## Chapter 5.6 Selection of Connection Sequence

In this Chapter, we take key problem into consideration: Choose Sequence for Join which relates to more than three Relations.

### Chapter 5.6.1 The Meaning of Left and Right Connection Parameters

***Introduction:***

When choose the sequence for Join, then we need to remember that Join Operator is mostly not symmetrical, therefore, seen from this kind of meaning, two represented Relations are totally different, the Join cost depends on which Relation represents which meaning.

***Example:***

For one trip Join, it read the smaller Relation into Main Memory, and form one type of structure, it called Hash Table, therefore it can be used conveniently to match tuples from other Relation. Then read other Relation, one block for each time, and start Join Operator for tuples in the Block and in the Main Memory.

***Definition:***

*Prerequisite:*

When try to choose one Physical Plan, we decide to use *One-Trip Join*.

* Choose the Smaller Relation and save it into Main Memory, then this kind of Relation is called *Construction Relation*.
* For Right Join Parameter, it is called *Query Relation*, and we try to match tuple in the Block with the already Saved Tuple in Main Memory.

***Here other Join Algorithm that are divided by parameters, included:***

1. Inner Loop Join, in that kind of Join, the left Parameter is the outer Loop Relation.
2. Index Join, in that kind of Join, there should have index in the Right Parameter.

### Chapter 5.6.2 Connection Tree

***Situation:***

When we try to Join two Relations, then we need to sort all Parameters. According to the normal custom, we need to choose the smaller Relation as the left parameter. The size of each parameter is important and has different meaning. This is normal, since the Query that relates to Join needs to choose at least one Attribute, and the Selection Operator makes estimation decreasing.

***Example:***

*For Query:*

SELECT movieTitle

FROM StarsIn, MovieStar

WHERE starName = name AND

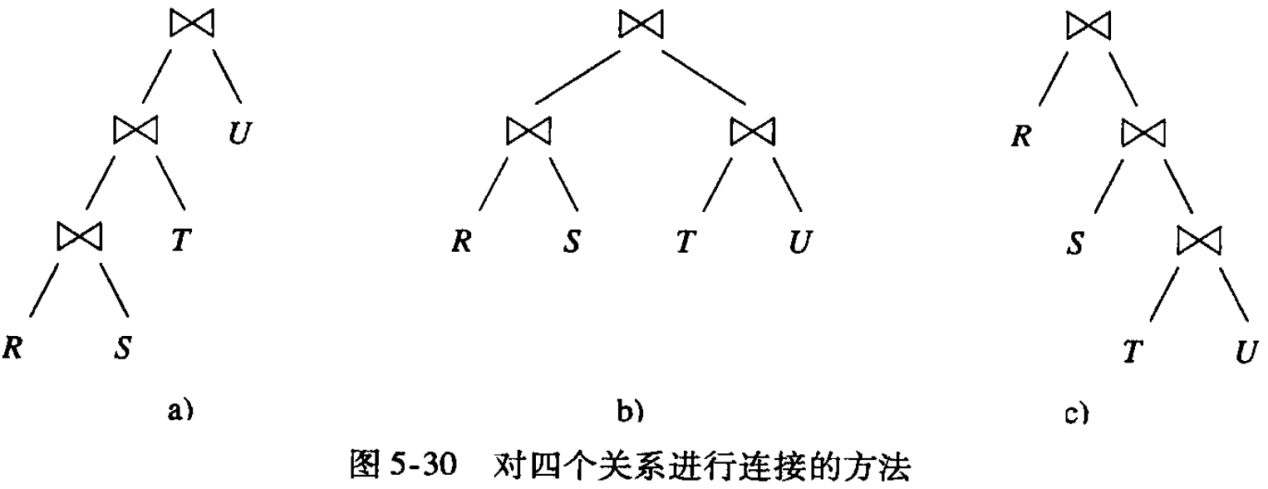
Birthdate LIKE ‘%1960’;

This is an example that we try to Join Relations include StatsIn and MovieStar, but there have not include any estimation size of Relations. We can think that one Selection will get almost 1/50 tuples among Relation MovieStar.

But since each Movie will include several Movie Stars, therefore we can assume that the size of StarsIn is far bigger than the size of MovieStar, and the second parameter of Join Operation *Selection ( birthdate LIKE ‘%1960’ ) MovieStar* is far less than the first Relation StarsIn.

As the result, we choose the MovieStar as the left Parameter which is different from the connection tree in 5 - 24.

When there have two Relations, and we can only choose one Relation as the left Parameter. When the choice relates to more than two Relations, and the number of Connection Tree will be increased quickly. For four Relations R, S, T and U, we try to connect these four trees and the possible tree structure as below. Normally, we will get 24 different trees.



### Chapter 5.6.3 Left Deep Connection Tree

### Chapter 5.6.4 Choose Connection Sequence and Grouping by Dynamic Programming

### Chapter 5.6.5 Dynamic Programming with Specific Cost Function

### Chapter 5.6.6 Choose Connection Sequence by Greedy Algorithm