So far: Querying with SQL

- Lots more functionality within SQL to support even more advanced querying
  - You can call functions in programming languages all within a SELECT statement
  - You can use "WINDOW" functions that allow you to perform computation across groups, e.g., capture a running/cumulative total
  - You can define special shortcuts for relations via Common Table Expressions (CTEs)
- We won't cover any of this for now…

Recap

- Data-savviness is the future!
- Notion of a DBMS
- The relational data model and algebra: bags and sets
- SQL Queries:
    - SFW and its semantics
    - LIKE, AS, *, %, …
    - Null values
    - Single and multiple relations
    - Subqueries
    - Bag algebra and set operations
    - Grouping
    - Ordering
    - Case
- Next: SQL Modifications

It's not sufficient to query…

- We also need to be able to **modify** the data in relations
- A modification command doesn't return the result in the same way a query does, but it changes the contents of the database

- Three kinds of modifications:
  - *Insert* a tuple or tuples
  - *Delete* an existing tuple (or tuples)
  - *Update* the value(s) of an existing tuple (or tuples)

Inserting One or More Tuples

- INSERT INTO Relation VALUES (<list of values>);

- Example:
  - Actor (actor_id, first_name, last_name, last_update)
  - INSERT INTO Actor VALUES (5000, 'Adam', 'Driver', 'Feb 3, 2020');
  - INSERT INTO Actor VALUES (5001, 'Brenda', 'Driver', 'Feb 3, 2020'), (5002, 'Charles', 'Driver', 'Feb 3, 2020');

Specifying Attributes in Insert

- INSERT INTO Actor VALUES (5001, 'Brenda', 'Driver', 'Feb 3, 2020'), (5002, 'Charles', 'Driver', 'Feb 3, 2020');
- The previous syntax requires you to remember order of attributes

- Can be more explicit:
  - INSERT INTO Actor (actor_id, first_name, last_name, last_update) VALUES (5003, 'Didi', 'Driver', 'Feb 3, 2020');
- Another benefit beyond remembering the order of attributes:
  - We don't have values for all attributes, and we want the system to fill in NULL values for the rest

Inserting Many Tuples

- We may insert the entire result of a query into a relation
- INSERT INTO Relation (<Subquery>);

- Actor (actor_id, first_name, last_name, last_update)
- INSERT INTO Actor (actor_id) (SELECT * FROM Film_Actor)
- Q: What do you think this query does?

Inserting Many Tuples

- We may insert the entire result of a query into a relation
- INSERT INTO Relation (<Subquery>);

- Actor (actor_id, first_name, last_name, last_update)
- INSERT INTO Actor (actor_id) (SELECT * FROM Film_Actor)
- Q: What do you think this query does?
- A: inserts all the actor_ids from the Film_Actor relation into the Actor relation

Deleting Tuples

- To delete tuples satisfying a condition from some relation:
- DELETE FROM Relation WHERE <condition>;

- Actor (actor_id, first_name, last_name, last_update)
- Q: To delete all the many "Driver"s — Adam, Brenda, Charles, and Didis from our Actor relation, what query would we use?

Deleting Tuples

- To delete tuples satisfying a condition from some relation:
- DELETE FROM Relation WHERE <condition>;

- Actor (actor_id, first_name, last_name, last_update)
- Q: To delete all the many "Driver"s — Adam, Brenda, Charles, and Didis from our Actor relation, what query would we use?
- DELETE FROM Actor WHERE last_name = 'Driver';

- To delete all tuples from a relation simply say:
- DELETE FROM Actor;

Updates

- To change certain attributes of certain tuples in a relation:

  UPDATE Relation

  SET <list of attribute assignments>

  WHERE <condition>

Example: Update

- Actor (actor_id, first_name, last_name, last_update)
- Change Adam Driver's actor_id to 6000:

```
UPDATE Actor
SET actor_id = 6000
WHERE first_name = 'Adam' AND last_name = 'Driver'
```


A Tricky Issue of Semantics

- Now, we know at a high level how INSERTS, DELETES, and UPDATES work
- There's a special issue that impacts both DELETEs and UPDATEs
- Suppose we have a relation Employee (Name, Salary, Address, Title)
- We run the following update to increase salaries of all database designers who have a Salary < 100k by 20%
  - Q: How would we do this?

A Tricky Issue of Semantics

- Now, we know at a high level how INSERTS, DELETES, and UPDATES work
- There's a special issue that impacts both DELETEs and UPDATEs
- Suppose we have a relation Employee (Name, Salary, Address, Title)
- We run the following update to increase salaries of all database designers who have a Salary < 100k by 20%

```
UPDATE Employee SET Salary = Salary * 1.2
WHERE Title= 'Database Designer' AND Salary <100000
```

A Tricky Issue of Semantics

```
UPDATE Employee SET Salary = Salary * 1.2
WHERE Title = 'Database Designer' AND Salary <100000
```

- Suppose the Employee relation has a tuple corresponding to a Database Designer "Bob" with Salary <100000.
- When the UPDATE statement hits Bob's tuple, it changes that Salary to 120% of the original one.
- If we're not careful, we could keep increasing Bob's Salary…
- Q: when will we stop?

A Tricky Issue of Semantics

```
UPDATE Employee SET Salary = Salary * 1.2
WHERE Title = 'Database Designer' AND Salary <100000
```

- Suppose the Employee relation has a tuple corresponding to a Database Designer "Bob" with Salary <100000.
- When the UPDATE statement hits Bob's tuple, it changes that Salary to 120% of the original one.
- If we're not careful, we could keep increasing Bob's Salary…
- Q: when will we stop?
- We will stop when the Salary hits 100000 or above.

A Tricky Issue of Semantics

UPDATE Employee SET Salary = Salary * 1.2

WHERE Title = 'Database Designer' AND Salary <100000

- How do we deal with this issue?
- We do UPDATES (and DELETES) in two phases:
  - Phase 1:
    - We look at all the tuples and evaluate their WHERE condition
    - If the tuple matches the WHERE condition, we mark it for an update or delete
  - Phase 2:
    - We actually update/delete all the marked tuples
- Homework: Convince yourselves that this always returns the same answer independent of the order of processing of tuples

Quick Demo

- Insert into the Actor relation

  INSERT INTO Actor VALUES (5000, 'Adam', 'Driver', 'Feb 3, 2020');

  INSERT INTO Actor VALUES (5001, 'Brenda', 'Driver', 'Feb 3, 2020'), (5002, 'Charles', 'Driver', 'Feb 3, 2020');

  INSERT INTO Actor (actor_id, first_name, last_name, last_update) VALUES (5003, 'Didi', 'Driver', 'Feb 3, 2020');

- Updates to the Actor relation

  UPDATE Actor

  SET actor_id = 6000

  WHERE first_name = 'Adam' AND last_name = 'Driver'

- Deletes from the Actor relation

  DELETE FROM Actor WHERE last_name = 'Driver';

Recap

- Data-savviness is the future!
- Notion of a DBMS
- The relational data model and algebra: bags and sets
- SQL Queries:
    - SFW and its semantics
    - LIKE, AS, *, %, …
    - Null values
    - Single and multiple relations
    - Subqueries
    - Bag algebra and set operations
    - Grouping
    - Ordering
    - Case
- SQL Modifications