CSCD 327: Relational Database Systems

Multitable Queries

Instructor: Dr. Dan Li

An Example

- *List all orders, showing the order number and amount, and the name and credit limit of the customer who placed it.*
  - Order number and amount come from ORDERS
  - Customer name and credit limit come from CUSTOMERS
  - There is a connection between ORDERS and CUSTOMERS
    - CUST_NUM in CUSTOMERS is a primary key
    - CUST in ORDERS is a foreign key which refers to CUST_NUM in CUSTOMERS
    - They build a PARENT (the table containing the primary key) / CHILD (the table containing the foreign key) relationship
- How would you get the query results by hand?

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.
  - Need to select from TWO tables
- The pairs of rows are found by matching the data values in *corresponding columns* from the tables.
  - Need to specify such condition in WHERE clause
- Solution

```
SELECT ORDER_NUM, AMOUNT, COMPANY, CREDIT_LIMIT
FROM ORDERS, CUSTOMERS
WHERE CUST = CUST_NUM;
```

Parent/Child Relationship

- Parent: the table containing the primary key (CUSTOMER)
- Child: the table containing the foreign key (ORDER)
- One-to-many relationship
  - Each order is associated with exactly one customer.
  - Each customer can have many associated orders.
- Identify parent and child
  - List each salesperson and the city and region where they work.

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

SALESPERSONS is the child;
OFFICES is the parent.

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

```
SELECT CITY, NAME, TITLE
FROM OFFICES, SALESREPS
WHERE MGR = EMPL_NUM;
```

SALESPERSONS is the parent;
OFFICES is the child.

JOIN … ON to Replace WHERE

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

```
SELECT NAME, CITY, REGION
FROM SALESREPS JOIN OFFICES
ON REP_OFFICE = OFFICE;
```

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

```
SELECT CITY, NAME, TITLE
FROM OFFICES JOIN SALESREPS
ON MGR = EMPL_NUM;
```

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 MGR = EMPL_NUM
AND TARGET > 600000.00;
```

Same as:

```
SELECT CITY, NAME, TITLE
FROM OFFICES JOIN SALESREPS
ON MGR = EMPL_NUM
WHERE TARGET > 600000.00;
```

Multiple Parent/Child Relationships

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

```
SELECT ORDER_NUM, AMOUNT, DESCRIPTION
FROM ORDERS, PRODUCTS
WHERE MFR = MFR_ID
AND PRODUCT = PRODUCT_ID;
```

Same as:

```
SELECT ORDER_NUM, AMOUNT, DESCRIPTION
FROM ORDERS JOIN PRODUCTS
ON MFR = MFR_ID
AND PRODUCT = PRODUCT_ID;
```

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.

```
SELECT ORDER_NUM, AMOUNT, DESCRIPTION
FROM ORDERS NATURAL JOIN PRODUCTS;
```

```
SELECT ORDER_NUM, AMOUNT, DESCRIPTION
FROM ORDERS JOIN PRODUCTS
USING (MFR, PRODUCT);
```

```
SELECT ORDER_NUM, AMOUNT, DESCRIPTION
FROM ORDERS JOIN PRODUCTS
ON ORDERS.MFR = PRODUCTS.MFR
AND ORDERS.PRODUCT = PRODUCTS.PRODUCT;
```

```
SELECT ORDER_NUM, AMOUNT, DESCRIPTION
FROM ORDERS, PRODUCTS
WHERE ORDERS.MFR = PRODUCTS.MFR
AND ORDERS.PRODUCT = PRODUCTS.PRODUCT;
```

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

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 ORDERS, CUSTOMERS, SALESREPS
WHERE CUST = CUST_NUM
AND REP = EMPL_NUM
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 ORDERS JOIN CUSTOMERS ON CUST = CUST_NUM
JOIN SALESREPS ON REP = EMPL_NUM
WHERE AMOUNT > 25000.00;
```

Is Parent/Child Relationship a Must?

- NO.
- Any columns can serve as matching columns, provided they have comparable data types.
- *Find all orders received on a day when a new salesperson was hired.*

```
SELECT ORDER_NUM, AMOUNT, ORDER_DATE, NAME
FROM ORDERS, SALESREPS
WHERE ORDER_DATE = HIRE_DATE;
```

This example generates a many-to-many relationship:
Each potential order_date can be associated with more new hires.
Each potential hire_date can be associated with more orders.

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 QUOTA > TARGET;
```

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;
```

It won't work, because SALES appear in both tables.

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

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

Self-Joins

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

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

This SELECT statement is illegal because of the duplicate reference to the SALESREPS table in the FROM clause.

How about eliminating one reference?

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

This query is legal, but it won't do what you want it to do!

Correct solution:

```
SELECT EMPS.NAME, MGRS.NAME
FROM SALESREPS EMPS, SALESREPS MGRS
WHERE EMPS.MANAGER = MGRS.EMPL_NUM;
```

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.

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.

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

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 $(A_1, A_1, \ldots, A_n)$

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-347

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*

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

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

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*

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