

Database Systems
Dr. Sreenivasa Kumar
Department of Computer Science and Engineering
Indian Institute of Technology-Madras

Lecture-16
Aggregate functions

Ok, so let us get started. So in the last lecture, we were talking about various features in SQL for expressing queries.

(Refer Slide Time: 00:35)

Aggregation of Data

Data analysis

- to get info on summary and trends in certain attributes
- need for computing aggregate values for data
- total value, average value etc

Aggregate functions in SQL

- five aggregate function are provided in SQL
- AVG, SUM, COUNT, MAX, MIN
- can be applied to any column of a table
- can be used in the *select* clause of SQL queries

NPTEL

Prof P Sreenivasa Kumar
Department of CSE, IITM

37

We also talked about the operations that are available for combining sets of tuples. So we looked at union insert and except keywords. And the use of these will always result in a table which has by default the which removes the duplicates by default because the results are sets of tuples ok. So if you want to actually for some reason you know retain the duplicates in the results.

Then you can actually you know attach a keyword called all to this operators union, all, intersect all and accept all. So if you attach this key word then you kind of indicating that I want the duplicates to be present ok, so that is one additional point that I want to convey. Now moving on we have a very important aspects of data, which is what is data analysis we want to sometimes want to get some summary information or the trends in certain kind of attributes.

Like for example, amount of sales done or the number of units sold or things like that so. So for this we often do not want the database typically has detailed information, salary of every person etc. So but we want aggregation information, summarized information, what is the average salary you know what is the maximum salary we are paying, what is the minimum salary we are paying things like that.

So oftentimes we want to compute these aggregates they are called aggregates. So how does SQL support that is the next topic that we are going to talk about. So the following aggregate functions are kind of available in SQL. So SQL gives you these 5 aggregate functions and they are written like this. So this stands for, as you can guess it is average and then sum of values and count of the number of items in a particular sub query or yeah typically even an attribute.

And then maximum value, the minimum value is independent. So the max min obviously can be applied to attributes for which they are applicable right, so numeric kind of attributes. So in general, these things can be applied to any column of a table and then they are used in the select clause of SQL to kind of report those values. So we will see how exactly we will use this features.

(Refer Slide Time: 04:25)

Aggregate functions

- **AVG ([DISTINCT]A):**
computes the average of (distinct) values in column A
- **SUM ([DISTINCT]A):**
computes the sum of (distinct) values in column A
- **COUNT ([DISTINCT]A):**
computes the number of (distinct) values in column A or no. of tuples in result
- **MAX (A):** computes the maximum of values in column A
- **MIN (A):** computes the minimum of values in column A

Optional keyword

NPTEL

Prof P. Srinivasan Kumar
Department of CS&IT, IITM


38

Now let me give you a little bit more about this aggregate functions. So average can be within braces you will need to mention the attribute name. And you can optionally you know prefix it

with this word called distinct. If you want to compute the average of the distinct values in that particular column. Or if you want to so depending on whether you want all values in the column or only the distinct values in the column, you can use this key word to indicate that.

So it is an optional keyword where in a similar way computes the sum of values in a column. And count computes the number of values in a column or number of tuples in a result. And MAX computes the maximum value in the column A, maximum of values in the column A and minimum of values in column A. So these are the aggregate functions that are available. If you need any other aggregate function, then the only way is for you to kind of supply your own function and then kind of invoke that but we will see that much later ok.

(Refer Slide Time: 05:51)



Examples involving aggregate functions (1/2)

Suppose data about GATE exam in a particular year is available as a table with schema

gateMarks(regNo, name, sex, branch, city, state, marks)

Obtain the total number of students who have taken GATE in CS and their average marks

```
Select count(regNo) as CsTotal, avg(marks) as CsAvg
from gateMarks
where branch = 'CS'
```


CsTotal	CsAvg

Get the maximum, minimum and average marks obtained by Students from the city of Hyderabad

```
Select max(marks), min(marks), avg(marks)
from gateMarks
where city = 'Hyderabad';
```

Prof P. Sreenivas Kumar
Department of CSE, IITM

39



So let us start looking at some examples involving these aggregate functions in order to clearly understand what is there. So we let us say for this discussion let me introduce a particular relation. Let us say we have this you know the GATE exam the graduate aptitude test in engineering exam. Let us say the data about the exam in a particular year is available as a relation with this following schema.

So GATE marks, registration number is the key and then name, sex, branch what is the; is it computer science, ECE, etc civil engineering. So what is that branch of engineering in which the

GATE exam has been taken and what is the city of origin of that person. And what is the state of in which that city exists and what is the marks that is obtained by the student.

So this let us assume that this kind of data is available and we will see how exactly we can make use of this to compute various items of interest. So there can be a query like this obtained the total number of students who have taken GATE in computer science and their average marks particularly say you want to get that. So the way you can express this is like this, so you want to your particular about the branch computer science.

So branch so where branch CS from the only relation is gate marks and now here we are using the aggregate functions. So count of the registration number and we can rename that as CS total. And here is again the aggregate function being used average of marks column as CS average ok. So this is how we will express this query and what will be the result of this what do you expect in the result.

Remember the results of SQL queries are always tables right. So there will be a 2 column table with CS total and CS average as the column names and the corresponding values, so that will be the output for the thing. So output will be CS total, CS average and appropriate values whatever is the values ok. So it is very simple to make use of this aggregate functions, you will use them in the select clause and then you can you know give new names.

Or if you do not give new names, it will simply put count underscore registration number here and then average underscore marks something like that. So the names will be automatically derived from the aggregate functions and the attribute names you have used ok. So the interpretation is as usual the where clause will be applied and for all the tuples that satisfy the where clause in that set of tuples we will compute these averages or whatever is ask for.

In this case count of the number of tuples and then also the average of the marks. Get the maximum minimum and average marks obtained by students from the city of Hyderabad. So now we do not bother about the branch but we just want to get the maximum minimum average


marks of students from the city of Hyderabad, so it is very simple similar. So in this case, I have chosen not to give any aliases I mean the really mean.

So max min average marks where city equals Hyderabad. Now you may want to actually you may not really be interested in just all the students from Hyderabad their max min average marks. But rather probably you are interested in you know can you give me the same information branch wise for all the branch for all the computer science people in Hyderabad, what is the maximum, what is the minimum, what is the average.

For all the civil engineering students from Hyderabad what is the maximum average. So you may want to do this is the kind of analysis that we are talking about. We may want to you know partition the data into appropriate groups and then apply these aggregate functions, that is another. So here right now we are talking about applying the aggregate function on the overall complete data that satisfies certain condition.

So let us see now how we can actually you know partition the data and then apply the aggregate function because that is also often of interest to us. In fact it is probably more of interest to us ok.

(Refer Slide Time: 11:20)



Examples involving aggregate functions (2/2)

Get the names of students who obtained the maximum marks in the branch of EC

```
select name, max(marks)
from gateMarks
where branch = 'EC'
```

Will not work


Only aggregate functions can be specified here. It does not make sense to include normal attributes ! (unless they are grouping attributes - to be seen later)

```
select regNo, name, marks
from gateMarks
where branch = 'EC' and marks = ANY
(select max(marks)
from gateMarks
where branch = 'EC');
```

Correct way of specifying the query

Prof P Srinivasu Kumar
Department of CSE, IITM

40



Before we go on to that, let us look at this query which illustrates a subtle point about using this aggregates. So get the names of students who obtained maximum marks in the branch of EC ok.

So we want names of students who obtained the maximum marks in the there will be several people who have obtained the maximum. Now in attempt like this where select name and max of marks where branches equal to easy, actually it does not work.

The reason is that when you are using aggregate functions, as you have seen in this previous slide. Whenever you are using aggregate functions, you only mention the aggregate functions here, projecting any other attribute actually does not make sense right. So if you say name or something like that here and then along with all these things, whose name are you referring to you know.

So it does not make sense to mix the ordinary attributes with these you know aggregate functions. So that is why this approach actually does not work and it is not correct to mention any attribute name. When you are using aggregate functions attribute names cannot be mentioned in this select clause unless there are what are called grouping attributes, I will talk to you about that in a little later.

So unless there are grouping attributes you cannot mention them here. So this will not work. So the correct way of expressing this query you could think about the correct way what do you think would be the correct way of expressing this get the names of students who obtain the maximum marks in branch of EC. You probably have to first get the maximum marks and then figure out who are those students who got that mark.


So this already I mentioned that aggregate functions only can be specified here. So the correct way of specifying this query is select registration number, name, marks from this GATE marks I think where branch is EC. Because we are interested only in EC students and marks of the student marks in the tuple is equal to what is this doing. This is a sub query that is actually computing the maximum marks again in branch EC.

So the correct way of handling this is to first or in a sub query find out the maximum marks. So which can be done easily as a select max marks from gate marks where branch equal to EC. So now you know what is that max marks, there is one value and so in the outer you are looking for

tuples whose branch is EC and the marks is equal to this maximum marks whatever is the maximum marks.

And if the tuple satisfies this conditions then you can project the registration number, name and again the marks. Of course this marks will be the same for all the people because that is the maximum marks. So this is how aggregate functions have to be used in queries of this kind.

(Refer Slide Time: 15:33)



Date Aggregation and Grouping


Grouping

- Partition the set of tuples in a relation into groups based on certain criteria and compute aggregate functions for each group
- All tuples that agree on a set of attributes (i.e have the same value for each of these attributes) are put into a group
- The specified aggregate functions are computed for each group
- Each group contributes one tuple to the output
- All the grouping attributes *must* also appear in the select clause
 - the result tuple of the group is listed along with the values of the grouping attributes of the group

Called the grouping attributes

Prof P Sreenivasu Kumar
Department of C&AI, IITM

41



Now let us move on to the issue of trying to you know group the attributes, group the data sorry. So partition the set of tuples in a relation or in an appropriate subset of the relation when selected using certain conditions into groups. Again this grouping again is based on a certain criteria and then compute the aggregate functions for each of these groups. So this criteria is not very complicated criteria, all that it does is to group all the tuples on a set of attributes and they are called the grouping attributes.

So how a grouping works is that, all the tuples that have the same values for these set of attributes, the corresponding attributes right or put into the same in the group. So let us say you have 2 attributes A, B, then you will examine all the tuples. And if the corresponding values of A and B are same for 2 tuples, the 2 tuples have the same A value and same B value, then they will be put into one group ok.

And how many groups will be there as many as there are distinct pairs of values for these attributes A, B. If the A, B together if the A B pair has say some 10 distinct values in the data, then there will be 10 groups 10 partitions. So that is how we partition the set of tuples and then for each of these partition, we will apply the whatever is specified as the aggregate function ok.

So now the number of tuples in the output will be equal to the number of groups that result after doing this grouping. So the specified attribute aggregate functions are going to be computed for each of these groups and each group contributes one tuple to the output. Now here is an interesting thing that these the grouping attributes must necessarily be available in the select clause.

Because they are the ones that you know kind of distinguish each group one the other group, their values distinguish the group from the other groups. Because all the tuples in one group have exactly the same values for this grouping attributes. So the grouping attributes will have to be reported and then what is the whatever aggregate function that was requested for the group can be reported.

So the aggregate functions will be reported group wise and each group is identified by the values of the grouping attributes. And that is in the schema output of the schema we will have the grouping attributes, we need to have the grouping attributes and then the aggregate function values. So the result of the tuple for the group is listed along with the values of the grouping attributes for that group ok.

(Refer Slide Time: 19:22)

Examples involving grouping(1/2)

Determine the maximum of the GATE CS marks obtained by students in each city, for all Cities. Assume 4 cities exist - Hyderabad, Chennai, Mysore and Bangalore.

```

Select city, max(marks) as maxMarks
from gateMarks
where branch = 'CS'
group by city;

```

Result:

City	maxMarks
Hyderabad	87
Chennai	88
Mysore	90
Bangalore	86

Prof P Sreenivasa Kumar
Department of CSE, IITM

NPTEL

42

So let me illustrate this with examples. Determine the maximum of the GATE computer science marks obtained by students in each city and for all cities. Assume that 4 cities exist right, I am just since we do not know the data I am giving you a hour of time that there are only 4 cities available Hyderabad, Chennai, Mysore, Bangalore let us say ok. So what does it mean now, how do we proceed here.

So we have the same gate marks, now we want to group the tuples along the city value and there say 4 distinct city values. And so all the tuples that have the same city value Hyderabad will be one group, all the tuples that have Chennai as the city value will be one group etc. So the 4 groups will be created. So in this case there is only one grouping attribute city you may actually have more grouping attributes also.

Let us say I have city and sex as a grouping pair of attributes, then what we asking for is group the doubles as Hyderabad boys, Hyderabad girls, Chennai boys, Chennai girls, Mysore boys, Mysore girls right like that ok, you get the point. So depending on the grouping attributes that are mentioned, group the tuples such that they have the same values for those grouping attributes.

So in this case if you just mention city the values are 4 distinct values, so you group them by just the you get 4 groups. And now what is it that we are asking for, we are asking for maximum of GATE computer science ok. So the query actually now looks like this select city, city is the

grouping attribute it must appear because this is the one that distinguishes one group from the other.

And max marks as this renamed max marks from where branch is computer science. This is a new clause that we are introducing, it is called the grouping group by clause. So group by, it is typically followed by so ok this clause has group by and the sequence of grouping attributes. In this case this is only one grouping attribute, so you mentioned only that ok. So to kind of summarize what SQL provides you is a way of grouping the partitioning the data tuples.

And their way of and the you know sorry, the allowed way of partitioning this tuples is basically by giving a list of attribute names of those tuples ok. Now the result for this would be simple something like this somewhere whatever values. So there are 4 distinct values here and 4 groups and for each group we are projecting the we are picking out the. So this of course will be computed because all the tuples that have city value Hyderabad would be considered and their maximum will be computed etc.

Now in case you add one more grouping attribute here same branch computer science and city and sex if you put. Then city, sex, max marks will come. So then Hyderabad male whatever is the maximum, Hyderabad female whatever is the maximum like that the result will change ok. So this is one thing that is often used the grouping feature is often used by data analyze to figure out how the data is behaving in various you know subgroups ok.

(Refer Slide Time: 24:18)



Examples involving grouping(2/2)

In the University database, for each department, obtain the name, deptId and the total number of four credit courses offered by the department

```
Select deptId, name, count(*) as totalCourses
from department, course
where deptId = deptNo and credits = 4
group by deptId, name;
```

Prof P Sivasubramanian Kumar
Department of CS&E, IITM

43



Moving on, here is another example of using grouping attributes grouping. In the university database for each department obtain the name, department Id and the total number of 4 credit courses offered by the department. What is the total number of 4 credit courses offered by the department, get it for each department. Now you can see where is the data coming from, you want department name, department Id, so the department relation has to be used.

And where is the information about 4 credits of the course, number of credits for the course, it is in the course table. So you need to bring in course table and then check whether the course has 4 credits. And then group it by department ok, here is how you can mention obtain this thing. So select department Id, name and count star say count star is used if you, you know do not have to bother about which attribute to count.

But basically you are counting all the counting the tuples if counting the tuples, so count star as total courses. From the information is there from department and course and what is the appropriate combination here department Id should be equal to the department number. So department uses department Id, course uses department number to indicate us to what is the offering department for the course and credit should be equal to 4.


So this will give you all the combinations of departments and courses in which the course is being offered by that particular department and the course has 4 credits. Now this entire chunk of

data is being grouped, how is it being grouped, it is grouped by department Id, name. So it is now all the tuples that have the same department Id and the name value, department Id, name value will be put into one group.

And then you obtain apply the aggregate function. So what is the aggregate function is simply count here that so that is how we are finding out how many number of 4 credit courses are being offered by that particular department. Now since you want to report name and department Id together in the output, you need to put them as grouping attributes. It is actually the same groups will result even if you just do a group by department Id.

Because it is a key right but then you want to report name also and so you will put that as a grouping attribute. So what will be the schema for this output department Id, name and total courses and there will be how many tuples, as there are a number of departments which is some 13 you know, 14.

(Refer Slide Time: 28:01)



Having clause

After performing grouping, is it possible to report information about only a subset of the groups ?


- Yes, with the help of *having clause* which is always used in conjunction with Group By clause

Report the total enrollment in each course in the even semester of 2014; include only the courses with a minimum enrollment of 10.

```
Select courseId, count(rollNo) as Enrollment
from enrollment
where sem = even and year = 2014
group by courseId
having count(rollNo) ≥ 10;
```

Prof P Sreenivas Kumar
Department of CSE, IITM

44



Now after performing this grouping, is it possible to report the information for only a subset of the groups, that is also another interesting thing. You have a lot of data and you have partition date. Now I am interested in only some partitions, I am not really interested in all the groups, so for this we have a new clause called as the having clause. So the having clause will allow us to

specify one more condition which the group has to satisfy in order that group to be reported in the result ok.

So we are one more level we are going, so report the total enrollment in each course in the even semester of 2014 ok, total enrollment in each course. So we want course wise enrollment details, insert in so and so semester. But then we are not interested in those courses that do not have a minimum of 10 students do not bother about small kind of courses, report to me about the enrollment of courses that have at least 10 students.

Now obviously this condition that include only courses with minimum enrollment of 10 can only be applied after you get the aggregate function after you get the count of the number of students in that course, right. So after you do the grouping, then you get the value, then that value should satisfy certain condition, that is what we are querying. So the usual where clause will not be able to handle this, where the where clause can put conditions on the normal attributes.

It cannot put conditions on this aggregated values because they are not available in the data. They have to be first computed and only then you can use them for any checking of the conditions. So that is why we have a new clause called the having clause and what the having clause allows you to do is to select those groups which satisfies certain condition and then report information about them.


So in that condition obviously the grouping attributes as well as these aggregate or function values can be used in specifying those conditions. So here is how it can be done, so here is a new clause, so let us first understand the entire query. Select course Id, count of roll numbers as the enrollment total. The information is all available in the enrollment relation where the semester is even and the year is 2014.

And having selected the appropriate enrollment tuples do a group by course Id to get the group wise I mean sorry course wise enrollment in information. And then the count of that will be applied. So while reporting, since there is a having clause while reporting you report only those groups that have this specific value count roll number is at least, that is what this is. So the result

of this will be course Id and enrollment for all those courses offered in even semester 2014, provided those courses have at least 10 students ok.

Now one has to be careful about you know specifying these conditions, there may be a little bit confusion about whether the condition should be specified in the where clause or the having clause. But if you understand you know the flow of how the query will be, you know evaluated, **you would** you would not have this confusion.

(Refer Slide Time: 32:43)




Where clause versus Having clause

- Where clause
 - Performs tests on rows and eliminates rows not satisfying the specified condition
 - Performed *before* any grouping of rows is done
- Having clause
 - Always performed *after* grouping
 - Performs tests on groups and eliminates groups not satisfying the specified condition
 - Tests can only involve grouping attributes and aggregate functions

```
select courseId, count(rollno) as enrollment
from enrollment
where sem = 2 and year = 2014
group by courseId
having count(rollno) >= 10;
```

Prof P. Srinivasan Kumar
Department of CSE, IITM

45




So if where clause and having clause are both present, how exactly this happens. So where clause performs tests on rows and eliminate rows that are not satisfying the specified condition as usual. And this one is performed before grouping of the rows is done ok, this testing is done before any grouping of the rows is done. Whereas the having clause is always performed after the grouping.

And it performs tests on groups and eliminates groups that are not satisfying the specified condition. And these tests can only involve the grouping attributes as well as the aggregate functions. Now in this test you cannot use the normal attributes because they do not make sense, the groups do not have those attributes, right. It they only have this grouping attributes and the aggregate function values.

So these tests can be specified using these, so that is what we have done here count of roll number. So try out some more queries using so these are the interesting you know constructs definitely very useful constructs for analyzing data. And you typically analyze data by you know doing a lot of grouping and aggregation. And then get to know the trends and things like that. So you want to find out what is the city that contributed most to the sales of cosmetic items of in the last quarter.

So you want to figure out first you know figure out the last quarter means last 3 months, I mean whatever is the quarter in definitions. And then cosmetic type of things, there is all come in where clause. And then you do a group by the city and then figure out what is the city and then city wise you report then you can figure out which city has contributed most. So a lot of these analysis kind of queries require you to use group by and having clauses and they are very important kind of constructs in no SQL.

(Refer Slide Time: 32:30)



String Operators in SQL


- Specify strings by enclosing them in single quotes
e.g., 'Chennai'

Common operations on strings –

- Pattern matching – using 'LIKE' comparison operator
 - specify patterns using special characters –
- Character '%' (percent) matches any Substring
e.g., 'Ram%' matches any string starting with "Ram"
- Character '_' (underscore) matches any single character
e.g., (a) '____nagar' matches with any string ending with "nagar", with any 3 characters before that.
(b) '____' matches any string with exactly four characters

Prof P Sreenivasan Kumar
Department of CSE, IITM

46

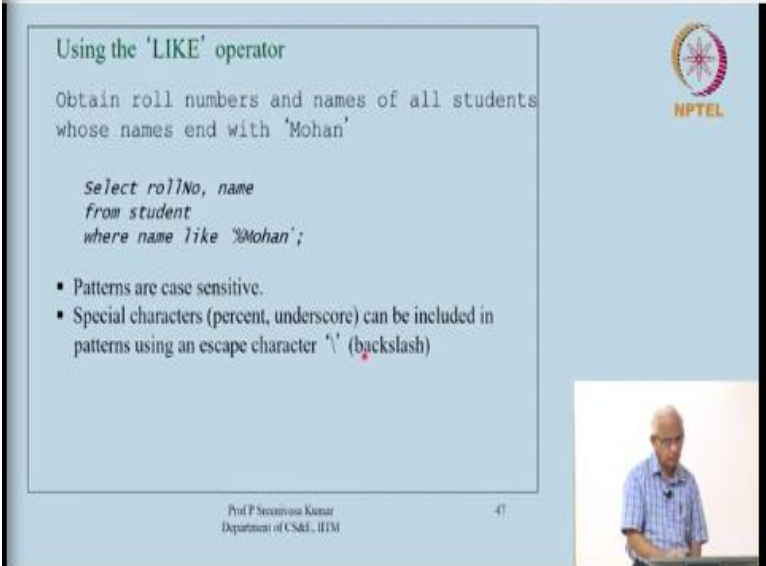


Now we are kind of more or less done with the main you know features of SQL. So let me just wind up with few of these other things that we can do using SQL, you can specify, you can use strings. There are some string operators in SQL and so you can do pattern matching using like which is a comparison operator and then specify patterns using. So this like comparison operator can be used to compare an attribute with a pattern.

So and in the pattern, you can also use you can use percentage symbol which matches any substring. For example, if you can say Ram% matches any string that starts with Ram, Ram Kumar, Ram, Prabhu whatever. And then the underscore character matches any single character, so there are certain pattern matching features. So these 3 underscores followed by nagar matches any string ending with nagar but has exactly 3 characters before that like Ram nagar something like that.

So this underscore 4 underscores matches any string with exactly 4 characters, so you can use these various functions all these like operator and the patterns while checking the value of an attribute whether it is equal to something or you know it is like this pattern.

(Refer Slide Time: 37:26)



Using the 'LIKE' operator

Obtain roll numbers and names of all students whose names end with 'Mohan'

```
Select rollNo, name
from student
where name like 'Mohan';
```

- Patterns are case sensitive.
- Special characters (percent, underscore) can be included in patterns using an escape character '\ ' (backslash)

NPTEL

Prof P Sreenivas Kumar
Department of CS&E, IITM

47

The slide features a light blue background with a white rectangular box containing the title and query. The NPTEL logo is in the top right corner. A small video inset in the bottom right shows a man in a blue shirt. The footer contains the professor's name, department, and slide number.

So here is some examples, obtain the role number names of all students whose names end with Mohan or begin with Mohan whatever we can specify like this. So these patterns are of course, case sensitive and then these special characters this percentage underscore. Sometimes if you want to use them then you need an escape character. So backslash is the escape character for using them within the patterns itself ok, so that is like operator.

(Refer Slide Time: 38:04)

Join Operation

In SQL, usually joining of tuples from different relations is implicitly specified in the 'where' clause



Get the names of professors working in CSE dept.

```
Select f.name  
from professor as f, department as d  
where f.deptNo = d.deptId and  
d.name = 'CSE';
```

The above query specifies joining of professor and department relations on condition $f.deptNo = d.deptId$ and selection on department relation using $d.name = 'CSE'$ •

Prof P Sreenivasa Kumar
Department of CSE, IITM

48



So here is a feature that you know in some sense goes slightly away from a spirit of SQL but nevertheless it is there. So far we have seen that we do not you know indicate as to which condition is a join condition, which condition is a select condition things like that, all conditions are there in the where clause. So for example if you say get the names of professors working in the CSE department.

So this is how you the query can be specified and you very well know that I mean from our background now of having studied relational algebra and all that. We know that f dot department number because d dot department Id is actually a join condition, this is a joint condition. Whereas d dot name equals CSE is a selection condition. So while it is not you know highly recommended.

(Refer Slide Time: 39:14)

Explicit Specification of Joining in 'From' Clause


```
select f.name  
from (professor as f join department as d on  
      f.deptNo = d.deptId)  
where d.name = 'CSE';
```

Join types:

1. inner join (default):
from (r₁ inner join r₂ on <predicate>)
use of just 'join' is equivalent to 'inner join'
2. left outer join:
from (r₁ left outer join r₂ on <predicate>)
3. right outer join:
from (r₁ right outer join r₂ on <predicate>)
4. full outer join:
from (r₁ full outer join r₂ on <predicate>)

Prof P. Srinivasan Kumar
Department of CS&IT, IITM

NPTEL




But SQL does allow you to kind of introduce explicit specification of joining in the from clause. So select f dot name from here is a you know kind of a temporary relation being created on the fly by doing a join between 2 tables with the join condition. So professor has f join, so this is a keyword join department as d, on this is also a keyword and the join condition. And then you go ahead give the other conditions like.

So this is how that the same query can be alternately expressed but we do not you know routinely recommend this kind of thing to be done. This feature I think is essentially required when you want to you know perform other kinds of joins rather than the normal inner join. The inner join can be taken care of by specifying the joining conditions etc in the where clause itself.

But if you want to really do the other joints, the outer joints then there is no appropriate way to specify that in SQL. So the default is inner join and one can specify outer joins like left outer join, right outer join and full outer join like this. So in the from clause from r 1 left outer join r 2 on predicate ok. So when you are doing this like left outer join, right outer join, full outer join, then there is really no alternate row other than specifying it like this.

Unless of course you can use sub queries in order to produce this outer joins also. But it is kind of convenient to say that ok we are interested in keeping all the attributes of the left relation and that is the kind of joins that we are looking for, so left outer join.

(Refer Slide Time: 41:37)



Natural join

The adjective 'natural' can be used with any of the join types to specify natural join.

FROM (r_1 NATURAL <join type> r_2 [USING <attr. list>])


- natural join by default considers all common attributes
- a subset of common attributes can be specified in an optional USING <attr. list> phrase

REMARKS

- Specifying join operation explicitly goes against the spirit of declarative style of query specification
- But the queries may be easier to understand
- The feature is to be used judiciously

Prof P Sreenivas Kumar
Department of CSE, IITM

90



And natural join can be used with any of these join types by specifying the keyword natural, natural and then the join type and then r_1 natural, join type r_2 . And then there is an optional keyword called using attribute list in case you want to we will consider that a little later. But what natural join basically does is considers all the common attributes between r_1 and r_2 and then enforces equality.

So in case you want to take a subset of these common attributes then that can be specified using this optional using keyword and you can specify a natural join also. So these specifying this join operation kind of explicitly goes as you can see goes against the spirit of you know declarative specification of queries in SQL. But the positive side is that sometimes the queries will be easier to understand.

So use this feature judiciously do not go about using it just because it is there, do not go about kind of using it without a proper reason for using ok.