Database Management and Performance Tuning

Query Tuning II

Pei Li

University of Zurich
Institute of Informatics

Unit 3

Acknowledgements: The slides are provided by Nikolaus Augsten and adapted from "Database Tuning" by Dennis Shasha and Philippe Bonnet.

Outline

Outline

About Query Tuning

- `DISTINCT` removes duplicate tuples from the query result.
- Goal: avoid `DISTINCT` if possible!
- How to know if `DISTINCT` is necessary?
- We use the notions of
  - privileged tables and
  - reachability

  to decide whether there can be duplicates in the query result.

Privileged Tables

- Privileged table: Attributes returned by SELECT clause contain a key.
- Example: Get the social security numbers of all employees that work in a technical department.

```
SELECT ssnum
FROM Employee, Techdept
WHERE Employee.dept = Techdept.dept
```

- Employee is a privileged table:
  - the SELECT clause projects the attribute ssnum
  - ssnum is a key of Employee

Reachability

- R and S are tables
- R reaches S if
  - R and S are joined on equality and
  - the join attribute in R is a key of R
- Intuition: A tuple from S is joined to at most one tuple from R.
- Reachability is transitive: if A reaches B and B reaches C then A reaches C.

Reachability – Example

- Previous Example: Get the social security numbers of all employees that work in a technical department.

```
SELECT ssnum
FROM Employee, Techdept
WHERE Employee.dept = Techdept.dept
```

- `Techdept` reaches `Employee`:
  - `Techdept` and `Employee` are joined on equality
  - `dept` is a key of `Techdept`

No-Duplicate Guarantee

- A query returns no duplicates if the following conditions hold:
  - Every attribute in the `SELECT` clause is from a privileged table.
  - Every unprivileged table reaches at least one privileged one.

No-Duplicate Guarantee – Examples

- Does this query return duplicates?

```
SELECT ssnum
FROM Employee, Techdept
WHERE Employee.manager = Techdept.manager
```

- **YES**
- Reason:
  - `manager` is not a key of `Techdept`
  - thus `Techdept` does not reach privileged table `Employee`

No-Duplicate Guarantee – Examples

- Does this query return duplicates?

```
SELECT ssnum, Techdept.dept
FROM Employee, Techdept
WHERE Employee.manager = Techdept.manager
```

- **NO**
- Reason: different from previous example,
  - both `Techdept` and `Employee` are privileged table

No-Duplicate Guarantee – Examples

- Does this query return duplicates?

```
SELECT ssnum, Techdept.dept
FROM Employee, Techdept
```

- **NO**
- Reason: as before,
  - both `Techdept` and `Employee` are privileged table

No-Duplicate Guarantee – Examples

- Does this query return duplicates?
  (note that `Student.name` is not a key)

```
SELECT Student.ssnum
FROM Student, Employee, Techdept
WHERE Student.name = Employee.name
AND Employee.dept = Techdept.dept
```

- **NO**
- Reason:
  - join attribute `Employee.name` is a key, thus `Employee` reaches privileged table `Student`
  - join attribute `Techdept.dept` is a key thus `Techdept` reaches `Employee`
  - transitivity: `Techdept` reaches `Employee` and `Employee` reaches `Student`, thus `Techdept` reaches `Student`

No-Duplicate Guarantee – Examples

- Does this query return duplicates?
  (note that `Student.name` is a key)

```
SELECT Student.ssnum
FROM Student, Employee, Techdept
WHERE Student.name = Employee.name
AND Employee.manager = Techdept.manager
```

- **YES**
- Reason:
  - join attribute `Techdept.manager` is not key
  - thus `Techdept` does not reach `Employee` (and `Student`)

No-Duplicate Guarantee – Examples

- Try the example queries on the following instance (keys underlined):
  - `Employee(ssnum, name, manager, dept)`

ssnum	name	manager	dept
1	Peter	John	IT
2	Rose	Mary	Development

  - `Techdept(dept, manager)`

dept	manager
IT	John
Development	Mary
Production	John

  - `Students(ssnum, name)`

ssnum	name
5	Peter
6	Peter

Outline

Types of Nested Queries

- Uncorrelated subqueries
  - with aggregates in the inner query

```
SELECT ssnum
FROM Employee
WHERE salary > (SELECT AVG(salary) FROM Employee)
```

  - without aggregates in the inner query

```
SELECT ssnum
FROM Employee
WHERE dept IN (SELECT dept FROM Techdept)
```

Types of Nested Queries

- Correlated subqueries
  - with aggregates in the inner query

```
SELECT ssnum
FROM Employee e1, Techdept
WHERE salary = (SELECT AVG(e2.salary)
                FROM Employee e2, Techdept
                WHERE e2.dept = e1.dept
                AND e2.dept = Techdept.dept)
```

  - without aggregates in the inner query (uncommon)

Uncorrelated Subquery with Aggregates

- Uncorrelated subqueries with aggregate in the inner query:

```
SELECT ssnum
FROM Employee
WHERE salary > (SELECT AVG(salary) FROM Employee)
```

- Not problematic:
  - Result of inner query is a single value (constant).
  - Most systems will first execute the inner query and then substitute it with the resulting constant.

Uncorrelated Subquery without Aggregates

- Uncorrelated subqueries without aggregate in the inner query:

```
SELECT ssnum
FROM Employee
WHERE dept IN (SELECT dept FROM Techdept)
```

- Some systems might not use index on `Employee.dept`.
- Unnested query:

```
SELECT ssnum
FROM Employee, Techdept
WHERE Employee.dept = Techdept.dept
```

Uncorrelated Subquery without Aggregates

- Unnesting strategy:
  1. Combine the arguments of the two `FROM` clauses.
  2. `AND` together the `WHERE` clauses.
  3. Replace "`outer.attr1 IN (SELECT inner.attr2 ...)`" with "`outer.attr1 = inner.attr2`" in the `WHERE` clause.
  4. Retain the `SELECT` clause from the outer block.
- Strategy works for nesting of any depth.
- Note: If `inner` table does not reach `outer` table in new join condition, new duplicates may appear.

Duplicates in Unnested Queries – Examples

- Nested query:

```
SELECT AVG(salary)
FROM Employee
WHERE dept IN (SELECT dept FROM Techdept)
```

- Unnested query:

```
SELECT AVG(salary)
FROM Employee, Techdept
WHERE Employee.dept = Techdept.dept
```

- Unnesting is correct:
  - `Techdept` reaches `Employee`, thus no duplicates are introduced
  - each salary appears once in average

Duplicates in Unnested Queries – Examples

- Nested query:

```
SELECT AVG(salary)
FROM Employee
WHERE manager IN (SELECT manager FROM Techdept)
```

- Unnested query:

```
SELECT AVG(salary)
FROM Employee, Techdept
WHERE Employee.manager = Techdept.manager
```

- Unnesting is not correct:
  - `Techdept` does not reach `Employee`, thus duplicates possible
  - some salaries might appears multiple times in the average
- Note: Duplicates do not matter for aggregates like `MIN` and `MAX`.

Duplicates in Unnested Queries – Examples

- Solution for following query?

```
SELECT AVG(salary)
FROM Employee
WHERE manager IN (SELECT manager FROM Techdept)
```

- Create temporary table!

```
SELECT DISTINCT manager INTO Temp
FROM Techdept

SELECT AVG(salary)
FROM Employee, Temp
WHERE Employee.manager = Temp.manager
```

Correlated Subqueries with Aggregates

- Correlated subquery with aggregates in the inner query:

```
SELECT ssnum
FROM Employee e1, Techdept
WHERE salary = (SELECT AVG(e2.salary)
                FROM Employee e2, Techdept
                WHERE e2.dept = e1.dept
                AND e2.dept = Techdept.dept)
```

- Inefficient in many systems.

Strategy for Rewriting Query

```
SELECT ssnum
FROM Employee e1, Techdept
WHERE salary = (SELECT AVG(e2.salary)
                FROM Employee e2, Techdept
                WHERE e2.dept = e1.dept
                AND e2.dept = Techdept.dept)
```

1. Create temporary table:
   - GROUP BY on correlated attribute of inner query (must be equality!).
   - Use uncorrelated qualifications of inner query for WHERE clause.

```
SELECT AVG(salary) as avsalary, Employee.dept INTO Temp
FROM Employee, Techdept
WHERE Employee.dept = Techdept.dept
GROUP BY Employee.dept
```

Strategy for Rewriting Query

```
SELECT ssnum
FROM Employee e1, Techdept
WHERE salary = (SELECT AVG(e2.salary) ... )
```

```
SELECT AVG(salary) as avsalary, Employee.dept INTO Temp
FROM Employee, Techdept
WHERE Employee.dept = Techdept.dept
GROUP BY Employee.dept
```

2. Join temporary table with outer query:
   - Condition on the grouped attribute replaces correlation condition.
   - Depending attribute of grouping replaces subquery.
   - All other qualifications of outer query remain (none in example).

```
SELECT ssnum
FROM Employee, Temp
WHERE salary = avsalary
AND Employee.dept = Temp.dept;
```

The Count Bug

- Correlated subquery with COUNT aggregate in the inner query:

```
SELECT ssnum
FROM Employee e1, Techdept
WHERE numfriends = COUNT(SELECT e2.ssnum
                         FROM Employee e2, Techdept
                         WHERE e2.dept = e1.dept
                         AND e2.dept = Techdept.dept)
```

- Rewrite with temporary table:

```
SELECT COUNT(ssnum) as numcolleagues, Employee.dept INTO Temp
FROM Employee, Techdept
WHERE Employee.dept = Techdept.dept
GROUP BY Employee.dept
```

```
SELECT ssnum
FROM Employee, Temp
WHERE numfriends = numcolleagues
AND Employee.dept = Temp.dept;
```

- What is going wrong?

The Count Bug

- Consider for example an employee Jane:
  - Jane is not in a technical department (`Techdept`).
  - Jane has no friends (`Employee.numfriends = 0`)
- Original (nested) query:
  - since Jane is not in a technical department, inner query is empty
  - but `COUNT(`$\emptyset$`)=0`, thus Jane is in the result set!
- Rewritten query with temporary table:
  - Jane not in a technical department and does not survive the join
  - thus Jane is not in the result set