**2025-3-5 COMPSCI751**

**英文原文 & 中文翻译**

第一段 (Paragraph 1)

English:

So, yesterday we are progress very fast, but things like we have covered normally the basics of the database. The relational model would cover the first part of the algorithm, including single theory of the relation name, all writers, filter slides, combine the relations such as “select.” You guys remember the “less,” which is the signal and the combination, and then followed by one relation name, and that’s going to choose the roles of the relation based on the predicate with the conditioner. And the project, we choose the collars denoted by a capitalized pipe, and then you listed the attributes you would like for your instant, followed by another expression or a relation itself.

中文：

那么，昨天我们进展得非常快，不过我们通常已经涵盖了数据库的基础知识。关系模型会涵盖算法的第一部分，包括关于关系名的单一理论、所有写法、过滤展示，以及合并关系（例如“select”）。你们记得那个“less”吗？它是一个信号加组合，然后后面跟着一个关系名，用来根据带有条件的谓词来选择关系中的行。而投影这个操作，我们用一个大写的竖线来表示，然后你会列出你想要的属性，接着再跟一个表达式或者直接跟一个关系本身。

⸻

第二段 (Paragraph 2)

English:

And cross product: so you have two relations, and you want to create a new relation where the attribute set of the new relation is—it’s like a multi-set union. It’s like combining all the attributes of two relations together, and if there are duplicated attributes, in order to duplicate it, you just indicate the source. Remember that? So here, instructor ID, course ID, and then you know you will have one relation has A attributes, another relation has B attributes, then you have A plus B attributes. In terms of the number of tuples—attributes is column, tuples is row—and in terms of number of tuples, the number of tuples in the first relation times the number of tuples in the second relation.

中文：

再来说说笛卡儿积（cross product）：如果你有两个关系，你想要创建一个新的关系，那么新关系的属性集合就像一个多重集的并集，类似于把两个关系所有的属性合并在一起。如果遇到重复的属性，为了区分重复，你只需要指明它们各自的来源。还记得吗？比如这里有instructor ID和course ID，然后其中一个关系有A个属性，另一个关系有B个属性，那么合起来就有A+B个属性。至于元组的数量（属性是列，元组是行），新关系中的元组数就是第一个关系元组数乘以第二个关系的元组数。

⸻

第三段 (Paragraph 3)

English:

They also talk about natural join. There are two properties in the natural join: (1) they have an implicit condition where common attributes should have the same value, and (2) the attributes with the same name will be removed—I mean, they will only keep one copy of these attributes because they are the same. Anyway, “θ-join,” on the other hand, is a selection of a condition (any condition) over a condition product, but with this expression, it’s abbreviated using this expression where you have expression1, expression2, both times with the subscription detail of synthetic pollution. The more you ask me what the θ be: the θ could be something just like here, something like this, the common name equals this, and this salary is greater than 9,000, and this kind of pollution that is composed with comparisons and logic connectives.

中文：

他们还谈到了自然连接（natural join）。自然连接有两个特性：(1) 它有一个隐含的条件，即公共属性的值必须相同；(2) 对于同名的属性，它们会被合并——也就是说，这些同名属性只保留一份副本。无论如何，“θ连接”则是对条件笛卡儿积进行选择，也就是说，你可以在这个表达式上加一个条件，这个写法可以用表达式的形式简写，你会看到expression1和expression2，中间加上下标，表示一些比较复杂的语法。至于你问 θ (theta) 可以是什么？它可以像这里一样，比如“公共名字相等并且薪水大于9000”之类的条件，这种条件通常是由比较和逻辑连结组成。

⸻

第四段 (Paragraph 4)

English:

With this, we are able to do a little bit of exercise. So, today we want to start off with a strong exercise, where I’m giving you 3 relations: one is “employee,” and employee has ID. So, this ID should be the private key of the single E, right? Next level, it should be the primary key of the single E. And then you have this person’s name. This screen and say: you guys remember what does it mean by primary, or what does it mean by key? And the benefit is unique, right? So you can identify this tuple by this key value. And this ID is obviously a key. The “works” is a relation describing a person works for some company with a salary, right? So, ID, person’s name, company name, and salary. And then you have relations, right? The company itself. So every company has a company name, and it’s… But let’s think about the three queries I’m going to give; then we’ll come to discuss.

中文：

有了这些内容，我们就可以做一些小练习了。今天我们想开始做一个比较有挑战性的练习，我会给你们3个关系：一个是“employee”，这个employee里有ID。这个ID应该是这个雇员实体的私有键，对吧？再往下一层，它应该是雇员的主键。接着你还会有这位雇员的名字。这里我也提到屏幕，你们还记得主键或者键是什么意思吗？它的好处就是唯一性，对吧？可以用这个键来标识一个元组。这个ID显然就是一个键。然后还有“works”这个关系，描述了一个人在哪个公司工作以及他的薪水。所以这个表里有ID、人的名字、公司名称和薪水。然后还有另一个关系，就是“company”本身。每个公司都有一个公司名，然后……不过让我们想想我要给你们的三个查询，然后我们再讨论。

⸻

第五段 (Paragraph 5)

English:

Okay, I’m going to use this. I think the educator doesn’t work using document camera to do the demonstration. You guys can hear me cleaning. So for the first query, anyone can tell me your answer? “Yes, π ID, name, (brackets, bracket) = = to, speed, big bank…” I’m not going to write it here. Another question of “works.” That’s it, right? Yep. How many of you agree with this? Yep, perfect. So, I think this is correct because in this query, all we need to know is: so the ID and name are the columns, are the projecting operator, and then, would you like to select employees who work for Big Bank? Fortunately—oh, this one is not that updated—please look at the… so the other one has ID, so this printed is not the correct version. So with this, you will have the ID and the name of the Employee Natural… so in this case, we only need to focus on one relation, which is “works.” Nice, very good.

中文：

好的，我要用这个。我想这个演示仪器可能不能用文档摄像机来演示。你们应该能听到我在清理。所以关于第一个查询，有人能告诉我答案吗？“是的，π ID, name，（一系列括号）== speed, big bank…” 我就不在这里写出来了。然后还有“works”的另一个问题。对吧？对。有多少人同意这个做法？好的，完美。所以我认为这是对的，因为在这个查询里，我们只需要知道的是：ID和name是投影操作符所投影的列，然后我们想筛选那些在Big Bank工作的员工。幸运的是——哦，这里打印出来的版本不是最新的——请看……另一个版本里有ID，所以现在显示的版本不太正确。但不管怎样，这样你就能拿到Employee的ID和name的自然……在这种情况下，我们只需要关注一个关系，也就是“works”就行了。很好，非常好。

⸻

第六段 (Paragraph 6)

English:

How about B? ID, name, company name, equals your Big Bank, and then “work works and employee and employee and logic and and record.” So, this “and” operator—“and” operator can only connect two conditions. Yeah, so the standard operator is a logical operator that can only be used in predicates or filter conditions. So if you want to combine two relations, you can either use condition product or natural join or θ-join. So if you use condition product, then you need to specify the condition that the ID should agree with each other. But if you use a natural join, then you don’t have to do that because it’s implied. So the sickest way is to use a natural join. Well, also you can use a condition product with a selection or a θ-join. Yeah, they all work. “Yes, so on for the what’s not company name… That one, like the work… sorry, is a company name. After Sigma, do we need to ‘and’ it, like what’s not company? Employee doesn’t have a company name, what dog company?” So the thing is, if you do the both talking—because we haven’t joined companies in this instruction, right? So all we are focused on are just these two relations. And in these two relations, there is only one attribute called company name, and in this case, you don’t have to disambiguate because there is only one company name. So this is one possible answer. Well, you have other alternatives as long as they are stating the conditions correctly.

中文：

那B呢？ID, name, company name，等于你的Big Bank，然后“work works和employee和employee和逻辑and and record”。其实这个“and”操作符——“and”操作符只能连接两个条件。对，所以这个标准操作符是一个逻辑操作符，只能用在谓词或者过滤条件中。如果你想组合两个关系，你要么用条件笛卡儿积(condition product)，要么用自然连接(natural join)，或者用θ连接(θ-join)。如果你用条件笛卡儿积，那你就得指定一个条件，确保它们的ID相等。但是如果用自然连接，就不需要这样做，因为它本身就隐含这个条件。所以最简洁的方式就是用自然连接。当然，你也可以用条件笛卡儿积结合一个选择操作，或者用θ连接。对，这些都可以。“是啊，那对于company name怎么处理……比如说work……啊不好意思，是company name。做完Sigma以后，我们需要加and之类的吗？Employee里并没有company name，那company这个表里如何？” 其实是这样的：如果你把它们都写在一起——因为在这一步我们并没有和“company”表做连接，对吧？我们现在只关注这两个关系。而在这两个关系里，只有一个属性叫company name，所以这时候就不需要做歧义处理，因为只有一个company name。这是一种可能的答案。只要你的条件表达正确，其他写法也是可以的。

⸻

第七段 (Paragraph 7)

English:

How about C? Well, if you look at this, the difference between B and C is street address. So in the project, in the projection, you need to add a street address, and it earns more than 10,000, right? So there are two differences: one, you need to add a street name here—okay, I’m not gonna write it here—and another one is to, like, make sure the salary is more than 10,000 when you do the selection here over the join. Okay, so here you should add up “and then salary > 10,000.” Oh, that’s the little exercise. But the reason we are doing this exercise is: relational algebra is a formal way we press our query, while later on, we demonstrate the SQL query that they’re equivalent. They’re basically equivalent, and the logic behind this is the same. So you would like to express some conditions, selections over the relation, and what they are doing is exactly the same. “What if the C—if I remove the city from the employees—what is the same? If we remove the city from employees…” So things like the city of residence, but because you haven’t… so in these three queries, you only use the two relations, so this city will not incur any complication here because this relation has not been used. Later on, we will have another example to use this city.

中文：

那C呢？如果你看看，B和C之间的区别是街道地址。所以在投影的时候，你需要把街道地址也加进去，而且要求薪水大于1万，对吗？所以有两个差别：第一，你得在这里添加一个街道名，好吧，我就不在这儿写了；第二，你要确保在连接时做选择的时候，薪水大于1万。好的，所以你应该在条件里加一个“并且薪水 > 10,000”。哦，这就是一个小练习。我们之所以做这个练习，是因为关系代数是一种对查询进行形式化表达的方式，之后我们再示范对应的SQL查询时，你会发现它们是等价的，基本上是等价的，背后的逻辑是一样的。你会想以某种方式表达过滤条件、选择条件，这些操作实际做的事情是一样的。“那如果C——如果我把员工表的city这个字段去掉呢——还会一样吗？如果在employee表里删掉city……”类似居住的城市之类的字段，但是因为你在这三个查询里只用了另外两个关系，所以这里city不会带来任何影响，因为这个关系并没有被用到。等之后我们就会有其他例子来用这个city。

⸻

第八段 (Paragraph 8)

English:

Have we talked about the separation in the condition product, in the natural join, in the θ-join? We are combining two relations in a horizontal way. Why is it horizontal? Because we’re expanding the width of the relation. So the resulting relation will have more attributes than any particular relation because we’re combining them. Especially when you look at the condition products, the width of the resulting relation will be, you know, the addition of the width of the two relations together. So we call it “horizontally combine two relations.” So sometimes we would like to combine the relations vertically—vertical in the way that these two relations have exactly the same schema. You can union them, or like, do set operations like intersections and differences, so supports. So basically, not only relational algebra but also SQL queries are all providing set operations.

中文：

我们之前有没有谈到在条件笛卡儿积、自然连接以及θ连接里对关系的区分？其实我们是在“水平”地合并两个关系。为什么说是“水平”的呢？因为我们在扩展这个关系的宽度。所以合并后的关系往往会包含比任何一个源关系都多的属性，因为我们把它们合并到一起了。尤其是在条件笛卡儿积里，结果关系的属性数就是两个关系属性数相加。所以我们称之为“水平”合并两个关系。有时候，我们也想“垂直”地合并关系——所谓“垂直”，就是这两个关系有完全相同的模式(schema)，那你就可以对它们做并集(union)或者像交集、差集之类的集合操作。所以从本质上来看，不仅关系代数能做这些操作，SQL查询也都提供了这些集合运算。

⸻

第九段 (Paragraph 9)

English:

Here is an example: let’s say, find the names of the instructors and the names of the departments. What we do is: we do a projection of the instructor to the name—that’s the instructor’s name—well, it can be Einstein or whoever, Gould or Susskind, and project department to department’s name, and then we union them together. So here, it’s a little bit problematic because they don’t share the same schema. I mean, later on, we’ll go introduce a new mechanic called rename to make sure that the two relations are having the same schema before they do the union. So here, it’s generally logically correct and not syntax. Similarly, we can go intersection. So, find a name that is both an instructor name and a common name—who could this be? Well, this is a toy example, but who could this be? Maybe Washington or Turing or these, like, names they commonly name after famous people, right? So what kind of query could that be? You just do projection, same, and do… But the third operation is the set difference operation. The example is like if you want to find the name of the department that has no instructors, what would you do? Well, some of you may have already realized, you project the department to the department name, and you project instructor to their department name, and it could be different, because some department may have no instructors. And then you do the set difference by going to find the department name that appears in one set and not the other set, and to indicate to throw out those ones. So this is the expression.

中文：

这里举个例子：比如说，找到所有教师（instructor）的名字和所有系（department）的名字。做法是，我们先对instructor做投影，投影到“name”这个属性——那就是教师的名字，可能是Einstein或者别的，Gould或者Susskind什么的；然后再对department做投影，投影到department’s name；接着把它们做并集。所以这里会出现一个小问题，因为它们在模式上并不相同。我的意思是，之后我们会介绍一个叫“rename”的新机制，让两个关系在做并集之前模式相同。所以这里在逻辑上没问题，但从语法上还不够严谨。相似地，我们也可以做交集。比如，找一个既是教师名字又是常见名字的人——这可能是谁呢？当然这是个玩具例子，不过也可能是Washington或者Turing之类，因为这些名字经常会被用作命名，对吧？那这种查询怎么写？其实你只要投影同样的列，然后做……但第三个操作就是差集操作。举个例子，如果你想找没有任何教师的系的名字，该怎么做？有些同学也许已经想到了，你先投影department这个关系得到所有department name，然后投影instructor得到instructor的department name，它们可能会不同，因为有的系可能没有教师。然后做差集，也就是从前者当中去掉后者当中出现过的系名，这样就能得到那些没有教师的系。这就是对应的表达式。

⸻

第十段 (Paragraph 10)

English:

Let me emphasize that these three operations are somehow, in terms of expressiveness, not that we begin—why we can just do away with this intersection to see how. Say, let’s say you have a set A and you have a set B, okay? We would like to use the set operations like intersection or union. We can address the problem of not matching schemas, right? We can just rename their schemas into the same one. So for example, we’re going to rename this relation to X, X is the relation name, A is the X is the name, and then the same to the other one, and then these two relations are sharing the same schema. Now they can be unioned or intersected or set-differenced without any problem. This is only a minor balance. There is a very important property (or benefit) brought by this rename operation: self-join. What does it mean by self-join? Just consider a query asking you to enumerate all the instructor pairs coming from the same department. Okay, we want you to give me the pair of instructors in the instructor’s name, or the instructors themselves (the tuple); they are from the same department. Well, if you think about it, it will require you to do this product, right? In this Cartesian product, it will be completely killed because they are from the same relation, and even if you want to distinguish where the ID is coming from, they are all coming from “instructor.” So how do you want to make a difference, or indicate which is which? Here you must use a rename. Okay, you must use a rename.

中文：

我想强调，这三个集合操作在表达能力上其实都是类似的，我们可以不用“intersection”也是能做的，看是怎么处理。假设你有一个集合A，还有一个集合B，对吧？我们要用并集或交集之类的集合操作时，确实会遇到模式不匹配的问题，但我们可以先用rename把它们的模式改成相同。举个例子，我们把这个关系改名成X，X是关系名，然后它的属性像A就变成X的属性名，另一个关系也做同样的处理，这样就能让它们的模式一致。现在它们就可以做并集、交集或者差集，都不会有问题。这只是rename在集合操作里解决的小问题。但rename带来的一个非常重要的好处是：自连接（self-join）。什么叫自连接？举个例子，如果你有一个查询，想要列举所有来自同一个系的教师对儿。OK，我们需要的是把这些教师成对儿给出来，或者列出他们各自的名字或者其本身的元组，但他们都来自同一个系。如果你仔细想，想要列出这样的对儿就得做一个笛卡儿积，对吗？可是在做笛卡儿积时，这两个关系实际上都是“instructor”本身。虽然你想区分哪一个ID来自第一个instructor，哪一个ID来自第二个instructor，但都是同一个表，所以怎么区分呢？此时就必须用到rename。对，必须用rename。

⸻

第十一段 (Paragraph 11)

English:

So by using a rename, we are using relation R1 with 4 attributes to replace the ID, name, department name, and salary, which is A1, B1, C1, D1, and we just rename it “this instructor.” The second one we use R2, A2, B2, C2, D2, and with this cross product (Cartesian product), we only need to apply a selection operator to make sure that C1 equals C2. We can say C1, C2 are or indicating to the same department. More conveniently, we can use natural join, and then by renaming these two relations’ identical attributes using the same name (so they are both C), we can do the natural join. So here is a little exercise. So with this expression, you will have some “anomaly pairs,” like Einstein will be paired up with Einstein, and Einstein paired up with… So how to remove this same scenario? So you want to add another selection, right, to make sure that they have different names or different IDs. Different IDs. So make sure that A1 does not equal A2. And there is another complicated scenario saying that Einstein has a pair with, let’s say, Gould, from the same department, and there is another pair which is Gould and Einstein. How are you going to remove that duplication? There is a very simple way of doing this, very simple: we require that A1 must be smaller than A2, and then you’re going to eliminate both scenarios.

中文：

因此，通过rename操作，我们就可以把这个名为R1且含有4个属性的关系，用来表示ID、name、department name和salary，这些属性分别叫A1、B1、C1、D1，我们可以把它重命名为“this instructor”。然后第二个我们用R2，属性是A2、B2、C2、D2。再通过它们做笛卡儿积（Cartesian product）后，只要加一个选择操作，让C1等于C2，就能确保它们来自同一个系。更简单的方法是用自然连接（natural join），然后把这两个关系里需要匹配的属性都改成同样的名字（比如都叫C），这样就能直接做自然连接。这里还有一个小练习：使用这个表达式后，会出现一些“异常配对”，比如Einstein会和Einstein自己配对，还有Einstein和……这些情况怎么去除？你可以再加一个选择条件，比如说它们的ID不能相同，让A1不等于A2就行。另一个情况是Einstein和Gould在一个系，那么就会同时出现Einstein-Gould和Gould-Einstein这两对，你怎么去除这个重复呢？其实很简单，只要要求A1必须小于A2，就可以一次性去掉这两种重复。

⸻

第十二段 (Paragraph 12)

English:

The beauty of the rename operation—self-join—also played a very important role in the join. Sometimes we are interested in the relation itself, so that’s why the rename operator is very well in its expressiveness. It has to replace—it’s a necessary operator here. So, assignment operation is just the operation that breaks down the relational algebra expressions to their parts. Instead of one lined expression, a very long relational expression, we can break it into shorter and more clean-up ways. So, for example, here, if we want to union the instructors who are just the instructors from Physics and Music, then we can just assign this selection of instructors to “Physics,” which is a new relation, and then the selection of instructors to “Music,” and then we union these two relations. So those, yeah, just make the expression easier to read.

中文：

rename操作和自连接的妙处，也在连接中发挥了非常重要的作用。有时我们只关注同一个关系本身，所以这就是为什么rename操作在表达能力上很有用——它是一个必要的操作。接着，assignment（赋值）操作则是把一个长长的关系代数表达式拆分成多个部分。与其写在一条非常长的表达式里，我们可以把它拆分成更短、更清晰的方式。举个例子，如果我们要对“Physics”系的教师和“Music”系的教师做并集，那么就可以先把满足“Physics”条件的教师选出来，赋给一个名为“Physics”的新关系；再把满足“Music”条件的教师选出来，赋给一个名为“Music”的新关系，最后再对它们做并集。这样一来，表达就变得更容易阅读了。

⸻

第十三段 (Paragraph 13)

English:

So, so far, we have introduced all the expressions and the expression tree. It is not like introducing new expressiveness to the whole relational algebra expression; it’s just a different way of viewing and showing the structure of this expression. So instead of just a plain line of expression, now you are going to see how these operations are organized. In this example, you will see a tree, and all the tree leaves are just relations, and the internals are the operators—operators either unary operator or binary operator. The thing is, I made a little simplification: this natural join actually joins these two and then joins the other one, or these two together, so the order actually I didn’t specify, but this is just the natural join. So after we do the natural join, we assume that all the attributes with the same name are agreed with each other, and then we perform a selection “building = 3 or 3.” If you think about it, this selection only applies to time, right? You actually can push it down to make this relation smaller at the very beginning before the join, because the join is pretty expensive if you think about the multiplication of the number of tuples, worst case. So it is equivalent to this experiment. So you’re pushing the selection down, and then you perform the join later on, and eventually you perform…

中文：

到目前为止，我们已经介绍了所有的操作以及表达式树。这并不是给整个关系代数添加新的表达能力，而只是一种展示表达式结构的不同方法。也就是说，不再是简简单单地把表达式写在一行里，而是让你能看到这些操作是如何组织在一起的。在这个例子里，你会看到一棵树，树的叶子节点都是关系，而内部节点都是操作符——有一元操作符，也有二元操作符。需要说明的是，我这里做了一点简化：这个自然连接实际上可以是先连接其中两个关系，再去连接另一个，或者直接把它们都连起来，其实我没有明确指出顺序，但它就是一个自然连接的过程。做完自然连接后，我们假设所有同名属性已经对齐，然后再做一个选择操作，比如“building = 3或3”。如果你仔细想，这个选择操作其实只作用在某一个关系上，对吧？那我们完全可以把它下推到连接前面，这样可以让要参与连接的关系在一开始就缩小规模，因为连接一旦做起来，最坏情况下是两边元组数量的乘积，开销会很大。所以把选择下推也能得到相同的结果。这就是所谓的等价转换。也就是你先把选择做完，再做连接，然后最后再……

⸻

第十四段 (Paragraph 14)

English:

Okay, so, so far, a little bit of a wrap-up. We have introduced several operators of relational algebra, such as the pretty essential operators like selection, projection, union, set differences, Cartesian product, natural join, inner joins we need, and we also show different ways of expressing relational algebra expression, like assignments and then expression tree. Here comes the tail part with a request: I’m going to give you, let’s say, 3 minutes to finish. Okay, coming up, who can tell me the result? I think it’s much easier this time, right, than the last time? Who would like to share the result?

中文：

好的，现在做一个小结吧。我们已经介绍了关系代数中的几个运算，比如非常重要的选择、投影、并、差、笛卡儿积、自然连接，以及所需的内部连接，还展示了几种表达关系代数的方法，比如赋值和表达式树。接下来我给你们一个练习，给你们3分钟做一下。好，谁能告诉我结果？我觉得这次比上次简单多了吧？谁愿意分享一下结果呢？

⸻

第十五段 (Paragraph 15)

English:

“Yes, hi, ID, person name, first name, θ, I… I thought city = = two, θ, yeah, Sigma, Sigma… oh, Sigma, yeah, Sigma, right, yeah. I.city\_employee = R2… rE… does = 2… city\_employee… dash employee, yeah, equals R2.city\_company, yeah. Then roll, roll, relay, yeah, R1. So R1, yeah, ID, person name, street, city\_employee… it’s a bracket, you have a bracket, yeah, bracket ID, ID, person name, person name, street, city\_employee… employees relation… ‘ah no no city no dash in employee… that means I want to change the first city to city\_employee, city… oh, oh okay, so you are renaming, yeah, the city is sitting here, right, yeah, okay… then employ, employ, employ relation, employ relation, natural join, natural join, natural join, natural join works, works, yeah, natural join R2, R2, yeah, yeah, and we name this city to city\_company, so this city also is company, yeah, company relation, and so everything the same except the city to be city\_company. City… is this the end? Yeah. So we do natural join. What should be the same? ID and person name, ID, person name, and company name. Company name should be the same. How about the city? And the city, we’d want it to be the same because they have different names.”

So basically, you have tried the assignment operator, which is very nice, and you tried the next one, which is also nice, and this is a very… you know how to draft statements right now. There is only one minor—it’s just a small minor: here, if you rename the city of employee and city of company to be different, which means that they don’t have to agree with each other, then how to satisfy the employee who lives in the same city as the company for which he works? “Find… Sigma… Sigma, yeah… oh, very much, I think this is correct, this is correct, this is correct. I mean, it’s an exercise for everything we’ve learned today. Yeah, I think it’s pretty, pretty nice exercise, although we have probably a simpler version… oh yeah, I have time, I know that, that’s fine. But this is correct. This is correct. So when you rename this, and you later on, yeah, indicate they need to be the same… and it’s also possible, like, you don’t rename them, yeah, and just a natural… just a normal join, and they will automatically indicate that they should be the same. Yeah. So, well, perfect, this is correct.

中文：

“是的，嗨，ID，person name，first name，θ，我……我以为city == 2，θ，对，Sigma，Sigma……哦，Sigma，对，Sigma，对的。I.city\_employee = R2…… rE…… does = 2…… city\_employee…… dash employee，对，等于R2.city\_company，是的。然后roll，roll，relay，是的，R1。所以R1里就是ID，person name，street，city\_employee……这里有个括号，对，你那儿有个括号，对，括ID、ID、person name、person name、street、city\_employee…… employees relation……‘啊不不，这里employee里没有city dash之类的……意思是我想把第一个city改成city\_employee，city……哦，好吧，你是在做rename，对吧？城市字段在这里，对吧，对，好……然后employ，employ，employ relation，employ relation，natural join，natural join，natural join，natural join works，works，对，然后natural join R2，R2，对，对，然后把这个city重命名成city\_company，所以这个city也就是company，对，公司关系，然后其他都不变，只是city变成city\_company。city……结束了吗？嗯。所以我们做自然连接。哪些要相同？ID和person name，对，ID，person name，还有company name。company name也要相同。那city呢？city我们也要相同吗？因为它们名字不一样嘛。”

基本上，你用了赋值操作符，很不错，然后你用了下一个操作符，也挺好，你现在知道怎么写这些语句了。只有一个小问题——真的是小问题：如果你把employee的city和company的city重命名成不同的名字，那就意味着它们不需要相同，那又如何表示“员工居住的城市和他所工作的公司所在城市相同”呢？“找……Sigma……Sigma，对……嗯，我觉得这样是对的，对的，对的。我的意思是，这是对我们今天学的内容的一个练习。对，我觉得这是个挺好的小练习，虽然我们也许能写个更简单的版本……哦，是的，我有空，我明白，这是可以的。但这确实是对的。对，所以当你做了rename，然后在后面再指定它们必须相等……也可以有其他方法，比如你压根不把它们改名，直接用自然连接就会自动要求它们必须相同。对。好吧，完美，这样是对的。”

⸻

第十六段 (Paragraph 16)

English:

And I think that would be for today. So, next lecture will be tomorrow. Tomorrow, we will have a short introduction to SQL, and then we will have a demo, and then we will have a demo to prepare for this. You are encouraged to bring your laptop, and I have uploaded a specification for you to set up your own laptop with a light database, either SQLite or DuckDB. SQLite is a very light DB, and DuckDB, on the other hand, is very trendy, a fashionable DB for data mining purposes, and it has quite a well-known database originated by the Netherlands. And this database currently has been widely used because it’s very easy to be adapted to machine learning and this kind of analytic tasks, and combined with data management tools. And that’s why we are including the database to be, you know, an experimental database. Well, thank you very much. I’ll see you tomorrow.

中文：

我想今天就到这里吧。下一堂课在明天。明天我们会先简要介绍一下SQL，然后我们会做一个演示，再做一个小实验来准备这部分内容。建议你们带上自己的笔记本电脑。我已经上传了一个文档，告诉你们怎么在自己的笔记本上安装轻量级数据库，比如SQLite或者DuckDB。SQLite是一个非常轻量级的数据库，而DuckDB则是现在非常流行的一种数据库，主要面向数据挖掘用途。它最初来自荷兰，现在用得很广，因为它很容易跟机器学习或者其他分析任务结合，也能跟数据管理工具搭配使用。这就是我们将它也纳入实验数据库的原因。好了，非常感谢大家，明天见。