

## 02 – Advanced SQL (CMU Databases Sys...

---

### 02-01



00:14-00:20 虚生花

So it's Wednesday night fights over ~~—————~~ Mack banged Oakland, so is the other guy.

今天是周三，比赛已经结束了

00:20-00:25 虚生花

I second lecture all over again in a motel room.

今天我会在旅馆里给你们上第二节课

00:25-00:30

I just that it would be better just to show you guys what I did in 2018

我觉得向你们展示下我在2018年所做的事情比较好

0.30-0.37

because that was front of a live audience and I felt that I did it, I could do a better job and whatever do by myself.

因为通过去年这个课现场大家的眼神，我觉得自己做到了，所以我可以做得更好，无论我自己做什么

00:37-00:42 虚生花

So with that's why I'm just really showing the 2018 lecture here for the second lecture

这就是为什么我将2018年的讲座录播作为第二节课的原因了

0.42-0.47

and then when then we'll be heading back to CMU, and then starting Wednesday next week we'll have live lectures again.

在我们回到CMU后，我们会在下周三再次开始线上课堂

00:48-00:52 虚生花

So, take care.

So, 保重



00:52-0:55 虚生花

Today's lecture is on advance SQL

今天课程内容主要是高级SQL相关

0.55-1.01 虚生花

and by advance I mean going beyond what you may or may not already know about basic SQL.

我说的高级就是那些你可能已经知道或者还不知道的SQL基础以外的知识

01:01-01:10 虚生花

Right, it's 2018 SQL was invented in 1973, I imagine most you have seen some SQL throughout your life.

今年是2018年，SQL是在1973年问世，我可以想象你们中大部分人在日常生活中已经见过了SQL

01:10-01:13 虚生花

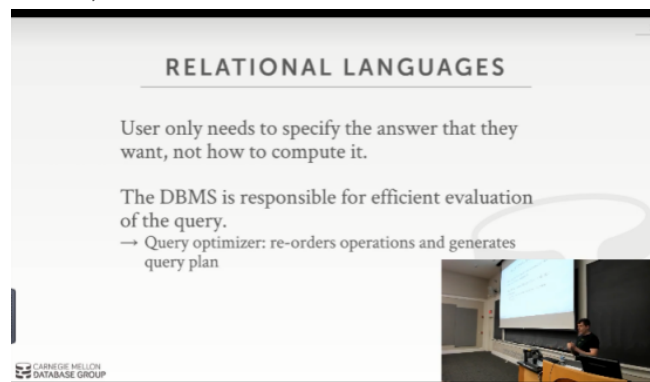
So I don't feel it's necessary to teach you the basics of it.

因此，我觉得没必要去教你那些SQL基础

01:13-01:17 虚生花

So I'm gonna spend time actually talking about the more complicated or interesting things you can do with SQL.

实际上，我想花点时间来讨论下那些你可以用SQL来做到的更为复杂且有趣的事情



01:17-01:25 虚生花

Right, so just to pick up where we left last class, we were discussing the relational model and relational algebra.

回顾下我们上节课所讨论的东西，我们讨论了关系模型以及关系代数

01:25–01:29 虚生花

And we sort of mentioned that the with relational algebra  
我们提到了关系代数

01:29–01:40

The goal was sort of at a high-level to describe what the answer we wanted that the database system would compute rather than the exact steps of actually how to do it.  
我们的目标是站在一个高级层面去告诉数据库系统计算出我们想要的答案，而不是告诉它具体该怎么做

01:40–01:45

All right, so the way to think about this is like say we want to sort our data.  
我们以对我们的数据进行排序为例来思考这个问题

01:45–01:52

If we have to tell the database system exactly what to do, we have to provide it you know with the quick sort of bubble sort algorithm.  
如果我们必须告诉数据库系统具体该怎么做的的话，我们就必须提供给它例如你知道的冒泡排序算法

01:52–01:59

But with a high-level language or declarative-language would you say, hey we want you to sort this, we don't care how you actually do it.  
但如果我们用的是高级语言或者声明式语言（知秋注：比如SQL），我们就会说，hey，我们想让你对数据进行排序，我们并不介意你实际上是怎么完成它的

16

01:59–02:02

But this is the answer that we want.  
但这就是我们想要的答案

17

02:02–02:06

And we'll see this throughout today's lecture and then going forward throughout later in the semester.  
我们不仅会在这节课上看到这个，并且它会贯穿我们之后的整个学期

02:06–02:10 虚生花

This is one of the advances of using something like SQL or declarative-languages.  
这是使用某种类似于SQL或者声明式语言的其中一个优点

02:10–02:15 虚生花

we don't have to tell database system exactly how to do Things, it can figure it out on its own.  
我们无须告诉数据库具体该怎么做，它自己就能搞定

02:15–02:24 虚生花

And that frees it up to figure out what the most optimal way, it is to actually execute the query that you want to execute based on the data that you have and and it's a harbor that's available to you.

这就能让数据库根据你给出的数据自己去找到执行查询的最佳方案

从而腾出时间来找到最佳优化方案，让数据库根据实际的你想要的方式来对数据进行查询

21

02:24-02:28 虚生花

So we'll see this later when we talk about the query optimizer

So我们会在之后讨论查询优化时看到这个

2.28-2.37 虚生花

but the query optimizer is this complicated piece of machinery inside of a database system, that's gonna take our SQL query and convert it to the most efficient plan.

查询优化是数据库系统中最复杂的一部分，它将我们的SQL查询转换为某种最有效的查询方案

02:37-02:40 虚生花

And we'll cover how they actually do this later on.

之后我们会去介绍它是如何做的

23

02:40-02:50 虚生花

But I will say is that if you know, if you get involved in doing research on query optimization or just help you know working on Query optimizers, you can get a job yesterday.

但我想说的是如果你对查询优化做过研究或者你做过查询优化方面的工作，那么昨天你可以得到一份工作

02:50-02:58 虚生花

Because this is the one thing that all my friends at Davis company companies email me about open over again, do I have any students to do any query optimization.

因为昨天我有个在Davis公司工作的朋友发邮件问我这里是否有学生做过查询优化相关的工作

02:58-03:01 虚生花

But has some and they all get off go off and do awesome jobs.

但确实有部分人在这方面做得很好

26

03:01-03:05 虚生花

This is the one thing that everybody wants because it's really hard to do.

这是每个人都想要的东西，因为查询优化这块真的很难做

27

03:05-03:06 虚生花

So that's not the focus here.

当然，这并不是今天的重点

3.06-3.13

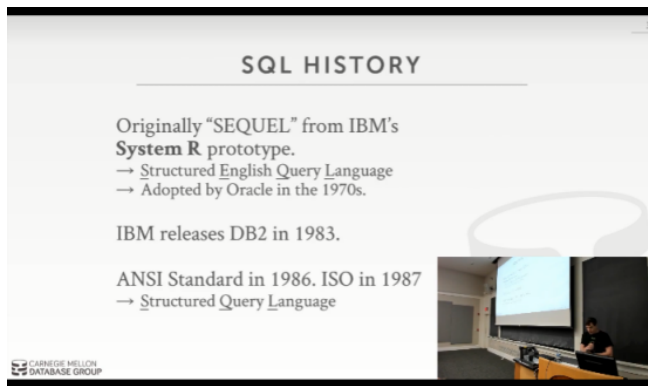
Right, just in the back your mind will see examples as we go along where a query optimizer could try out different things to try to come up with different plans.

回顾下我们脑海中以前看过的例子，查询优化器可以尝试不同策略来对查询进行优化

03:13-03:18 虚生花

Well we'll discuss how you actually do query optimization later on.

Well，我们之后会讨论你们该如何进行查询优化



03:18-03:25 虚生花

So the history of SQL as I said goes back into the early 1970s.

我之前说过SQL的历史可以追溯到1970年代早期

03:25-03:30 虚生花

So you may notice that I say SQL, some people say S-Q-L

你们可能注意到了，关于SQL的念法，我说的是SEQUEL，也有人读作SQL

3.30-3.39 虚生花

right, the part of the reason I say SQL, because in not that I was alive in 1970s. But

the original name of the language was actually spelled out S,E,Q,U,E,L sequel(SQL).

我之所以把SQL读作SEQUEL的部分原因并不是因为我出生在1970年代，只是它一开始的名字的拼写就是SEQUEL

03:39-03:44 虚生花

Right ,this was invented by IBM, as part of the system R project.

它是由IBM所发明，并且作为System R项目的一部分

03:44-03:47 虚生花

Right, so it stood for the Structured English Query Language.

它的意思是结构化英语查询语言

03:47-03:52 虚生花

So if you remember from the Ted Codd paper, I said that Ted.Codd was a mathematician.

如果你记得Ted Codd所发表的论文的话，我曾说过他是一名数学家

03:52-04:00 虚生花

Right, he devised that relational algebra and relational model, but he didn't actually define the programming language you would use to write queries on it.

他设计了关系代数以及关系模型，但实际上他并没有定义任何你能够用来编写查询语句的编程语言

04:00-04:01 虚生花

Right, you can't write queries using relational algebra.

你没办法使用关系代数来编写查询

04:01-04:07 虚生花

Right, it's sort of you know there's no it's very difficult to write that in the keyboard to do that.

这很难用键盘去敲出来

04:07-04:13 虚生花

He did later define or come up with his own query language called alpha, but that was much later in 1970s.

但在1970年代末期，他才定义了他自己的查询语言Alpha

04:13-04:18 虚生花

So back then when people said, hey there's a relational model idea we should actually try to build a system to do this.

因此在那时，人们表示因为有了关系模型，我们应该试着构建一个系统来做这些事情

04:18-04:23 虚生花

People had to come up with their own language that could implement relational calculus or relational algebra.

人们必须拿出他们自己的语言，这些语言能够实现关系演算或者关系代数

04:23-04:26 虚生花

So at IBM they came out with SQL

因此，IBM他们推出了SQL

4.26-4.33 虚生花

this is part of the system R project which is a sort of one of the first relational database systems that people were trying to build in the 1970s.

这是System R项目的一部分，它也是人们在1970年代第一个尝试构建的关系型数据库系统

04:34-04:38 虚生花

The other major one was ingres that came out of Berkeley.

另一个主流则是由伯克利所推出的Ingres

04:38-04:40 虚生花

So you've ever heard of PostgreSQL.

你们可能已经听说过PostgreSQL

44

04:40-04:43 虚生花

Right ,PostgreSQL was invented by the same guy that did ingres.

PostgreSQL是由开发Ingres的那批人所开发的

04:43-04:46 虚生花

So it's called PostgreSQL as in post ingres the thing that came after ingres.

它之所以被称为PostgreSQL，是因为它是在Ingres之后所开发出来的

04:46-04:50

So the ingres guys had their own language called quel

开发Ingres的那群人有他们自己的语言Quel

4.50-4.53

and this was developed by one of my advisers Mike Stormbreaker.

它是由我的导师Mike Stormbreaker所开发的

04:53-4.56

He claims it was much better than SQL

他声称这要比SQL要好得多

4.56–4.57

and the IBM guys didn't know what they were doing

IBM那群人根本不知道他们在干什么

4.57–5.00

of course most people have never heard a quel.

当然，大部分人从来没有听说过Quel

05:00–05:04

Right, I said it didn't actually win IBM won.

我并没有说它赢了IBM

05:04–05:07

So back then again it was spelled out as SEQUEL

在那时候，它的拼写是SEQUEL

5.07–5.19

IBM later got sued for, I think you know copyright infringement

trademark infringement, there was some guy in England that had the term SQL spelt out in the full English word for his programming language.

之后因为英国个人为他的编程语言也取名SQL，IBM为此与他产生了版权和商标纠纷

05:19–05:23

So it is shortened it to be SQL.

因此，它被简称为SQL

05:23–05:31

So what happened was the reason why we use SQL today, because IBM it's not back then IBM is and what it is now.

我们现在使用SQL的原因是因为当时IBM的地位和现在的地位完全不同

52

05:31–05:35 虚生花

Right, everyone thinks the big tech companies like Microsoft Amazon and Google.

现在可能每个人都觉得微软，亚马逊和谷歌才是大型科技公司

53

05:35–05:38 虚生花

Back in the 1970s –1980s, IBM was the juggernaut.

在1970–1980年代，IBM的地位至高无上

05:38–05:44

So essentially whatever IBM did or said, this is the way we're gonna do it that ended up being the standard.

因此从本质上来讲，不管IBM说了什么或者做了什么，最终都成为了行业标准

55

05:44–05:49

So when IBM first released their first commercial relational database system DB2.

当IBM首先发布了他们第一个商用关系型数据库DB2时

56

05:49–05:54

So they never actually released system R was just sort of a research prototype  
他们实际上从未放出System R，他们只是放出了研究原型

5.54-6.00

but then finally made DB2 or at least that DB2 supported SQL, So that  
essentially became the standard.

但他们最终做出了DB2，DB2支持SQL，因此SQL成为了标准

06:02-06:08 虚生花

And the reason why Oracle sort of took off and got as big as it is today is, they were  
sort of copying what IBM was doing in the 1970s.

之所以Oracle起飞并成为当今巨头的原因是因为他们抄袭了IBM在1970年代时所作的成果

06:08-06:12 虚生花

In more ways than one we could talk to her that later.

这其中原因有很多，我们会在之后讨论

06:12-06:22 虚生花

And they did they had SQL, so when IBM came out with DB2 and had SQL , Oracle was  
right at the right place at the right time said we now supports we know we support  
SQL too.

当IBM推出了DB2和SQL时，Oracle在正确的时间和正确的地点表示他们自己同样也支持了SQL

61

06:22-06:28 虚生花

So it became an ANSI standard in 1986 and became an international standard in 1987

因此，这在1986年成为ANSI标准，同时在1987年成为国际标准

6.28-6.31 winston

and now the short burden just means the Structured Query Language.

现在简称结构化查询语言（SQL）

SQL HISTORY

Current standard is **SQL:2016**

- **SQL:2016** → JSON, Polymorphic tables
- **SQL:2011** → Temporal DBs, Pipelined DML
- **SQL:2008** → TRUNCATE, Fancy ORDER
- **SQL:2003** → XML, windows, sequences, auto-generated IDs.
- **SQL:1999** → Regex, triggers, OO

Most DBMSs at least support **SQL-92**

→ System Comparison: <http://troels.arvin.dk>

CARNEGIE MELLON  
DATABASE GROUP

62

06:32-06:38 虚生花

So SQL even though it's from 1970s, it's not a dead language, it's not certainly static  
尽管SQL是1970年代的产物，但它并不是什么没人用的语言，它也不是静态类型语言

6.38-6.42 虚生花

, it's sort of like you know C++ keep new specifications every so often

就和你知道的C++那样，它一直与时俱进



6.42-6.49 虚生花

it's the same thing in SQL every so often there's a new specification where they add in new features and new functionalities to to the basic language.

对于SQL也是一样，人们经常为它添加新的特性和新的功能

06:49-06:53 虚生花

So the latest standard is defined in SQL 2016

最新的标准是在SQL2016版本时定制的

6.53-6.58

and you can see over the years at the add new versions they add new features.

你可以看到随着时间的推移，开发人员会在新的版本中添加新的特性

06:59-07:06 虚生花

Right, so 2016, they added JSON, Polymorphic tables, they add XML stuff and then 2003,1999 added Regex and triggers.

在2016版本中，他们添加了对JSON和多态表的支持，2003版本中则添加了对XML的支持，在1999版本中，则支持了正则和触发器

07:06-07:13

Typically what happens is there's a standards body that the members are all from the major database companies

通常，这会有一个标准，制定这种标准的成员都是来自主流数据库公司

7.13-7.17

and the major database companies come up with their own proprietary features and extensions,

主流的数据库公司会推出他们自己专有的功能和扩展

7.17-7.22

and then they go in the standards body and push to try to get their version of certain functionality as part of the standard.

然后他们会试着将他们的某些功能纳入标准

7.23-7.29

Right, so this is although there is a SQL standard, nobody actually follows it to the T. 这就是为什么虽然有SQL标准，但没人去遵守的原因了

07:28-07:34

Right, because everyone sort of has their own proprietary things that got invented before the standard came out said, this is how you should do things.

因为在标准出来之前（即该标准指引你应该怎么做），每个公司就已经发明了他们自己专有的东西

07:34-07:39 虚生花

So if you're gonna claim that your database system supports SQL

因此，如果你声明你的数据库系统支持SQL

7.39-7.43 虚生花

the bare minimum you need to have is actually what is defined in the SQL-92 standard.

那么，你所需要满足的最低要求也得是SQL-92标准

07:43-07:53 虚生花

So this is what the basic SQL that we know about today select, insert, update, delete, create tables, transactions things like that all that's defined in SQL-92.

这也就是我们如今所熟知的基本SQL，例如select，insert，update，delete，创建表，事务这些东西都在SQL-92中定义了

07:53-07:57 虚生花

So again, if someone says their databases system supports SQL, chances are they that really mean this.

因此，如果有人说他们的数据库系统支持SQL，他们所说的其实就是这个意思

07:57-08:07 虚生花

And then the more advanced databases both an open source and the commercial ones, they add a bunch of more features from the newer standards.

无论是开源的还是商用的高级数据库系统，它们都添加了很多新标准中的特性

08:07-08:18

And then there's this great website here it's a bit dated now, but it's going at its some random dude who basically looked at sort of the top four, top five database systems.

这里有个很棒的网站，上面有人列出了前四或者前五的数据库系统

08:18-08:24

And just look to see how they differ on various SQL features it compares SQL functionalities.

上面可以看到不同数据库间SQL特性以及功能的比较

08:24-08:26

And we'll see this as we go through today

我们会在今天的课上看到这个

8.26-8.33

there'll be some examples where the standard says one thing different database systems do other things it's usually MySQL.

这其中会涉及到一些例子，比如，标准SQL下一件事情在不同数据库系统间是怎么做的，通常我们看MySQL就可以了

例如，标准SQL是如何做的，其他的数据库系统又是如何处理的，通常我们看MySQL就可以了

08:33-08:40

And this is just an even though there's a standard nobody actually follows it exactly.

尽管有标准实现，但实际上并没有人去按照标准去做

08:40-08:45

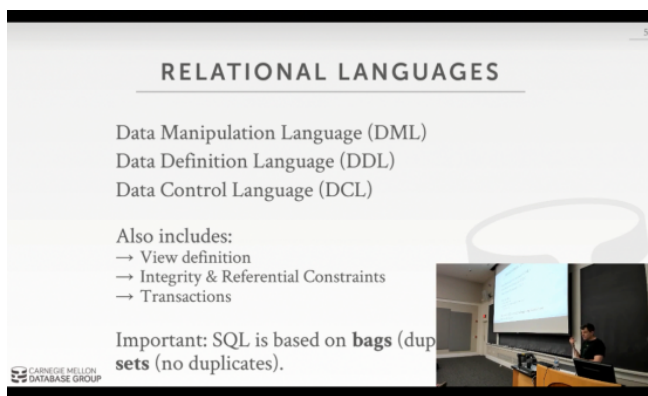
Right, there's no database system that I'm aware of that would claim that they're you know certify for SQL:2016.

据我所知，目前没有任何数据库系统通过了SQL 2016标准的认证

08:45-08:47

Right, they have bits and pieces of things.

它们只支持了一些很零碎的功能



注：

DML数据库操作语言

DDL数据库定义语言

DCL数据库控制语言

08:49-08:55 虚生花

So SQL itself technically is not a single language, it's sort of a collection of things.

从技术层面而言，SQL并不是一门单一语言，它是某些东西的集合

08:55-09:00 wisnton

In a particular, it's a collection of a DML, DDL and DCL commands.

具体一点,它是一个DML，DDL和DCL命令的集合。

09:00-09:11 虚生花

So DML will be would be the Data Manipulation Language would be the commands, like insert, update, delete, selects like the things that actually manipulate the data that you can store in your database.

DML其实就是数据操作语言，它是一种命令，例如insert、update、delete、select这些命令来操作你存在数据库中数据

09:11-09:17 虚生花

The DDL is the way you create tables actually define **schemas** to actually store things.

DDL是一种你通过创建schema来存储东西的方式

DDL是一种你可以通过定义**schemas** 来创建表存储数据的方式

09:17-09:24 虚生花

And then the DCL is the way you sort of do security authorization to grant you know who's allowed to read what data.

然后是DCL，它是某种关于安全性授权的东西，它可以用来控制哪些人可以读取哪些数据

09:24-09:31 虚生花

There's a bunch of other things like, how to define views, how to define integrity and strange referential constraints.

当然，这里面还包含了一些其他东西，例如如何定义视图，如何定义完整性以及奇怪的参照约束

09:31-09:34 虚生花

As well as transactions

当然还有事务相关

9.34-9.36 虚生花

these are all part of the umbrella of SQL.

这就是SQL的组成了

09:36–09:40 虚生花

Right, and within that there's these different categories of commands.

其中有很多不同种类的命令

09:40–09:51

So the one important thing that I want to point out here, and we'll see this throughout today's lecture is unlike in relational algebra which is based on set theory or sets.

我想指出一件重要的事，这也是我们今天课上的内容，它和关系代数不同，它是基于集合论的某种东西

09:51–09:54 虚生花

SQL is actually based on bag algebra.

SQL实际上是基于bag algebra

09:54–09:59 虚生花

So the way it sort of thing about this you could have like lists sets or bags.

你可以把它当成list, set或者bag

09:59–10:01

So a list can have duplicates

在一个list中可以有重复元素

10.01–10.07

but there's a defined orders, if I push something to my list that's its position in that list.

但它里面是有顺序的，如果我将某个元素添加到list上，添加的位置就是它在list中的位置

10:07–10:09

A set is unordered

set中的元素顺序是无序的

10.09–10.13

meaning the elements don't have a position, but you can't have duplicates

这意味着元素并没有固定位置，并且你不能有重复元素

10.13–10.18

if I try to insert the same thing into a set, it just gets over the old run gets overwritten.

如果我试着向一个set中插入相同元素，它就会将原来的元素覆盖掉

10:18–10:25

A bag has neither a set position or ordering, but it also allows for duplicates.

在bag中既没有固定位置也没有顺序，但它允许元素重复

10:25–10:29

Right, and we'll see why we've got to do this as we go along

随着我们的讨论，我们会看到我们为什么必须这样做

10.29–10.40

because if we want to actually define order on our elements, or if you want to make sure that we don't have duplicates, essentially the database system has to do extra work

to to provide that for you.

因为如果我们想让我们的元素有顺序，或者你想保证没有重复元素出现，那么本质上来说，数据库必须为我们做一些额外工作才能提供这种功能

!!!!!!!

10:40–10:49

And so the idea is that only if you explicitly ask the database system to provide you ordering and provide you to remove duplicates.

只有你显式地要求数据库系统为你提供排序以及移除重复元素

即只有当你明确的告诉数据库系统：为我提供排序，为我删除重复元素

10:49–10:53 winston

It won't actually do it and this actually make things be more efficient.

它做不到这样，实际上是更高效处理事情。

此时它真的做不到，而实际上，如果能做到，真的可以让事情处理的更高效



10:54–11:03 Winston

All right, so the outline for today is we're going to cover Aggregations + Group Bys, a bunch of operations on Strings / Dates and Times, then we had to do Output control.

今天的课纲，我们要讲 聚合函数 + Group By；处理字符串、日期和时间类型；然后还有 Output control。

11:03–11:08 Winston

And then the more complicated things have been Nested Queries, Common Table Expressions and window functions.

接着是更复杂的Nested Queries、Common Table Expressions(CTE)和window functions。

97

11:08–11:14

So for homework one, you'll need to use all of these except for the window functions.

在作业1中，你会需要用到除了window函数以外的所有功能

98

11:14–11:16

All right, it's gonna be doing homework one on SQLite.

你们会使用SQLite来完成作业1

11:16–11:22 虚生花

And only actually the latest version of SQLite as of last week, just added support for window functions

实际上，在上周放出的SQLite最新版中，它添加了对window函数的支持

11.22-11.24 虚生花

but everything else SQLite should be able to support.


但对于其他东西，SQLite都支持

## EXAMPLE DATABASE

| sid   | name   | login      | age | gpa |
|-------|--------|------------|-----|-----|
| 53666 | Kanye  | kayne@cs   | 39  | 4.0 |
| 53688 | Bieber | jbieber@cs | 22  | 3.9 |
| 53655 | Tupac  | shakur@cs  | 26  | 3.5 |

| sid   | cid    | grade |
|-------|--------|-------|
| 53666 | 15-445 | C     |
| 53688 | 15-721 | A     |
| 53688 | 15-826 | B     |
| 53655 | 15-445 | B     |
| 53666 | 15-721 | C     |

| cid    | name                         |
|--------|------------------------------|
| 15-445 | Database Systems             |
| 15-721 | Advanced Database Systems    |
| 15-826 | Data Mining                  |
| 15-823 | Advanced Topics in Databases |



100

11:29-11:34 Winston

Okay all right, so for this we're needs a sample database comprised of three tables.

我们需要一个含有三张表的简单数据库

101

11:34-11:37 Winston

So sort of it's a mock University.

像这样，它是一个虚拟的大学（名）。

11:37-11:41 虚生花

So we're gonna have a student table where students have student IDs, names, logins and GPA

我们会有一张student表，上面有学生的id，name，login以及GPA

11:41-11:44 虚生花

will have a course table with course cids and name.

然后有一张course表，上面有课程id以及name

11:44-11:46 虚生花

And then we'll have an enrolled table

接着，我们会有一张enrolled表

11:46-11:54 虚生花

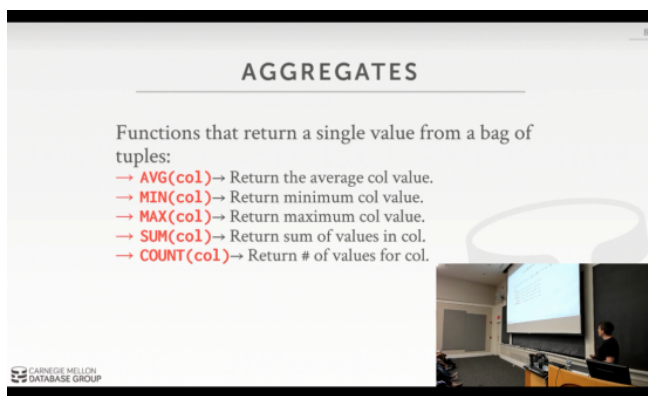
where we have a certain foreign key reference from the student table, and the course table and along with the grade that the student got in the class.

在这张表中我们使用student表中的sid作为外键使用，里面有course表中的cid以及学生在这门课中所得的分数

11:54-11:59

All right, we'll use this as our running example as we go along.

我们会将它作为我们讨论的例子使用



12:00–12:03 虚生花

All right, so the first thing we talk about our aggregations.

首先我们要讨论聚合函数

12:03–12:05 虚生花

Right, and these are pretty simple to understand

这些很容易去理解

12:05–12:10 虚生花

it's basically a function that you define in the output list of your select statement

它基本上就是一个函数，它会返回你所select语句执行的所得到的结果

12:10–12:15 虚生花

that's going to take as input multiple tuples a set of tuples

它将多个tuple作为输入

12:15–12:20 \*\*\*\*\*

and it's gonna compute some kind of aggregation on top of that and produced a single result.

它会在在此基础上计算某种聚合并产生单个结果

12:20–12:25

Right, so the SQL-92 standard defines AVG(),MIN(),MAX(),SUM() and COUNT().

在SQL-92标准中定义了AVG()、MIN()、MAX()、SUM() 以及 COUNT()函数

12:25–12:36

And again think of this is like in case of count, you're gonna take a bag of tuples as the input, and you're gonna count the number of tuples there are and that you're a produces a single output that has that that count.

例如，在COUNT()函数中，你将一组tuple作为输入，然后你要去数出tuple的个数，接着你将数出来的结果作为单个值输出

12:36–12:49

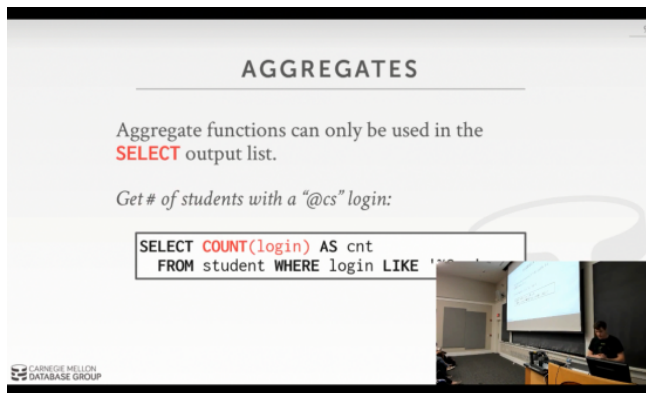
Right, it's again so this is what's in the basic standard, the the later version of the standard and in other database systems they'll have other things like median, mode,standard deviation.

这只是一中基础标准，在之后的标准和其他数据库系统中，它们还提供了例如中位数，取模以及标准偏差之类的东西

12:49–12:54

Right, in the github different aggregates and some of them actually that allow you to find your own aggregations as well.

某些数据库系统也允许你使用你自己的聚合函数



12:55–12:57 winston

So let's look an example like this.

我们来看下这个例子。

12:57–13:05

Right, so say we want to count the number of students in this student table where the login ends with "@CS".

假设我们想去数出student表中login字段中以"@cs"结尾的学生数量

113

13:05–13:08

I'll cover what like is means later on

我会在之后介绍LIKE的意思

13.08–13.09

but essentially it's just looking for a wild card here.

但本质上来说，这里它是用做通配符使用

13:09–13:18 winston

Right, so the first thing the most important thing to remember about aggregations is that the aggregation can only appear in the output list of the select statement.

首先记住聚合函数最重要的事：聚合函数只能出现在 select 子句中作输出列

13:18–13:24 winston

Right, so I have it here in the in this it's saying, this is I want to produce at the output, I can't have it in these in these other parts here.

这里它（PPT）说，这就是我想要产生的输出（结果），输出中不能有其他字段

我这在输出部分产生（结果），

还不能有它的其余字段。

13:24–13:26 winston

Right, cuz it doesn't really make sense.

它（其余部分在这）没意义



13:26-13:32

Right, because you're computing aggregation after you've sort of applied a filter to figure out what tuples actually match your where clause.

根据 where 子句过滤条件找出实际匹配的tuple，之后聚合运算

13:32-13:39 虚生花

But the same point out here is that in case of count(), again we just want to count the number of tuples.

但同样，在此处的COUNT()例子中，我们只想去数出tuple的数量

13:39-13:43 虚生花

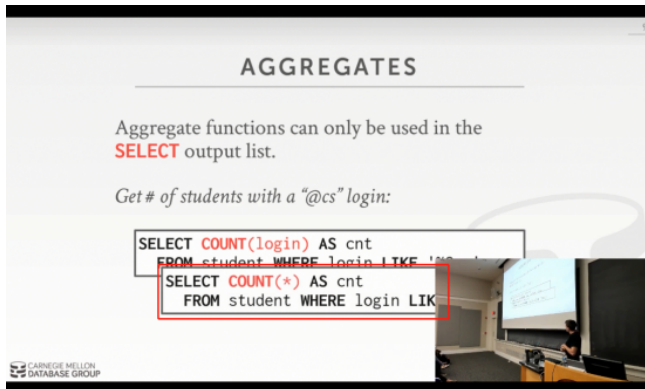
The login field here doesn't actually mean anything.

这里的login字段实际上没有任何意义

13:43-13:51 虚生花

Right, because we're just counting the number of tuples, it doesn't matter whether you know what the login actually is at the point after we do the filtering.

因为我们只想数出过滤完后的tuple的个数，这与login字段的内容是什么并没有关系



13:51-13:55

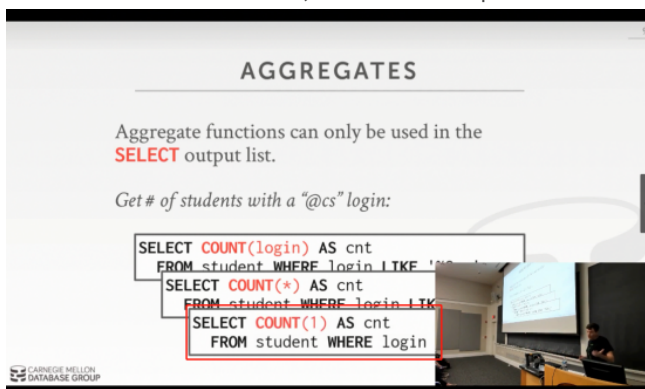
So we can rewrite this just to have a \*(star).

So，我们可以使用一个"\*"来重写这个

13:55-14:02

Right, the star is a special you know special keyword in SQL the basically says all attributes for the tuple.

"\*"是SQL中的特殊关键字，它代表了该tuple中的所有属性



14:02-14:07

Begin go even further and say we can actually replace the star with a one.

实际上更进一步，我们可以将这个“\*”用1替换

14:07-14:12

Right, count the number of tuples by just adding one every single time.

即每数一个tuple，tuple的数量就加1

14:12-14:21

Right, so this is a good example

因此，这是一个好例子

where we have three different queries that all produce or semantically the same to produce the same result.

因为我们有三种不同的查询，但它们都能产生语义上相同的结果

14:21-14:28

But the database system could choose different flavours or different variations of this, in order to derive that answer

但为了得出答案，数据库系统可以去选择这种查询的不同变体

14.28-14.30

and some of them may have different performance differences.

它们中的某些在性能上可能有不同的差异

14:30-14:31

This one's pretty simple

这种可能相当简单

14.31-14.35

so the most differences would be smart to realize,

我们能很快意识到此处最大的差别在哪

14.35-14.36

I don't need a copy around the login here,

我不需要在COUNT中传入login

14.36-14.38

I can just replace that with a one.

我可以用1去替换此处的login

14:38-14:39

Right, most of them will do that.

大部分数据库系统都会这么做

14:39-14:42

But from what complicating things it may not always work.


但如果遇到复杂的场景，它就不一定会一直有效

10

## MULTIPLE AGGREGATES

Get the number of students and their average GPA that have a "@cs" login.

```
SELECT AVG(gpa), COUNT(sid)
FROM student WHERE login LIKE '@cs'
```



CARNEGIE MELLON  
DATABASE GROUP

14:44–14:47

We actually can combine the aggregate multiple a grits together in a single query.

实际上，我们可以在单个查询中放入多个聚合函数

14:47–14:54

So say for this one we want to get the number of students and their average GPA where their login ends with "@CS".

假设，在这个例子中，我们想去得到login字段中以"@cs"结尾的学生的数量以及他们的平均GPA


10

## MULTIPLE AGGREGATES

Get the number of students and their average GPA that have a "@cs" login.

```
SELECT AVG(gpa), COUNT(sid)
FROM student WHERE login LIKE '@cs'
```

| AVG(gpa) | COUNT(sid) |
|----------|------------|
| 3.25     | 12         |



CARNEGIE MELLON  
DATABASE GROUP

132

14:54–15:02 虚生花

Right, so now you see I've combined it average GPA and count in my output list for my select statement.

现在你可以看到，我在SELECT语句中将AVG(GPA)和COUNT(sid)都放在了一起

15:02–15:04 winston

And then it'll produce my my result like that.

然后它产生的结果就像我这里展示的这样

15:04–15:06 winston

Right, pretty straightforward.

很简单。

11


## DISTINCT AGGREGATES

COUNT, SUM, AVG support DISTINCT

Get the number of unique students that have an "@cs" login.

```
SELECT COUNT(DISTINCT login)
FROM student WHERE login LIKE '@cs'
```

CARNEGIE MELLON  
DATABASE GROUP



115

15:07–15:16

I can also add the distinct keyword to tell it to only count the distinct elements, or values of attributes for my tuples.

我也可以在SQL语句中加入DISTINCT关键字来告诉数据库，让它只去数我tuple中出现的不重复的元素，或者是不重复的属性值

15:16–15:24

So this is saying count the number of unique students with unique logins from the student table where the login ends with "@CS".

即计算在Student表中，login字段中以"@cs"结尾的不重复的学生个数

15:24–15:28 虚生花

Right, and so I had the distinct side of account.

这样，我就得到了不重复的账号个数

15:28–15:30 虚生花

Now this one's sort of nonsensical to some way, 某种意义上来说，这可能有点荒谬

15:30–15:38 虚生花

because presumably no two students can have the same login, you know login account otherwise you'd have problems.

因为，想必不存在有两个学生的登录账户是相同的情况，不然这就出问题了

15:38–15:46 虚生花

But in other cases you know, you can apply the same kind of thing for other scenarios and it would work the way you wanted to work.

但在其他例子中，你可以将这种运用在其他相同的场景之中，并且它也会按照你想的那样去做

11

## DISTINCT AGGREGATES

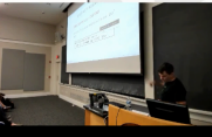
COUNT, SUM, AVG support DISTINCT

Get the number of unique students that have an "@cs" login.

```
SELECT COUNT(DISTINCT login)
FROM student WHERE login LIKE '@cs'
```

| COUNT(DISTINCT login) |
|-----------------------|
| 10                    |

CARNEGIE MELLON  
DATABASE GROUP



15:46–15:50

Right, in this case here we produce the same result.

在这个例子中，我们产生了相同的结果

15:50–15:56 虚生花

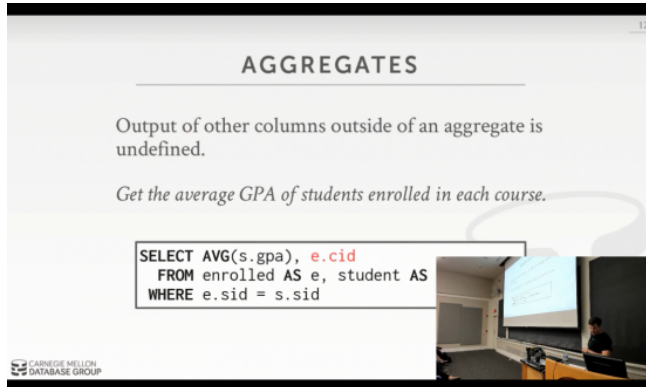
So the one thing that now you may want to try to start doing is now that I'm start doing aggregations

现在我要开始计算聚合

15:56–16:02 虚生花

I want to get additional information about my data, outside of just what I'm computing in my aggregate.

除了我现在计算的聚合以外，我还想从我的数据中得到额外的信息



16:03–16:13 虚生花

Right, so say that I want to get the average GPA of the students that are enrolled in the course, and I want to know what that e.cid was.

假设，我想知道选这门课学生的平均GPA以及这门课的cid

16:13–16:19 虚生花

Right, so in this case here I've now added the e.cid to my output list outside of my aggregation.

因此，在这个例子中，我将e.cid加到了聚合之外的输出列表中

16:19–16:23 winston

Right, did you take a guess what would happen here.

你们猜下这个SQL执行结果

16:28–16:29 winston

Would this work or not ?

这个SQL 能不能运行?

16:32–16:36 winston

Raise your hand, you think would work.

Raise your hand, if you think it wouldn't work.

举下手，你认为这能运行。

举下手，如果你认为这不能运行。

16:37-16:38 winston

See of you, why?

来，就你，为什么？

128

16:43-16:52

(Judy 说的)

16:52-17:00 虚生花

Correct, yes, so he said is there's not a single e.cid for all my tuples that are competing my average on.

没错，他说的对，在我用来计算平均GPA的所有tuple中，它们的cid并不是一样的

129

17:00-17:05

Right, it's all the students are taking all the different classes

所有的学生上的课都是不同的

what e.cid to actually put is the output.

那么在输出结果中的e.cid实际上是什么呢？

17:05-17:14 虚生花

Right, it's the SQL standard say, this is actually undefined and in most systems you actually will get an error with this.

按照SQL的标准来讲，这实际上并没有被定义，在大部分的系统中你得到关于这个的报错

17:14-17:16 虚生花

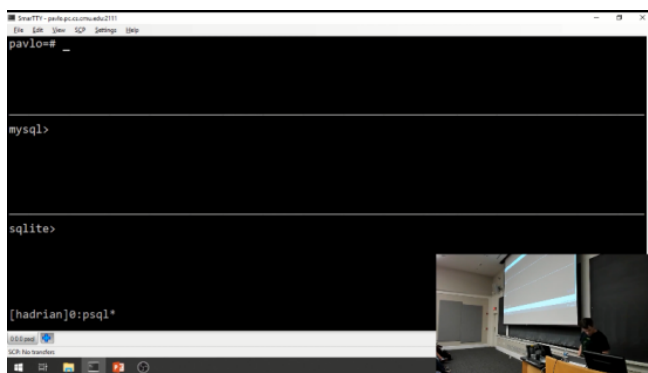
We can actually test this.

我们其实可以测试一下

17:16-17:21 虚生花

Right, so in this ,here we go.

开始



17:22-17:24 winston

I have three three terminal setup.

我开了3个终端

17:24-17:33 winston

All right, so this is running a machine back in my office, I have three panels, the top one is PostgreSQL, the bottom one or the middle one is MySQL, and the bottom one is SQLite.  
这是在我办公室运行的机器,  
我有三个面板,  
上面的是PostgreSQL,  
中间的是MySQL,  
然后是SQLite。

17:35-17:39

It's as much easier for you to type from this machine here so I can log into that.  
对于你们来说, 用这台机器进行输入要更简单, 我可以在这里登录

17:42-17:48

Right, okay, so the query was we wanted to select, where was it.  
此处我们想进行select操作

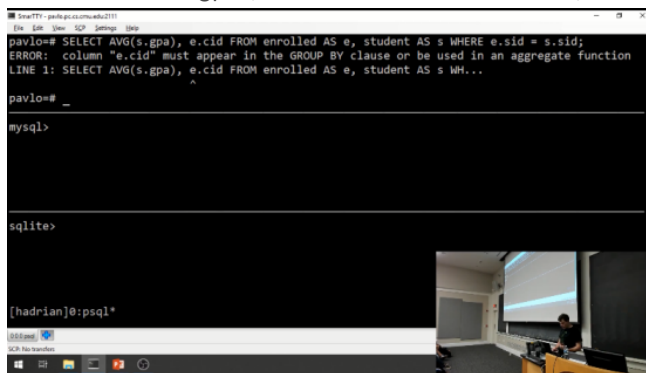
17:50-17:53

Just get the average GPA of students enrolled in each course.  
找到每门课程注册学生的平均GPA

17:53-18:14

Right, so 'SELECT AVG(s.gpa), e.cid FROM enrolled AS e, student AS s WHERE e.sid = s.sid;

SELECT AVG(s.gpa), e.cid FROM enrolled AS e, student AS s WHERE e.sid = s.sid;



139

18:14-18:23 winston

Right,so PostgreSQL says, you can't do this because as he said the e.cid and I'm highlighting here,so you can see it.  
PostgreSQL说, 你不能这样做, 因为它在e.cid这有标记, 你们看。

18:23-18:27 虚生花

So the cid is not defined it's not part of the aggregation.  
此处的cid并没有被定义, 它并不是聚合的一部分

18:27-18:30 虚生花

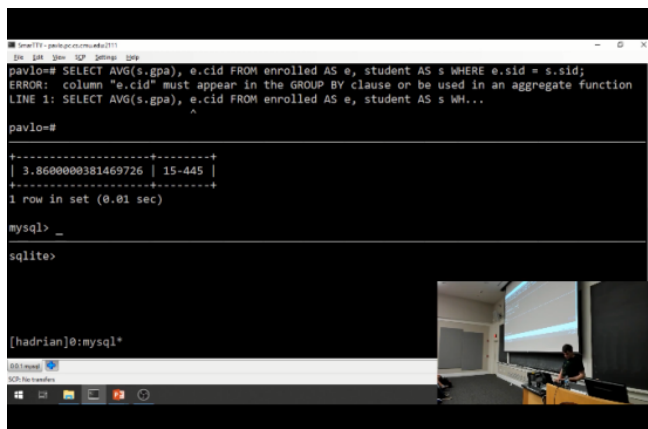
Right, so it doesn't know which cid you actually want.

因此，PostgreSQL并不知道你实际想要的是哪个cid

18:30–18:35 winston

So now if we go down and try this in MySQL.

现在我们来试一下MySQL



```
pavlo=# SELECT AVG(s.gpa), e.cid FROM enrolled AS e, student AS s WHERE e.sid = s.sid;
ERROR: column "e.cid" must appear in the GROUP BY clause or be used in an aggregate function
LINE 1: SELECT AVG(s.gpa), e.cid FROM enrolled AS e, student AS s WH...
^
pavlo=#
+-----+-----+
| AVG(s.gpa) | cid |
+-----+-----+
| 3.8600000381469726 | 15-445 |
+-----+-----+
1 row in set (0.01 sec)

mysql>
sqlite>

[hadrian]@mysql*
```

18:40–18:41 winston

MySQL gave us an answer.

MySQL 给了我们一个结果。

144

18:43–18:46 winston

Right, but is that correct?

但是正确的吗？

145

18:46–18:51 winston

No right, because what cid do the pick I picked a random one.

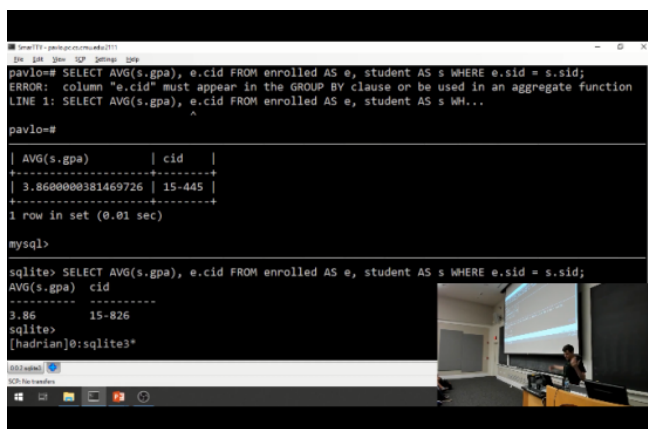
不对，因为在cid中给我随便挑了个。

146

18:52–18:56 winston

Right, and then now we can try in SQLite same thing.

现在我们同样来试SQLite



```
pavlo=# SELECT AVG(s.gpa), e.cid FROM enrolled AS e, student AS s WHERE e.sid = s.sid;
ERROR: column "e.cid" must appear in the GROUP BY clause or be used in an aggregate function
LINE 1: SELECT AVG(s.gpa), e.cid FROM enrolled AS e, student AS s WH...
^
pavlo=#
+-----+-----+
| AVG(s.gpa) | cid |
+-----+-----+
| 3.8600000381469726 | 15-445 |
+-----+-----+
1 row in set (0.01 sec)

mysql>
sqlite> SELECT AVG(s.gpa), e.cid FROM enrolled AS e, student AS s WHERE e.sid = s.sid;
AVG(s.gpa) cid
-----
3.86      15-826
sqlite>

[hadrian]@sqlite3*
```



18:57-19:00 (winston)

SQLite gave us a different cid.

SQLite 给了我们不同的cid

19:00-19:04 虚生花

So you see they both computed the correct average, but they chose different cid.

你可以看到MySQL和SQLite都算出了正确的平均值，但它们选择了不同的cid进行计算

19:05-19:10

All right, and just because I know that MySQL guys watch these videos and complain.

All right, 我知道看视频的小伙伴在这里会对MySQL有疑惑

19:12-19:19 winston

I will say that so this is running MySQL in and, so MySQL traditionally allowed you to do loosey goosey things like this.

我要说MySQL运行时，MySQL原则上允许你干不合规的事

19:19-19:26 winston

Right, so this would be running it what they would call it **traditional mode**, but you can set the **SQL mode** to be more strict.

他们把这种执行称为严格模式，但你可以设置**SQL mode**让它更严格。

在MySQL中，这种运行模式我们称之为传统模式，但你也可以将SQL的模式设置的更为严格

19:27-19:30 虚生花

So now if I run that same query, it throws the same error that PostgreSQL did.

现在，如果我执行相同的查询，它会报出和PostgreSQL同样的错误

19:30-19:37 虚生花

So by default the least MySQL 5.7 will now throw errors in older versions they didn't do that .

默认情况下，MySQL 5.7及以上版本会报错，然而老版本的MySQL并不会报错

**GROUP BY**

Project tuples into subsets and calculate aggregates against each subset.

```
SELECT AVG(s.gpa), e.cid
FROM enrolled AS e, student AS s
WHERE e.sid = s.sid
GROUP BY e.cid
```

CARNEGIE MELLON  
DATABASE GROUP

19:40-19:45 虚生花

OK, So a way to fix this is to do GROUP BY.

修复这个问题的一个方法就是使用GROUP BY

19:45-19:58

So with GROUP BY I what's gonna happen is, now we're gonna define how we want to essentially bucket it together, the tuples in our output based on one attribute.

GROUP BY所做的就是，基于某个属性将我们想要的tuple放在一起，即物以类聚

19:58-20:02

And then now we can then compute the aggregation on the tuples in each bucket.

接着，现在我们可以对每个bucket中的tuple进行聚合计算

158

20:03-20:07

Right so again, what I wanted to do was get the AVG(s.GPA) per course

我现在想做的就是得到每门课的平均分

20:07-20:11 winston

I add now in my group by clause the course ID.

我现在把课程ID加在group by子句中