Recap

- Data-savviness is the future!
- Notion of a DBMS
- The relational data model and algebra: bags and sets
- SQL:
  - SFW and its semantics
  - LIKE, AS, *, %, …
  - Null values
  - Single and multiple relations
  - **Next: subqueries**

Subqueries

- A parenthesized SELECT-FROM-WHERE statement (a *subquery*) can be used as a value in various places, including FROM and WHERE clauses
- First version: if a subquery returns a single tuple with a single attribute value, it can be treated as a scalar in expressions
  - Runtime error if used incorrectly
- Second version: checking if a query is non-empty via EXISTS
- *Other versions: IN, ANY, ALL: homework!*

Subquery as a Scalar

- Length of the movie African Egg: 130m
  - SELECT length
  - FROM Film
  - WHERE title = 'African Egg’
- Find movies that are longer:
  - SELECT * FROM Film WHERE length >= (SELECT length FROM Film WHERE title = 'African Egg');

Subquery as a Set: EXISTS

- EXISTS <relation> is true iff <relation> is not empty
    - NOT EXISTS is opposite
- Can appear in WHERE clauses
- Inventory (inventory_id, film_id, store_id, last_update)
    - SELECT * FROM Inventory;
- Rental (rental_id, rental_date, inventory_id, customer_id, return_date, staff_id, last_update)
- Find items in the inventory that have not been rented before:
    - SELECT Inventory.inventory_id FROM Inventory
    - WHERE NOT EXISTS
        - (SELECT * FROM Rental WHERE Inventory.inventory_id = Rental.inventory_id)
    - **Notice scoping rules!**

Recap

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

Recall: Operations on Bags

- Selection: preserve # of occurrences
- Projection: preserve # of occurrences (duplicates not removed)
- Cartesian product, join: preserve # of occurrences
- Union {a, b, b, c} U {a, b, c, d} = {a, a, b, b, b, c, c, d}
- Difference {a, a, a, b, c} — {a, b, b} = {a, a, c}

SFW: Bag Semantics

- Select-From-Where uses Bag semantics
  - Select: preserve number of occurrences [projection]
  - From: preserve number of occurrences [cross-product]
  - Where: preserve number of occurrences [selection]

Set Operations: Set (Not Bag) Semantics

- Union, difference, intersection are expressed as follows:
  - (subquery) UNION (subquery)
  - (subquery) EXCEPT (subquery)
  - (subquery) INTERSECT (subquery)
- Find items in the inventory that have not been rented before:
  - (SELECT inventory_id FROM Inventory)
  - EXCEPT
  - (SELECT inventory_id FROM Rental)
- **Set operations use set semantics by default**
  - Thus, duplicates are eliminated at the end of the operation

Why this weird mix of sets and bags?

- When doing projection in relational algebra, it is easier to avoid eliminating duplicates
  - Just work tuple-at-a-time
  - So, use bag semantics for SFW
- When doing intersection or difference, it is most efficient to sort the relations first.
  - At this point, might as well eliminate the duplicates
  - Even though union can be done simpler with bags, it is lumped together intersection and difference since it is also set-oriented

Forcing Different Behavior

- Force result to be a set by
  - SELECT DISTINCT release_year FROM Film;
  - Contrast with SELECT release_year FROM Film;

- Force result to be a bag using ALL
  - … UNION ALL …