MySQL数据库开发教程

第一节 MySQL系统简介

01

MySQL的简介

MySQL介绍

MySQL

- MySQL是一个关系型数据库管理系统，由瑞典MySQL AB 公司开发，目前属于Oracle 旗下产品。MySQL 是最流行的关系型数据库管理系统之一，在 WEB 应用方面，MySQL是最好的 RDBMS (Relational Database Management System，关系数据库管理系统) 应用软件之一。
- MySQL是一种关系数据库管理系统，关系数据库将数据保存在不同的表中，而不是将所有数据放在一个大仓库内，这样就增加了速度并提高了灵活性。

MySQL的安装

本教案使用MySQL 5.x作为教学软件

- 实战1： 安装MySQL软件。
- 实战2： 安装图形客户端软件。
- 实战3： 注册成后台服务软件。

02

MySQL的日常使用

MySQL的日常使用

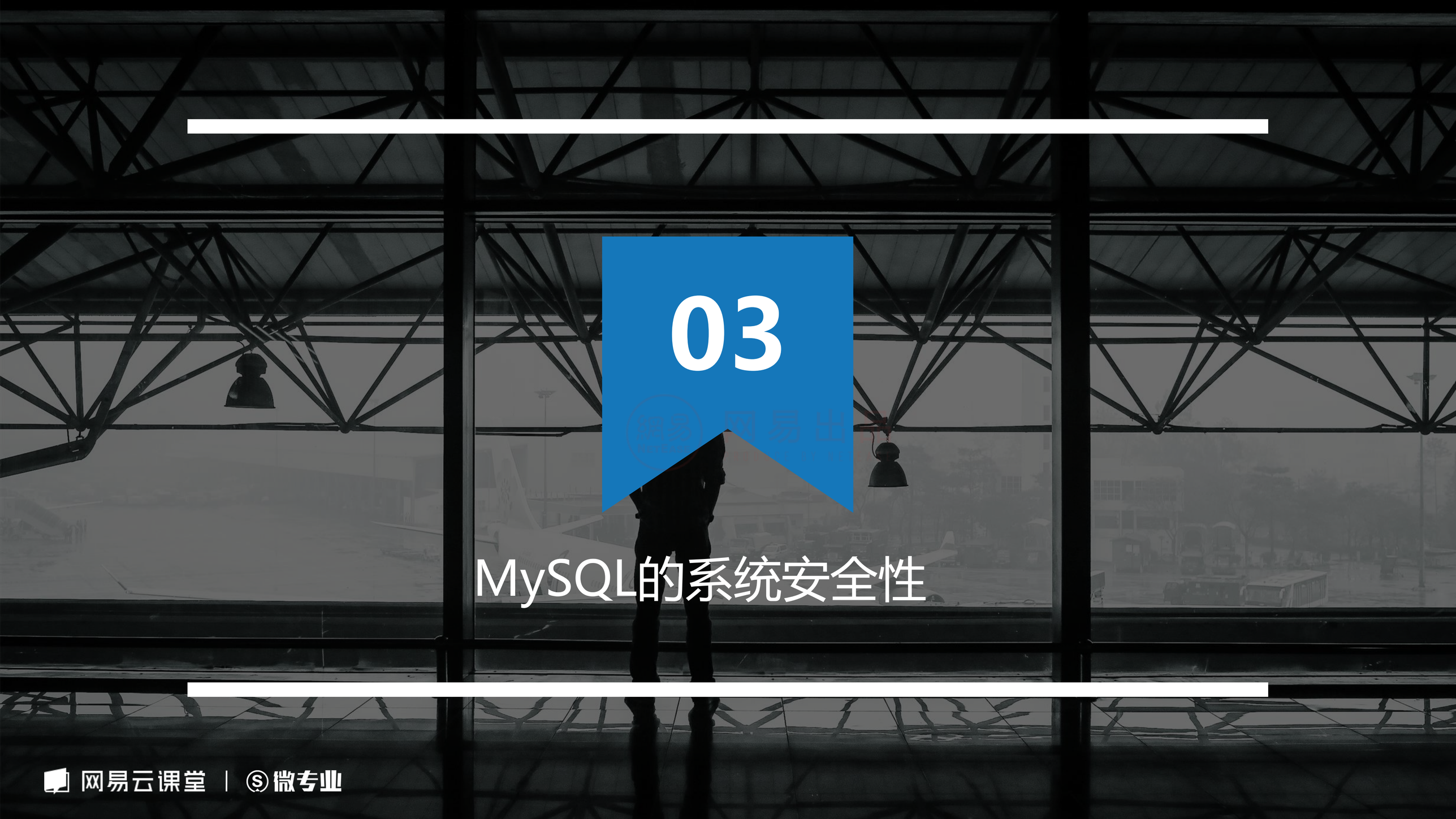
MySQL客户端连接

- 为了连接到MySQL，需要以下信息：
- 主机名（计算机名）——如果连接到本地MySQL服务器，为localhost；
- 端口（如果使用默认端口3306之外的端口）；
- 一个合法的用户名；
- 用户口令（如果需要）。

MySQL的日常使用

实战：了解MySQL的数据库和表

- 实战1：show databases
- 实战2：show tables
- 实战3：show columns from xxx table
- 实战4：show status
- 实战5：show grants
- 实战6：show errors / warings



03

MySQL的系统安全性

MySQL的系统安全性

MySQL安全原则

- 数据库服务器通常包含关键的数据，确保这些数据的安全和完整需要利用访问控制
- MySQL服务器的安全基础是：用户应该对他们需要的数据具有适当的访问权，既不能多也不能少。换句话说，用户不能对过多的数据具有过多的访问权。

MySQL的系统安全性

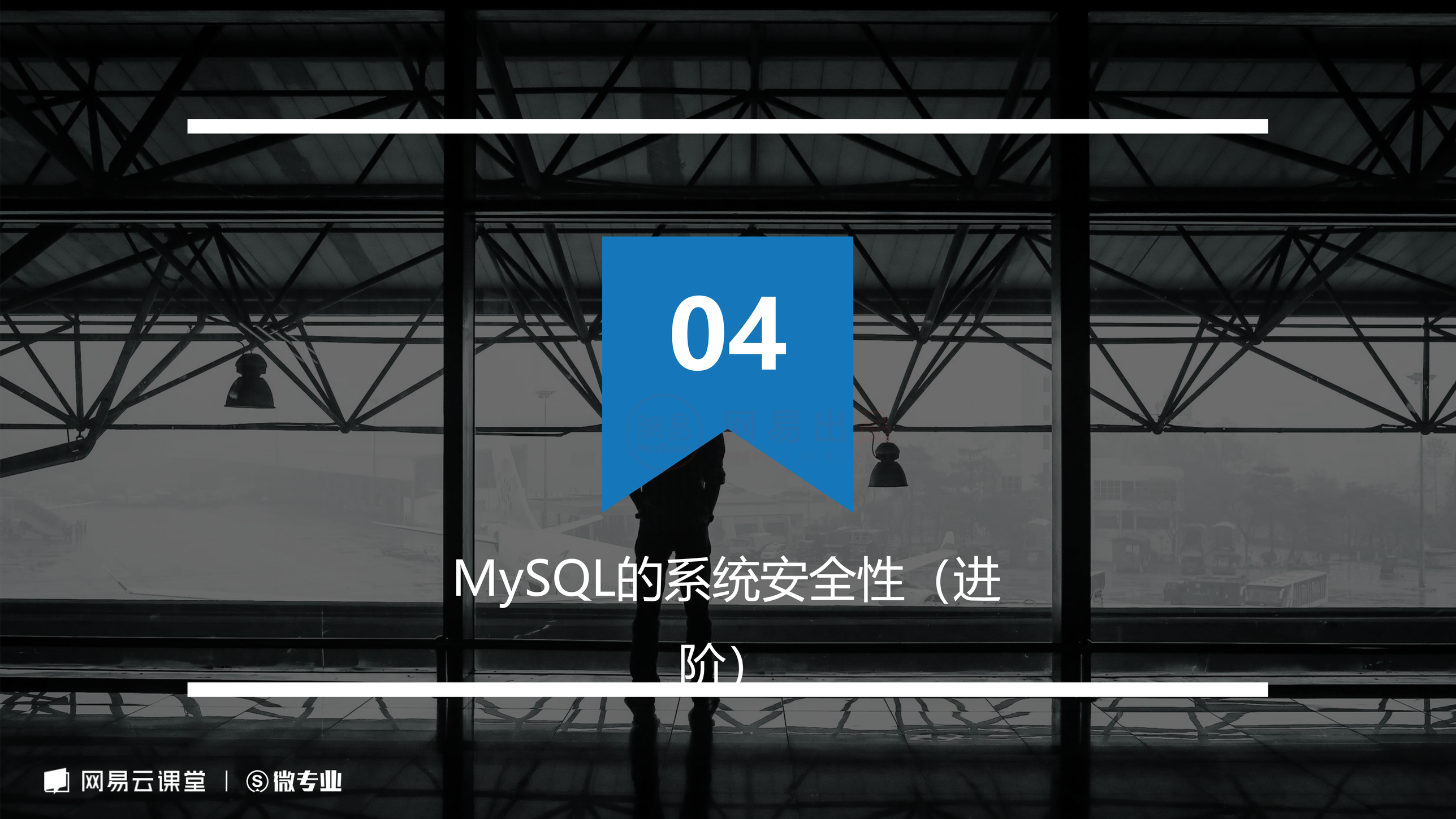
MySQL安全原则

- 不要使用root 应该严肃对待root登录的使用。仅在绝对需要时使用它（或许在你不能登录其他管理账号时使用）。不应该在日常的MySQL操作中使用root。

MySQL的系统安全性

实战：了解MySQL的安全性

- 实战1：管理用户
- 实战2：创建用户账号
- 实战3：删除账号
- 实战4：设置访问权限
- 实战5：更改密码



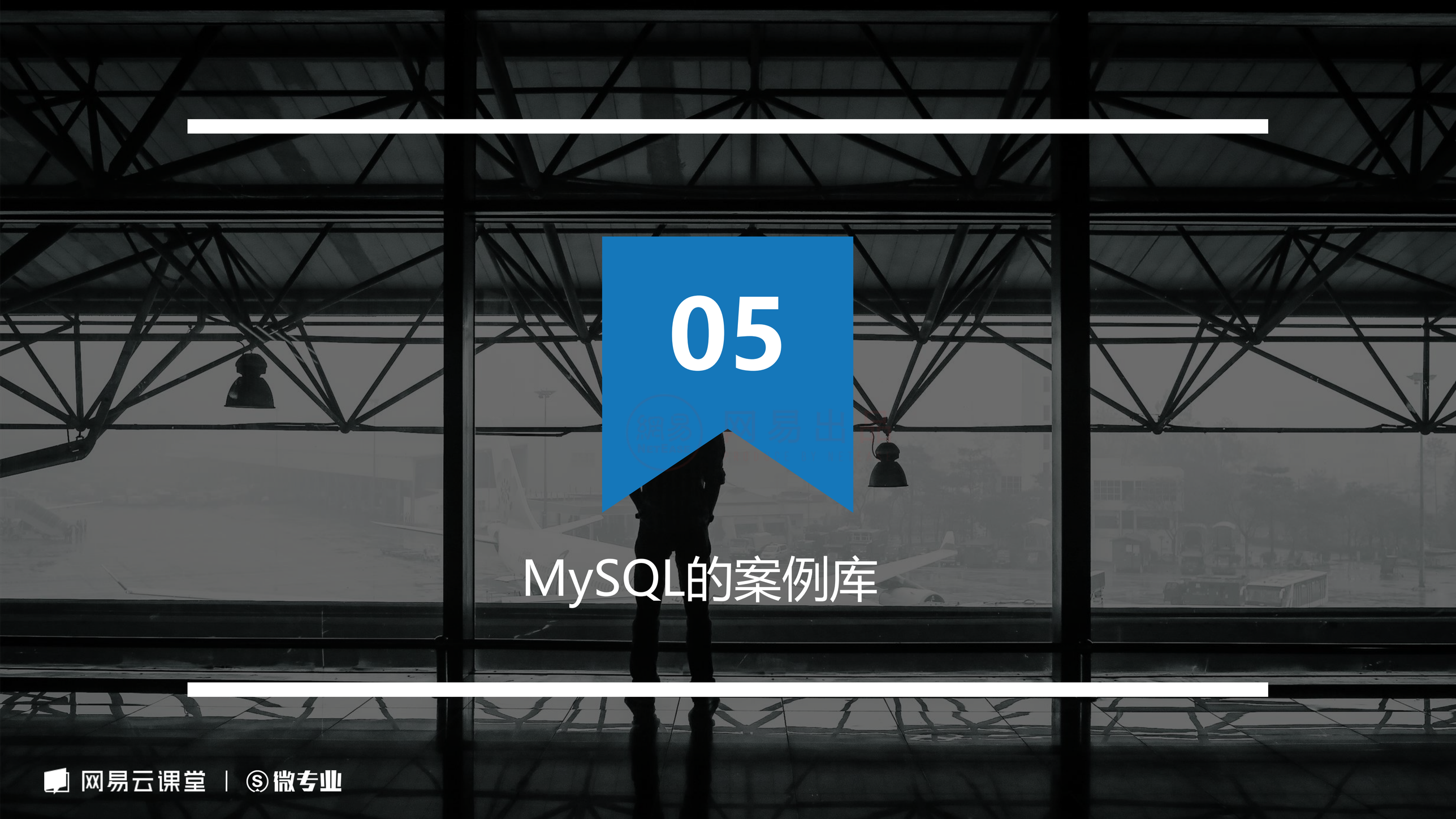
04

MySQL的系统安全性（进阶）

MySQL的系统安全性

实战：了解MySQL的安全性

- 实战1：普通用户密码丢失情况下的恢复
- 实战2：Root 密码丢失情况下的恢复



05

MySQL的案例库

MySQL的案例库

了解MySQL的案例数据库

- 本课程中使用的样例表为一个想象的随身物品推销商使用的订单录入系统
- 这些表用来完成以下几个任务：
- 管理供应商；
- 管理产品目录；
- 管理顾客列表；
- 录入顾客订单。

MySQL的案例库

了解MySQL的案例数据库

- 要完成这几个任务需要作为关系数据库设计成分的紧密联系的6个
- 表。以下几节描述各个表。
- vendors表
- vendors表存储销售产品的供应商。每个供应商在这个表中有一个记
- 录，供应商ID（vend_id）列用来匹配产品和供应商。

MySQL的案例库

了解MySQL的案例数据库

- 要完成这几个任务需要作为关系数据库设计成分的紧密联系的6个
- 表。以下几节描述各个表。
- products表
- products表包含产品目录，每行一个产品。每个产品有唯一的ID
- （prod_id列），通过vend_id（供应商的唯一ID）关联到它的供应商。

MySQL的案例库

了解MySQL的案例数据库

- 要完成这几个任务需要作为关系数据库设计成分的紧密联系的6个
- 表。以下几节描述各个表。
- customers表
- customers表存储所有顾客的信息。每个顾客有唯一的ID（cust_id
- 列）。

MySQL的案例库

了解MySQL的案例数据库

- 要完成这几个任务需要作为关系数据库设计成分的紧密联系的6个
- 表。以下几节描述各个表。
- orders表
- orders表存储顾客订单（但不是订单细节）。每个订单唯一地编号
- （order_num列）。订单用cust_id列（它关联到customer表的顾客唯一ID）
- 与相应的顾客关联。

MySQL的案例库

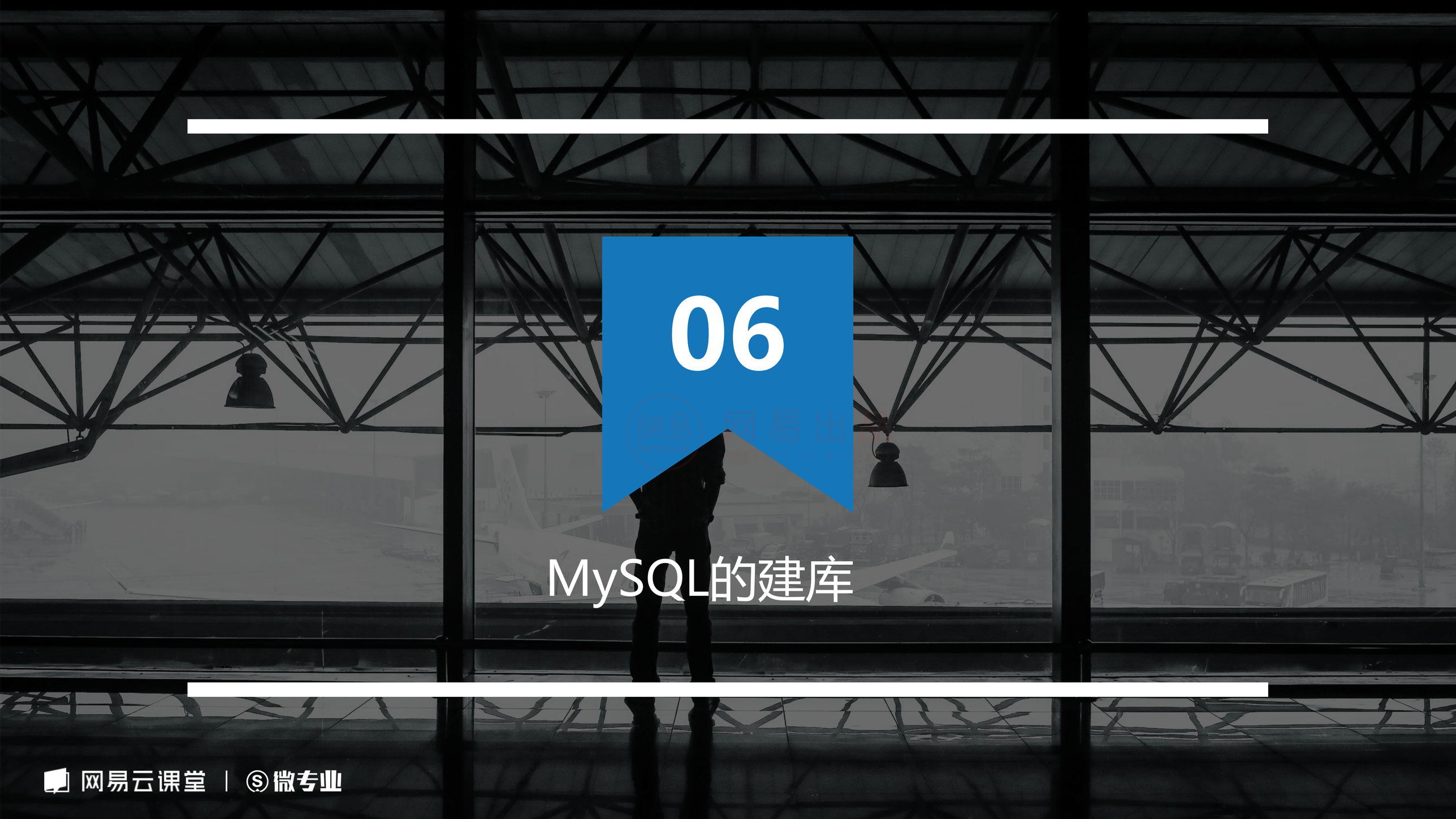
了解MySQL的案例数据库

- 要完成这几个任务需要作为关系数据库设计成分的紧密联系的6个
- 表。以下几节描述各个表。
- orderitems表
- orderitems表存储每个订单中的实际物品，每个订单的每个物品占
- 一行。对orders中的每一行，orderitems中有一行或多行。每个订单物
- 品由订单号加订单物品（第一个物品、第二个物品等）唯一标识。

MySQL的案例库

了解MySQL的案例数据库

- 要完成这几个任务需要作为关系数据库设计成分的紧密联系的6个
- 表。以下几节描述各个表。
- productnotes表
- productnotes表存储与特定产品有关的注释。并非所有产品都有相
- 关的注释，而有的产品可能有许多相关的注释。



06

MySQL的建库

MySQL的建库

了解MySQL的建库语句

- CREATE DATABASE [IF NOT EXISTS] <数据库名>
- [[DEFAULT] CHARACTER SET <字符集名>] [[DEFAULT] COLLATE <校对规则名>];

MySQL的建库

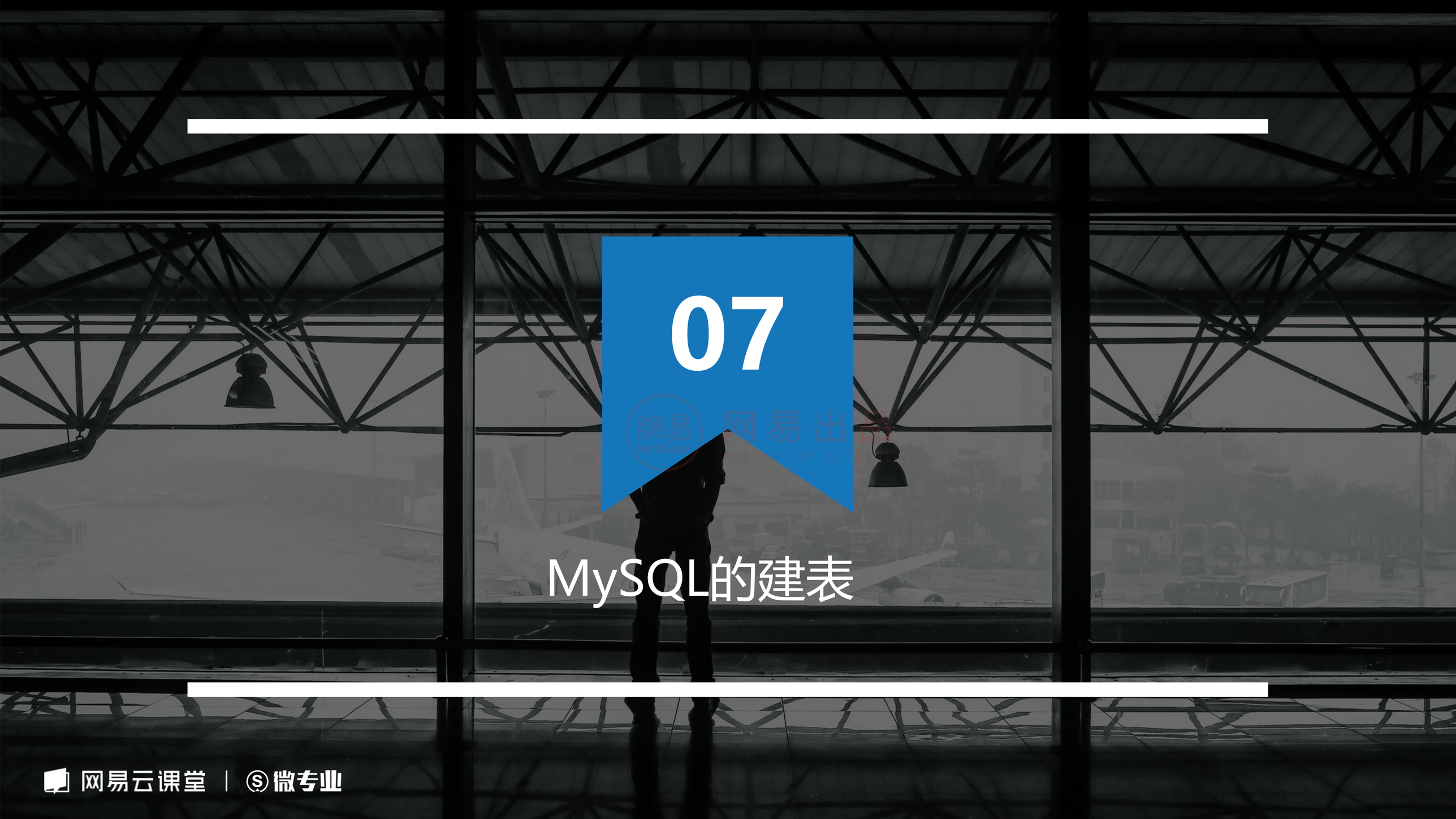
了解MySQL的建库语句

- [DEFAULT] CHARACTER SET：指定数据库的默认字符集。
- [DEFAULT] COLLATE：指定字符集的默认校对规则。
- MySQL 的字符集（CHARACTER）和校对规则（COLLATION）两个不同的概念：字符集是用来定义 MySQL 存储字符串的方式，校对规则定义了比较字符串的方式，解决排序和字符分组的问题。

MySQL的建库

实战案例 MySQL的建库

- 案例1：创建一个数据库
- 案例2：创建一个指定字符集和校对规则的数据库
- 案例3：使用模糊匹配查找数据库
- 案例4：修改现有数据库的字符集
- 案例5： 删除数据库



07

MySQL的建表

MySQL的建表

了解MySQL的建表语句

- 利用CREATE TABLE创建表，必须给出下列信息：
- 新表的名字，在关键字CREATE TABLE之后给出；
- 表列的名字和定义，用逗号分隔。

MySQL的建表

了解MySQL的建表语句

```
CREATE TABLE customers
(
  cust_id      int       NOT NULL AUTO_INCREMENT,
  cust_name    char(50)  NOT NULL ,
  cust_address char(50)  NULL ,
  cust_city    char(50)  NULL ,
  cust_state   char(5)   NULL ,
  cust_zip     char(10)  NULL ,
  cust_country char(50)  NULL ,
  cust_contact char(50)  NULL ,
  cust_email   char(255) NULL ,
  PRIMARY KEY (cust_id)
) ENGINE=InnoDB;
```

MySQL的建表

了解MySQL的主键

- 主键值必须唯一。即，表中的每个行必须具有唯一的主键值。如果主键使用单个列，则它的值必须唯一。如果使用多个列，则这些列的组合值必须唯一。

- 迄今为止我们看到的CREATE TABLE例子都是用单个列作为主键。

MySQL的建表

了解MySQL的主键

- **创建由多个列组成的主键，应该以逗号分隔的列表给出各列名**

```
CREATE TABLE orderitems
(
  order_num  int           NOT NULL ,
  order_item int           NOT NULL ,
  prod_id    char(10)      NOT NULL ,
  quantity   int           NOT NULL ,
  item_price decimal(8,2) NOT NULL ,
  PRIMARY KEY (order_num, order_item)
) ENGINE=InnoDB;
```

MySQL的建表

了解MySQL的主键

- 主键和NULL值 第1章介绍过，主键为其值唯一标识表中每个行的列。主键中只能使用不允许NULL值的列。允许NULL值的列不能作为唯一标识。

MySQL的建表

了解MySQL的AUTO_INCREMENT

- 再次考察customers和orders表。customers表中的顾客由列cust_id唯一标识，每个顾客有一个唯一编号。
- 使用的最简单的编号是下一个编号，所谓下一个编号是大于当前最大编号的编号。例如，如果cust_id的最大编号为10005，则插入表中的下一个顾客可以具有等于10006的cust_id。
- AUTO_INCREMENT告诉MySQL，本列每当增加一行时自动增量。每次执行一个INSERT操作时，MySQL自动对该列增量

MySQL的建表

了解MySQL的默认值

- 如果在插入行时没有给出值，MySQL允许指定此时使用的默认值。默认值用CREATE TABLE语句的列定义中的DEFAULT关键字指定。

MySQL的建表

了解MySQL的存储引擎

- 与其他DBMS一样，MySQL有一个具体管理和处理数据的内部引擎。在你使用CREATE TABLE语句时，该引擎具体创建表，而在你使用SELECT语句或进行其他数据库处理时，该引擎在内部处理你的请求。多数时候，此引擎都隐藏在DBMS内，不需要过多关注它。

MySQL的建表

了解MySQL的存储引擎

- 以下是几个需要知道的引擎:
- InnoDB是一个可靠的事务处理引擎（参见第26章），它不支持全文
- 本搜索;
- MEMORY在功能等同于MyISAM，但由于数据存储在内存（不是磁盘）
- 中，速度很快（特别适合于临时表）；
- MyISAM是一个性能极高的引擎，它支持全文本搜索但不支持事务处理
- 。

MySQL的建表

了解MySQL的建表语句

- 为更新表定义，可使用ALTER TABLE语句。但是，理想状态下，当表中存储数据以后，该表就不应该再被更新。在表的设计过程中需要花费大量时间来考虑，以便后期不对该表进行大的改动。
- 为了使用ALTER TABLE更改表结构，必须给出下面的信息：
- 在ALTER TABLE之后给出要更改的表名（该表必须存在，否则将出错）；
- 所做更改的列表。

MySQL的建表

了解MySQL的建表语句

- ALTER TABLE的一种常见用途是定义外键。下面是用来定义表所用的外键的代码：

```
ALTER TABLE orderitems
ADD CONSTRAINT fk_orderitems_orders
FOREIGN KEY (order_num) REFERENCES orders (order_num);

ALTER TABLE orderitems
ADD CONSTRAINT fk_orderitems_products FOREIGN KEY (prod_id)
REFERENCES products (prod_id);

ALTER TABLE orders
ADD CONSTRAINT fk_orders_customers FOREIGN KEY (cust_id)
REFERENCES customers (cust_id);

ALTER TABLE products
ADD CONSTRAINT fk_products_vendors
FOREIGN KEY (vend_id) REFERENCES vendors (vend_id);
```

MySQL的建表

了解MySQL的建表语句

- 复杂的表结构更改一般需要手动删除过程，它涉及以下步骤：
- 用新的列布局创建一个新表；
- 使用INSERT SELECT语句从旧表复制数据到新表。如果有必要，可使用转换函数和计算字段；
- 检验包含所需数据的新表；
- 重命名旧表（如果确定，可以删除它）；
- 用旧表原来的名字重命名新表；

MySQL的建表

了解MySQL的建表语句

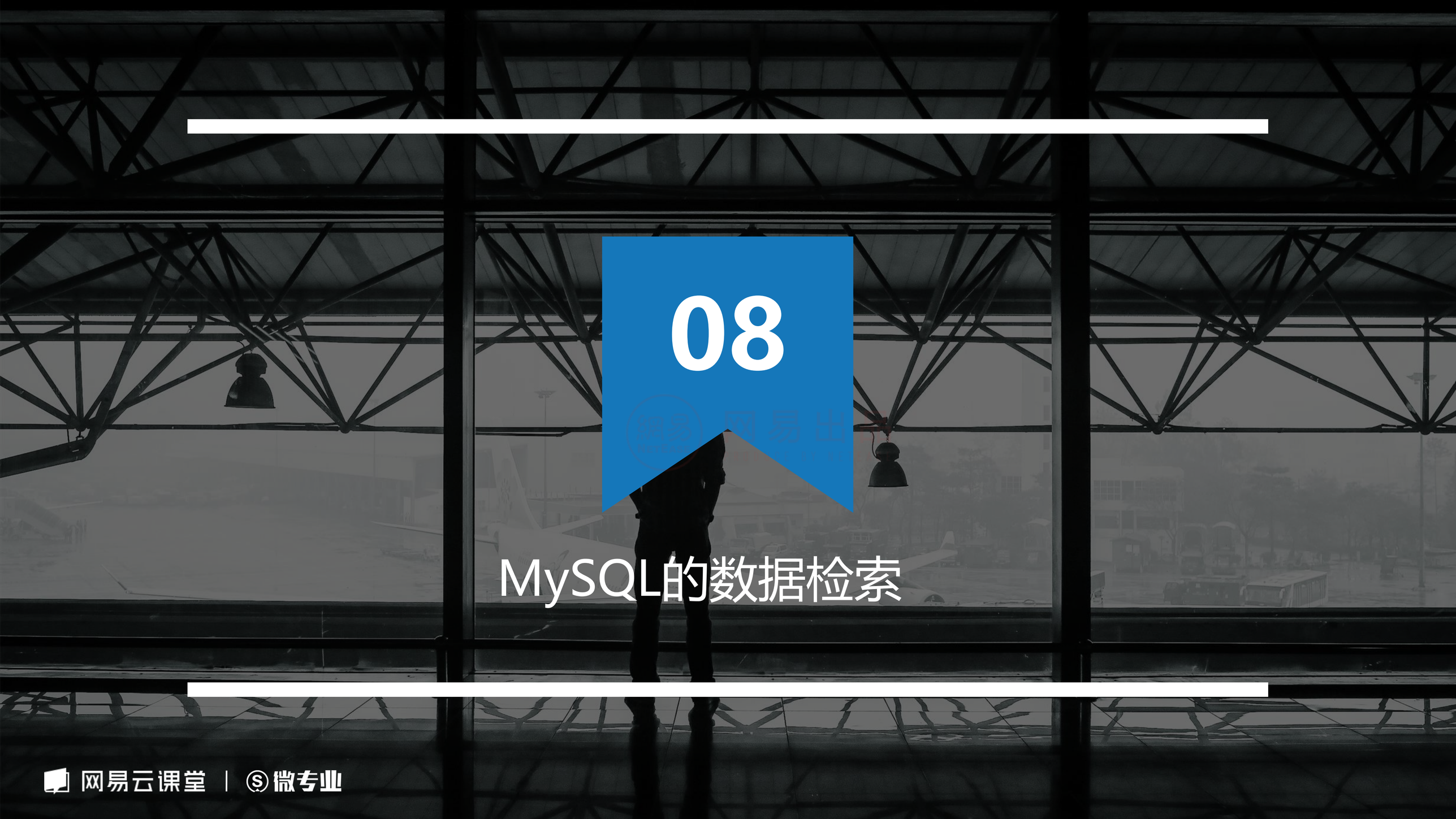
- 删除表（删除整个表而不是其内容）非常简单，使用DROP TABLE语句即可

- 重命名表使用RENAME TABLE语句可以重命名一个表句即可

MySQL的建表

实战案例 MySQL的建库

- 案例1：创建一个表
- 案例2：创建一个PK
- 案例3：创建一个复合PK
- 案例4：创建带有默认值的表
- 案例5： GUI状态下创建表



MySQL的数据检索

了解MySQL的Select语句

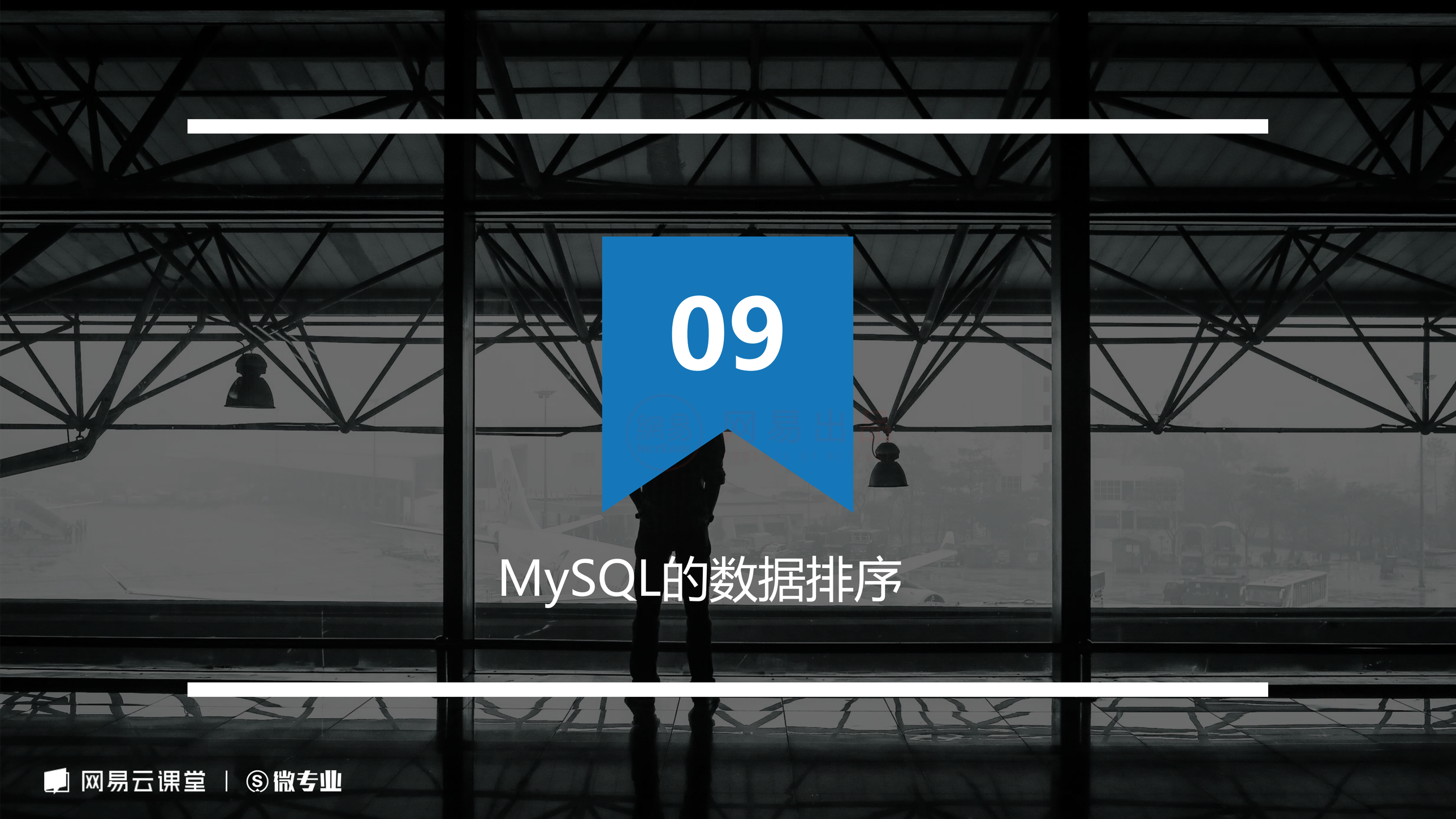
- SQL语句是由简单的英语单词构成的。这些单词称为关键字，每个SQL语句都是由一个或多个关键字构成的。大概，最经常使用的SQL语句就是SELECT语句了。它的用途是从一个或多个表中检索信息。

- 为了使用SELECT检索表数据，必须至少给出两条信息——想选择什么，以及从什么地方选择。

MySQL的数据检索

实战案例 MySQL的数据检索

- 案例1：Select 语句语法
- 案例2：Select 检索所有列
- 案例3：Select 检索单个列
- 案例4：Select 检索不同行
- 案例5：Select 的结果限定



MySQL的数据排序

了解MySQL的数据排序

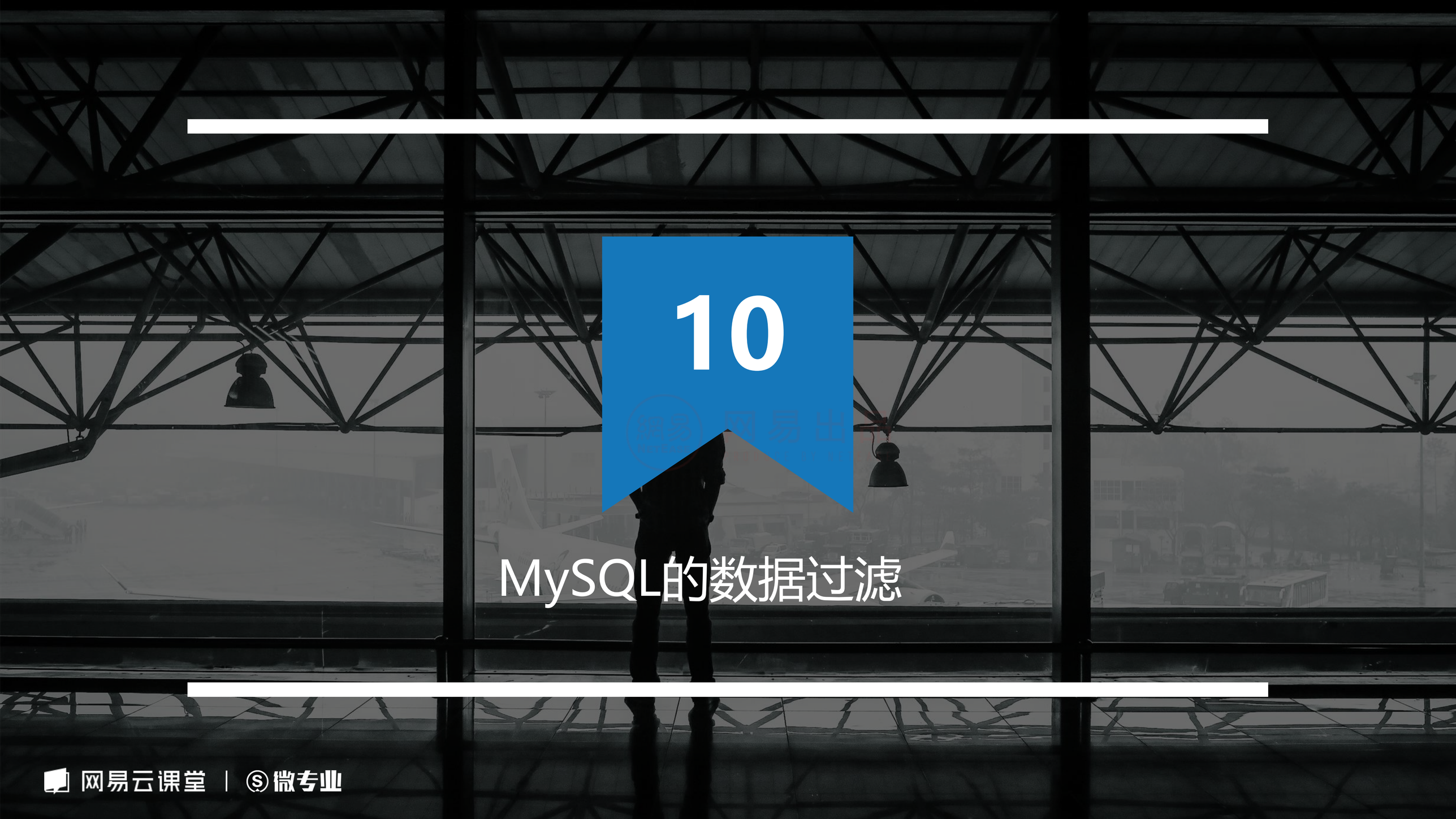
- 本节讲授如何组合WHERE子句以建立功能更强的更高级的搜索条件。
- 我们还将学习如何使用NOT和IN操作符。

- 所有WHERE子句在过滤数据时使用的都是单一的条件。为了进行更强的过滤控制，MySQL允许给出多个WHERE子句。这些子句可以两种方式使用：以AND子句的方式或OR子句的方式使用。

MySQL的数据排序

实战案例 MySQL的数据检索排序

- 案例1：排序数据
- 案例2：按照多个列进行排序
- 案例3：制定排序的方向


10
MySQL的数据过滤

MySQL的数据过滤

了解MySQL的数据过滤

- 数据库表一般包含大量的数据，很少需要检索表中所有行。通常只会根据特定操作或报告的需要提取表数据的子集。只检索所需数据需要指定搜索条件（search criteria），
- 搜索条件也称为过滤条件（filter condition）。 在SELECT语句中，数据根据WHERE子句中指定的搜索条件进行过滤。WHERE子句在表名（FROM子句）之后给出

MySQL的实战

实战案例

- 案例1：使用Where子句
- 案例2：检查单个值
- 案例3：不匹配检查
- 案例4：范围值检查
- 案例5：空值检查

11

MySQL的数据过滤（二）

MySQL的数据排序

了解MySQL的数据过滤

- 本节讲授如何组合WHERE子句以建立功能更强的更高级的搜索条件。
- 我们还将学习如何使用NOT和IN操作符。

- 所有WHERE子句在过滤数据时使用的都是单一的条件。为了进行更强的过滤控制，MySQL允许给出多个WHERE子句。这些子句可以两种方式使用：以AND子句的方式或OR子句的方式使用。

MySQL的实战

实战案例

- 案例1：And操作
- 案例2：Or操作
- 案例3：计算次序
- 案例4：In操作
- 案例5：Not操作

12

MySQL的通配符数据过滤

MySQL的数据过滤

了解MySQL的数据过滤

- 前面介绍的所有操作符都是针对已知值进行过滤的。不管是匹配一个还是多个值，测试大于还是小于已知值，或者检查某个范围的值，共同点是过滤中使用的值都是已知的。但是，这种过滤方法并不是任何时候都好用。
- 例如，怎样搜索产品名中包含文本anvil的所有产品？用简单的比较操作符肯定不行，必须使用通配符。利用通配符可创建比较特定数据的搜索模式。
- 在这个例子中，如果你想找出名称包含anvil的所有产品，可构造一个通配符搜索模式，找出产品名中任何位置出现anvil的产品。

MySQL的数据过滤

通配符

- 通配符（wildcard） 用来匹配值的一部分的特殊字符。

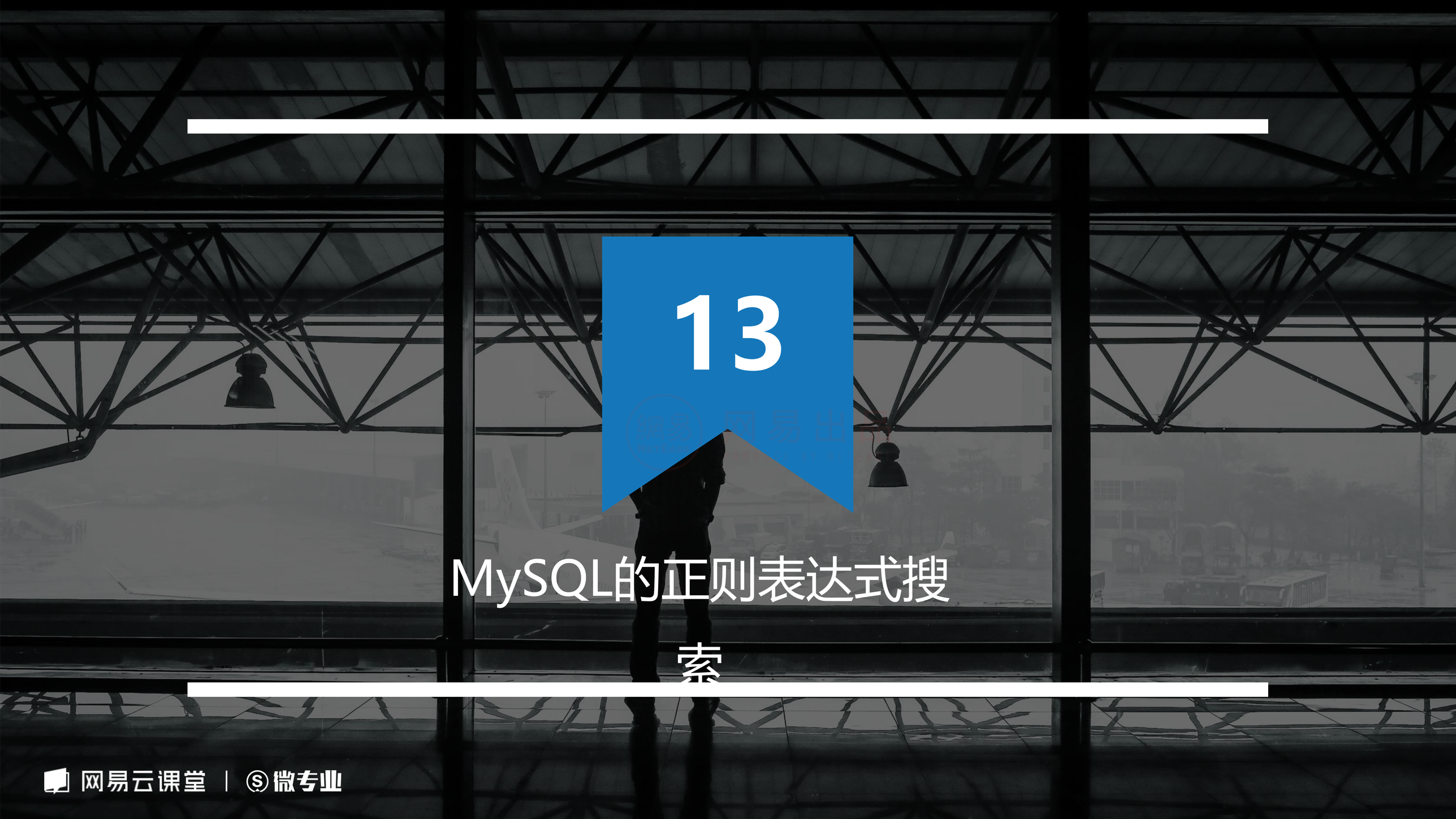
- 搜索模式（search pattern）① 由字面值、通配符或两者组合构成的搜索条件。

- 为在搜索子句中使用通配符，必须使用LIKE操作符。LIKE指示MySQL，后跟的搜索模式利用通配符匹配而不是直接相等匹配进行比较。

MySQL的实战

实战案例

- 案例1：Like操作
- 案例2：百分号%操作
- 案例3：下划线_操作
- 案例4：使用通配符的技巧



13

MySQL的正则表达式搜索

MySQL的正则表达式

了解MySQL的数据过滤

- 前两章中的过滤例子允许用匹配、比较和通配操作符寻找数据。对于基本的过滤（或者甚至是某些不那么基本的过滤），这样就足够了。但随着过滤条件的复杂性的增加，WHERE子句本身的复杂性也有必要增加。

- 所有种类的程序设计语言、文本编辑器、操作系统等都支持正则表达式。有见识的程序员和网络管理员已经关注作为他们技术工具重要内容的正则表达式很长时间了。

MySQL的正则表达式

了解MySQL的数据过滤

- 这也就是正则表达式变得有用的地方。正则表达式是用来匹配文本的特殊的串（字符集合）。如果你想从一个文本文件中提取电话号码，可以使用正则表达式。如果你需要查找名字中间有数字的所有文件，可以使用一个正则表达式。如果你想在一个文本块中找到所有重复的单词，可以使用一个正则表达式。如果你想替换一个页面中的所有URL为这些URL的实际HTML链接，也可以使用一个正则表达式（对于最后这个例子，或者是两个正则表达式）。

MySQL的正则表达式

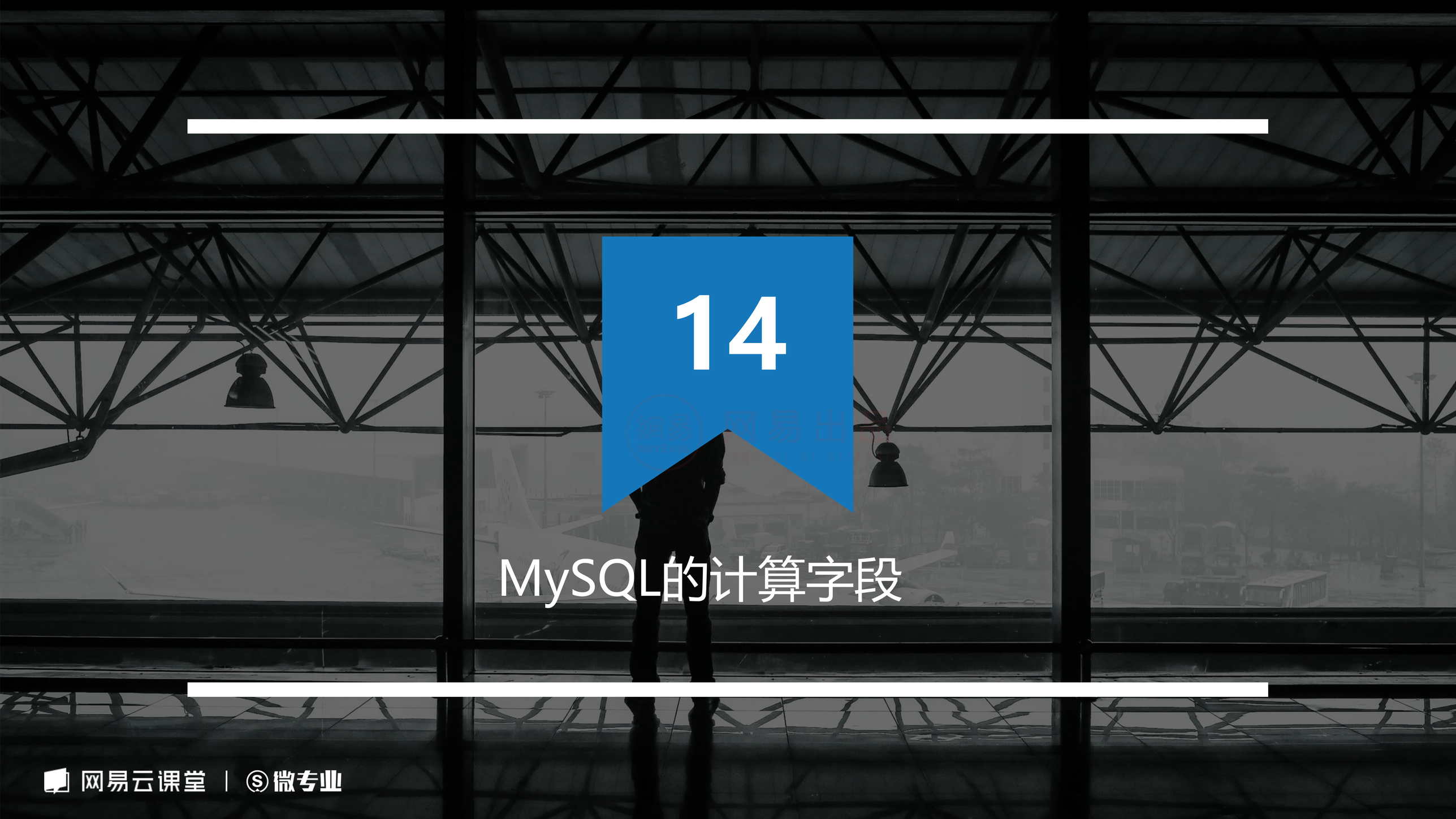
了解MySQL的数据过滤

- 那么，正则表达式与MySQL有何关系？已经说过，正则表达式的作用是匹配文本，将一个模式（正则表达式）与一个文本串进行比较。MySQL用WHERE子句对正则表达式提供了初步的支持，允许你指定正则表达式，过滤SELECT检索出的数据。

MySQL的实战

实战案例

- 案例1：基本字符串匹配
- 案例2：进行Or匹配
- 案例3：匹配几个字符之一
- 案例4：匹配范围
- 案例5：匹配特殊字符
- 案例6：匹配多个实例和定位符


14
MySQL的计算字段
网易云课堂 | 微专业

MySQL的计算字段

什么是计算字段

- 如果想在一个字段中既显示公司名，又显示公司的地址，但这两个信息一般包含在不同的表列中。
- 城市、州和邮政编码存储在不同的列中（应该这样），但邮件标签打印程序却需要把它们作为一个恰当格式的字段检索出来。
- 列数据是大小写混合的，但报表程序需要把所有数据按大写表示出来。

MySQL的计算字段

什么是计算字段

- 物品订单表存储物品的价格和数量，但不需要存储每个物品的总价格（用价格乘以数量即可）。为打印发票，需要物品的总价格。
- 需要根据表数据进行总数、平均数计算或其他计算。
- 在上述每个例子中，存储在表中的数据都不是应用程序所需要的。我们需要直接从数据库中检索出转换、计算或格式化过的数据；而不是检索出数据，然后再在客户机应用程序或报告程序中重新格式化。这就是计算字段发挥作用的所在了。计算字段并不实际存在于数据库表中。

MySQL的实战

实战案例

- 案例1：计算字段
- 案例2：拼接字段
- 案例3：算数计算

15

MySQL的数据处理函数

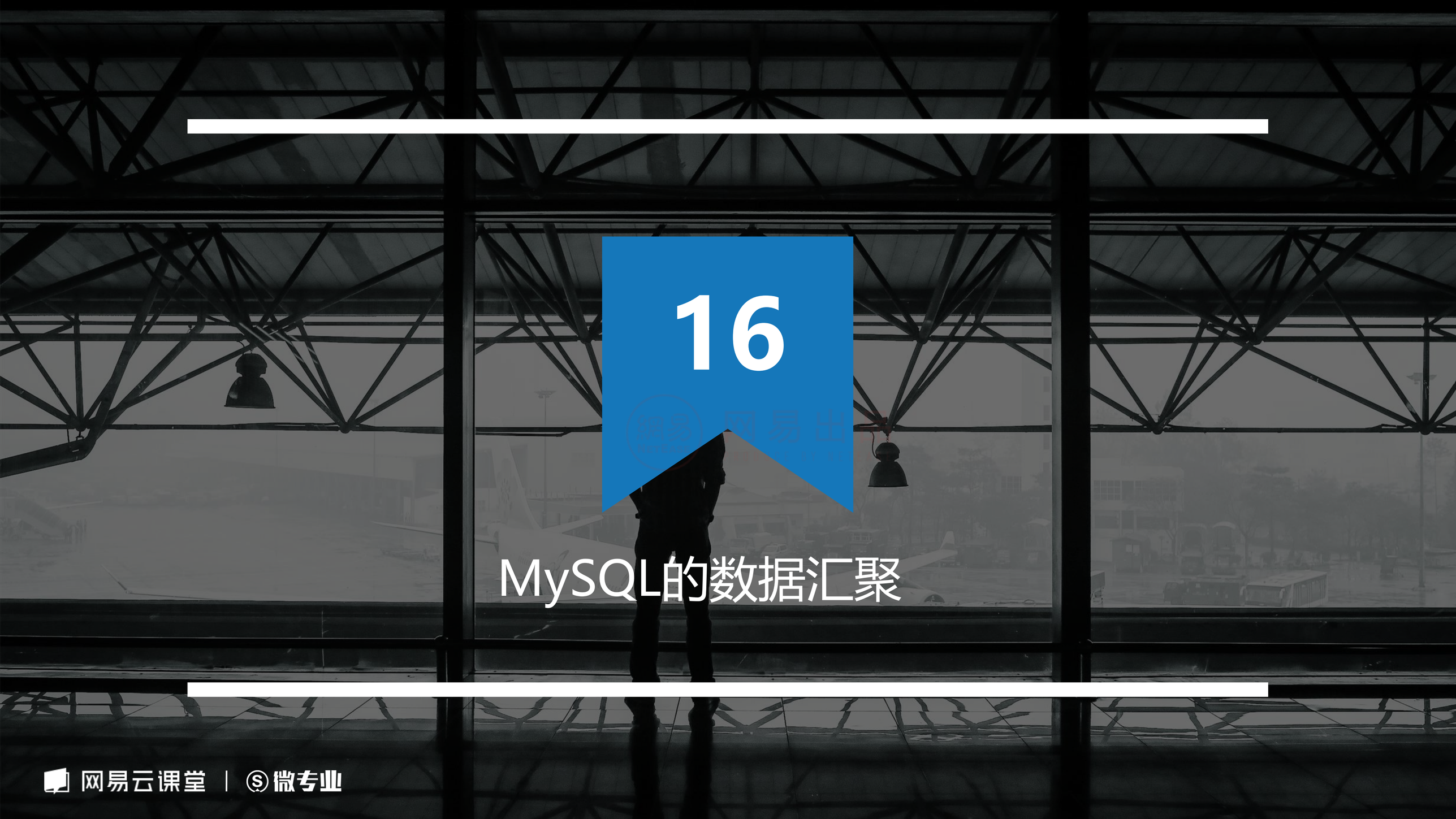
MySQL的数据处理函数

- 用于处理文本串（如删除或填充值，转换值为大写或小写）的文本函数。
- 用于在数值数据上进行算术操作（如返回绝对值，进行代数运算）的数值函数。
- 用于处理日期和时间值并从这些值中提取特定成分（例如，返回两个日期之差，检查日期有效性等）的日期和时间函数。
- 返回DBMS正使用的特殊信息（如返回用户登录信息，检查版本细节）的系统函数

MySQL的实战

实战案例

- 案例1：文本函数的使用
- 案例2：日期和时间函数的处理
- 案例3：数值函数的处理



MySQL的数据汇聚

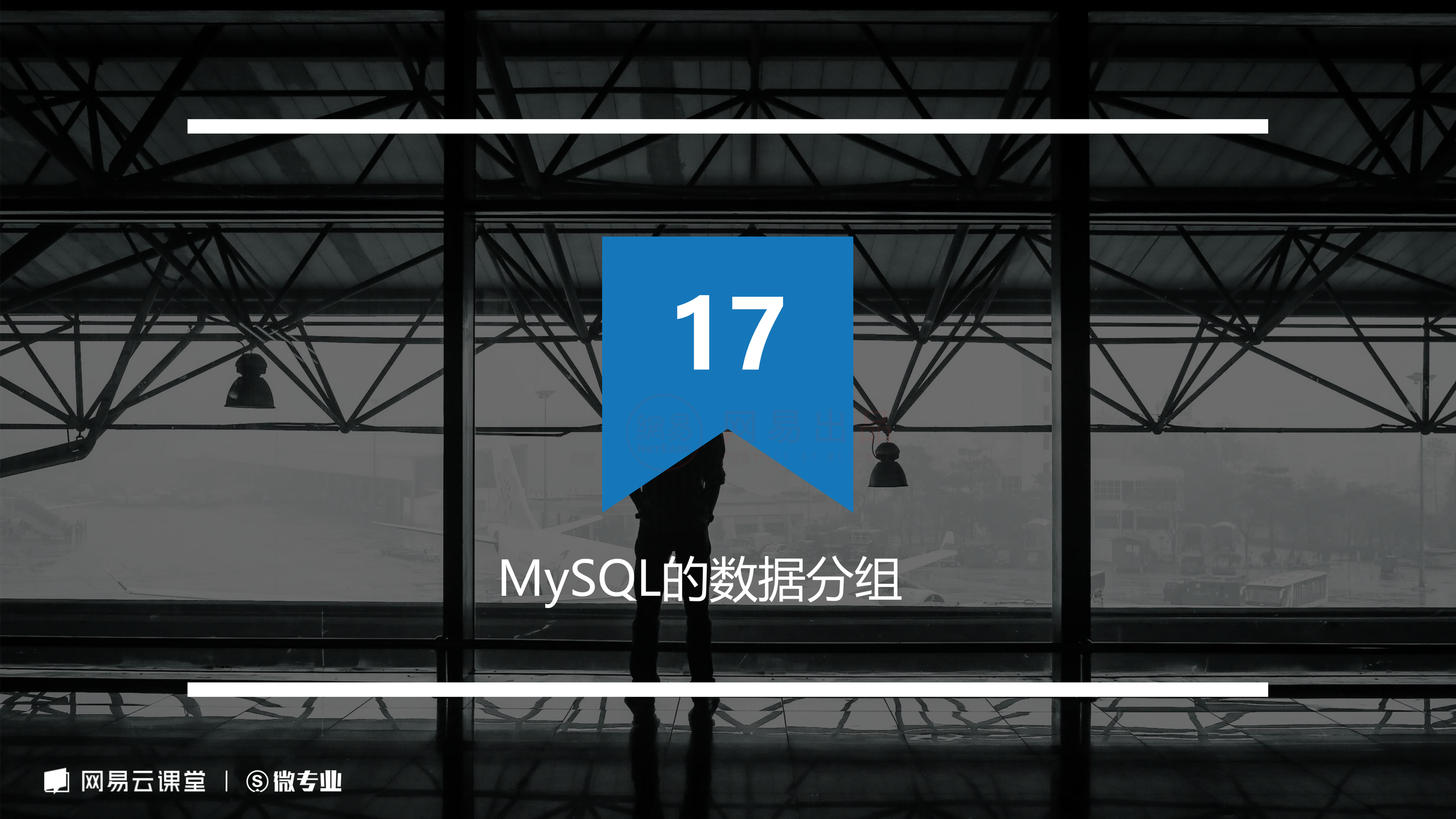
- 我们经常需要汇总数据而不用把它们实际检索出来，为此MySQL提供了专门的函数。使用这些函数，MySQL查询可用于检索数据，以便分析和报表生成。

- 确定表中行数（或者满足某个条件或包含某个特定值的行数）。
- 获得表中行组的和。
- 找出表列（或所有行或某些特定的行）的最大值、最小值和平均值。

MySQL的实战

实战案例

- 案例1：AVG聚合函数
- 案例2：count聚合函数
- 案例3：Max函数
- 案例4：Min函数
- 案例5：Sum函数
- 案例6：组合聚合函数



MySQL的数据分组

- SQL聚集函数可用来汇总数据。这使我们能够对行进行计数，计算和与平均数，获得最大和最小值而不用检索所有数据。

- 如果要返回每个供应商提供的产品数目怎么办？或者返回只提供单项产品的供应商所提供的产品，或返回提供10个以上产品的供应商怎么办？

- 这就是分组显身手的时候了。分组允许把数据分为多个逻辑组，以便能对每个组进行聚集计算。

MySQL的数据分组

- 使用了GROUP BY，就不必指定要计算和估值的每个组了。系统会自动完成。

- GROUP BY子句可以包含任意数目的列。这使得能对分组进行嵌套，为数据分组提供更细致的控制。
- 如果在GROUP BY子句中嵌套了分组，数据将在最后规定的分组上进行汇总。换句话说，在建立分组时，指定的所有列都一起计算（所以不能从个别的列取回数据）。

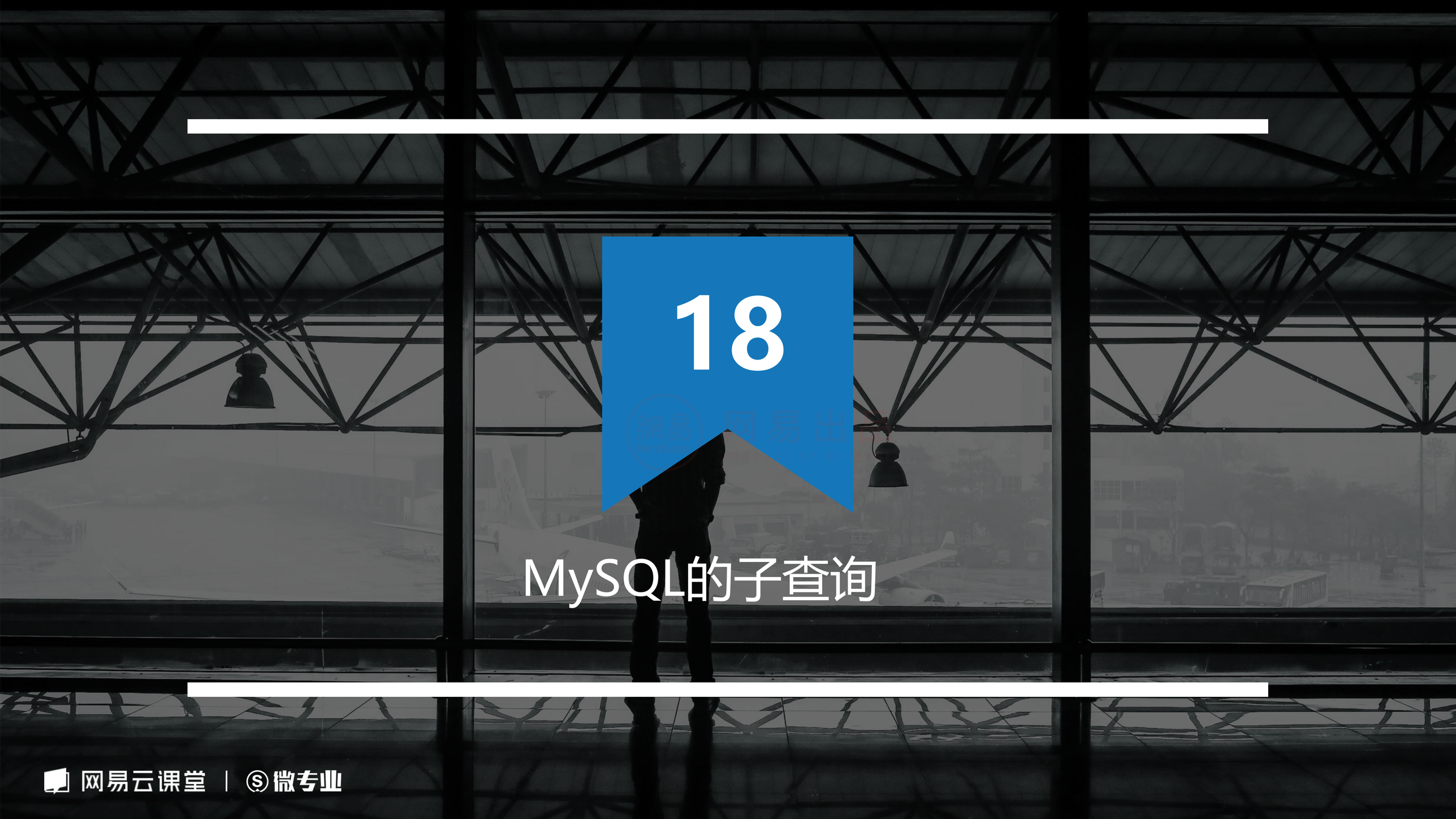
MySQL的数据分组

- GROUP BY子句中列出的每个列都必须是检索列或有效的表达式（但不能是聚集函数）。如果在SELECT中使用表达式，则必须在GROUP BY子句中指定相同的表达式。不能使用别名。
- 除聚集计算语句外，SELECT语句中的每个列都必须在GROUP BY子句中给出。
- 如果分组列中具有NULL值，则NULL将作为一个分组返回。如果列中有多行NULL值，它们将分为一组。
- GROUP BY子句必须出现在WHERE子句之后，ORDER BY子句之前。

MySQL的实战

实战案例

- 案例1：创建数据分组
- 案例2：过滤数据分组
- 案例3：分组和排序
- 案例4：Select子句的顺序



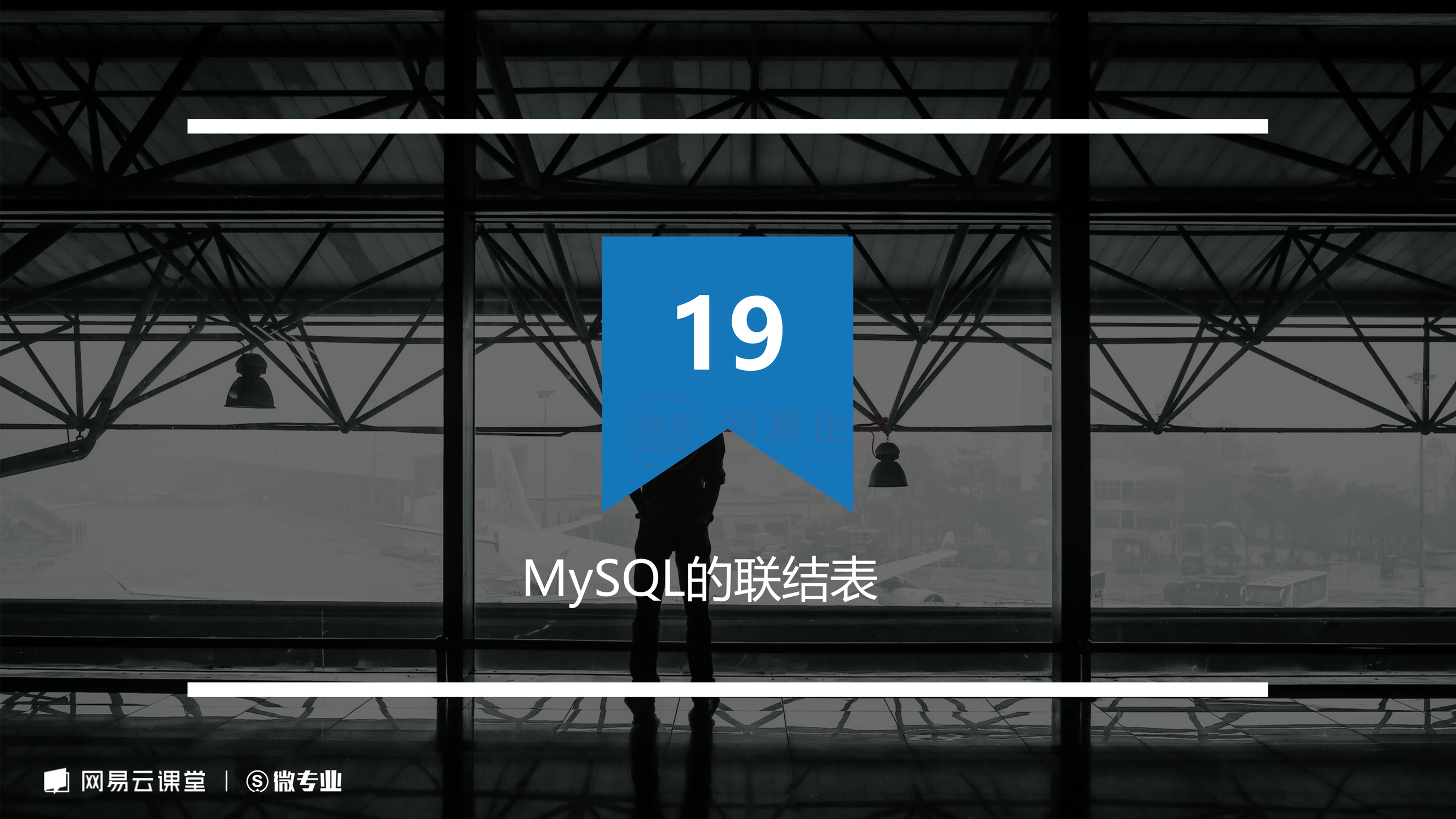
MySQL的子查询

- SELECT语句是SQL的查询。迄今为止我们所看到的所有SELECT语句都是简单查询，即从单个数据库表中检索数据的单条语句。
- SQL还允许创建子查询（subquery），即嵌套在其他查询中的查询。
- 作为计算字段使用子查询和利用子查询进行过滤

MySQL的实战

实战案例

- 案例1：使用简单子查询
- 案例2：利用子查询过滤数据
- 案例3：利用子查询作为计算字段使用



19

MySQL的联结表

MySQL的联结表

- SQL最强大的功能之一就是能在数据检索查询的执行中联结（join）表。联结是利用SQL的SELECT能执行的最重要的操作，很好地理解联结及其语法是学习SQL的一个极为重要的组成部分。

MySQL的联结表

- 理解关系表的最好方法是来看一个现实世界中的例子。
- 假如有一个包含产品目录的数据库表，其中每种类别的物品占一行。对于每种物品要存储的信息包括产品描述和价格，以及生产该产品的供
- 应商信息。
- 现在，假如有由同一供应商生产的多种物品，那么在何处存储供应商信息（如，供应商名、地址、联系方法等）呢?

MySQL的联结表

- 因为同一供应商生产的每个产品的供应商信息都是相同的，对每个产品重复此信息既浪费时间又浪费存储空间。
- 如果供应商信息改变（例如，供应商搬家或电话号码变动），只需改动一次即可。
- 如果有重复数据（即每种产品都存储供应商信息），很难保证每次输入该数据的方式都相同。不一致的数据在报表中很难利用。

MySQL的联结表

- 相同数据出现多次决不是一件好事，此因素是关系数据库设计的基础。关系表的设计就是要保证把信息分解成多个表，一类数据一个表。各表通过某些常用的值（即关系设计中的关系（relational））互相关联。

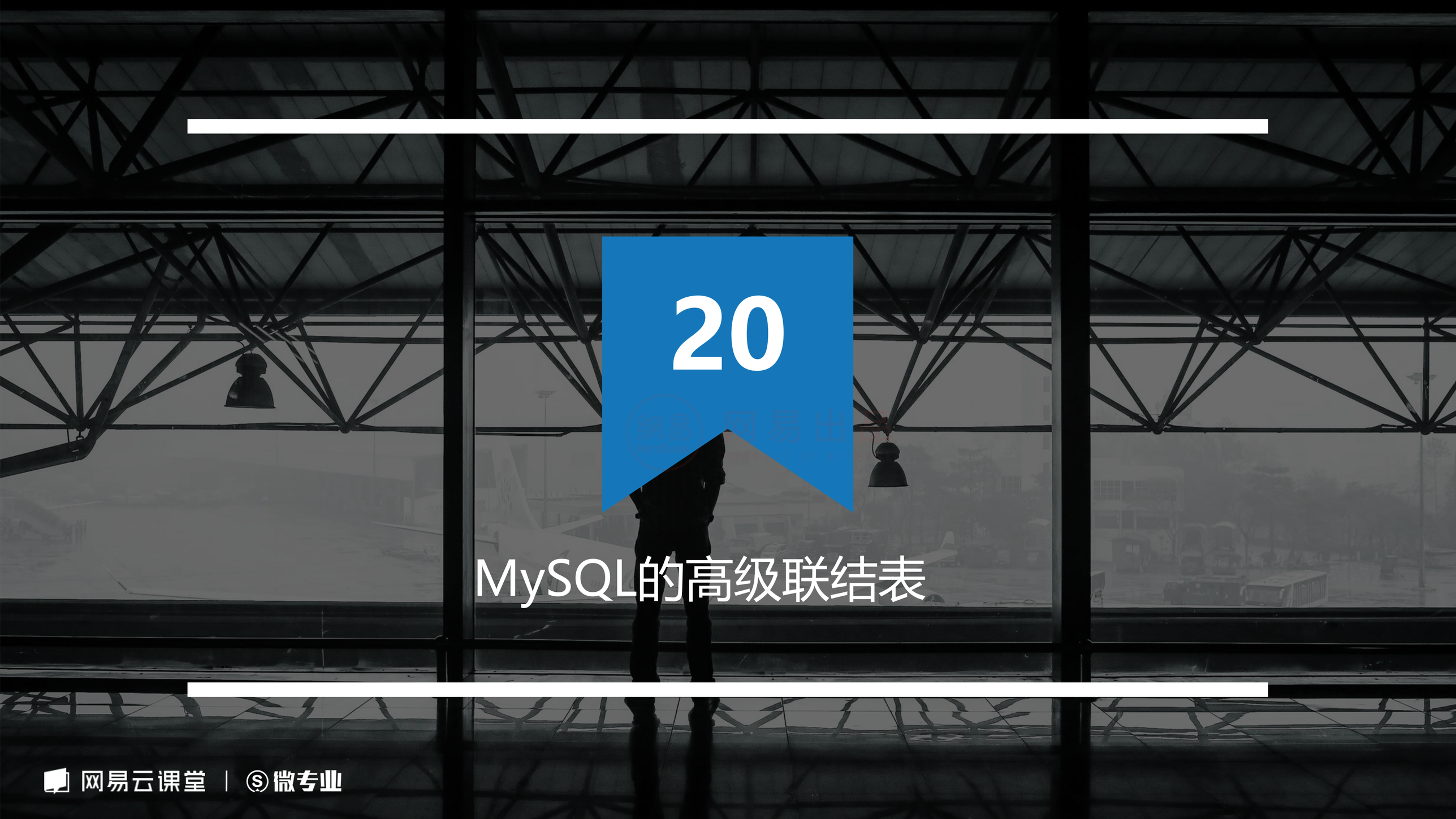
MySQL的联结表

- 外键（foreign key） 外键为某个表中的一列，它包含另一个表的主键值，定义了两个表之间的关系。这样做的好处如下：
- 供应商信息不重复，从而不浪费时间和空间；
- 如果供应商信息变动，可以只更新vendors表中的单个记录，相关表中的数据不用改动；
- 由于数据无重复，显然数据是一致的，这使得处理数据更简单。
- 总之，关系数据可以有效地存储和方便地处理。

MySQL的实战

实战案例

- 案例1：如何创建表直接的关联
- 案例2：如何利用Join进行表直接的连接
- 案例3：内联表
- 案例4：联结多个表
- 案例5：联结表时候Where的使用



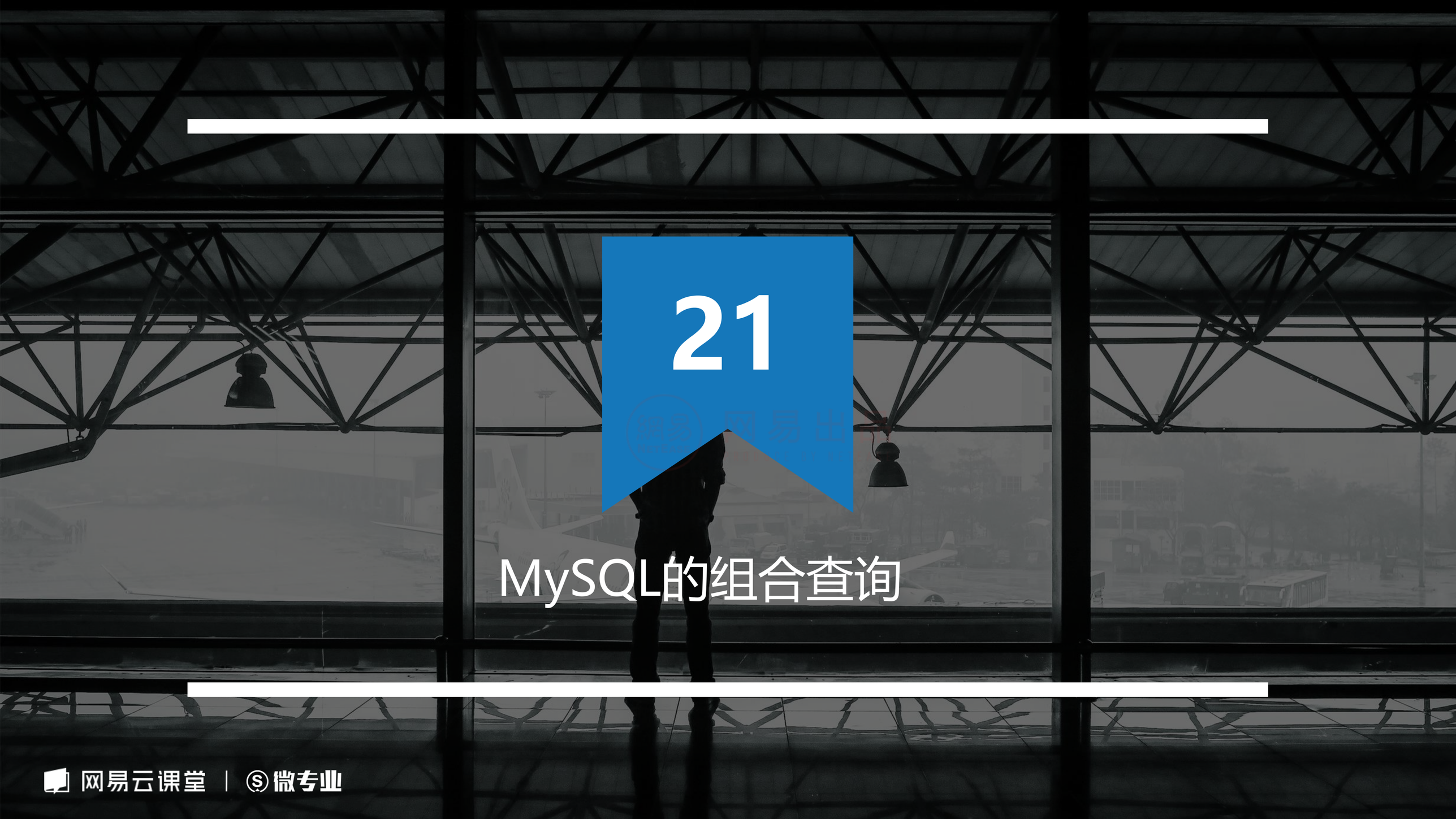
MySQL的联结表

- 迄今为止，我们使用的只是称为内部联结或等值联结（equijoin）的简单联结。
- 现在来看3种其他联结，它们分别是自联结、自然联结和外部联结。

MySQL的实战

实战案例

- 案例1：自联结语句
- 案例2：自然联结方式
- 案例3：外部联结方式



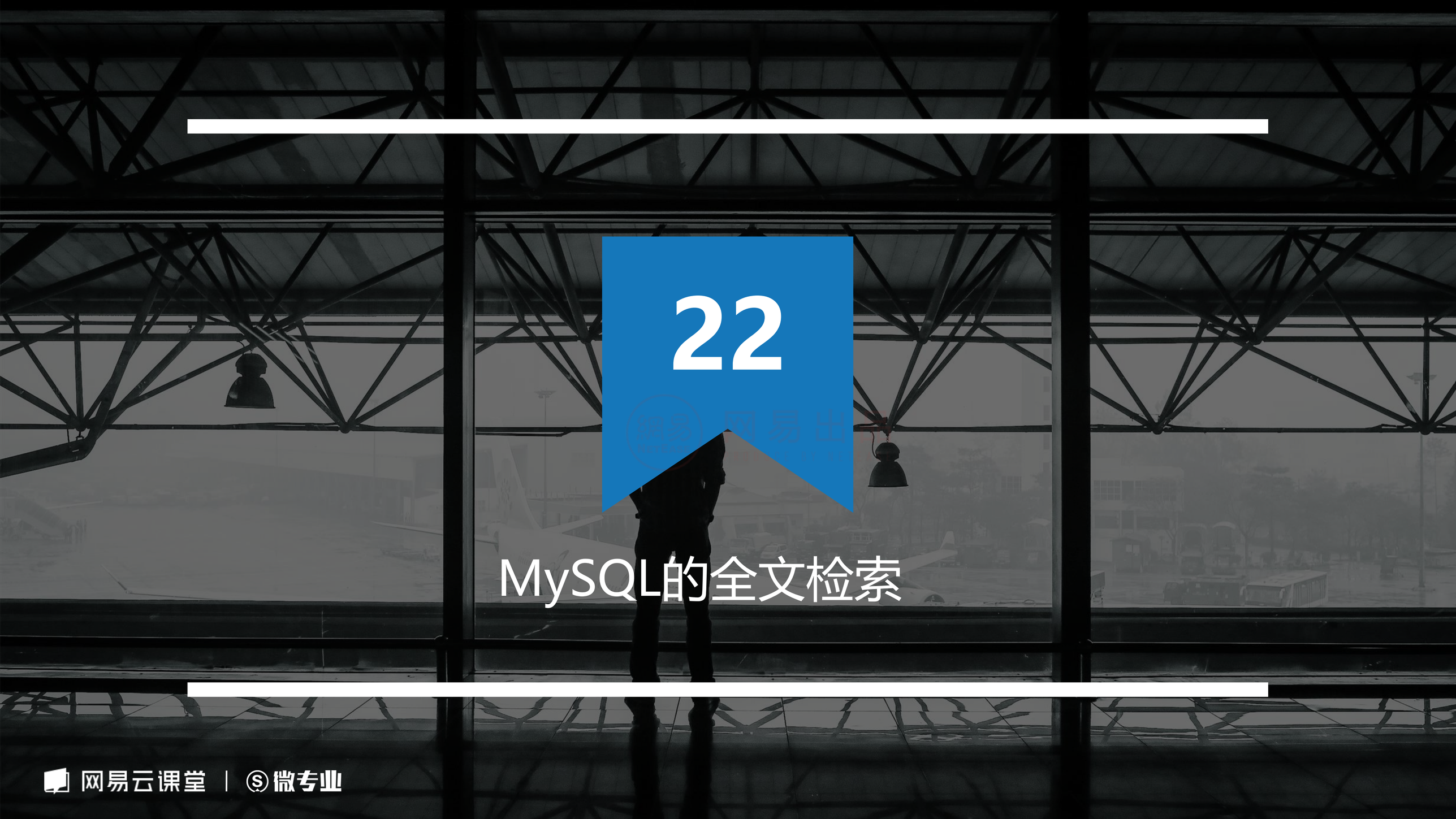
MySQL的组合查询

- 多数SQL查询都只包含从一个或多个表中返回数据的单条SELECT语句。MySQL也允许执行多个查询（多条SELECT语句），并将结果作为单个
- 查询结果集返回。这些组合查询通常称为并（union）或复合查询（compound query）。
- 有两种基本情况，其中需要使用组合查询：
- 在单个查询中从不同的表返回类似结构的数据；
- 对单个表执行多个查询，按单个查询返回数据。

MySQL的实战

实战案例

- 案例1：使用Union查询数据
- 案例2：Union规则
- 案例3：包含或取消重复的行号
- 案例4：对组合结果进行排序


22
MySQL的全文检索
网易云课堂 | 微专业

MySQL的全文检索

- 虽然之前章节学过的搜索机制非常有用，但存在几个重要的限制。
- 性能——通配符和正则表达式匹配通常要求MySQL尝试匹配表中所有行（而且这些搜索极少使用表索引）。因此，由于被搜索行数不断增加，这些搜索可能非常耗时。
- 明确控制——使用通配符和正则表达式匹配，很难（而且并不总是能）明确地控制匹配什么和不匹配什么。例如，指定一个词必须匹配，一个词必须不匹配，而一个词仅在第一个词确实匹配的情况下才可以匹配或者才可以不匹配。

MySQL的全文检索

- 所有这些限制以及更多的限制都可以用全文本搜索来解决。在使用全文本搜索时，MySQL不需要分别查看每个行，不需要分别分析和处理每个词。MySQL创建指定列中各词的一个索引，搜索可以针对这些词进行。这样，MySQL可以快速有效地决定哪些词匹配（哪些行包含它们），哪些词不匹配，它们匹配的频率，等等。

MySQL的实战

实战案例

- 案例1：启用全文索引
- 案例2：进行全文索引
- 案例3：使用查询扩展
- 案例4：布尔全文搜索
- 案例5：全文搜索使用说明

23

MySQL的插入数据

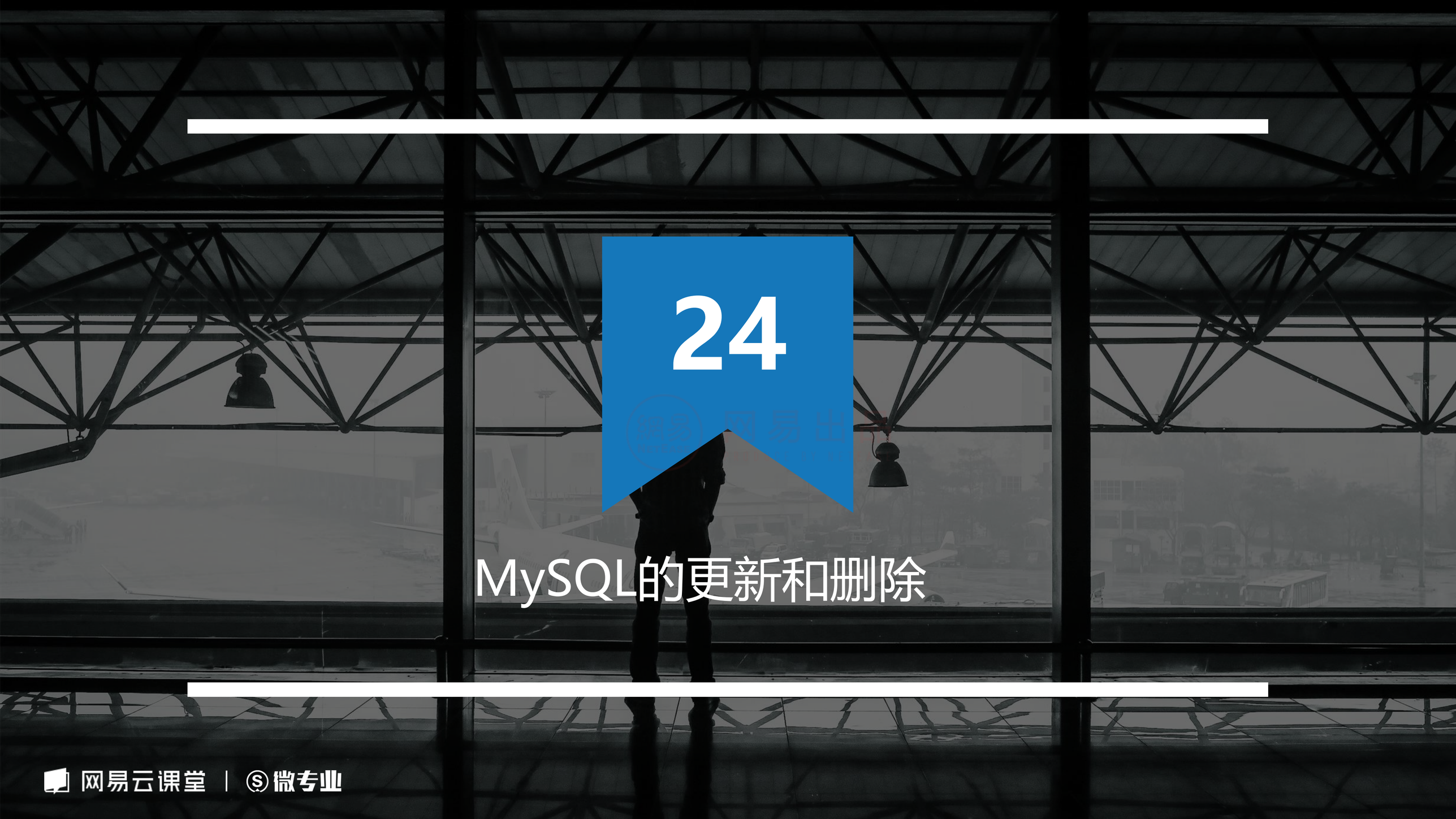
MySQL的数据插入

- 毫无疑问，SELECT是最常使用的SQL语句了（这就是为什么之前讲的都是它的原因）。但是，还有其他3个经常使用的SQL语句需要学习。
- 第一个就是INSERT（下一章介绍另外两个）。顾名思义，INSERT是用来插入（或添加）行到数据库表的。插入可以用几种方式使用：
- 插入完整的行；
- 插入行的一部分；
- 插入多行；
- 插入某些查询的结果。

MySQL的实战

实战案例

- 案例1：Insert语句的使用
- 案例2：插入完整的一行数据
- 案例3：插入多个行
- 案例4：插入检索出的数据



24

MySQL的更新和删除

MySQL的数据更新和删除

- 为了更新（修改）表中的数据，可使用UPDATE语句。可采用两种方
- 式使用UPDATE：

- 更新表中特定行
- 更新表中所有行

MySQL的数据更新和删除

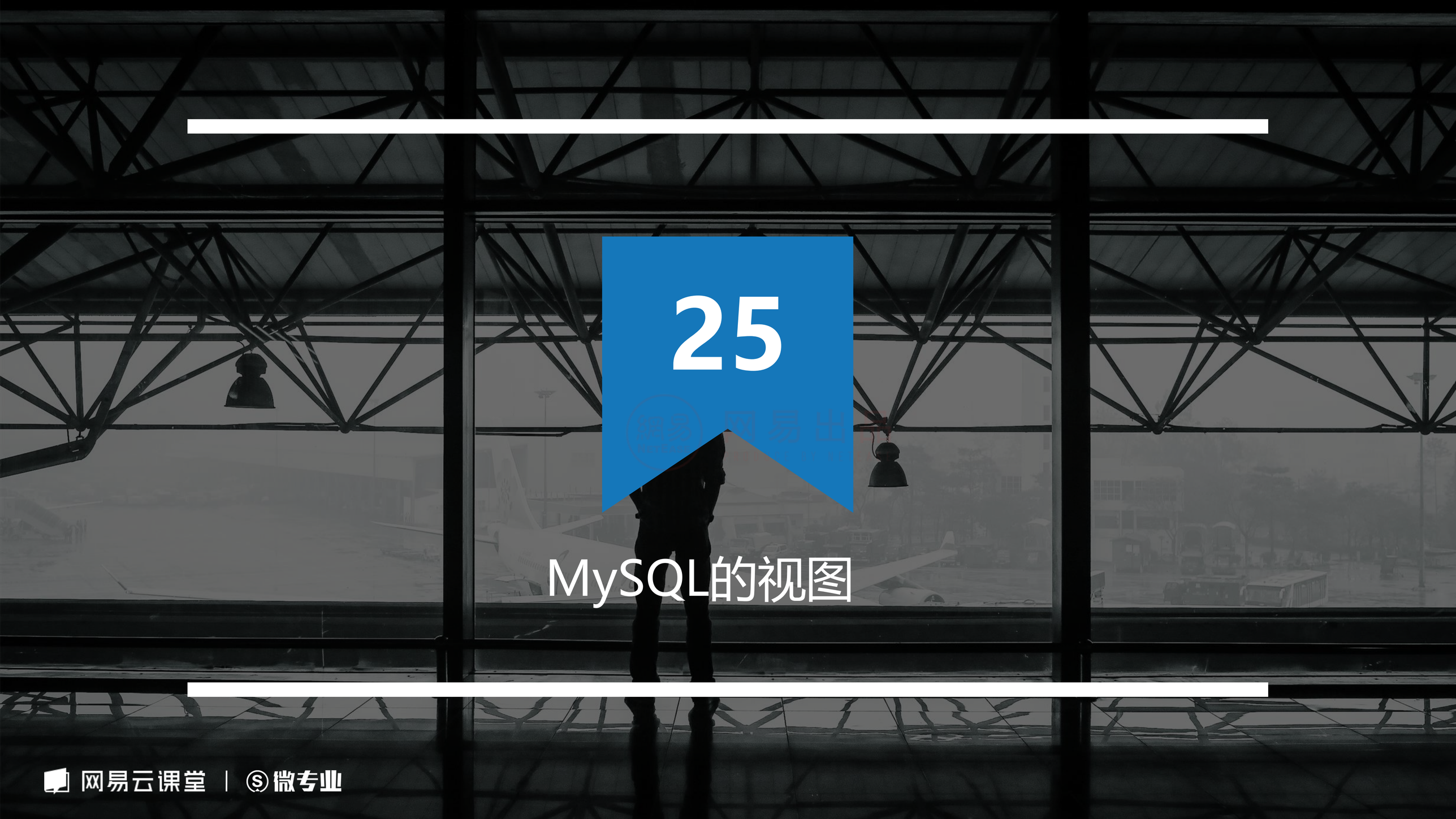
- 为了从一个表中删除（去掉）数据，使用DELETE语句。可以两种方
- 式使用DELETE：

- 从表中删除特定的行；
- 从表中删除所有行。

MySQL的实战

实战案例

- 案例1：Update语句的使用
- 案例2：更新多行数据
- 案例3：Delete语句使用
- 案例4：更新和删除的指导



25

MySQL的视图

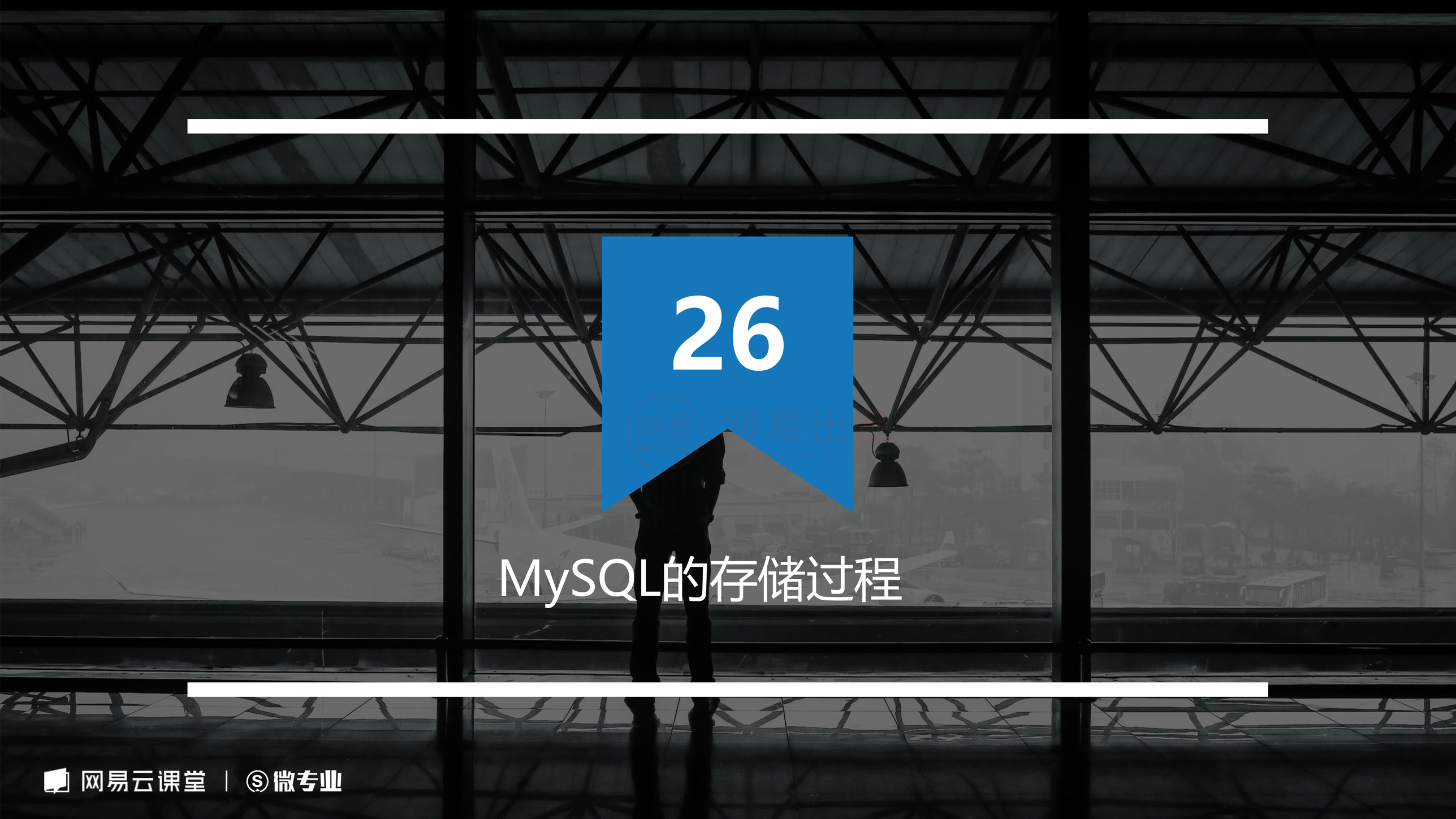
MySQL的视图

- 视图是虚拟的表。与包含数据的表不一样，视图只包含使用时动态检索数据的查询。重用SQL语句。
- 简化复杂的SQL操作。在编写查询后，可以方便地重用它而不必知道它的基本查询细节。
- 使用表的组成部分而不是整个表。
- 保护数据。可以给用户授予表的特定部分的访问权限而不是整个表的访问权限。
- 更改数据格式和表示。视图可返回与底层表的表示和格式不同的数据。

MySQL的实战

实战案例

- 案例1：利用视图简化联结语句
- 案例2：用视图重新格式化数据
- 案例3：用视图过滤不想要的数据
- 案例4：使用视图和计算字段



26

MySQL的存储过程

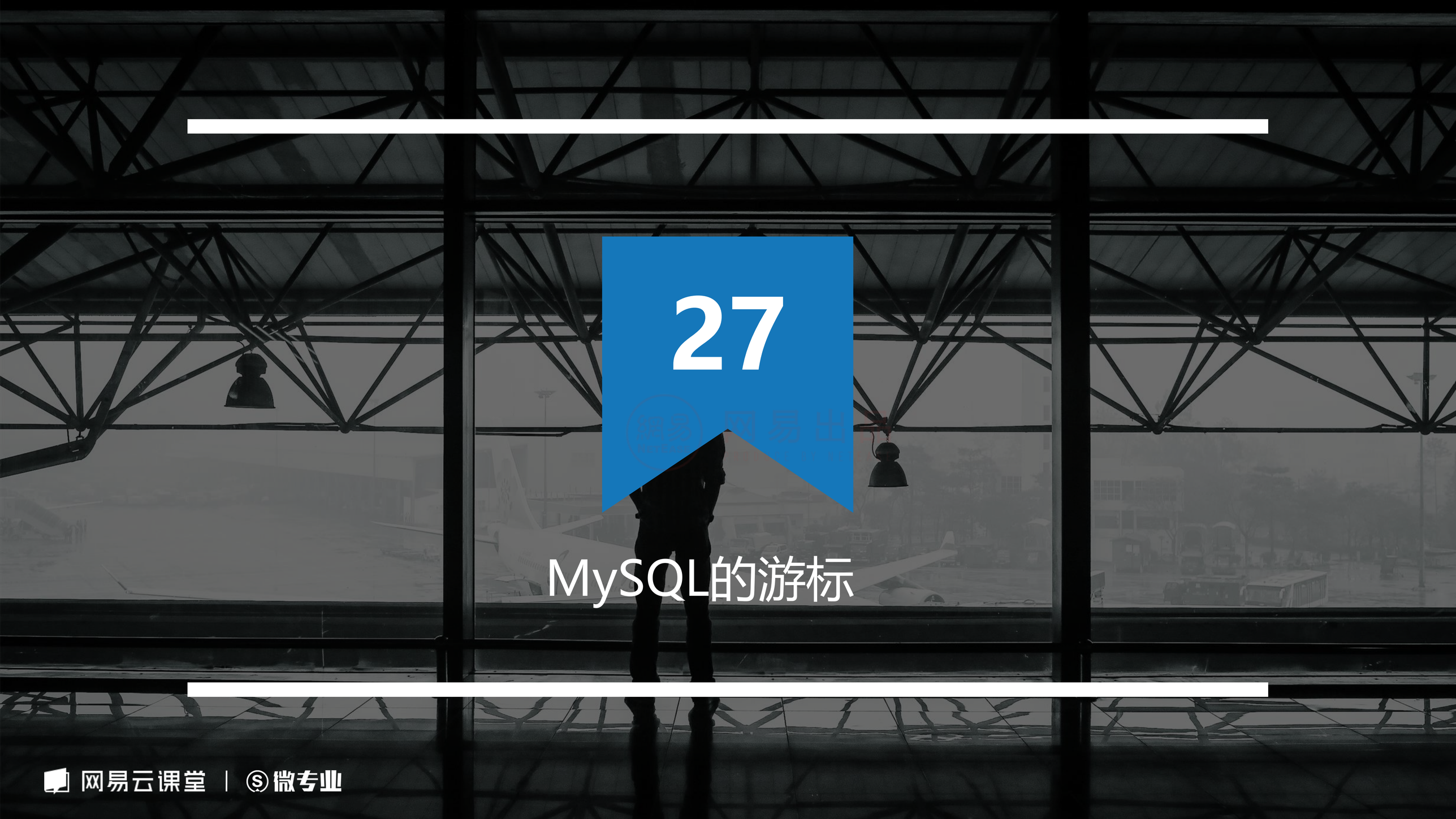
MySQL的存储过程

- 迄今为止，使用的大多数SQL语句都是针对一个或多个表的单条语句。并非所有操作都这么简单，经常会有一个完整的操作需要多条语句才能完成。例如，考虑以下的情形。
- 为了处理订单，需要核对以保证库存中有相应的物品。
- 如果库存有物品，这些物品需要预定以便不将它们再卖给别的人，并且要减少可用的物品数量以反映正确的库存量。
- 库存中没有的物品需要订购，这需要与供应商进行某种交互。
- 关于哪些物品入库和哪些物品退订，需要通知相应的客户。

MySQL的实战

实战案例

- 案例1：创建存储过程
- 案例2：执行存储过程
- 案例3：删除存储过程
- 案例4：检查存储过程



MySQL的游标

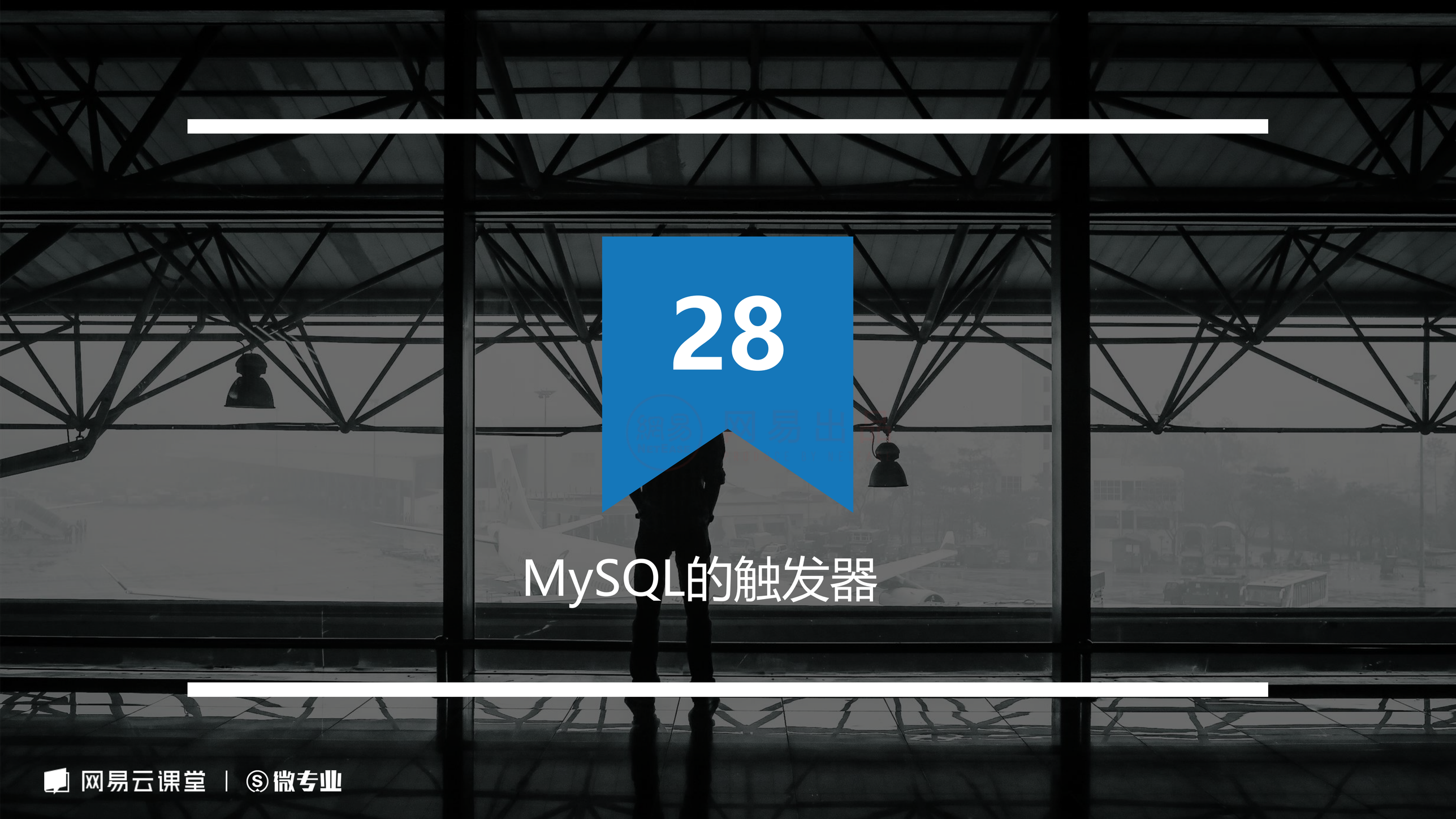
- 有时，需要在检索出来的行中前进或后退一行或多行。这就是使用游标的原因。游标（cursor）是一个存储在MySQL服务器上的数据库查询，它不是一条SELECT语句，而是被该语句检索出来的结果集。在存储了游标之后，应用程序可以根据需要滚动或浏览其中的数据。

- 游标主要用于交互式应用，其中用户需要滚动屏幕上的数据，并对数据进行浏览或做出更改。

MySQL的实战

实战案例

- 案例1：创建游标
- 案例2：打开关闭游标
- 案例3：使用游标数据



28

MySQL的触发器

MySQL的触发器

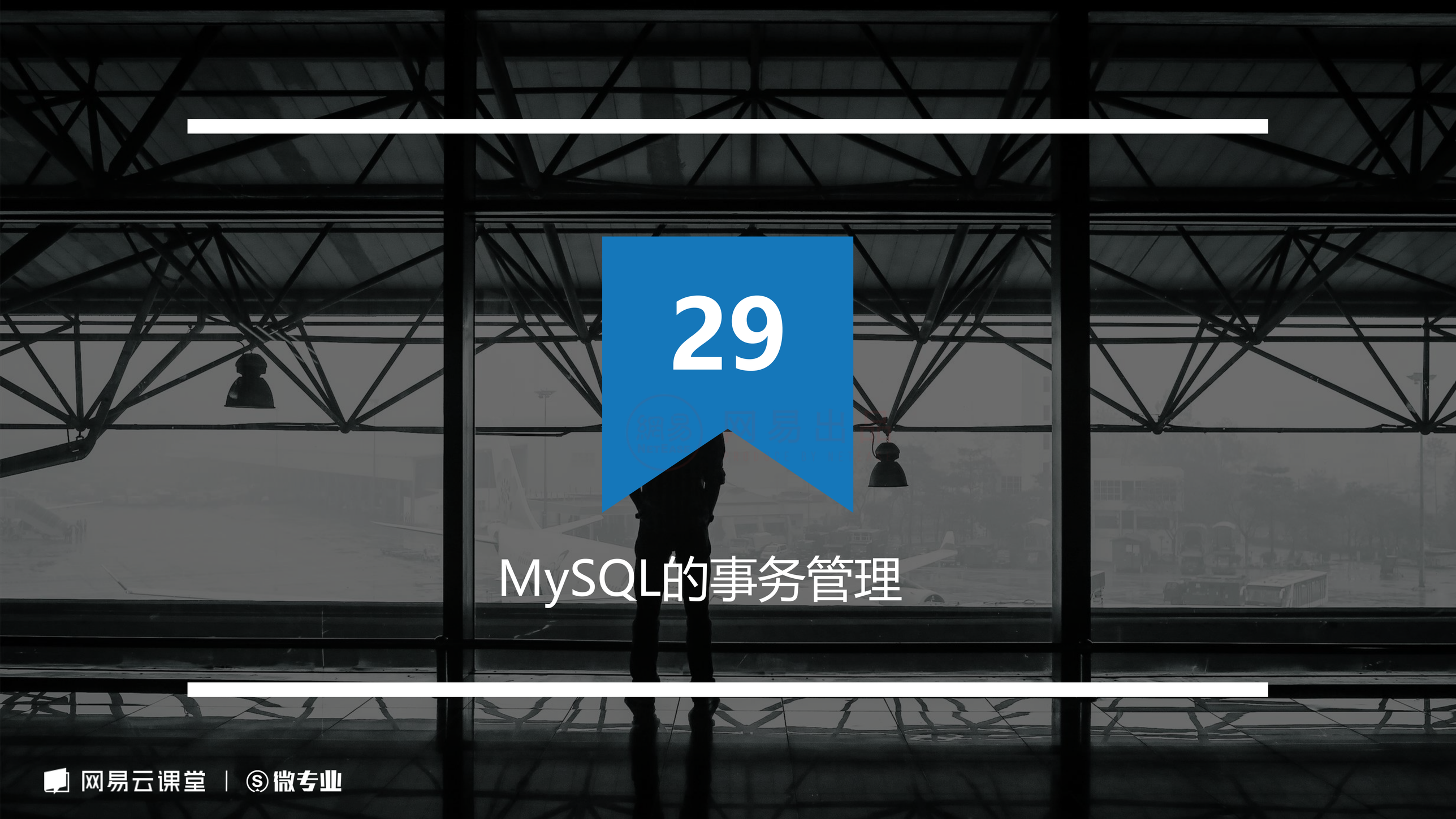
- MySQL语句在需要时被执行，存储过程也是如此。但是，如果你想要某条语句（或某些语句）在事件发生时自动执行，怎么办呢？例如：
- 每当增加一个顾客到某个数据库表时，都检查其电话号码格式是
- 否正确，州的缩写是否为大写；
- 每当订购一个产品时，都从库存数量中减去订购的数量；
- 无论何时删除一行，都在某个存档表中保留一个副本。

- 触发器是MySQL响应以下任意语句而自动执行的一条MySQL语句

MySQL的实战

实战案例

- 案例1：创建触发器
- 案例2：使用触发器
- 案例3：删除触发器



29

MySQL的事务管理

MySQL的事务

- 在使用事务和事务处理时，有几个关键词汇反复出现。下面是关于事务处理需要知道的几个术语：
- 事务（transaction）指一组SQL语句；
- 回退（rollback）指撤销指定SQL语句的过程；
- 提交（commit）指将未存储的SQL语句结果写入数据库表；
- 保留点（savepoint）指事务处理中设置的临时占位符（place_x0002_holder），你可以对它发布回退（与回退整个事务处理不同）。

MySQL的事务

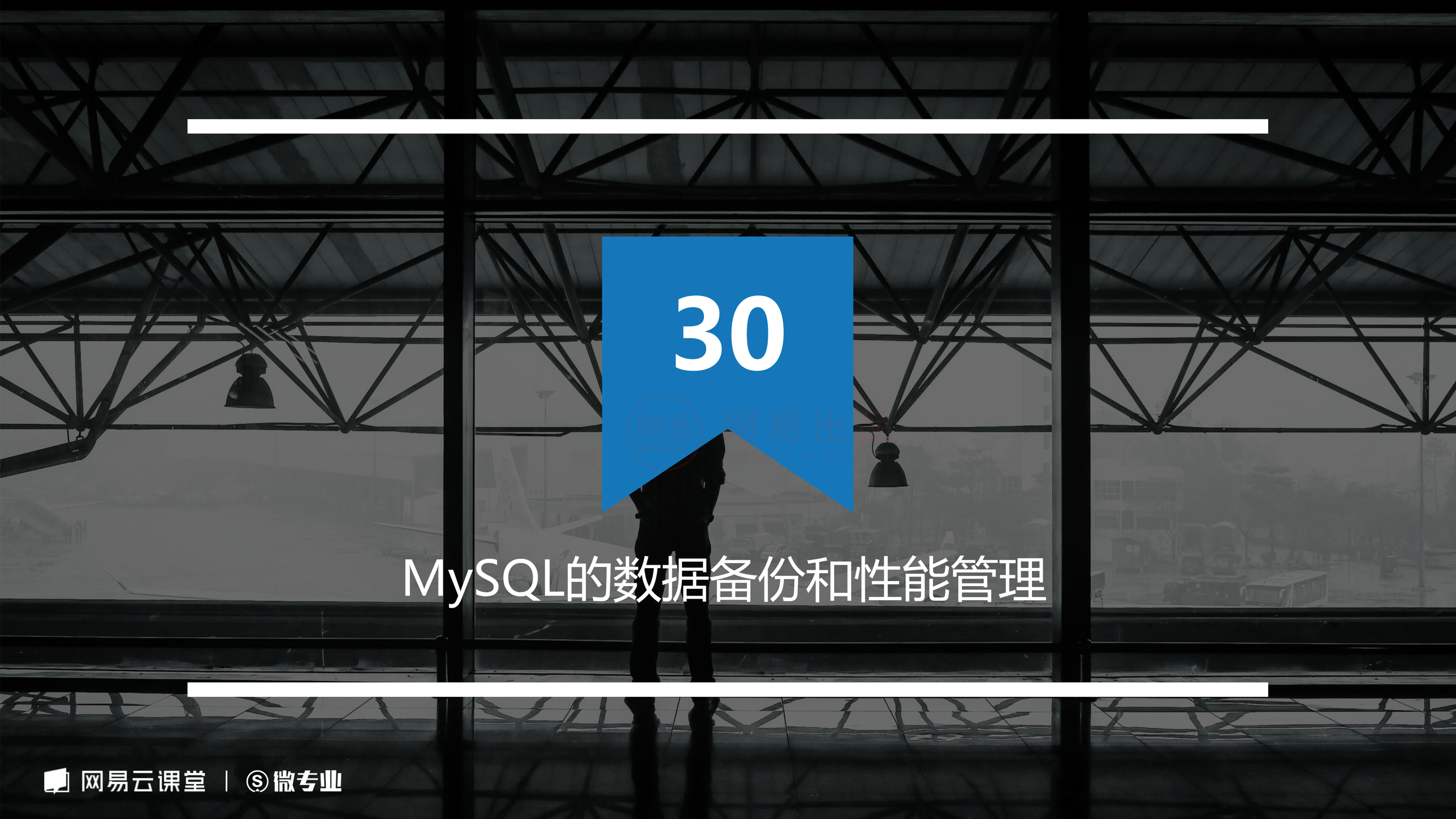
- 事务处理是一种机制，用来管理必须成批执行的MySQL操作，以保证数据库不包含不完整的操作结果。利用事务处理，可以保证一组操作不会中途停止，它们或者作为整体执行，或者完全不执行（除非明确指示）。

- 如果没有错误发生，整组语句提交给（写到）数据库表。如果发生错误，则进行回退（撤销）以恢复数据库到某个已知且安全的状态。

MySQL的实战

实战案例

- 案例1：事务提交
- 案例2：事务回滚
- 案例3：保留点



MySQL的数据备份

- 像所有数据一样，MySQL的数据也必须经常备份。由于MySQL数据库是基于磁盘的文件，普通的备份系统和例程就能备份MySQL的数据。但是，由于这些文件总是处于打开和使用状态，普通的文件副本备份不一定总是有效。
- 使用命令行实用程序mysqldump转储所有数据库内容到某个外部文件。在进行常规备份前这个实用程序应该正常运行，以便能正确地备份转储文件。
- 可用命令行实用程序mysqlhotcopy从一个数据库复制所有数据（并非所有数据库引擎都支持这个实用程序）。
- 可以使用MySQL的BACKUP TABLE或SELECT INTO OUTFILE转储所有数据到某个外部文件。

MySQL的性能改善

- 数据库工作人员把他们工作中的相当一部份时间花在了性能调整上、试验改善DBMS性能。
- 在诊断应用的滞缓现象和性能问题时，性能不良的数据库（以及数据库查询）通常是最常见的祸因。
- 利用索引Index可以大幅改善查询时候的性能问题。
- 利用一些操作系统的参数也可以提供数据访问的性能。
- 使用Explain语句让MySQL解释它将如何执行一条SELECT语句。
- 一般来说，存储过程执行得比一条一条地执行其中的各条MySQL语句快

MySQL的实战

实战案例

- 案例1：Mysql的数据备份
- 案例2：Mysql的性能改善

谢谢观看