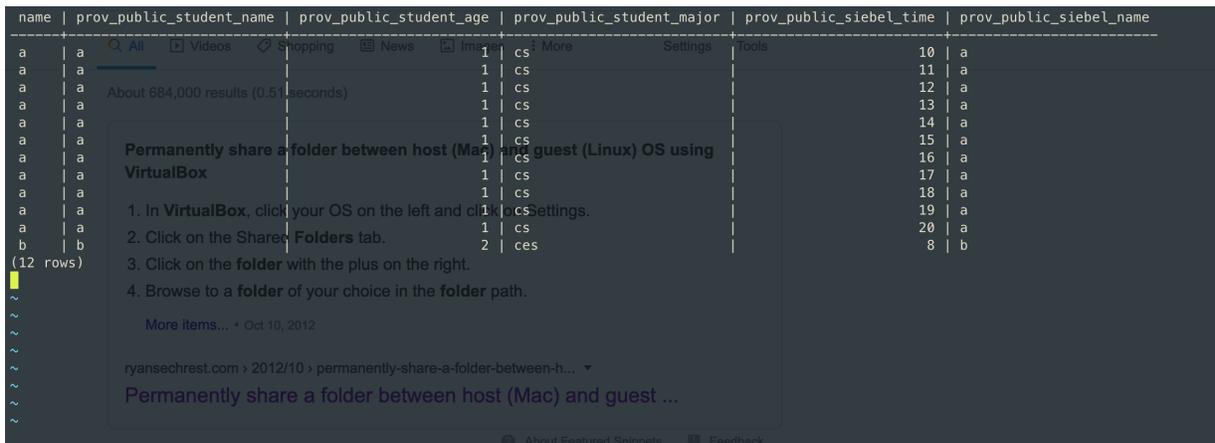
```
student (  
    name    varchar,  
    age     integer,  
    major   car char  
)
```

```
siebel (  
    name    archer,  
    time    integer  
)
```

## QUERY TYPES:

### 1. SELECT

```
select provenance s.name  
from student as s join siebel as b  
on s.name = b.name;
```



name	prov_public_student_name	prov_public_student_age	prov_public_student_major	prov_public_siebel_time	prov_public_siebel_name
a	a	1	cs	10	a
a	a	1	cs	11	a
a	a	1	cs	12	a
a	a	1	cs	13	a
a	a	1	cs	14	a
a	a	1	cs	15	a
a	a	1	cs	16	a
a	a	1	cs	17	a
a	a	1	cs	18	a
a	a	1	cs	19	a
a	a	1	cs	20	a
b	b	2	ces	8	b

### 2. SELECT WITH GROUP BY

```
select provenance count(s.name)  
from student as s join siebel as b  
on s.name = b.name  
group by s.major
```

```

count | prov_public_student_name | prov_public_student_age | prov_public_student_major | prov_public_siebel_time | prov_public_siebel_name | 2020-
11 | a | 10 | cs | 10 | a
11 | a | 10 | cs | 11 | a
11 | a | 10 | cs | 12 | a
11 | a | 10 | cs | 13 | a
11 | a | 10 | cs | 14 | a
11 | a | 10 | cs | 15 | a
11 | a | 10 | cs | 16 | a
11 | a | 10 | cs | 17 | a
11 | a | 10 | cs | 18 | a
11 | a | 10 | cs | 19 | a
11 | a | 10 | cs | 20 | a
1 | b | 8 | b
(12 rows)

```

3. SELECT WITH >  
 select provenance s.name  
 from student as s join siebel as b  
 on s.name = b.name and b.time > 18  
 group by s.major

```

name | prov_public_student_name | prov_public_student_age | prov_public_student_major | prov_public_siebel_time | prov_public_siebel_name | 2020-
a | a | 10 | cs | 20 | a
a | a | 10 | cs | 19 | a
(2 rows)

```

Found ways to switch between modes after reading the original paper for PERM.

LINK: <http://cs.iit.edu/%7edbgroup/assets/pdfpubls/G10a.pdf>

From page 140, section 5.1 SQLPLE.

## 5.1.2 Provenance Computation

A user can request provenance computation for an SQL query  $q$  in *SQL-PLE* by adding the keyword *PROVENANCE* to the select clause of  $q$ . A query  $q$  that is marked by this keyword is substituted with a query that computes the relational representation of the provenance of  $q$ . For example:

```
SELECT PROVENANCE name FROM shops ;
```

The default is to compute provenance according to *PI-CS*. If another *data* provenance *CS* type is desired the optional *ON CONTRIBUTION* parameter can be used to indicate this. For instance, if the *CDC-CS* provenance of the example query presented above should be computed this is expressed in *SQL-PLE* as:

```
SELECT PROVENANCE ON CONTRIBUTION (COPY COMPLETE NONTRANSITIVE) name
FROM shops ;
```

There are different modes:

CS Category	Abbreviation	Description
Input-CS	IN-CS	The provenance of a data item $d$ includes all input data items of the transformation(s) that produced $d$ .
Copy-CS	C-CS	The provenance of a data item $d$ includes data items which have been copied (partially or completely) to $d$ .
Influence-CS	I-CS	The provenance of a data item $d$ includes all data items that have influenced the creation of $d$ in some way.

Figure 2.2: Data Provenance Contribution Semantics Categories

Influence and Copy tested. Input seems not working.

Influence mode will collect all tuples which contributes to the result tuple.

Copy mode has multiple sub-modes.

CS type	Description
<b>Complete-Direct-Copy-CS (CDC-CS)</b>	Only tuples that have been copied directly as a whole from the input to the output of a query are considered to belong to the provenance.
<b>Partial-Direct-Copy-CS (PDC-CS)</b>	Only tuples from which at least one attribute value has been copied directly from the input to the output belong to the provenance.
<b>Complete-Transitive-Copy-CS (CTC-CS)</b>	<i>CTC-CS</i> contains all directly copied tuples. In addition, implied equalities enforced by selection conditions are handled as copy operations.
<b>Partial-Transitive-Copy-CS (PTC-CS)</b>	Like <i>PDC-CS</i> , but implied equalities are considered as copy operations.

Figure 3.12: C-CS types

We could see that influence mode is kind of like a sub-mode of Copy mode.

### 2.1.1 Data Provenance

*Data provenance* is information about the input data items that were used to create (*contributed* to) an output data item *d*. *Data provenance* approaches can be classified according to which input data items are considered to belong to the provenance of *d*, which we refer to as *contribution semantics* (*CS*). Several definitions of *contribution semantics* have been proposed in the literature. Here we present a higher level classification of *CS* types and postpone the discussion of concrete definitions to section 2.3. We consider a *CS* definition to belong to the class of *input-CS* (*IN-CS*) if it considers all input data items of a transformation to belong to the provenance of an output of this transformation. Instances of this class of *CS* are easy to define and compute, because no knowledge about the inner workings of a transformation is required. *IN-CS* is used by provenance-aware workflow systems and in grid-computing approaches where transformations are modeled as black boxes. For most use cases *IN-CS* does not provide enough detail to be useful in practice, but for some application scenarios it is the only type of *CS* for which automatic provenance computation can be provided. Under *copy-CS* (*C-CS*) a data item is considered to belong to the provenance of an output data item, if it has been copied literally from the input to the output (we do not specify if complete or just partial copying is required). Thus, the provenance of a data item *d* includes the data items that contributed values to *d*. *C-CS* class contribution semantics are generally used with fine-granular data items and allow to trace back the origin of values in the result of a transformation. This class of *CS* is not well suited for transformations that generate output data items without copying input values (e.g., aggregation of values). An extension to *C-CS* is *influence-CS* (*I-CS*). Contribution semantics belonging to this class include all data items in the provenance of an output data item *d*, that had some influence on the creation of *d*. For instance, data items that were used in a filter condition in the transformation that produced *d*. In

Definitions for these three mode.

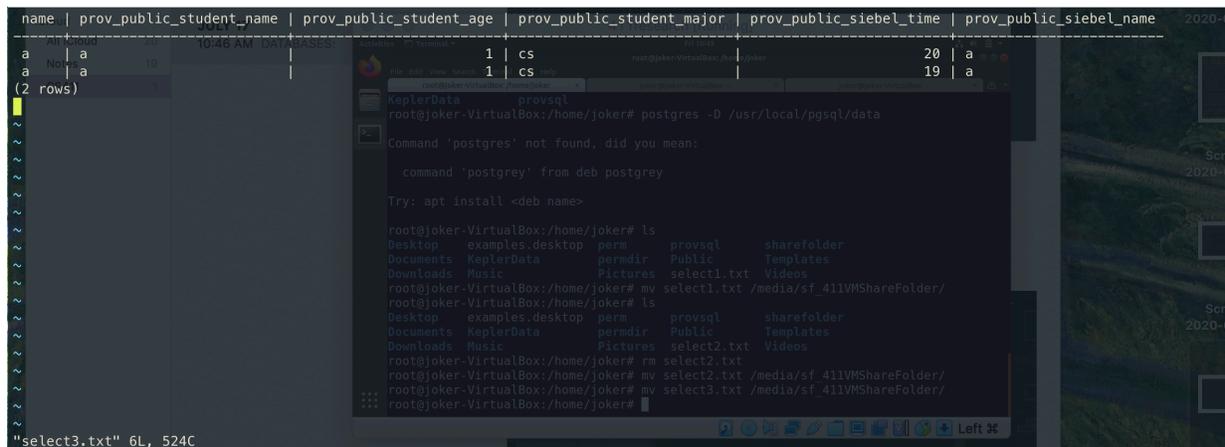
Input on line 6

Copy on line 12

Influence mode on line 3 from bottom to top.

Now try the third query with different modes.

Influence: (default)



Copy:

copy (default is copy complete nontransitive)

```

name | prov_public_student_name | prov_public_student_age | prov_public_student_major | prov_public_siebel_time | prov_public_siebel_name
-----
a | a | a | a | a | a
a | a | a | a | a | a
(2 rows)

root@joker-VirtualBox:/home/joker# cat select4.txt
name | prov_public_student_name | prov_public_student_age | prov_public_student_major | prov_public_siebel_time | prov_public_siebel_name
-----
a | a | a | a | a | a
a | a | a | a | a | a
(2 rows)

root@joker-VirtualBox:/home/joker# mv copy_partial.txt /media/sf_411VMSHareFolder/
root@joker-VirtualBox:/home/joker# mv copy.txt /media/sf_411VMSHareFolder/
root@joker-VirtualBox:/home/joker#

```

Only select s.name, thus no tuple is completely copied, thus none provenance showed.

copy partial transitive/nontransitive

```

name | prov_public_student_name | prov_public_student_age | prov_public_student_major | prov_public_siebel_time | prov_public_siebel_name
-----
a | a | a | a | a | a
a | a | a | a | a | a
(2 rows)

root@joker-VirtualBox:/home/joker# cat select4.txt
name | prov_public_student_name | prov_public_student_age | prov_public_student_major | prov_public_siebel_time | prov_public_siebel_name
-----
a | a | a | a | a | a
a | a | a | a | a | a
(2 rows)

root@joker-VirtualBox:/home/joker# mv copy_partial.txt /media/sf_411VMSHareFolder/
root@joker-VirtualBox:/home/joker#

```

Use partial, and student.name is copied, thus student's provenance is shown.

Copy + partial/complete/none + transitive/none(nontransitive)

Below is operations supported.

5.1.2 Provenance Computation

A user can request provenance computation for *PROVENANCE* to the select clause of *q*. A query that computes the relational representation

`SELECT PROVENANCE`

The default is to compute provenance according to the optional *ON CONTRIBUTION* parameter can be used to force provenance of the example query presented above

`SELECT PROVENANCE ON CONTRIBUTION FROM shops;`

Construct	Description
<code>PROVENANCE</code>	Marks a query for provenance computation.
<code>ON CONTRIBUTION</code> ( <i>cs_type</i> )	Instructs <i>Perm</i> to use a certain <i>CS</i> type.
<code>BASERELATION</code>	Handle a <i>FROM</i> clause item as if it were a base relation.
<code>PROVENANCE</code> ( <i>attr_list</i> )	Handle attributes from <i>attr_list</i> as provenance attributes.
<code>TRANSPROV/TRANSSQL/TRANSXML</code>	Mark a query for transformation provenance computation.
<code>EXPLAIN SQLTEXT</code>	Return the (rewritten) SQL text of a query.
<code>EXPLAIN GRAPH</code>	Return an algebra tree for a query (as a dot-language script).

Figure 5.2: SQL-PLE language constructs

To do:

- Support for different types of provenance:
  - **Why-Provenance:** Perm-Influence Contribution Semantics (PI-CS)
  - **Where-Provenance:** Copy Contribution Semantics (C-CS), Where-Provenance (Buneman et al.)
  - **How-Provenance:** Provenance Polynomials
  - **Transformation Provenance**

Find information about how to switch to How Provenance in <http://cs.iit.edu/%7edbgrouppubls/G10a.pdf>. Tried but failed.

Need database and students' queries from homework to study more complex queries.