CSCD 327: RELATIONAL DATABASE SYSTEMS

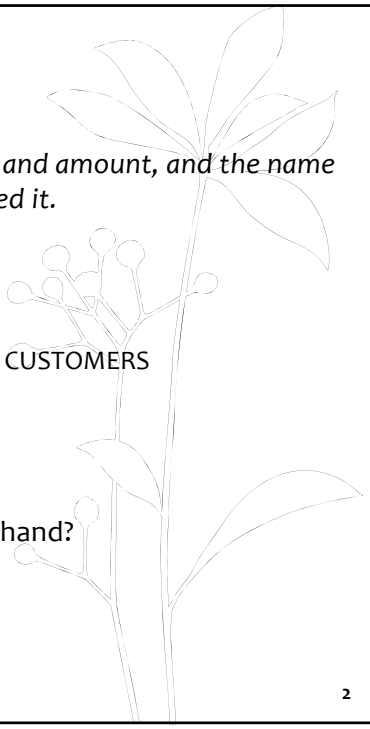
MULTITABLE QUERIES

Instructor: Dr. Dan Li

1

An Example

- *List all orders, showing the order number and amount, and the name and credit limit of the customer who placed it.*
  - There is a connection between ORDERS and CUSTOMERS
- How would you get the query results by hand?



2

Two Findings

- Each row of query results draws its data from a specific *pair of rows, one from the* ORDERS table and one from the CUSTOMERS table.
- The pairs of rows are found by matching the data values in *corresponding columns* from the tables.
- Solution

3

Parent/Child Relationship

- Parent: the table containing the primary key (CUSTOMER)
- Child: the table containing the foreign key (ORDER)
- One-to-many relationship
- Identify parent and child
  - List each salesperson and the city and region where they work.
  - List the offices and the names and titles of their managers.

4

JOIN … ON to Replace WHERE

- List each salesperson and the city and region where they work.

```
SELECT NAME, CITY, REGION
FROM … JOIN …
ON …;
```

- List the offices and the names and titles of their managers.

```
SELECT CITY, NAME, TITLE
FROM … JOIN …
ON …;
```

5

Joins with Row Selection Criteria

- *List the offices with a target over $600,000 and their manager information.*

```
SELECT CITY, NAME, TITLE
FROM OFFICES, SALESREPS
WHERE …
AND …;
```

Same as:

```
SELECT CITY, NAME, TITLE
FROM OFFICES JOIN SALESREPS
ON …
WHERE …;
```

6

Multiple Parent/Child Relationships

- *List all the orders, showing amounts and product descriptions.*

```
SELECT ORDER_NUM, AMOUNT, DESCRIPTION
FROM ORDERS, PRODUCTS
WHERE …
AND …;

Same as:
SELECT ORDER_NUM, AMOUNT, DESCRIPTION
FROM ORDERS JOIN PRODUCTS
ON …
AND …;
```

7

NATURAL JOIN

- Can only be used when the two matching columns have the exactly same name in both tables.
  - Not the case in previous examples
  - Now let's make some changes
    - Change MFR_ID in PRODUCTS into MFR
    - Change Product_ID in PRODUCTS into Product
- Match ALL column pairs with the same names.
- All of the following four statements are the same.

Q: which one is preferred?
A: JOIN… USING
Why?

8

JOIN Involving Three or More Tables

- *List orders over $25,000, including the name of the salesperson who took the order and the company name of the customer who placed it.*

```
SELECT ORDER_NUM, AMOUNT, COMPANY, NAME
FROM …, …, …
WHERE …
AND …
AND AMOUNT > 25000.00;
```

CUST in ORDERS is a foreign key to the CUSTOMERS table (CUST_NUM);
REP in ORDERS is a foreign key to the SALESREPS table (EMPL_NUM).

Same as:

```
SELECT ORDER_NUM, AMOUNT, COMPANY, NAME
FROM … JOIN … ON …
JOIN … ON …
WHERE AMOUNT > 25000.00;
```

9

Is Parent/Child Relationship a Must?

- NO.

- *Find all orders received on a day when a new salesperson was hired.*

```
SELECT ORDER_NUM, AMOUNT, ORDER_DATE, NAME
FROM …, …
WHERE …;
```

This example generates a many-to-many relationship:

10

Is Equality Condition a Must?

- No
- *List all combinations of salespeople and offices where the salesperson's quota is more than that office's target, regardless of whether the salesperson works there.*

```
SELECT NAME, QUOTA, CITY, TARGET
FROM SALESREPS, OFFICES
WHERE ...;
```

11

Qualified Column Names to Avoid Ambiguity

- *Show the name, sales, and office for each salesperson.*

```
SELECT NAME, SALES, CITY
FROM SALESREPS, OFFICES
WHERE REP_OFFICE = OFFICE;
```

Will this work?

- To avoid ambiguity, use dot (.) operator to explicitly list table name.

```
SELECT NAME, ..., CITY
FROM SALESREPS, OFFICES
WHERE REP_OFFICE = OFFICE;
```

12

Self-Joins

- *List the names of salespeople and their managers.*

```
SELECT NAME, NAME
FROM SALESREPS, SALESREPS
WHERE MANAGER = EMPL_NUM;
```

How about eliminating one reference?

```
SELECT NAME, NAME
FROM SALESREPS
WHERE MANAGER = EMPL_NUM;
```

Correct solution:

```
SELECT …, …
FROM …, …
WHERE …;
```

Use aliases to refer to the same table twice.
Can be considered as to virtual copies.
To simplify, only use one alias; the other one uses the original table name.

13

Multitable Query Processing

- The query is processed in this order:
  - Form the product of the tables named in the FROM clause.
  - Apply matching-column condition specified by ON clause.
  - Apply select condition specified by WHERE clause.
  - Keep the columns listed in SELECT clause.
  - Remove duplicates if SELECT DISTINCT is specified.
  - Sort the query results based on ORDER BY clause.

14

More JOINs

- OUTER JOIN to avoid loss of information
  - Computes the join and then adds tuples form one relation that does not match tuples in the other relation to the result of the join.
  - Uses *null* values.
  - LEFT OUTER JOIN
  - RIGHT OUTER JOIN
  - FULL OUTER JOIN (MySQL doesn't support it!)
- INNER JOIN

15

Joined Relations

- **Join operations** take two relations and return as a result another relation.
- These additional operations are typically used as subquery expressions in the **from** clause
- **Join condition** – defines which tuples in the two relations match, and what attributes are present in the result of the join.
- **Join type** – defines how tuples in each relation that do not match any tuple in the other relation (based on the join condition) are treated.

Join types
inner join
left outer join
right outer join
full outer join

Join Conditions
natural
on <predicate>
using $(A_1, A_1, \ldots, A_n)$

16

Sample Relations

- Relation *course*

course_id	*title*	*dept_name*	*credits*
BIO-301	Genetics	Biology	4
CS-190	Game Design	Comp. Sci.	4
CS-315	Robotics	Comp. Sci.	3

- Relation *prereq*

course_id	*prereq_id*
BIO-301	BIO-101
CS-190	CS-101
CS-347	CS-101

- Observe that
  prereq information is missing for CS-315 and
  course information is missing for CS-437

17

Left Outer Join

- *course* **natural left outer join** *prereq*

course_id	*title*	*dept_name*	*credits*	*prere_id*
BIO-301	Genetics	Biology	4	BIO-101
CS-190	Game Design	Comp. Sci.	4	CS-101
CS-315	Robotics	Comp. Sci.	3	*null*

18

Right Outer Join

- *course* **natural right outer join** *prereq*

course_id	*title*	*dept_name*	*credits*	*prere_id*
BIO-301	Genetics	Biology	4	BIO-101
CS-190	Game Design	Comp. Sci.	4	CS-101
CS-347	*null*	*null*	*null*	CS-101

19

Full Outer Join

- *course* **natural full outer join** *prereq*

course_id	*title*	*dept_name*	*credits*	*prere_id*
BIO-301	Genetics	Biology	4	BIO-101
CS-190	Game Design	Comp. Sci.	4	CS-101
CS-315	Robotics	Comp. Sci.	3	*null*
CS-347	*null*	*null*	*null*	CS-101

20

Inner Join & ON

- *course* **inner join** *prereq* **on** *course.course_id = prereq.course_id*

course_id	*title*	*dept_name*	*credits*	*prereq_id*	*course_id*
BIO-301	Genetics	Biology	4	BIO-101	BIO-301
CS-190	Game Design	Comp. Sci.	4	CS-101	CS-190

- What is the difference between the above, and a natural join?
- *course* **left outer join** *prereq* **on** *course.course_id = prereq.course_id*

course_id	*title*	*dept_name*	*credits*	*prereq_id*	*course_id*
BIO-301	Genetics	Biology	4	BIO-101	BIO-301
CS-190	Game Design	Comp. Sci.	4	CS-101	CS-190
CS-315	Robotics	Comp. Sci.	3	*null*	*null*

21

USING

- *course* **full outer join** *prereq* **using** (*course_id*)

course_id	*title*	*dept_name*	*credits*	*prere_id*
BIO-301	Genetics	Biology	4	BIO-101
CS-190	Game Design	Comp. Sci.	4	CS-101
CS-315	Robotics	Comp. Sci.	3	*null*
CS-347	*null*	*null*	*null*	CS-101

22