TA Section

April 8, 2010

Modifying the Database

Three kinds of modifications

- Insertions
- Deletions
- Updates

Sometimes they are all called “updates”

Insertions

General form:

```
INSERT INTO R(A1,…., An) VALUES (v1,…., vn)
```

Insertions

Product(name, listPrice, category)
Purchase(buyer, seller, product, price)

Example: Insert a new purchase to the database:

```
INSERT INTO Purchase(buyer, seller, product, price)
       VALUES ('Joe', 'Fred', 'wakeup-clock-espresso-machine',
                              'The Sharper Image')
```

Missing attribute → NULL.
May drop attribute names if give them in order.

Insertions

```
INSERT   INTO   PRODUCT(name)

    SELECT  DISTINCT  Purchase.product
    FROM      Purchase
    WHERE   Purchase.date > “10/26/01”
```

The query replaces the VALUES keyword.
Here we insert *many* tuples into PRODUCT

Deletions

Example:

```
DELETE    FROM    PURCHASE

WHERE    seller = 'Joe'   AND
                  product = 'Brooklyn Bridge'
```

Updates

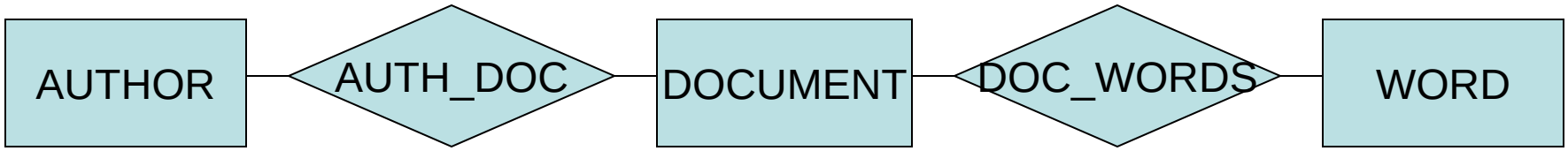
Example:

```
UPDATE   PRODUCT
SET    price = price/2
WHERE  Product.name  IN
                (SELECT product
                  FROM    Purchase
                  WHERE  Date ='Oct, 25, 1999');
```

Aggregate Queries

- Our Schema

  AUTHOR(aid, name)
  AUTH_DOC(aid, did)
  DOCUMENT (did, title)
  DOC_WORD (did, word)
  WORD(word)



- Find authors who wrote more than 20 docs

```
SELECT name FROM AUTHOR a
  WHERE(SELECT COUNT(*) FROM
AUTH_DOC ad WHERE ad.aid = a.aid) > 20

SELECT name FROM AUTHOR a, AUTH_DOC
ad WHERE a.aid = ad.aid GROUP BY a.aid,
a.name HAVING COUNT(*) > 20
```

- Find authors who have a vocabulary of more than 10,000 words

  SELECT name FROM AUTHOR WHERE (SELECT COUNT(DISTINCT word) FROM …) > 10000

  SELECT name FROM AUTHOR a, AUTH_DOC ad, DOC_WORDS dw WHERE a.aid = ad.aid AND ad.did = dw.did GROUP BY a.aid, a.name HAVING COUNT(DISTINCT word) > 10000

- Find authors who have written a total 10,000 words

  (same queries as on previous slide, but drop keyword DISTINCT)

- For each author, report the total number of words

```
SELECT aid, COUNT(*) num
FROM AUTHOR a, AUTH_DOC ad,
     DOC_WORDS dw
WHERE a.aid = ad.aid AND
     ad.did = dw.did
GROUP BY aid.
```

- For each author, report average number of words per paper.

```
SELECT aid, AVG(num) FROM (
    SELECT aid, did, COUNT(*) num
    FROM AUTHOR a, AUTH_DOC ad,
            DOC_WORDS dw
    WHERE …
    GROUP BY aid, did) t
GROUP BY aid
```

- Find author with highest average number of words per paper

  SELECT …
  WHERE NOT EXISTS (…)

- Find words used by at least 10 authors

```
SELECT word
FROM DOC_WORDS
    NATURAL JOIN AUTH_DOC
GROUP BY word
HAVING COUNT(DISTINCT aid) >= 10
```

- Find most frequently used word

  SELECT word FROM DOC_WORDS
  GROUP BY word
  HAVING (COUNT(*)) >= ALL(…)

- Find the largest document

  SELECT did FROM DOC_WORDS
  GROUP BY did
  HAVING COUNT(*) >= ALL(…)

  or

  HAVING NOT EXISTS(…)

- Find authors who have written the largest document

```
SELECT name
FROM author a
WHERE (
      SELECT COUNT(word)
      FROM DOC_WORDS dw, AUTH_DOC ad
      WHERE dw.did = ad.did AND ad.aid = a.aid)
= (SELECT …)
```

Existential and Universial Quantifiers

- Our Schema

  LIKES(drinker, beer)
  FREQUENTS(drinker, bar)
  SERVES(bar, beer)

- Find all drinkers that like some beer that is not served by the bar “Black Cat”

```
SELECT l.drinker
FROM LIKES l
WHERE l.beer NOT IN (
    SELECT s.beer FROM SERVES s
    WHERE S.bar = “Black Cat”)
```

- Find drinkers that frequent some bar that serves some beer they like

```
SELECT f.drinker
FROM FREQUENTS f, LIKES l, SERVES s
WHERE l.drinker = f.drinker
AND l.beer = s.beer AND s.bar = f.bar

SELECT f.drinker
FROM FREQUENT f
WHERE f.bar IN (
      SELECT bar FROM SERVES
      WHERE (drinker, beer) in LIKES)
```

- Find drinkers that frequent only bars that serves some beer they like

```
SELECT drinker
FROM FREQUENTS f
WHERE NOT EXISTS(
        SELECT beer FROM SERVES s
        WHERE s.bar = f.bar AND
        NOT EXISTS(
                SELECT drinker
                FROM LIKES l
                WHERE l.drinker = f.drinker
                AND l.beer = s.beer)
```

- Find drinkers that frequent some bar that serves only beers they like.

```
SELECT f.drinker
FROM FREQUENTS f
WHERE EXISTS(
        SELECT beer FROM SERVES s
        WHERE s.bar = f.bar AND
        NOT EXISTS (
                SELECT beer FROM Serves s2
                WHERE s2.bar = s.bar AND beer NOT IN (
                        SELECT beer FROM Likes WHERE
                                Likes.drinker = f.drinker)))
```

Can you improve this one?

- Find drinkers that frequent only bars that serve some beer they like

```
SELECT drinker
FROM FREQUENTS f
WHERE NOT EXISTS (
      SELECT beer FROM SERVES s
      WHERE s.bar = f.bar AND beer
      NOT IN(
            SELECT beer FROM Likes l
            WHERE l.drinker = f.drinker))
```