Views

- In some cases, it is not desirable for all users to see the entire logical model (i.e, all the actual relations stored in the database.)
- Consider a person who needs to know a customer's loan number but has no need to see the loan amount. This person should see a relation described, in SQL, by

  (**select** *customer_name*, *loan_number*
  from *borrower*, *loan*
  where *borrower.loan_number = loan.loan_number*)
- A **view** provides a mechanism to hide certain data from the view of certain users.
- Any relation that is not of the conceptual model but is made visible to a user as a "virtual relation" is called a **view**.

View Definition

- A view is defined using the **create view** statement which has the form

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

  where <query expression> is any legal SQL expression. The view name is represented by *v*.
- 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, a view definition causes the saving of an expression; the expression is substituted into queries using the view.

Views in SQL

- A view is a “virtual” table that is derived from other tables
- Allows for limited update operations (since the table may not physically be stored)
- Allows full query operations
- A convenience for expressing certain operations

Specification of Views

- SQL command: `CREATE VIEW`
  - a table (view) name
  - a possible list of attribute names (for example, when arithmetic operations are specified or when we want the names to be different from the attributes in the base relations)
  - a query to specify the table contents

Example Queries

- A view consisting of branches and their customers

```
create view all_customer as
        (select branch_name, customer_name
         from depositor, account
        where depositor.account_number =
                account.account_number )
        union
        (select branch_name, customer_name
        from borrower, loan
        where borrower.loan_number = loan.loan_number )
```

- Find all customers of the Perryridge branch

```
select customer_name
from all_customer
where branch_name = 'Perryridge'
```

Views Defined Using Other Views

- One view may be used in the expression defining another view
- A view relation v_1 is said to *depend directly* on a view relation v_2 if v_2 is used in the expression defining v_1
- A view relation v_1 is said to *depend on* view relation v_2 if either v_1 depends directly to v_2 or there is a path of dependencies from v_1 to v_2
- A view relation v is said to be *recursive* if it depends on itself.

SQL Views: Another Example

- Specify a different WORKS_ON table

```
CREATE TABLE WORKS_ON_NEW AS
SELECT      FNAME, LNAME, PNAME, HOURS
FROM        EMPLOYEE, PROJECT, WORKS_ON
WHERE       SSN=ESSN AND PNO=PNUMBER
GROUP BY    PNAME;
```

Using a Virtual Table

- We can specify SQL queries on a newly create view:

```
SELECT    FNAME, LNAME
FROM      WORKS_ON_NEW
WHERE     PNAME='Seena';
```

- When no longer needed, a view can be dropped:

```
DROP WORKS_ON_NEW;
```

Efficient View Implementation

- Query modification: present the view query in terms of a query on the underlying base tables
  - disadvantage: inefficient for views defined via complex queries (especially if additional queries are to be applied to the view within a short time period)

Efficient View Implementation

- View materialization: involves physically creating and keeping a temporary table
  - assumption: other queries on the view will follow
  - concerns: maintaining correspondence between the base table and the view when the base table is updated
  - strategy: incremental update

Update of a View

- Create a view of all loan data in the *loan* relation, hiding the *amount* attribute

  create view *branch_loan* **as**
  select *branch_name*, *loan_number*
  from *loan*

- Add a new tuple to *branch_loan*

  insert into *branch_loan*
  values (‘Perryridge’, ‘L-307’)

  This insertion must be represented by the insertion of the tuple

  (‘L-307’, ‘Perryridge’, *null*)

  into the *loan* relation

View Update

- Update on a single view without aggregate operations: update may map to an update on the underlying base table
- Views involving joins: an update *may* map to an update on the underlying base relations
  - not always possible

Updates Through Views (Cont.)

- Some updates through views are impossible to translate into updates on the database relations
  - **create view** *v* **as**
    select *branch_name* **from** *account*

    insert into *v* **values (**‘*L-99*’, ‘ Downtown’, ‘*23*’**)**
- Others cannot be translated uniquely
  - **insert into** *all_customer* **values** *(*‘ Perryridge’, ‘John’**)**
    - Have to choose loan or account, and create a new loan/account number!
- Most SQL implementations allow updates only on simple views (without aggregates) defined on a single relation

Un-updatable Views

- Views defined using groups and aggregate functions are not updateable
- Views defined on multiple tables using joins are generally not updateable
- WITH CHECK OPTION: must be added to the definition of a view if the view is to be updated
  - to allow check for updatability and to plan for an execution strategy