Correlated subqueries

```
select *
from overlap_sizes as ov1
where not exists
   (select *
    from overlap_sizes as ov2
    where ov2.count > ov1.count);
```

Results of inner select

SW104	3
UA101	16

ov1

Flightid	Count
SW118	2
UA128	1
UA117	1
SW104	3
SW116	2
UA101	16

ov2

Flightid	Count
SW118	2
UA128	1
UA117	1
SW104	3
SW116	2
UA101	16

Correlated subqueries

```
select *
from overlap_sizes as ov1
where not exists
   (select *
    from overlap_sizes as ov2
    where ov2.count > ov1.count);
```

ov1

Flightid	Count
SW118	2
UA128	1
UA117	1
SW104	3
SW116	2
UA101	16

ov2

Flightid	Count
SW118	2
UA128	1
UA117	1
SW104	3
SW116	2
UA101	16

Correlated subqueries

```
select *
from overlap_sizes as ov1
where not exists
   (select *
    from overlap_sizes as ov2
    where ov2.count > ov1.count);
```

Results of inner select

SW118	2
SW104	3
SW116	2
UA101	16

ov1

Flightid	Count
SW118	2
UA128	1
UA117	1
SW104	3
SW116	2
UA101	16

ov2

Flightid	Count
SW118	2
UA128	1
UA117	1
SW104	3
SW116	2
UA101	16

Correlated subqueries

```
select *
from overlap_sizes as ov1
where not exists
   (select *
    from overlap_sizes as ov2
    where ov2.count > ov1.count);
```

Results of inner select

ov1

Flightid	Count
SW118	2
UA128	1
UA117	1
SW104	3
SW116	2
UA101	16

ov2

Flightid	Count
SW118	2
UA128	1
UA117	1
SW104	3
SW116	2
UA101	16

Views

- Relation that is not defined in conceptual model but made visible to user as “virtual relation”
- Three main uses:
  - Expression in ‘with clause’ that you want to reuse across queries
  - Backwards compatibility
    - DBA wants to change the schema, but some applications are still expecting the old schema
  - Security/authorization
    - Don’t want to give user access to entire table, just subset
      - E.g. Create a view of customer table without social security number and credit card information

View Definition

- Defined using the **create view** statement:

  create view *v* **as** < query expression >

  create view *v* <*colnames*> **as** < query expression >

  <query expression> is any legal SQL expression. *v is the view name.*

- Once a view is defined, the view name can be used to refer to the virtual relation that the view generates.
- View definition is not the same as creating a new relation by evaluating the query expression
  - Rather, just expression is saved and substituted into queries using the view.
  - But sometimes DB will create a “materialized view” which does save the results in addition to the expression

Example Views

- ```
  create view cust_list(customerid) as
     (select customerid from flewon
      where flightid = 'DL119');
  ```

- ```
  create view flewon_overlap as
     (select * from flewon
      where customerid IN (select * from cust_list));
  ```

(note that flewon_overlap is defined using another view)

View Expansion

```
create view cust_list(customerid) as
   (select customerid from flewon
    where flightid = 'DL119');
```

```
create view flewon_overlap as
   (select * from flewon
    where customerid IN
          (select * from cust_list));
```

- ```
  select * from flewon_overlap;
  ```
- ```
  select * from
    (select * from flewon
      where customerid IN
            (select * from cust_list));
  ```
- ```
   select * from
    (select * from flewon
      where customerid IN
            (select * from (select customerid
                            from flewon
                            where flightid = 'DL119')));
  ```
  ```

Update of a View

- `insert into cust_list values ('cust263');`
  - View does not exist independently, so system will try to insert into the relations from which this view is derived
  - In this case, this insert fails because of a foreign-key / primary key dependency in flewon (cust263 doesn't exist in customers)

- `insert into cust_list values ('cust63');`
  - In this case, the insert succeeds, but nulls are inserted for the other fields of flewon

- Most views defined in terms of joins of multiple tables or aggregations are extremely different for the DB to figure out how to convert the update of the view to the update of the original relations
  - Therefore updates to these views usually return an error

Views vs. Creating a new table

	Views	Creating a new table
Creating	Create view V as (select * from A, B where …)	Create table T as (select * from A, B where …)
Can be used	In any select query. Only some update queries.	It's a new table. You can do what you want.
Maintained as	1. Materialized view: Evaluate the query and store it on disk as if a table. 2. Normal view: Don't store. Substitute in queries when referenced.	It's a new table. Stored on disk.
What if a tuple inserted in A ?	1. Materialized view: the stored table is automatically updated to be accurate. 2. Normal view: there is no need to do anything.	T is a separate table; there is no reason why DBMS should keep it updated. If you want that, you must define a trigger (see later slides).