**2025-3-6 COMPSCI751**

**英文原文 & 中文翻译**

English (Paragraph 1)

Um, hello everyone. Okay, so hello everyone. Yesterday, we talked about relational algebra; we finished all the introduction of the relational algebra operators, including selection, project, union, differences, Cartesian product, natural join, theta join, rename, assignment, and expression tree.

中文（第1段）

嗯，大家好。好的，大家好。昨天我们讨论了关系代数；我们完成了所有关于关系代数算子的介绍，包括选择、投影、并、差、笛卡儿积、自然连接、θ连接、重命名、赋值以及表达式树。

English (Paragraph 2)

With this, the reason we are introducing relational algebra is that it was paving the foundation of the very practical query language. The high-level query language that we’re introduced to today is called SQL—I mean, it’s SQL. Well, some people say it is like “sequel,” but I think it’s more professional to say SQL, right? So SQL is like a standard query language. It was—based on a model Edgar Codd proposed in the 1970s. And he not only proposed the relational model, relational algebra, and later, based on this, they developed the whole relational database system using the simple query. And, I mean, he received a Turing Award in 1981. Yeah, that’s the most prestigious award in computer science.

中文（第2段）

因此，我们之所以介绍关系代数，是因为它为一种非常实用的查询语言奠定了基础。我们今天介绍的高级查询语言叫做 SQL——我是说它就是 SQL。嗯，有些人会把它读成 “sequel”，但我觉得更专业的说法是 “SQL”，对吗？所以 SQL 就像是一种标准的查询语言。它是基于 Edgar Codd 在 1970 年代提出的模型而来的。他不仅提出了关系模型、关系代数，后来基于此，人们开发了整套利用简单查询的关系数据库系统。而且，他在 1981 年获得了图灵奖。对，那是计算机科学领域里最具声望的奖项。

English (Paragraph 3)

Okay, so SQL has a number of—so SQL, at the very beginning, it was just basic queries, and later on, it has been extended over and over again and become a standard SQL. So they have ISO standards; they have an ’86 version, ’89 version, ’92 version, and so on and so forth. So most of the commercial databases are using the ’92 version as the basic, and most of the ’92 functions will be provided. Some of the later standards will be added on but vary from one system to another. Some function may be provided by one system, not provided by the other system, so you need to understand different systems specifically.

中文（第3段）

好的，SQL 有很多内容——最初的 SQL 只是一些基础查询，后来不断扩展，最终成为了标准 SQL。因此它有 ISO 标准，有 1986 版、1989 版、1992 版等等。大多数商用数据库都使用 1992 版作为基础，并提供了 1992 版的大部分功能。一些之后的标准也会被添加进来，但在不同系统之间可能有所差异。有些功能可能在某个系统里提供，但另一个系统里却没有，所以需要针对不同系统进行具体了解。

English (Paragraph 4)

Okay, so just like today, we’re going to demonstrate using SQLite. SQLite is very lightweighted. SQL query language does not support some of the functions, like SOME or ANY or… that kind of function is not supported by SQLite. On the other hand, it was supported by DuckDB, or these things will be supported by other systems like PostgreSQL, and so on and so forth. Okay, so we’ll have a dip into this area today.

中文（第4段）

好的，就像我们今天要演示的那样，我们会使用 SQLite。SQLite 非常轻量化。SQL 查询语言在 SQLite 中并不支持某些函数，比如 SOME 或 ANY 等此类函数。另一方面，DuckDB 等数据库或者像 PostgreSQL 之类的系统就支持这些功能。好的，今天我们会在这方面进行一些探索。

English (Paragraph 5)

So SQL has mainly two parts. One is Data Definition Language, to define the relations, including the relation’s name, the relation’s attributes, the domain of the attributes, the constraints on the attributes. The constraint includes the key, primary key, foreign key, and some other integrity constraints. And based on the database defined in the data definition language, the data will be loaded and then be queried using Data Manipulation Language. Well, in a data manipulation language, it’s mostly SELECT ... FROM ... WHERE ... clause, and later on, we’ll see a lot of examples of this.

中文（第5段）

因此，SQL 主要由两部分组成。一部分是数据定义语言（Data Definition Language，DDL），用于定义关系，包括关系的名称、关系的属性、属性的域以及属性上的约束。约束包括键、主键、外键以及其他一些完整性约束。基于在数据定义语言中定义的数据库，我们会导入数据，然后使用数据操作语言（Data Manipulation Language，DML）进行查询。在数据操作语言中，最常见的就是 SELECT ... FROM ... WHERE ... 子句，之后我们会看到很多这方面的示例。

English (Paragraph 6)

So, in this example here, Data Definition Language is creating a table. The name is instructor. It has four attributes: ID, name, department\_name, and salary, with their domains. So ID is a fixed-length string of five characters. name and department\_name are variable-length strings of up to 20 characters, and salary is a numeric number that has eight digits, with at most two digits after the decimal point and six digits before the decimal point. So this is what is meant by this “CREATE TABLE” thing.

中文（第6段）

在这里的示例中，数据定义语言用来创建一个表。这个表名叫做 instructor，它有四个属性：ID、name、department\_name 和 salary，并且都有相应的域。比如 ID 是一个长度固定为 5 个字符的字符串，name 和 department\_name 是最多可包含 20 个字符的可变长度字符串，而 salary 是一个数值，最多有 8 位数字，其中小数点后最多 2 位，小数点前最多 6 位。这就是我们所说的 “CREATE TABLE” 的含义。

English (Paragraph 7)

We can similarly define other relations in this university toy example. For example, the student relation here has four attributes: ID, name, department\_name, and total\_credit. ID is a variable-length string with up to five characters. name and department\_name are variable-length strings with up to 20 characters, and name “NOT NULL” means, like, we don’t allow null value, so everybody needs to have a name. The total\_credit is a numeric number with three digits, with no digits after the decimal point.

中文（第7段）

在这个大学的示例数据库中，我们可以用类似的方式定义其他关系。比如，这里有一个 student 关系，包含四个属性：ID、name、department\_name 和 total\_credit。ID 是一个可变长度字符串，最多可包含 5 个字符；name 和 department\_name 是可变长度字符串，最多可包含 20 个字符；而 name 的 “NOT NULL” 表示不允许其值为 null，也就是说每个学生都必须有名字。total\_credit 是一个数值，最多 3 位数字，并且小数点后没有任何位数。

English (Paragraph 8)

Okay, ID is a primary key, and department\_name is a foreign key referring to the primary key of department. Okay, so this is what it means. Or with the takes relation, we are having more examples talking about ID, course\_id, section\_id, semester, year, grade, and primary key.

中文（第8段）

好的，ID 是一个主键，department\_name 是一个外键，引用了 department 的主键。好的，这就是它的含义。或者说，对于 takes 这个关系，我们还会有更多例子，涉及 ID、course\_id、section\_id、semester、year、grade 和主键等内容。

English (Paragraph 9)

So one relation will have at most one primary key, but it can have multiple foreign keys. You guys still remember the foreign key, right? Okay, cool. So foreign key ID references student, and these combinations reference the section.

中文（第9段）

因此，一个关系最多只能有一个主键，但是它可以有多个外键。大家还记得外键吗？好的，很棒。所以这里的外键 ID 引用的是 student，而这些组合引用的是 section。

English (Paragraph 10)

So that’s the Data Definition Language.

中文（第10段）

以上就是数据定义语言的部分。

English (Paragraph 11)

For Data Manipulation Language—that is also called query language—most of the time when we say “query language,” it’s Data Manipulation Language. It’s used for accessing and updating data. Normally, it has two parts: one is to access, one is to update. But the thing is, for the query, for each query for accessing data, the result itself is a relation. So that’s why we call it compositional.

中文（第11段）

接下来是数据操作语言（Data Manipulation Language），也称查询语言——多数情况下，当我们说“查询语言”时，指的就是数据操作语言。它用于访问和更新数据，通常有两部分：访问和更新。但是关键在于，对于查询来说，每次访问数据的结果本身就是一个关系。这也是为什么我们说它具有“可组合性”。

English (Paragraph 12)

For example, we use two original tables to do a query, and the result will become a relation. By using this relation, we can combine it with another relation and do another query and get a new relation. In other words, these can be embedded. So this can be embedded in a very complicated way. Yeah. So SQL is a declarative language in a way that we only need to specify what we want to get; we don’t need to specify how we are going to get it. So the database has full control over how you are going to execute the query. That’s what we call declarative. But declarative is like an API. This has a very nice separation from the user and the backend implementation.

中文（第12段）

举例来说，我们可以对两个原始表进行查询，结果会形成一个新的关系。利用这个新的关系，我们可以再与另一个关系结合，进行下一次查询，并得到又一个新的关系。换句话说，这些都可以嵌套使用，并且可以以非常复杂的方式进行嵌套。是的，所以 SQL 本质上是一种声明式语言，因为我们只需要指定想要得到什么，而不需要指定具体如何得到它。数据库可以完全自主地决定执行查询的方式，这就是我们所说的“声明式”。声明式就像一个 API，用户与后端实现之间有一个非常良好的分离。

English (Paragraph 13)

Okay, so the basic query structure—this is a brief introduction. We’re going to have a lot of examples today. The brief query structure will be SELECT A1, A2, ..., An FROM R1, R2, ..., Rm WHERE condition P should be satisfied. So these three things are called “SELECT clause,” “FROM clause,” and “WHERE clause” of the SELECT statement. So the whole thing is the statement; every part is called a clause. And the result of a single query, of course, is a relation. By “update,” you will have INSERT INTO, DELETE, DROP TABLE, ALTER, and so on and so forth. This is just a brief introduction.

中文（第13段）

好的，接下来介绍一下基本的查询结构——这是一个简要介绍，今天我们会举很多例子。基本的查询结构是：SELECT A1, A2, ..., An FROM R1, R2, ..., Rm WHERE condition P，其中需要满足给定的条件 P。这三个部分分别称为 SELECT 子句、FROM 子句和 WHERE 子句。整个就是一个 SELECT 语句；每个部分都叫做一个子句。而单个查询的结果当然是一个关系。至于“更新”操作，你会看到 INSERT INTO、DELETE、DROP TABLE、ALTER 等等。这只是一个简要的介绍。

English (Paragraph 14)

So normally how we are going to create and use a database is that—so we database people normally use this kind of diagram to represent the database itself. So normally, we use Data Definition Language to create a schema level, right? The different relation schemas, and after that, we load the data in. Okay, that’s the instances. We’ll load the data in. And once the data is in, we will allow the user to concurrently query or manipulate the data in the database. So for example, a user may pose a query Q1, get the result answer A1, and pose another query Q2, get another answer A2. Well, at the same time, another user may send a request: “I would like to modify something”—it’s a modification or update request—and database: “OK, I have done the change.” So these things are catered concurrently to multiple users—query and update. This is a general way of creating/using relational databases.

中文（第14段）

那么，通常我们是如何创建并使用一个数据库的呢？我们这些数据库相关的人通常会用某种图形来表示数据库本身。一般来说，我们先用数据定义语言来创建模式层，也就是定义各种关系模式，然后再导入数据。好的，这些就是实例，我们会把数据加载进来。一旦数据加载完成，我们就允许用户并发地查询或操作数据库。例如，一个用户可能发出查询 Q1，得到结果 A1，再发出查询 Q2，得到结果 A2；而与此同时，另一个用户可能发出“我想修改一些东西”的请求——这属于修改或更新请求，数据库会回应：“好的，我已经完成了更改。”这些查询和更新操作都会被同时处理。这就是创建/使用关系型数据库的一般方式。

English (Paragraph 15)

So this is a little bit of a recap of what we talked about today: Data Definition Language, Data Manipulation Language, basic query structure, and basic steps in creating and using a relational database.

中文（第15段）

这就是今天我们谈到的简要回顾：数据定义语言、数据操作语言、基本的查询结构，以及创建和使用关系数据库的基本步骤。

English (Paragraph 16)

Now we are going to move to the demo session. So the whole introduction of SQL will mainly be based on the demonstration over the following concepts, and I’m not going to read it now. So once we finish this demonstration, later you can come back and have a conceptual review of what we have demonstrated.

中文（第16段）

现在我们进入演示环节。我们对 SQL 的整体介绍主要会基于对以下概念的演示，我现在不会逐字去读。等我们演示完之后，大家可以再回头，从概念层面复习我们所做的演示。

English (Paragraph 17)

So one computer is going to show the query, and the recording computer will show my demonstration. And if you’re going to review—so these slides will be uploaded to Canvas. You can just do the corresponding reading by yourself.

中文（第17段）

我们会用一台电脑来显示查询，另一台电脑来进行录屏演示。如果你要复习的话——这些幻灯片会被上传到 Canvas 上，你可以自己对应着进行阅读。

English (Paragraph 18)

Okay, so this would be—hmm, what is not showing here? Okay, all good.

中文（第18段）

好的，现在这是——嗯，怎么有些东西没有显示？好的，看起来一切正常。

English (Paragraph 19)

So with this demonstration, I have—in the last lecture, we said we can use either SQLite or DuckDB; all the installation instructions have already been specified in Canvas. You can install it by yourself and perform all the demonstrations by yourself. You can also use the—I think the textbook provides an HTML-based interactive way that you can write a query into a text box and get a result. So, for example, the DB book has “Run SQL queries directly in your browser.” So now, if you have a computer or anything, or you haven’t installed the database, you can use this webpage to run the queries that we are going to demonstrate, to get the results, to see if we are doing this correctly.

中文（第19段）

关于这次演示，我们在上一节课就提到过，你可以使用 SQLite 或者 DuckDB；所有的安装说明都已经放在 Canvas 上了。你可以自行安装，然后自行完成演示。你也可以使用——我记得课本里提供了一种基于 HTML 的交互方式，你可以在一个文本框里写查询并得到结果。比如 DB book 网站上就有 “在浏览器里直接运行 SQL 查询” 的功能。所以，如果你有电脑或者其他设备，或者你还没安装数据库，你可以用这个网页来运行我们将演示的查询，看看结果是否正确。

English (Paragraph 20)

And if you have already—or if you have MacBook, MacBook has SQLite3 inbuilt, so you can just use the command line to run it. If you don’t use MacBook or you are using Windows, you want to install the other two databases, that’s also fine.

中文（第20段）

如果你已经装好了，或者如果你使用的是 MacBook，那么 MacBook 自带了 SQLite3，你可以直接用命令行来运行。如果你没有用 MacBook，或者你在使用 Windows，需要去安装其他两个数据库，也没有问题。

English (Paragraph 21)

Okay, I’m going to use the command line. Could you guys see the screen clearly? Is it big enough? Maybe larger. Okay, so is this demo—so in this directory, I have downloaded the ddl.sql, small\_relation\_inserting5.sql, and then I have created a database called university. In order to use this database, I just use sqlite3 university.

中文（第21段）

好的，我这里要用命令行。你们能看清屏幕吗？够大吗？也许要再调大一点。好，这个演示……在这个目录下，我已经下载了 ddl.sql、small\_relation\_inserting5.sql，然后我创建了一个名为 university 的数据库。为了使用这个数据库，我就输入 sqlite3 university。

English (Paragraph 22)

Okay, so then I’m opening the university. SQLite provides a dot-statement where you can input the command here. So, for example, if I want to see all the tables in the database, so I just .table, and then you can see we have advisor, course, instructor, section, takes, time\_slot, and so on and so forth in the toy example. That corresponds—so some of them are the schema. I’ve listed here to make the demonstration more convenient. Is this visibly clean? Okay.

中文（第22段）

好的，现在我已经打开了 university 数据库。SQLite 提供了一个以点开头的命令，你可以在这里输入命令。比如说，如果我想看这个数据库里的所有表，我只要输入 .table，然后你就能看到在这个示例里有 advisor、course、instructor、section、takes、time\_slot 等表。这些对应了示例中的一些关系；我在这里也列出了它们的模式，方便演示。看上去还清晰吗？好的。

English (Paragraph 23)

Recall that the basic SQL query will have three clauses: SELECT, FROM, WHERE. Now we are going to write queries based on that. Find the names of all instructors. Okay, so these three clauses can be mapped back. This is the relational algebra: the SELECT statement is actually the projection, which selects the columns, the attributes; the FROM clause is the Cartesian product (or the join), so you are listing the relations over there—they’re going to be joined together; the WHERE clause is a selection in the relational algebra, where you indicate the condition in P.

中文（第23段）

回忆一下，基本的 SQL 查询包含三个子句：SELECT、FROM、WHERE。现在我们就基于这三部分来写一些查询。先来看一个例子：找出所有教师（instructor）的名字。好的，这三部分其实可以映射回关系代数：SELECT 子句实际上相当于投影（projection），它用来选择列（属性）；FROM 子句相当于笛卡儿积（或连接），因为你在这里列出了要连接的关系；WHERE 子句则相当于关系代数中的选择（selection），在这里你会写出条件 P。

English (Paragraph 24)

Okay, so basically, the SELECT-FROM-WHERE clause is a combination of relational algebra expressions. You just put the right place—the relation, the conditions—into the right place.

中文（第24段）

好的，所以总体来说，SELECT-FROM-WHERE 这三个子句的组合就对应了关系代数表达式。你只需要把关系和条件放在正确的位置即可。

English (Paragraph 25)

Okay, so now we want to find the names of all instructors. SELECT name FROM instructor; And then follow by a semicolon. Okay, so this is all the instructors’ names. This SQLite allows you to show the result in different formats. If you would like to see the table, you can say .mode table, and then you perform the same query, and you see a table. So this is the name of all the instructors. So this result—it outputs a relation, right? This is a relation with a single column, and this column—a single attribute, the attribute name is “name.”

中文（第25段）

好了，现在我们要找出所有教师的名字。我们写 SELECT name FROM instructor;，然后以分号结束。好的，这里就是所有教师的姓名。SQLite 允许你以不同的格式查看结果。如果你想以表格形式显示，可以输入 .mode table，然后再执行相同的查询，就能看到一个表格。所以这就是所有教师的名字。这个结果就是一个关系，对吧？它是一个只有一列的关系，这一列就是一个属性，属性名是 “name”。

English (Paragraph 26)

The second query: “Find department names of all instructors and remove duplicates.” So this is SELECT department\_name FROM instructor. Note that this query may include duplicates. So here, I should mention that the SQL query and relational algebra have a difference. Relational algebra, by default, it’s a set—everything is a set—so there is no duplication there. SQL assumes a multiset, meaning that it allows duplication. So in order to remove duplication, you need to add a little bit of—so let’s say this one is—so this one, you will see that “Physics” appeared twice, “History,” you have two occurrences. And to remove the duplications, you just add a DISTINCT—SELECT DISTINCT department\_name FROM instructor;. So by adding “DISTINCT” after the SELECT, you’re removing the duplications.

中文（第26段）

第二个查询：“找出所有教师所属的系名，并去重。” 对应的 SQL 语句是 SELECT department\_name FROM instructor;。需要注意，这样的查询可能包含重复值。这里我要提一下，SQL 查询和关系代数之间有一个区别：关系代数默认是集合（set），因此没有重复；而 SQL 默认是多重集（multiset），因此允许重复。为了去掉重复值，你需要加上一点额外的东西——比如写成 SELECT DISTINCT department\_name FROM instructor;。在 SELECT 后面加上 “DISTINCT” 就可以去掉重复值了。所以如果你不加 DISTINCT，就会看到 “Physics” 出现了两次，“History” 也出现了两次。有了 DISTINCT 就不会重复了。

English (Paragraph 27)

So the next query: “Find the department names of all instructors, not removing duplicates.” Well, by default, we do not remove duplicates; that’s already done in the previous query.

中文（第27段）

接下来的查询：“找出所有教师所属的系名，不去重。” 其实默认就不去重，所以这和之前的查询相对应，默认情况就是包含重复值。

English (Paragraph 28)

“Find all attributes of instructors.” “Find all attributes of instructors—wait a minute, what is this?” Yes, so this is related to how you’re going to find the schema of the instructor. So in order to know the schema, it’s .schema instructor. Okay, so then you know all the attributes of the schema.

中文（第28段）

“查找教师表中所有的属性。” “查找教师表所有属性——等等，这是什么？” 是的，这其实与如何查看 instructor 表的模式（schema）有关。要知道模式的话，你可以用 .schema instructor。好的，这样你就能看到这个表里的所有属性了。

English (Paragraph 29)

“Find a relation that is the same as the instructor relation, except that the value of the attribute salary is divided by 12.” So in other words, we want a new relation. So this relation would be mostly the same as instructor; the only difference is, like, we want the salary to be divided by 12. Okay, so we’re going to SELECT ID, name, department\_name, salary/12 AS sal\_or\_just FROM instructor;. Okay, so in this way—so the thing is, like, this tells you that you can use the numeric operators to operate on the whole column in the SELECT clause: plus, minus, division, multiplication, and some kind of more complicated expression, like, for example, if you do textation, you want the whole column to be manipulated in a specific way. We can just do this and then report a new relation after this transformation.

中文（第29段）

“找到一个与 instructor 表几乎相同的关系，但其中 salary 属性的值要除以 12。” 换句话说，我们想要一个新关系。它和 instructor 几乎一样，只是 salary 这个属性需要被除以 12。那我们就写 SELECT ID, name, department\_name, salary/12 AS sal\_or\_just FROM instructor;。好的，这样就可以了。这说明，你可以在 SELECT 子句中对整列数据使用数值运算符，比如加、减、除、乘，以及一些更复杂的表达式，例如对文本的处理等。我们可以这样做，然后把转换后的结果作为一个新关系输出。

English (Paragraph 30)

The next query: “Find all instructors in the Computer Science department with salary greater than 70000.” So this query asks you to do some selection using the WHERE clause. So we do SELECT \* FROM instructor WHERE department\_name = 'Comp. Sci.' AND salary > 70000;. Okay, so this gives you the two instructors who are from Computer Science whose salary is greater than 70000. Any question regarding this? Oh, good.

中文（第30段）

下一个查询：“找出所有属于计算机科学系且工资大于 70000 的教师。” 这个查询需要在 WHERE 子句中指定过滤条件。我们写 SELECT \* FROM instructor WHERE department\_name = 'Comp. Sci.' AND salary > 70000;。好的，这样就能得到两位来自计算机科学系，薪资高于 70000 的教师。这部分有问题吗？哦，没有问题。

English (Paragraph 31)

“Find the names of all instructors who have taught some course, and the course ID.” I mean the corresponding course ID. So this is the instructor schema, this is the teaches schema. The instructor’s ID is—so this teaches.ID is a foreign key of instructor. So by combining these relations, you will be able to know what courses are delivered by one instructor, right, using, say, a natural join or a theta join (remember the relational algebra?). Yeah.

中文（第31段）

“找出所有上过某门课的教师姓名，以及该课程的课程号。” 我指的是相应的课程号。这里有 instructor 表，还有 teaches 表。teaches.ID 是 instructor 的外键。把这两个关系结合起来，我们就能知道哪个教师上了哪些课程，对吗？我们可以用自然连接（natural join）或者 θ 连接（theta join），还记得关系代数吗？对的。

English (Paragraph 32)

So: “Find the names of all instructors who have taught some course, and the course ID.” The thing is, some instructors may not have taught anything, so that’s why we need to find the instructor who are joinable to the teaches. Okay. So what we do is we SELECT a.ID, a.name, course\_id FROM instructor NATURAL JOIN teaches; or something like that. Actually, in SQLite, you might do SELECT instructor.ID, instructor.name, teaches.course\_id FROM instructor, teaches WHERE instructor.ID = teaches.ID;. Then we see the results.

中文（第32段）

所以：“找出所有上过某门课的教师姓名，以及该课程号。” 有些教师可能没有上任何课程，所以我们只找和 teaches 能够连接上的教师就行。好，我们可以这样写：SELECT a.ID, a.name, course\_id FROM instructor NATURAL JOIN teaches;。或者在 SQLite 里，你可以这样写：SELECT instructor.ID, instructor.name, teaches.course\_id FROM instructor, teaches WHERE instructor.ID = teaches.ID;。然后我们就可以看到结果。

English (Paragraph 33)

That’s how we use natural join, but alternatively, we can use theta join. So here, we just list the two relations. So when we list the two relations, it means we are performing Cartesian product—well, condition product—joining the two relations together, and do the filtering. It’s like we need to have WHERE instructor.ID = teaches.ID;. Okay, they are basically the same—oh, sorry—they are the same.

中文（第33段）

这就是我们如何使用自然连接。当然，我们也可以使用 θ 连接。在这种情况下，我们只是在 FROM 子句中列出两个关系，这实际上意味着先做笛卡儿积，然后通过 WHERE 子句进行过滤，比如说 WHERE instructor.ID = teaches.ID;。好的，它们本质上是一样的——哦，抱歉——确实就是一样的。

English (Paragraph 34)

The next one: “Which one? Different semester, different year, different—for sometimes it’s offering spring, sometimes it’s offering fall, sometimes it’s only this year and that year?” Good question. Okay.

中文（第34段）

下一个问题：“哪一个？不同的学期，不同的年份，不同的……因为有时是春季学期，有时是秋季学期，有时只在这一年或那一年？” 问得好。好的。

English (Paragraph 35)

The next one: “Find the names of all instructors in the Art department who have taught some course and the course ID.” So the difference between this query and the previous query is that the previous query didn’t indicate the department of the instructor, but this query would only be interested in instructors from the Art department, right? So we are going to make a small modification here: WHERE department\_name = 'Art'. I hope it works. There’s no result—why? Is there anyone from the Art department? So SELECT \* FROM instructor;—we don’t have Art. Okay, well, let’s say this is Computer Science then. Yeah, so if—sorry, the Art department doesn’t appear in our toy example. So let’s say shift the Art to Computer Science. So if that is the case, we have this result.

中文（第35段）

下一个查询：“找出所有在艺术系（Art）任教过某门课的教师的姓名，以及该课程号。” 这个查询和前一个的区别在于，前一个没有指定教师所属的院系，而这个要求教师来自艺术系，对吗？所以我们要做一点修改：WHERE department\_name = 'Art'。我希望它能工作。结果没有数据——为什么？艺术系有人吗？执行 SELECT \* FROM instructor;——发现没有艺术系。好吧，那我们就把 Art 换成计算机科学（Computer Science）。对，抱歉，我们这个示例里并没有艺术系。所以如果我们换成计算机科学，就能查到结果。

English (Paragraph 36)

So this query can be equivalently rewritten using natural join, but using natural join, you can remove this condition and just use “NATURAL JOIN” here.

中文（第36段）

所以，这个查询也可以用自然连接来实现。用自然连接的话，可以把这个条件去掉，直接写成 “NATURAL JOIN”，效果是一样的。

English (Paragraph 37)

Okay, so “Find the names of all instructors who have a higher salary than some instructor in Computer Science.” In other words, we want to find the instructor whose salary is larger than the lowest-salary instructor in Comp. Sci. How do you—how do you write this query? Okay.

中文（第37段）

好的，接下来是“找出薪资高于计算机科学系某位教师的所有教师姓名。” 换句话说，我们想找的是那些薪资比计算机科学系中最低薪资教师还高的教师。我们该怎么写这个查询呢？好的。

English (Paragraph 38)

So to do this, I’m going to say this “some” is actually a function that’s provided by some databases. So for example, in DuckDB, you can just write a plain statement as if—so the WHERE clause is like SELECT ... FROM instructor WHERE salary > SOME(...). And then you have a selection statement—I’m going to show you right now. So I need to first exit—first exit those SQL—and I’m going to do the DuckDB. Wait a minute, I think the pathway—where is the path? So I will reconfigure it later on.

中文（第38段）

要实现这个，我想说一下，这里 “some” 实际上在某些数据库里是一个关键字。比如说，在 DuckDB 里，你可以直接写一个类似 SELECT ... FROM instructor WHERE salary > SOME(...) 的语句，然后把子查询写在括号里。等会儿我来示范一下。我得先退出 SQLite，改用 DuckDB。等等，我看看路径在哪里？我稍后会重新配置一下。

English (Paragraph 39)

So—“Oh, DuckDB, did I—no way—” So we are going to do SELECT name FROM instructor AS a, instructor AS b WHERE a.salary > b.salary AND b.department\_name = 'Comp. Sci.';. Okay, so here, the thing is, like, now you are comparing two instructors, right, comparing two instructors. When you do this, potentially you are joining two instructor relations. So basically, you are having instructor AS a, instructor AS b. So this AS is a kind of renaming mechanism—remember the renaming in relational algebra—so it’s like a and b are the name of two instances of instructor. And then what we want is a.name, and we want the a.salary to be greater than b.salary, where b should come from Computer Science, right? b should come from Computer Science: WHERE b.department\_name = 'Comp. Sci.';.

中文（第39段）

然后——“哦，DuckDB，我……不会吧——” 我们来写：SELECT name FROM instructor AS a, instructor AS b WHERE a.salary > b.salary AND b.department\_name = 'Comp. Sci.';。好的，这里我们就是在比较两个教师，对吧，比较两个教师。当你这么做时，实际上就是把 instructor 这个表当作两个实例：一个叫 a，一个叫 b。AS 是一种重命名机制——还记得关系代数里的重命名吗？就像这样，a 和 b 分别代表 instructor 的两个实例。然后我们想要的是 a.name，并且要求 a.salary > b.salary，而且 b.department\_name = 'Comp. Sci.'，也就是 b 来自计算机科学系，对吧？

English (Paragraph 40)

Okay, so once we—oh, and then—okay, so it looks like a little bit duplicated. Then we add a DISTINCT here. Okay, so this is one way of doing it. I will probably show you in the next demonstration using DuckDB. There is a much simpler way of doing this, because you will use the SOME notation. So it will look like something:

中文（第40段）

好的，当我们这样做之后——哦，发现可能结果会有些重复。那我们就加上 DISTINCT。好了，这是一种做法。我可能会在接下来的 DuckDB 演示里再演示一次。有一个更简单的方法，就是用 SOME 这个关键字来做，就会写成这样：

English (Paragraph 41)

SELECT name

FROM instructor

WHERE salary > SOME

(

SELECT salary

FROM instructor AS a

WHERE a.department\_name = 'Comp. Sci.'

);

中文（第41段）

SELECT name

FROM instructor

WHERE salary > SOME

(

SELECT salary

FROM instructor AS a

WHERE a.department\_name = 'Comp. Sci.'

);

（这是一个示例，在这里我们用了 SOME 关键字，如果数据库支持的话，可以直接这样写。）

English (Paragraph 42)

Okay, so it depends on whether this database supports the SOME keyword. SQLite doesn’t support it. This will be clearer, but DuckDB supports it. So you can try later on using DuckDB. They will tell you that they don’t know what SOME is, if you do it in SQLite.

中文（第42段）

好的，这取决于数据库是否支持 SOME 这个关键字。SQLite 并不支持，所以如果在 SQLite 里这么写，它会告诉你不知道 SOME 是什么；但在 DuckDB 里就可以用。所以你之后可以尝试在 DuckDB 上运行这段查询。

English (Paragraph 43)

Okay, now we are going to do some demonstration on the SQL query with the string operators—string operations—like you want to do a pattern matching of the strings. So some of the values are strings, right, and you want to find some strings that are similar to some string you are looking for, or contain some substring that you are interested in.

中文（第43段）

好了，现在我们演示一下 SQL 查询中的字符串操作，比如要做字符串的模式匹配。有些字段是字符串，对吧？那么你可能想找与某个模式相似的字符串，或者包含某些子串的记录。

English (Paragraph 44)

So here, the example asks you to find the names of all instructors whose name includes the substring “in.” Okay, so in this query, we’ll need to use a keyword called LIKE. So what we will do is we’re going to SELECT name FROM instructor WHERE name LIKE '%in%';. And then you have a pattern, where this percentage means a string of arbitrary length—it can match to a string of arbitrary length—and then we need “in” and another string of arbitrary length. Okay, so then we will get the result as those three names that include “in.”

中文（第44段）

例如，这个示例想要找出所有名字中包含子串 “in” 的教师。我们需要用到一个叫 LIKE 的关键字。所以我们可以写 SELECT name FROM instructor WHERE name LIKE '%in%';。这里的模式中，% 表示任意长度的字符串，它可以匹配任意长度的字符，然后我们写上 “in”，再加上一个 % 表示之后也可以是任意长度的字符串。这样就能匹配到名字里含有 “in” 的教师。结果我们可以看到，有三个人的名字里包含 “in”。

English (Paragraph 45)

Okay, so apart from the percentage which means the string of arbitrary length, we can match the underscore with any character. So, for example, if you want to find all instructors whose name has four characters, what you would do is that you basically use the same statement, but here you add four underscores to match four characters. So in this case, we will return “Gold” and “Cutz.” The number can be four or five, or sometimes you want to know the name with at least four characters. Whose name has at least four characters—and in this case, we’re going to have four underscores followed by a percentage, and it means that it can be arbitrarily long, but it has to have at least four characters. So once you type it, you will get a whole bunch of names with at least four characters.

中文（第45段）

好的，除了 % 用来表示任意长度的字符串之外，我们还可以用下划线 \_ 来匹配任意单个字符。举例来说，如果你想找名字正好是四个字符的教师，你就可以在模式里写四个下划线，比如 LIKE '\_\_\_\_'，这样就匹配长度为 4 的名字。在这种情况下，可能结果里会有 “Gold” 和 “Cutz”。有时你想找名字至少为四个字符的，那就可以写四个下划线后再加一个 %，表示至少四个字符，但是可以更多。这样执行后，你就能查到所有名字长度至少 4 个字符的教师。

English (Paragraph 46)

Okay—ah, this should be hidden a little bit. Demonstration error here. So please ignore the 9 to 11, just look at the first one.

中文（第46段）

好的——啊，这里本来应该隐藏一点，演示上有点小错误。请忽略 9 到 11 的部分，只看第一个就好。

English (Paragraph 47)

“List in alphabetical order the names of all instructors.” So remember that the relation doesn’t have an order of the tuples that they are storing, but you can indicate the order you want to see when you do the selection. So the statement will be using ORDER BY clause. So here, let’s say SELECT name FROM instructor ORDER BY name;. Okay, ORDER BY name. So by default, this ORDER BY is using ascending order. So with this, you have B, C, E, G, K, M, S, W. So this is alphabetical order. If you want to reverse the order, you add DESC, and then you have a reverse order starting from W back into B.

中文（第47段）

“按字母顺序列出所有教师的名字。” 大家要记住，关系数据库本身对元组没有排序，但我们可以在查询时指定要以怎样的次序显示结果。我们可以用 ORDER BY 子句来实现。比如 SELECT name FROM instructor ORDER BY name;，这样就会按名字的字母顺序排列。缺省情况下是升序，所以我们会看到从 B、C、E、G、K、M、S、W 依次往下排。如果想要逆序，就加上 DESC，这样就会从 W 往 B 反向排列。

English (Paragraph 48)

So sometimes this ORDER BY can use a combination of multiple attributes to do the sorting. So if, for example, we would like to order the instructor based on the combination of their names and salary—which means that if they have the same name, I would like them to be ranked based on their salary in increasing order—and then the ORDER BY can be combined with “name, salary.” Well, you can show the salary as well. It doesn’t make much difference because here we don’t have ties. Let’s say we rank it from salary and name, so we will first rank it with the salary and then rank it with the name, and then I will see how I’m going to sort it. .mode table.

中文（第48段）

有时我们会用多个属性的组合来排序。比如说，如果我们想先按名字排序，然后在名字相同的情况下再按工资升序排，那么就可以在 ORDER BY 后面写 “name, salary”。当然，你也可以写成先按工资再按名字。因为示例数据里可能没有完全一样名字的人，所以区别不大。比如我们写先按薪资再按名字，那就先排序薪资，然后再对薪资相同的人按名字排。输入 .mode table 可以查看表格格式的输出。

English (Paragraph 49)

Okay, so in this case, all the instructors will be sorted based on their salary, so salary will have a non-decreasing order—or increasing order. When they have a tie, which is, for example, 80000, they are going to be ranked based on their name, so “Kim” comes before “Singh.” That’s why Kim comes before Singh.

中文（第49段）

好的，在这种情况下，所有教师都会按薪资从小到大排序。如果有并列，比如两个教师的薪资都是 80000，那么就再按照名字进行排序，所以 “Kim” 会排在 “Singh” 前面。这就是为什么 Kim 在 Singh 前面的原因。

English (Paragraph 50)

Now we are going to show you there is a keyword called BETWEEN in the WHERE clause. So when you specify the selection conditions in the WHERE clause, you can compare the numeric value to any—sorry, compare the attribute of numeric type to any numeric value. So for example, this query finds the names of all instructors with salary between 90000 and 100000. You can write a query like SELECT name FROM instructor WHERE salary > 90000 AND salary < 100000;. So we get “Allison” and “Brando.” But alternatively, we can state it this way: salary BETWEEN 90000 AND 100000;. So inclusive—so the BETWEEN is inclusive—so if we add greater than or equal to and less than or equal to, then we will have exactly the same result.

中文（第50段）

现在我们要演示一下 WHERE 子句里的 BETWEEN 关键字。当你在 WHERE 子句里指定选择条件时，可以让数值属性与某个范围进行比较。比如说，这个查询要找所有薪资在 90000 到 100000 之间的教师，你可以写 SELECT name FROM instructor WHERE salary > 90000 AND salary < 100000;，这样就能查到 “Allison” 和 “Brando”。不过你也可以换种写法：salary BETWEEN 90000 AND 100000;。这里的 BETWEEN 是包含端点的，也就是说它相当于大于等于 90000 并且小于等于 100000。如果你手动写上大于等于和小于等于，也会得到同样结果。

English (Paragraph 51)

“Find the courses that ran in Fall 2017 or in Spring 2018.” This is about the set operations—remember the set operations? So we can construct one set and union it with another set. In this case, it is the same. So we can first define the courses that are delivered in Fall 2017, and then we find the courses that are delivered in Spring 2018, and then we union the results together, right? So what we’ll write is:

中文（第51段）

“找出所有在 2017 年秋季（Fall 2017）或 2018 年春季（Spring 2018）开设的课程。” 这就涉及到了集合运算，还记得吗？我们可以分别把 2017 年秋季的课程和 2018 年春季的课程抽取出来，然后把它们做一个并集（union）。实现方式是先定义好 2017 秋季开的那些课程，然后定义 2018 春季开的课程，最后把结果并在一起。我们可以这样写：

English (Paragraph 52)

SELECT \*

FROM section

WHERE year = 2017 AND semester = 'Fall'

UNION

SELECT \*

FROM section

WHERE year = 2018 AND semester = 'Spring';

中文（第52段）

SELECT \*

FROM section

WHERE year = 2017 AND semester = 'Fall'

UNION

SELECT \*

FROM section

WHERE year = 2018 AND semester = 'Spring';

（这就把 2017 秋季和 2018 春季所开的所有课程做了并集。）

English (Paragraph 53)

Okay, so then we will have all these courses. You can see it’s either Spring 2018 or Fall 2017, right? Cool. The set operation means, like, this “OR” means union, “AND” means intersection, right, and there’s also set differences. To do set differences, we use EXCEPT. So here we see if we want to find the courses that were given both in Spring 2018 and Fall 2017, we are going to do an intersection. There’s no course given by both like 2017 and 2018. It’s random—fall.

中文（第53段）

好的，这样就能查到这些课程，你可以看到要么是 2018 春季开的，要么是 2017 秋季开的。对吧？挺好。集合运算里，“OR” 类似于并集，“AND” 类似于交集。我们也可以做差集，比如说 EXCEPT。如果我们想找同时在 2017 秋季和 2018 春季都开设的课程，就可以做一个交集。结果可能发现并没有课程既在 2017 又在 2018 开，或者是某种随机情况——总之，这里演示的是这个思路。

English (Paragraph 54)

How about EXCEPT? The set difference—okay, so the EXCEPT is the set difference between the two results, and you will see that these three courses were exclusively delivered in Fall 2017 but not in Spring 2018. We’ll probably see how far we can go—this is the intersection, and this is the set difference.

中文（第54段）

那 EXCEPT 呢？就是做集合差运算。好，用 EXCEPT 来表示两个结果集之间的差集，你就会看到，有三门课只在 2017 年秋季开了，而没有在 2018 年春季开。我们可以看看结果，这就是交集和差集的用法示例。

English (Paragraph 55)

So “Find courses that ran in Fall 2018 or in Spring 2018, retaining all duplications.” So remember that the set operations in an SQL statement do not allow duplications by default, but if you want to retain the duplication, you can add ALL. So here, let’s say SELECT ... UNION ALL SELECT .... By indicating UNION ALL, it means, like, you are going to include all the duplications in the result.

中文（第55段）

接下来，“找出所有在 2018 年秋季或 2018 年春季开的课程，并且保留所有重复项。” 要知道，在 SQL 里的集合运算默认是不保留重复的，但如果你想保留重复值，就要加上 ALL。比如 SELECT ... UNION ALL SELECT ...。UNION ALL 的意思是把重复的记录也都保留下来。

English (Paragraph 56)

Do we have duplications? Hmm, how—hmm—CS101 Fall 2017, maybe the building will be different, right? But the easiest way is, like, I’m not going to say SELECT \*; I’m going to say SELECT course\_id. If you say SELECT course\_id ... UNION ... SELECT course\_id ..., the schema must match, of course. So you do that, you see they are automatically removing all the duplications, but if we want to retain those duplications, we just write UNION ALL. Okay, so in this case, you know, CS319 has been duplicated twice, right, and CS101 has also been duplicated.

中文（第56段）

有没有重复？嗯，怎么说呢，像 CS101 在 2017 秋季开，可能教室不一样之类的。但最简单的方式是，我们不写 SELECT \*，而是只写 SELECT course\_id。如果我们写 SELECT course\_id ... UNION ... SELECT course\_id ...，那么在模式匹配的情况下，默认会去重。如果我们想保留重复，那就要写 UNION ALL。这样的话，比如 CS319 可能出现两次，CS101 也可能重复出现。

English (Paragraph 57)

Well, I’m going to leave the next 9 to 11 to the next lecture—it is all about how to deal with null values. Okay, yep, that’ll be all for today. Thank you very much.

中文（第57段）

好的，关于“第 9 到 11 点”之类的内容，我就留到下一次课再讲——那部分主要是关于如何处理空值（null values）的。好了，今天就到这里。非常感谢大家。

English (Paragraph 58)

“About the recording—oh, what’s the problem with the recording?”

“So only… I can only consider first and fourth. It will take a day to appear on the system. So yesterday’s recording will appear at the same time. Oh, yesterday time is in the afternoon, right? Yeah, they will appear today in the afternoon.”

“Oh, thank you.”

“You’re welcome. Let me tell you—this is the stakehouse, the big longest, you are using small instance—yeah, so but… but you have to be the same. You don’t have to—if you are correct, you have to be the same. If they are not, you won’t be correct. I mean, but even if it is correct, it doesn’t mean that you are not correct.”

中文（第58段）

“关于录制……哦，录制有什么问题吗？”

“所以只能……我只能考虑第一和第四。它要过一天才会出现在系统上。所以昨天的录制也会在同一时间出现。哦，昨天是下午录的吗？对，那应该会在今天下午出现。”

“哦，谢谢。”

“不客气。我告诉你——这是牛排馆，最长的那家……你在用一个小实例——是的，但是……但是你要保持一致。如果你是对的，就要保持一致。如果不一致，那就不对。我的意思是，即使它是对的，也并不意味着你就不会错。”