Introduction to Database Systems

CSE 444

Lecture #4 Jan 17 2001

Announcements – I

 Special Lecture
 △ At Sieg 134 on Friday January 19th from 330-450PM

 △ Topic: Building SQL Applications
 ○ Important For
 ○ Programming Assignment
 ○ Course Project

Announcement II

೫ Mid Term ⊠In Class ⊠All material except Transactions

SQL (Contd.)

Reading: Sec 5 (except 5.10) Sec 7 (except 7.2 – to be covered later)

Views

#A view is just a relation, but we store a definition (query), rather than a set of tuples.

⊠Can rename columns

CREATE VIEW YoungActiveStudents (Yname, Ygrade) AS SELECT S.name, E.grade FROM Students S, Enrolled E WHERE S.sid = E.sid and S.age<21

* Views can be dropped using the DROP VIEW command.

Uses for Views

%Views can be used to present necessary information (or a summary), while hiding details in underlying relation(s) (security).

Views also useful for maintaining logical data independence when the conceptual schema changes.

6

Views vs. Relations

#Logical distinctions:

□<u>Updates not always possible</u> to a view □View updates must be unambiguously mappable to base relation updates in order to be allowed

% Physical distinctions:

☑Relations must be physically stored somewhere ☑Views are logical entities

Is it possible to rewrite using Views?

Find companies who manufacture products bought by Joe Blow.

SELECT Product.Company FROM Product WHERE Product.company = "Bazzar" AND Product.name IN (SELECT product FROM Purchase WHERE buyer = "Joe Blow");

Is it possible to rewrite using Views?

Product (pname, price, category, maker) Find products that are more expensive than all those produced By "Gizmo-Works"

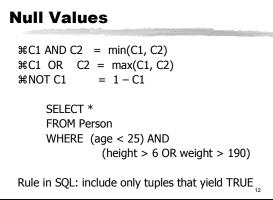
SELECT name FROM Product WHERE price > ALL (SELECT price FROM Purchase WHERE maker="Gizmo-Works") Is it possible to rewrite using Views?

Product (pname, price, category, maker, year)

%Find products (and their manufacturers) that are more expensive than all products made by the same manufacturer before 1972

SELECT pname, maker FROM Product AS x WHERE price > ALL (SELECT price FROM Product AS y WHERE x.maker = y.maker AND y.year < 1972);

Null Values#If x=Null then 4*(3-x)/7 is still NULL#If x=Null then x=``Joe'' is UNKNOWN#If x=Null then x=``Joe'' is UNKNOWN#If x=Null then x=``Joe'' is UNKNOWN#If x=Null then x=0#If x=Null then



10

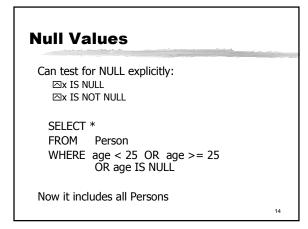
Null Values

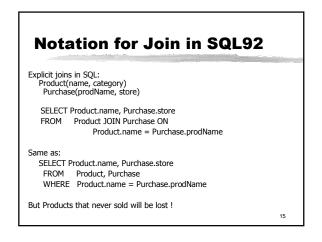
Unexpected behavior:

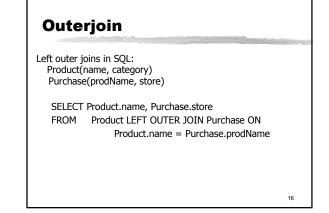
SELECT * FROM Person WHERE age < 25 OR age >= 25

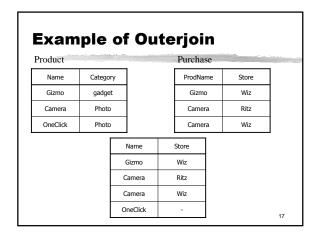
13

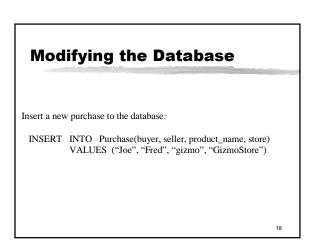
Some Persons are not included !

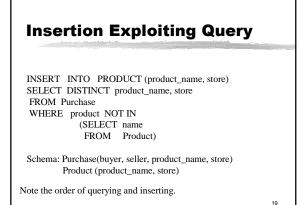


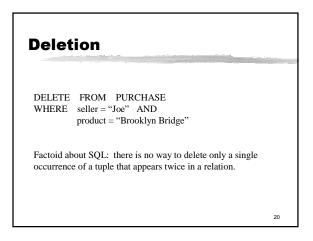












Updates

UPDATE PRODUCT SET price = price/2 WHERE Product.name IN (SELECT product FROM Sales WHERE Date = today);

Updating Complex Views

21

23

How can I insert a tuple into a table that doesn't exist?

CREATE VIEW bon-purchase AS SELECT store, seller, product FROM Purchase WHERE store = "The Bon Marche"

If we make the following insertion:

INSERT INTO bon-purchase VALUES ("the Bon Marche", Joe, "Denby Mug") We can simply add a tuple ("the Bon Marche", Joe, NULL, "Denby Mug") to relation Purchase.

Example of Non-Updatable Views

CREATE VIEW Seattle-view AS

 SELECT
 seller, product, store

 FROM
 Person, Purchase

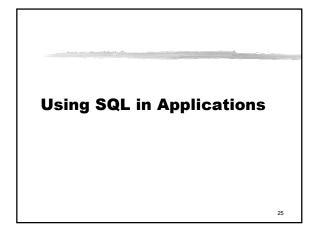
 WHERE
 Person.city = "Seattle"

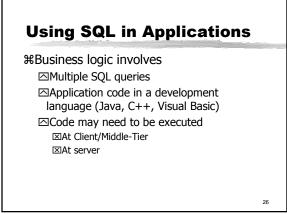
 AND
 Person.name = Purchase.buyer

How can we add the following tuple to the view above? Think about null semantics..

(Joe, "Shoe Model 12345", "Nine West")

24





Using SQL in Applications (2)

Data Type issues (Mapping of Types)
Reconcile Explicit iteration in Programming Language with set-oriented processing in SQL (Cursors)
SQL generated on-the-fly (Dynamic SQL)
Connectivity of client code to database server

27

Mapping Types

%char=> character (length, char set)
%varchar=> character varying (length, char
set)
%short=> smallint
%Long=> integer
%Float=> real
%Double= double precision

28

30

Getting Data Out

#Application languages deals with a row at a time

⊡Not set of rows

#How to consume result of a SQL query?

#SQL supports cursors

□Like a pointer that traverses a collection of rows <u>one at a time</u>

29

Cursors

- 1. Declare the cursor
- 2. Open the cursor
- 3. Fetch rows one by one
- 4. Update/Delete "current" tuples
- 5. Close the cursor

Declare - Example

Declare cursor1 cursor for Select current_sales_price, our_cost From movie_titles Where current_sales_price > :minprice Order By current-Sales_price

Open/Fetch/Close

Open cursor_name

Fetch [Next| Prior| First | Last | Absolute <k> | Relative <k>] cursor_name into :struct1

32

34

Close cursor_name

Update/Delete

Delete from table_name where current of cursor_name

Update table_name Set set_list where current of cursor_name

Update movie_titles Set our_cost = our_cost/2 where current of cursor1

33

31

Revisiting Declare

%DECLARE cursor-name □[INSENSTIVE] [SCROLL] CURSOR FOR □Query_expression □ORDER BY sort_expression □updatability

Declare (Contd)

¥Updatability

- ☑ Read Only no update/delete on cursor allowed
 ☑ Update restricted to specific fields
- Solution Specific fields Section (and the section of the section
 - Select current_sales_price, our_cost From movie_titles
 - For update of current_sales_price

35

<section-header><section-header><section-header><section-header><text><text><text><text>

Declare (Contd)

Scrollable Cursors △Additional syntax in Fetch enabled △Otherwise, only "next" tuple is available △But scroll forces cursor to be read-only!

Connectivity - ODBC Section reacts to establish a connection to to server Section connection handle - unique identification Section statements Section statements Section connection interface (CLI) to SQL stores

ODBC Details

- ${\tt {\# SQLDriverConnect}} \ {\rm -- \ opens \ a \ connection}$
- **#SQLExecDirect** -- executes a sql statement **#SQLBindCol** -- binds a program variable to a
- column in the result of a sql statement
- **#SQLFetch** -- fetches the next row in the current result set
- ℜ SQLMoreResults -- returns true if more result sets are yet to be consumed (e.g., useful for a batch of queries)
- **#SQLError** -- returns information about the last error (for the specified connection)

39

37

Friday's (Jan 19) Special Lecture

- ₩ More on Connectivity
 ₩ Building a front-end using ASP
 ₩ Relevant for
 □ Programming Assignment
- △Project
 ※ Note time and place
 △Sieg 134
 △3.30-4.50pm
- ₩Please be there!