Advanced SQL Topics

Alessandro Bozzon
TI1506: Web and Database Technology
ti1506-ewi@tudelft.nl

TU Delft

Course overview [DB]

1. Introduction to Database Systems
2. The Relational Model
3. Introduction to SQL
4. **Advanced SQL Topics**
5. Introduction to NoSQL database systems
6. Conceptual Data Modelling with ER Diagrams
7. Database Conceptual Design
8. Database Logical Design

Warmup Exercise

- Provide the SQL statement (only one!) that returns the correct answers to the following query

How many audio tracks in total were bought by German customers?
And what was the total price paid for them?

Solution

```
SELECT COUNT(t.trackId), SUM(l.unitprice)
FROM invoiceLine l, invoice i, customer c, track t,
     mediaType m
WHERE l.invoiceId = i.invoiceId
    AND i.customerId = c.customerId
    AND c.country = 'Germany'
    AND l.trackId = t.trackId
    AND t.mediaTypeId = m.mediaTypeId
    AND m.name LIKE '%audio%'
```

Nested Queries

Subqueries

- A parenthesised `SELECT-FROM-WHERE` statement can be used as a value in a number of places, including `FROM`, `WHERE`, and `HAVING` clauses
  - **subquery** or **nested query**
  - Example: in place of a relation in the `FROM` clause, we can place another query, and then assert a condition on its results

- A common use of subqueries is to perform tests for *set membership*, *set comparisons*, and *set cardinality*

- The use of nested queries may produce less declarative queries, but they often improve readability

- Complex queries can become very difficult to understand

The `IN` Operator

- `<tuple> IN <relation>` is **true if and only if** the tuple is a member of the relations
  - `<tuple> NOT IN <relation>` means the opposite

- **`IN`** expression can appear in **`WHERE`** clauses
  - The relation is often a subquery
    - e.g. `WHERE firstName IN ( SELECT lastName FROM EMPLOYEE)`
  - It is possible to specify a pre-defined list of values
    - e.g. `WHERE firstName IN  ('Jan', 'Arie', 'Henk')`

- It allows problem decomposition, typically with a “bottom-up” approach

Example / I

DB2

Find the name of the suppliers that supply product P2

Two Subproblems
- Code of P2 suppliers
- Name of such suppliers

With Join

```
SELECT NameS
FROM Supplier, Supply
WHERE Supplier.CodeS=Supply.CodeS AND CodeP = "P2"
```

With Nested Queries

```
SELECT NameS
FROM Supplier
WHERE CodeS IN (SELECT CodeS
                FROM Supply
                WHERE CodeP = "P2")
```

Example /2

DB2

Find the name of the suppliers that supply at least one red product

Three subproblems
- Code of red products,
- Code of suppliers that supply such products
- Name of such suppliers

With Join

```
SELECT NameS
FROM Supplier, Supply, Product
WHERE Supplier.CodeS=Supply.CodeS AND Supply.CodeP = Product.CodeP AND
      Product.CodeP = "P2"
```

With Nested Queries

```
SELECT NameS
FROM Supplier
WHERE CodeS IN (SELECT CodeS
                FROM Supply
                WHERE CodeP IN( SELECT CodeP
                                FROM Products
                                WHERE Color="red"))
```

ASQ Exercise

- Provide the SQL statement that returns the correct answers to the following query

Return the name of artists that played neither rock nor pop songs

Solution

```
SELECT DISTINCT Artist.Name
FROM Artist JOIN Album ON Album.ArtistId =
Artist.ArtistId
WHERE Artist.ArtistId NOT IN (
         SELECT Album.ArtistId
         FROM Track JOIN Album ON Album.AlbumId =
Track.AlbumId
         WHERE GenreId = 1 OR GenreId = 6
    )
```

Nested Queries That Return One Tuple

- If a subquery is guaranteed to produce one tuple, then the result of the subquery **can be used as a value**
  - Typically, a single tuple is guaranteed by key constraints of attributes `SELECT`ed by the subquery

- A run-time error occurs if there is no tuple or more than one tuple

- Usually, the tuple has one attribute, but with a tuple constructor we might have many => **row subquery**

Example / I

DB2

Find the code of suppliers having their office in the same city as `S1`

With Join

```
SELECT CodeS
FROM Supplier AS Su1, Supplier AS Su2
WHERE Su1.Office=Su2.Office AND Su1.CodeS = "S1"
```

With Nested Queries

```
SELECT CodeS
FROM Supplier
WHERE Office = (SELECT Office
                FROM Supplier
                WHERE CodeS = "S1")
```

Example /2

DB2

Find the code of suppliers with less shareholders than the supplier having the maximum number of shareholders

With Nested Queries

```
SELECT CodeS
FROM Supplier
WHERE Shareholders < (SELECT Max(Shareholders)
                      FROM Supplier)
```

With Join??

Maybe. Think about it :)

Example /3

DB2

Find the name of the suppliers that supply at least one product supplied by suppliers of red products

Difficult to express with joins

- Code of red products
- Code of suppliers that supply such products
- Code of products supplied by suppliers of red products
- Code of the suppliers of the products supplied by suppliers of red products
- Name of such suppliers

```
SELECT NameS
FROM Supplier
WHERE CodeS IN (SELECT CodeS
                FROM Supply
                WHERE CodeP IN( SELECT CodeP
                                FROM Supply
                                WHERE CodeS IN (SELECT CodeS
                                                FROM Supply
                                                WHERE CodeP IN ( SELECT CodeP
                                                                 FROM Product
                                                                 WHERE Color = "Red"))))
```

Example /3 with `JOIN`

DB2

Find the name of the suppliers that supply at least one product supplied by suppliers of red products

```
SELECT NameS
FROM Supplier, Supply as SA, Supply as SB, Supply as SBC, Product
WHERE Supplier.CodeS = SA.CodeS AND
      SA.CodeP = SB.CodeP AND
      SB.CodeS = SC.CodeS AND
      SC.CodeP = Product.CodeP AND
      Color = "Red"
```

ASQ Exercise

When an SQL query combines three or more AND and OR conditions in the WHERE clause, which of the following SQL operations would you use?

1) LIKE

2) IN

3) NOT IN

4) Both IN and NOT IN

Correct =>

Example with `NOT IN`

DB2

Find the name of the suppliers that do not supply product P2

- Can you express it with `JOIN`?

```
SELECT NameS
FROM Supplier, Supply
WHERE Supplier.CodeS = Supply.CodeS AND CodeP <> "P2"
```

- **WRONG!!!**
- **This would answer the query:**

Find the name of the suppliers that supply at least one product different from P2

Example with `NOT IN/2`

DB2

- We need to exclude from the result set suppliers that supply P2

```
SELECT NameS
FROM Supplier
WHERE CodeS NOT IN ( SELECT CodeS
                     FROM Supply
                     WHERE CodeP = "P2")
```

Example with `NOT IN` /3

DB2

Find the name of the suppliers that supply ONLY product P2

- We need to exclude from the result set suppliers that supply products different from P2
- But only keep the ones that actually supply something

```
SELECT NameS
FROM Supplier
WHERE CodeS NOT IN ( SELECT CodeS
                     FROM Supply
                     WHERE CodeP < >  "P2")
      AND Supplier.CodeS IN (Select CodeS
                             FROM Supply)
```

Example with `NOT IN` /4

DB2

Find the name of the suppliers that DO NOT supply red products

- We need to exclude from the result set suppliers of red products

```
SELECT NameS
FROM Supplier
WHERE CodeS NOT IN ( SELECT CodeS
                     FROM Supply
                     WHERE CodeP IN ( SELECT CodeP
                                      FROM Product
                                      WHERE Colour = "Red"))
```

ASQ Exercise

- Provide the SQL statement that returns the correct answers to the following query

Find the names of customers who live in the same city as the top employee (The one not managed by anyone).

Solution

```
SELECT firstname, lastname
FROM customer
WHERE city =
     (SELECT city
      FROM employee
      WHERE reportsTo IS NULL);
```

Scope of Attributes, Variables, and correlated queries

- A subquery can use attributes and/or variables defined by outermost queries in their WHERE clause
  - this is sometimes referred to as “transfer of bindings”

- The two queries are said to be correlated

- Semantics: the nested query is evaluated **for each row of the external query**

The EXISTS Operator/1

DB2

Find the name of the suppliers for whom it exists at least one supply of P2

```
SELECT NameS
FROM Supplier
WHERE EXISTS ( SELECT *
               FROM Supply
               WHERE CodeP = "P2" AND Supplier.CodeS = Supply.CodeS)
```

- !!! We need to test the existence of a *supply* for the evaluation of *suppliers*
- **`Suppliers.CodeS = Supply.CodeS`** impose a correlation between external and internal query

The EXISTS Operator /2

Find all the homonyms, i.e., persons who have the same first name and surname, but different BSN

```
SELECT *
FROM Person P
WHERE EXISTS (SELECT *
              FROM Person P1
              WHERE P1.FirstName = P.FirstName
                    AND P1.Surname = P.Surname
                    AND P1.BSN < > P.BSN)
```

ASQ Exercise

- How could you rewrite the same query without nested queries?

Find the names of customers who live in the same city as the top employee (The one not managed by anyone).

Solution

```
SELECT *
FROM customer JOIN employee
           ON customer.City = employee.City
WHERE employee.ReportsTo IS NULL
```

Limitations

- A query cannot refer to attributes in a subquery, or in a query at the same level of nesting

```
SELECT *
FROM Employee
WHERE Dept IN (SELECT DeptName
               FROM Department D1
               WHERE DeptName = "Production") OR
      Dept IN (SELECT DeptName
               FROM Department D2
               WHERE D2.City = D1.City)
```

- The query is incorrect because variable `D1` is not visible in the second nested query

Example of `NOT EXISTS` operator

Find the name of the suppliers that DO NOT supply P2 **or**

Find the name of the suppliers for whom it does not exist at least one supply of P2

```
SELECT NameS
FROM Supplier
WHERE NOT EXISTS ( SELECT *
                   FROM Supply
                   WHERE CodeP = "P2" AND
                         Supplier.CodeS = Supply.CodeS)
```

ASQ Exercise

- Provide the SQL statement that returns the correct answers to the following query

Find the name of songs that have never been sold to USA customers

Solution

```
SELECT count(DISTINCT T.Name)
FROM Track AS T
WHERE NOT EXISTS (
    SELECT InvoiceLine.TrackId
    FROM Invoice JOIN InvoiceLine
          ON Invoice.InvoiceId = InvoiceLine.InvoiceId
        JOIN Customer
          ON Invoice.CustomerId = Customer.CustomerId
    WHERE Customer.Country = "USA"
        AND InvoiceLine.TrackId = T.TrackId
    )
```

Example of `NOT EXISTS` operator

Find all the persons who do not have homonyms

```
SELECT *
FROM Person P
WHERE NOT EXISTS (SELECT *
                  FROM Person P1
                  WHERE P1.FirstName = P.FirstName
                        AND P1.Surname = P.Surname
                        AND P1.BSN < > P.BSN)
```

Question 2_BI/I

ASQ Exercise

What operator tests column for the absence of data?

Correct =>

1) EXISTS operator

2) NOT operator

3) IS NULL operator

4) None of these

Tuple Constructor

- The comparison with the nested query may involve more than one attributes
- The attributes must be enclosed within a pair of curved brackets (tuple constructor)
- The query in the previous slide can be expressed as:

```
SELECT *
FROM Person P
WHERE (FirstName,Surname) NOT IN ( SELECT FirstName, Surname
                                   FROM Person P1
                                   WHERE P1.BSN < > P.BSN)
```

ANY Operator /1

Find the employees who work in departments in London

```
SELECT FirstName, Surname
FROM Employee
WHERE Dept = ANY ( SELECT DeptName
                   FROM Department
                   WHERE City = "London")
```

- But also

```
SELECT FirstName, Surname
FROM Employee, Department D
WHERE Dept = DeptName AND D.City = "London"
```

ANY Operator /2

Find the employees of the Planning department, having the same first name as a member of the Production department

```
SELECT FirstName, Surname
FROM Employee
WHERE Dept = "Planning"
      AND FirstName = ANY ( SELECT FirstName
                            FROM Employee
                            WHERE Dept = "Production")
```

- But also

```
SELECT E1.FirstName, E1.Surname
FROM Employee E1, Employee E2
WHERE E1.FirstName = E2.FirstName AND E2.Dept="Production"
      AND E1.Dept = "Planning"
```

Example `MIN` and `MAX`

- Queries using the aggregate operators max and min can be expressed with nested queries

```
SELECT Dept
FROM Employee
WHERE Salary >= ALL ( SELECT Salary
                      FROM Employee)
```

- But also

```
SELECT Dept
FROM Employee
WHERE Salary IN ( SELECT max(Salary)
                  FROM Employee)
```

Example of negation with `ALL`

Find the departments in which there is no one named Brown

```
SELECT DeptName
FROM Department
WHERE DeptName < > ALL ( SELECT Dept
                         FROM Employee
                         WHERE Surname = "Brown")
```

- Alternatively (more next)

```
  SELECT DeptName
  FROM Department

EXCEPT

  SELECT Dept
  FROM Employee
  WHERE Surname = "Brown"
```

Nested and Aggregated Queries

Find the code of the suppliers that supply ALL products

```
SELECT CodeS
FROM Supply
GROUP BY CodeS
HAVING COUNT(*) = ( SELECT COUNT(*)
                    FROM Product)
```

- All the products that can be supplied are in the **`Product`** table
- A supplier supply all the products if the number of distinct supplied product is the same as the number of available products

ASQ Exercise

- Provide the SQL statement that returns the correct answers to the following query

Return the audio tracks which have a length longer than the average length of all the audio tracks

Solution

```
SELECT Track.Name
FROM Track,MediaType
WHERE Track.MediaTypeId = MediaType.MediaTypeId
AND MediaType.Name LIKE '%audio%'
AND Track.Milliseconds > (
    SELECT AVG(Milliseconds)
     FROM Track,MediaType
      WHERE Track.MediaTypeId=MediaType.MediaTypeId
      AND MediaType.Name LIKE '%audio%')
```

Nested and Aggregated Queries /2

Find the code of the suppliers that supply at least ALL the products supplied by S2

```
SELECT CodeS
FROM Supply
WHERE CodeP IN ( SELECT CodeP
                 FROM Supply
                 WHERE CodeS' = "S2")
GROUP BY CodeS
HAVING COUNT(*) = ( SELECT COUNT(*)
                    FROM Supply
                    WHERE CodeS = "S2")
```

- The number of products supplied by S2, should be equal to the number of products supplied by a supplier

Question 2_B2/2

ASQ Exercise

- Provide the SQL statement that returns the correct answers to the following query

Find the "TV Shows" tracks that sold more than the average "Drama" track

Solution

```
SELECT track.TrackId,
    SUM(InvoiceLine.UnitPrice * InvoiceLine.Quantity) as Total
FROM track JOIN InvoiceLine
             ON InvoiceLine.TrackId = track.TrackId
WHERE track.genreId = 19
GROUP BY track.TrackId
HAVING Total >= (
     SELECT avg(InvoiceLine.UnitPrice * InvoiceLine.Quantity)
     FROM track JOIN InvoiceLine ON InvoiceLine.TrackId = track.TrackId
     WHERE track.genreId = 21)
ORDER BY Total DESC
```

Set Queries

Set Queries

- Union, intersection, and difference of relations are expressed by the following forms, each involving subqueries:

- `(subquery) INTERSECT [ALL] (subquery)`
- `(subquery) EXCEPT [ALL] (subquery)`
- `(subquery) UNION [ALL] (subquery)`

Can be expressed with other operators (typically sub-queries)

Enhancement of the expressive power

Set Semantic of Set Queries

- Although the `SELECT-FROM-WHERE` statement uses bag semantics, the default for union, intersection, and difference is set semantics.
  - That is, duplicates are eliminated as the operation is applied

- Motivation: **Efficiency**
  - When *projecting* attributes, it is easier to avoid eliminating duplicates. Just work tuple-at-a-time.
  - When doing intersection or difference, it is most efficient to sort the relations first. At that point you may as well eliminate the duplicates anyway

UNION

- A single **`SELECT`** cannot represent unions of values from two or more tables

```
A UNION B
```

- It executes the union of two relational expressions
  - Expressions generated by **`SELECT`** clauses
  - **A and B must have compatible schema**
    - select queries must have the same number of output fields, in the same order, and with the same or compatible data types
  - Duplicate removal
    - **`UNION`** removes duplicates
    - **`UNION ALL`** does not remove duplicates

UNION Example

DB1

Find the code of red products **OR** products supplied by S2 (or both)

```
SELECT CodeP
FROM Product
WHERE Colour = "Red"

UNION

SELECT CodeP
FROM Supply
WHERE CodeS = "S2"
```

UNION Example

DB1

Find the code of red products **OR** products supplied by S2 (or both)

Products

CodeP	NameP	Color	Size	Storehouse
P1	Sweater	Red	40	Amsterdam
P2	Jeans	Green	48	Den Haag
P3	Shirt	Blu	48	Rotterdam
P4	Shirt	Blu	44	Amsterdam
P5	Skirt	Blu	40	Den Haag
P6	Coat	Red	42	Amsterdam

Supply

CodeS	CodeP	Amount
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P3	200
S4	P4	300
S4	P5	400

CodeP
P1
P6

CodeP
P1
P2

CodeP
P1
P2
P6

UNION Example

DB2

Find the first names and surnames of the employees

```
SELECT FirstName as Name
FROM Employee

UNION

 SELECT Surname
 FROM Employee
```

Question 2_B3/1

ASQ Exercise

How can we get all tuples (redundant as well as non-redundant) from UNION operator?

Correct =>

1) Using ‘ALL’ operator with UNION.

2) Using ‘Distinct’ operator with UNION

3) We get all records (redundant as well as non-redundant) with UNION operator by default.

4) None of the above.

INTERSECTION

```
A INTERSECT B
```

- Intersection of two subqueries
- As for the **`UNION`** operator, schema must be compatible
- Not supported by MySQL

INTERSECT Example

DB1

Find cities hosting both suppliers' offices and products' storehouses

```
SELECT Office
FROM Supplier

INTERSECT

SELECT Storehouse
FROM Product
```

- Equivalent to

```
SELECT Office
FROM Supplier, Product
WHERE Office = Storehouse
```

INTERSECT Example /2

DB2

Find the surnames of employees that are also first names

```
SELECT FirstName as Name
FROM Employee

INTERSECT

SELECT Surname
FROM Employee
```

- Equivalent to

```
SELECT E1.FirstName as Name
FROM Employee E1, Employee E2
WHERE E1.FirstName = E2.Surname
```

EXCEPT

```
A EXCEPT B
```

- *Difference* set operator
- Not supported by MySQL

ASQ Exercise

Which statement best describes the meaning of the following SQL query on the CHINOOK database?

```
SELECT artistId, name FROM artist
   EXCEPT
SELECT artist.artistId, artist.name
FROM artist, album
WHERE artist.artistId = album.artistId
```

Correct =>

1) Retrieve the id and name of artists that published at least one album

2) Retrieve the id and name of artists that published more than one albums

3) Retrieve the id and name of artists that published no album

4) Retrieve the id and name of artists that published with other artists

EXCEPT Example

DB1

Find cities hosting suppliers' offices, but not products' storehouses

```
SELECT Office
 FROM Supplier

EXCEPT

SELECT Storehouse
FROM Product
```

- Can be represented with a nested query using the NOT IN operator

```
SELECT Office
FROM Supplier
WHERE Office NOT IN (SELECT Storehouse
                     FROM Product)
```

EXCEPT Example /2

DB2

Find the surnames of employees that are not also first names

```
SELECT FirstName as Name
FROM Employee
EXCEPT

 SELECT Surname
 FROM Employee
```

ASQ Exercise

- Provide the SQL statement that returns the correct answers to the following query

Which artists did not record any tracks of the Latin genre?

Solution

```
SELECT artistId, name FROM artist

EXCEPT

SELECT artist.artistId, artist.name
FROM artist, album, track, genre
WHERE artist.artistId = album.artistId
    AND album.albumId = track.albumId
    AND track.genreId = genre.genreId
    AND genre.name = ‘Latin'
```

Today we covered

- Nested Queries
- Set Queries

Readings

- **Database Book**
  - *Chapter 4*
  - *Chapter 5*
- **Suggested Readings on Blackboard**

End of Lecture

ASQ Exercise

- Provide the SQL statement that returns the correct answers to the following query

List the genre of tracks which is contained in the most playlist

Solution

```
SELECT Genre.GenreId,  Genre.Name, COUNT(Distinct PlaylistId)
FROM Genre, Track, PlaylistTrack
WHERE Genre.GenreId=Track.GenreID AND
      Track.TrackId = PlaylistTrack.TrackId
GROUP BY Genre.GenreId,  Genre.Name
HAVING COUNT(Distinct PlaylistId) = (
   SELECT MAX(aux.count)
= PlaylistTrack.TrackId
FROM (SELECT  COUNT(Distinct PlaylistId) as count
      FROM Genre, Track, PlaylistTrack
      WHERE Genre.GenreId=Track.GenreID AND Track.TrackId
      GROUP BY Genre.GenreId,  Genre.Name) AS aux)
```

ASQ Exercise

- Provide the SQL statement that returns the correct answers to the following query

Return the name of the playlist(s) that contain the largest number of pop tracks

Solution

```
SELECT P.PlaylistId, P.Name
FROM Playlist as P, PlaylistTrack as PL, Track, MediaType, Genre
WHERE MediaType.Name LIKE '%audio%' AND P.PlaylistId = PL.PlaylistId
      AND PL.TrackId = Track.TrackId
      AND Track.MediaTypeId = MediaType.MediaTypeId
      AND Track.GenreId = Genre.GenreID AND Genre.Name='Pop'
GROUP BY P.PlaylistId, P.Name
HAVING COUNT(*) =  (SELECT MAX(aux.count)
                    FROM (SELECT COUNT(*) as count
                          FROM Playlist as P, PlaylistTrack as PL, Track, MediaType,
Genre
                          WHERE MediaType.Name LIKE '%audio%' AND P.PlaylistId =
PL.PlaylistId
                          AND PL.TrackId = Track.TrackId
                          AND Track.MediaTypeId = MediaType.MediaTypeId
                          AND Track.GenreId = Genre.GenreID AND Genre.Name='Pop'
                          GROUP BY P.PlaylistId      ) AS aux)
```

ASQ Exercise

- Provide the SQL statement that returns the correct answers to the following query

Find the playlist(s) which contains most tracks by artist ”AC/DC

Solution

```
SELECT P.Name, COUNT(*) as count
FROM Playlist AS P, PlaylistTrack AS PL,Track,Album,Artist
WHERE P.PlaylistId = PL.PlaylistId
      AND PL.TrackId = Track.TrackId
      AND Album.AlbumId = Track.Albumid
      AND Artist.ArtistId = Album.ArtistId
      AND Artist.Name='AC/DC'
GROUP BY P.PlaylistId, P.Name
HAVING COUNT(*) = (
        SELECT MAX(AUX.count)
        FROM (
            SELECT COUNT(*) as count
            FROM Playlist AS P, PlaylistTrack AS PL,Track,Album,Artist
            WHERE P.PlaylistId = PL.PlaylistId
                  AND PL.TrackId = Track.TrackId
                  AND Album.AlbumId = Track.Albumid
                  AND Artist.ArtistId = Album.ArtistId
                  AND Artist.Name='AC/DC'
                  GROUP BY P.PlaylistId) AS AUX)
```