5 数据库管理与保护

目录

5.1 数据库事务

回顾DBMS的数据库管理功能

事务机制是基础!

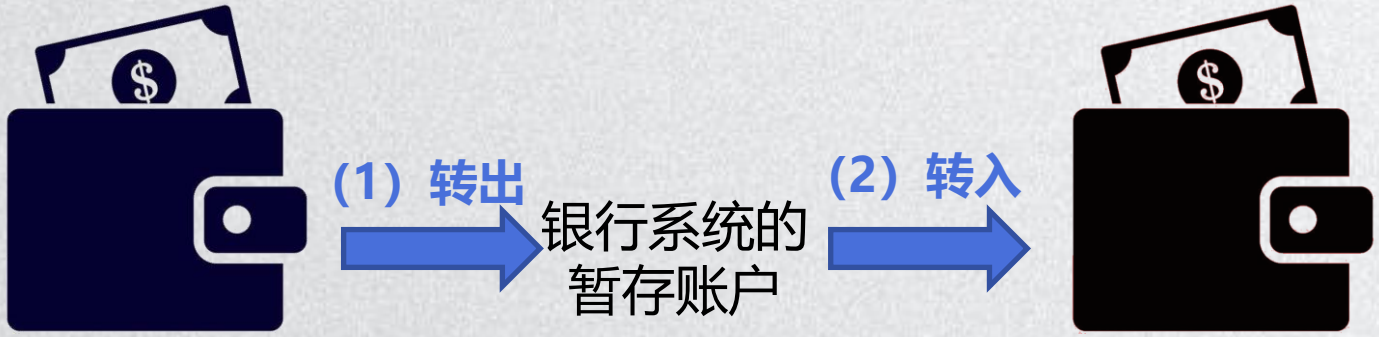
引例1 资金转账

妈妈给上大学的孩子汇生活费的场景。

（1）钱从妈妈的银行账户**转出**。

（2）将钱**转入**孩子的银行账户。

这些操作**涉及对后台数据库的修改。**



钱从妈妈的账户转出后而未转入孩子账户的情况是不可接受的。

对数据库的某些操作应该是一个整体，不能分割！

引例2 教务信息的维护

例5.1 将学时为32的课程学分均设置为2

```
UPDATE Course SET Credits=2 WHERE Hours=32;
```

包含多个操作：修改每门32课时的课程学分是1个操作，有多少门就有多少个操作。

例5.2 学生选课

```
INSERT INTO courseenroll(StudentCode,CourseCode)
VALUES(1001, 'C008');

UPDATE course
SET StudentNum=StudentNum+1
WHERE CourseCode='C008';
```

包含两个操作：
（1）插入一个选课记录
（2）课程表中选课人数+1

对数据库的某些操作应该是一个整体，不能分割！

什么是事务?

事务（Transaction）是一个包含了一组数据库操作命令的序列。这个序列作为一个整体，一起向系统提交或撤销，命令要么都执行，要么都不执行。

- 事务是一个不可分割的逻辑工作单元，是数据库运行的最小逻辑工作单位。
- 在关系数据库中，一个事务可以是一条或一组SQL语句。

事务的特性(ACID)

- ✓ **原子性(Atomic)** 组成事务的多个数据库操作是一个不可分割的单元。
- ✓ **一致性(Consistency)** 事务完成时，必须使相关的数据仍然保持一致状态。
- ✓ **隔离性(Isolation)** 多个事务并发执行时，彼此互不干扰。
- ✓ **持久性(Durability)** 事务完成后对数据库的所有修改永久性有效。

发生下列情况时，事务ACID特性可能遭到破坏：

- 多事务并行运行时，不同事务的操作交叉执行；
- 事务在运行过程中被强制停止。

KingbaseES的事务管理机制

KingbaseES通过**并发控制技术**和**故障恢复技术**，保证事务符合**ACID**特性。

锁定机制

Isolation

加锁使事务相互隔离。

多版本并发控制

Consistency

维护数据的多个版本,实现并发事务的隔离和并发执行。

预写式日志系统

Atomic
Durability

详细记录事务操作、实现回滚和重做机制，以确保日志先于数据持久化到磁盘。

KingbaseES事务类型

所有SQL语句都在事务中执行。根据事务提交方式的不同，KingbaseES数据库事务可分为**隐式事务**和**显式事务**。

不在事务块中

隐式事务是单语句事务，由数据库系统自动开启，执行成功则自动提交，执行失败则自动回滚。

例5.1中的UPDATE语句，系统自动通过事务保证该语句正确完成对所有相关课程的修改，如果中间出错，则会撤销已经完成的修改。

显式事务由用户明确指定事务的开始和结束语句。开始于第一个SQL语句或BEGIN语句，以COMMIT（事务提交）或ROLLBACK（事务回滚）语句结束来显式提交，在两者之间的SQL语句序列形成一个**事务块**，事务块作为一个整体执行。

例5.2的操作可定义为一个显示事务，通过DBMS强制两条SQL语句作为整体执行。

使用SQL语句定义事务

显式事务的定义语法：

```
[BEGIN];                    -- 开始事务
事务块
ROLLBACK;                   -- 回滚并结束事务
COMMIT;                     -- 提交并结束事务
```

事务块中若有任何一条语句失败，则通过ROLLBACK回滚事务来撤销已经执行的SQL操作，并且结束事务，使数据库恢复到事务开始之前的状态；只有当所有SQL语句都成功执行时，才通过COMMIT提交事务，确认所有数据库更新并结束当前事务。

自定义事务举例：学生选课操作

【例】编写存储过程，利用事务机制完成选课操作。

```
CREATE PROCEDURE proc_courseenroll(stcode int, cscode char(4))
AS
BEGIN
   -- 插入选课信息到 courseenroll 表中，事务自动开启
   INSERT INTO courseenroll (StudentCode, CourseCode) VALUES (stcode, cscode);
   -- 更新 course 表中的选课人数
   UPDATE course SET StudentNum = StudentNum + 1
                     WHERE CourseCode = cscode;
   RAISE INFO '执行过程正常！';
   COMMIT;                    -- 提交事务
EXCEPTION
   WHEN OTHERS THEN
      ROLLBACK;               -- 回滚事务
      -- 输出错误信息，可根据实际情况调整输出方式
      RAISE WARNING '执行出现异常，错误代码: %，错误信息: %', SQLCODE, SQLERRM;
END;
```

测试事务执行情况：学生选课操作

观察下列操作是否成功执行：

① 调用存储过程CALL proc_courseenroll(1012, 'C002')，检查针对两个表的更改操作是否成功执行。

② 重复运行CALL proc_courseenroll(1012, 'C002')，观察执行情况并解释原因。

③ 在course表上建立触发器，限定选课人数StudentNum的最大值（例如6）。运行事务CALL proc_courseenroll(1026, 'C002')，观察执行情况并解释原因。

执行情况：

① 成功

② 失败。向courseenroll表中插入同一个学号记录，记录不能插入，course表中的选课人数修改也不进行。

③ 失败，回滚。对course表中选课人数的修改操作失败，对courseenroll表的插入操作也被回滚，两个表都不发生变化。

小结：事务

事务是一个包含了一组数据库操作命令的序列。事务的操作命令
要么都执行，要么都不执行。

显式事务开始于第一个SQL语句或BEGIN语句，以COMMIT（事务提交）或ROLLBACK （事务回滚）语句结束。

隐式事务是单语句事务，由数据库系统自动开启，执行成功则自动提交，执行失败则自动回滚。

事务机制是基础！

5.2 数据库的并发访问控制

ACID特性：
- ✓ 原子性(Atomic)
- ✓ 一致性(Consistency)
- ✓ 隔离性(Isolation)
- ✓ 持久性(Durability)

为什么要并发访问数据库?

思考：串行执行（排队依次执行）存在哪些不足?

并行执行的优势：

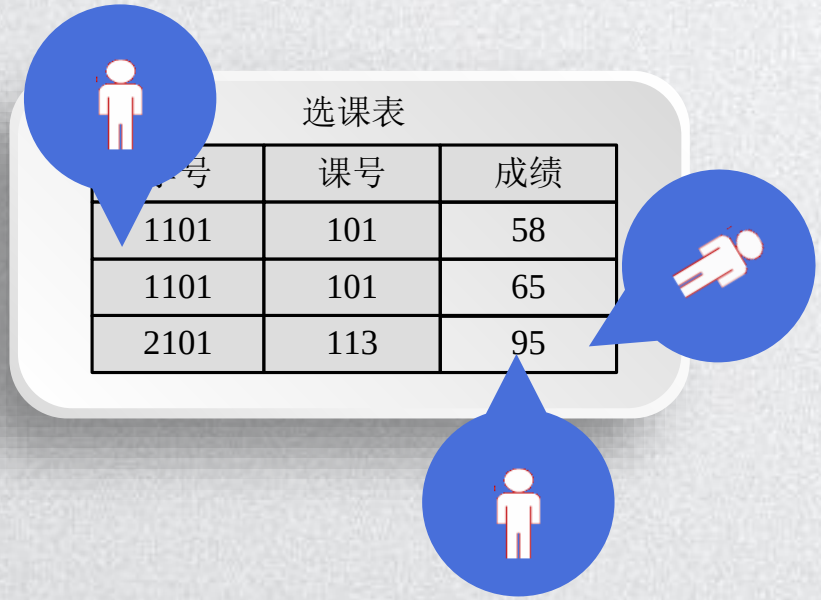
✓ 提高吞吐量和资源利用率

✓ 减少等待时间（减少平均响应时间）

当前计算机领域的一些发展趋势带来大量可能的并发性（多处理器或多核的优势）。

数据库的并发访问

并发访问是指多个事务同时访问数据库，且同时操作同一张表、甚至同一条记录、同一数据项。



并发问题1：丢失修改（Lost Update）

丢失修改

两个或多个事务同时对同一数据进行修改，其中一个事务的修改结果被另一个事务的修改结果覆盖，导致部分修改丢失。

举例：

时刻	事务T1 操作	事务T2操作	A
t1	读取 A=10		10
t2	写入 A=A*2		20
t3		写入 A=100	100
t4	读取 A=100		100

T1的修改丢失！

并发问题2：读脏数据（Dirty Read）

读脏数据

一事务读取了另一事务在回滚前修改的数据。（因与实际数据不符，称为“脏数据”）

举例：银行账户余额修改

时刻	事务T1 操作	事务T2操作	A帐户余额（数据库）
t1	开始事务		100
t2	读取A=100		100
t3	写入A=A*2		200
t4		读A=200	200
t5	ROLLBACK	展示余额 200	100（恢复原值）

T2读脏数据！

并发问题3：不可重读（Non-repeatable Read）

不可重读

一个事务读取某数据后，该数据被另一事务更新，当它再读进行校验时，发现数据不一致。

举例：

时刻	事务T1操作	事务T2操作
t1	读取A=10	
t2		读取A=10
t3	写入A=A-5	
t4		读取A=5

并发问题4：幻读（Non-repeatable Read）

幻读

在一个事务内两次查询的数据条数不一致。

举例：

时刻	事务T1 操作	事务T2操作
t1	插入记录	
t2		查询
t3	插入记录	
t4		查询

T2两次查询结果数据不一致

并发问题5：串行化异常（Serialization Exception）

串行化异常

成功提交一组事务的结果与这些事务所有可能的串行执行结果都不一致。

举例：

时刻	事务T1	事务T2
t1	读取 A=10	读取 A=10
t2	写入 A=A-5	
t3		写入A=A-3
t4		

T1和T2串行结果是2，
并行结果可能是5或7

事务隔离级别

级别低导致并发性高；
级别越高并发性越弱

SQL标准中的数据库事务的隔离有4个级别

低

隔离级别

高

(1)读未提交 允许事务读取其他事务未被提交的数据更改。

(2)读已提交 确保事务仅读取其他事务已提交的数据更改。

(3)可重复读 确保在同一事务中多次读取同一数据时的结果是一致的，即使其他事务对这些数据进行了修改。

(4)可串行化 事务执行顺序是可串行化的，当某个事务未提交时，其他事务只能等待。

事务隔离级别与各种并发问题的关系

SQL92事务隔离级别与并发访问问题发生可能性的关系

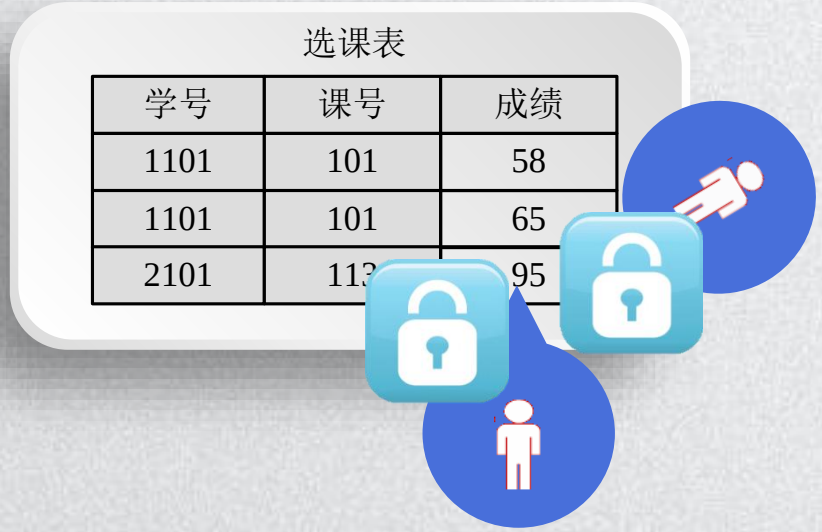
隔离级别	丢失修改	脏　　读	不 可 重 读	幻　　读
读未提交	不允许	允许	允许	允许
读已提交	不允许	不允许	允许	允许
可重复读	不允许	不允许	不允许	允许
可串行化	不允许	不会允许	不允许	不允许

KingbaseES 数据库默认的事务隔离级别是读已提交。

并发访问控制：锁机制和多版本并发控制策略

kingbaseES通过锁机制实现事务隔离

- 锁用于表明事务与资源相关性，由DBMS在内部管理，并基于事务所执行的操作分配和释放。
- KingbaseES遵循严格的两段锁协议。



kingbaseES实现多版本并发控制策略

- 通过维护数据对象的多个版本信息来**实现事务的高效并发**。

锁定管理包括：加锁、锁定和解锁。

锁定的资源粒度

即锁定的数据资源范围。

KingbaseES可以锁定的资源粒度

资 源	锁定描述
行	锁定表中的一条或几条记录
页	对一个数据页或索引页锁定，粒度介于行与表之间
表	对包括所有数据和索引在内的整个表锁定

因为锁定多行需要控制更多的锁，容易发生“死锁”（即两个事务互相阻塞）

锁定资源粒度小，事务等待时间短，并发性高，但系统开销大；
锁定资源粒度大，事务等待时间长，并发性低，但系统开销小。

KingbaseES 锁定模式

并发事务访问资源的方式。不同的锁定模式，其强度和适用场合不同。

序	锁定模式	描　述
1	读取共享锁 Access Share Lock	表级锁。任何只读取表的查询都获此锁。它允许其他事务同时读取表，以及获得8号之外的其他锁。对应SELECT操作
2	行共享锁 Row Share Lock	行级锁。在读取特定行时，防止其他事务对这些行进行排他性修改。它允许其他事务同时读取这些行，以及获得7、8号之外的其他锁。对应SELECT操作
3	行排他锁 Row Exclusive Lock	行级锁。需要对特定行进行更新操作时获此锁，它阻止其他事务获取5、6、7、8号锁。以确保不被干扰。对应UPDATE、DELETE、INSERT操作
4	共享更新排他锁 Share Update Exclusive Lock	表级锁。用于可能会更新表的操作，但允许其他事务进行读操作。它阻止其他事务获取4、5、6、7、8号锁以，防止多个事务同时更新表。对应VACUUM(不带FULL)、ANALYZE、CREATE INDEX CONCURRENTLY、REINDEX CONCURRENTLY, CREATE STATISTICS等操作
5	共享锁 Shared Lock	表级锁。对表进行共享访问时获此锁，同一时间允许多个事务持有该锁读取表，但不允许其他事务获取3、4、6、7、8号排他锁对表进行修改。对应CREATE INDEX（不带CONCURRENTLY）操作
6	共享行排他锁 Share Row Exclusive Lock	表级锁。该锁比Shared Lock的排他性更强，同一时间只允许一个事务持有该锁。不允许其他事务获取3、4、5、6、7、8号锁。对应CREATE TRIGGE，ALTER TABLE（部分）操作
7	排他锁 Exclusive Lock	表级锁。具有很强的排他性，只允许并发的1号锁，即只有来自于表的读操作可以与一个持有该锁模式的事务并行处理。对应REFRESH MATERIALIZED VIEW（不带CONCURRENTLY）操作
8	读取排他锁 Access Exclusive Lock	表级锁。排他性最强，与所有其他锁模式都冲突。它阻止其他事务对表进行任何访问，以保证操作的独占性。对应DROP TABLE、TRUNCATE/ ALTER TABLE（部分）、REINDEX、CLUSTER、VACUUM FULL

小结：锁定机制的选择

如何选择锁粒度和锁模式呢?

初级用户：

让DBMS自动选择和处理!

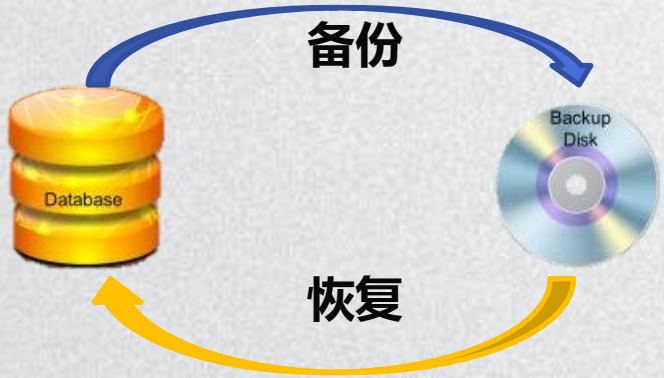
高级用户：

在程序中自定义锁定模式、粒度、持续时间等（通过SQL语句或数据库访问API）。

5.3 数据库备份和转移

数据库备份与恢复简介

备份是指对数据库部分或全部内容进行处理，**生成一个副本**的过程。
数据库备份记录了在进行数据库备份操作时的所有数据状态。



- ✓ 备份内容：可包括数据库中的所有对象
- ✓ 备份设备：磁盘或磁带机
- ✓ 备份格式：备份文件

KingbaseES提供了**物理数据库备份与恢复**和**逻辑备份与恢复**两种机制。

数据库物理备份

物理备份：备份KingbaseES数据库集簇的所有物理文件，包括数据文件、控制文件、WAL日志文件和归档日志文件等。

物理恢复：根据备份集和归档日志将数据库恢复到特定的时间点或事务ID。

用途：物理备份可以用于快速恢复整个数据库，特别是在严重故障如硬件损坏的情况下。

工具：**sys_rman**

物理备份的三种类型

全量备份	差异备份	增量备份
对所有数据库对象进行备份。可以独立恢复整个数据库。	以上一次全量备份为“基准”，仅记录自上次完整备份后更改过的数据。	依赖上一次全量备份，仅记录自上一次全量备份或最后一次差异备份之后发生变化的数据
占空间大，备份时间长，可独立恢复	占空间小，备份时间短，不可独立恢复	占空间小，备份时间短，不可独立恢复

数据库逻辑备份

逻辑备份：将KingbaseES数据库中**数据和结构导出为二进制文件或SQL文件**。

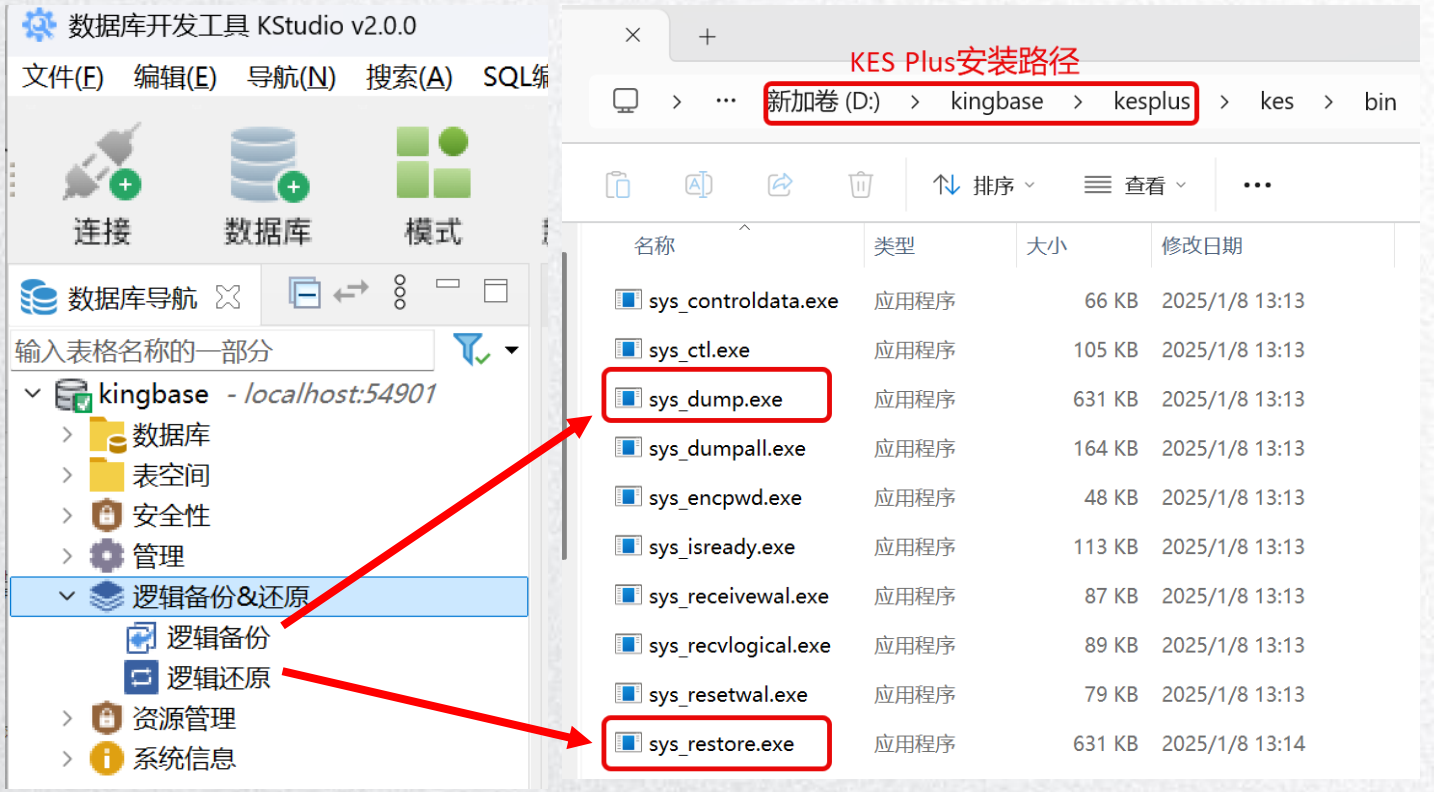
逻辑恢复：将这些备份文件重新加载到KingbaseES数据库中，恢复数据和结构。

用途：方便数据库在不同机器上的迁移，逻辑备份灵活性强、跨平台跨版本兼容，但速度相对慢。

工具：

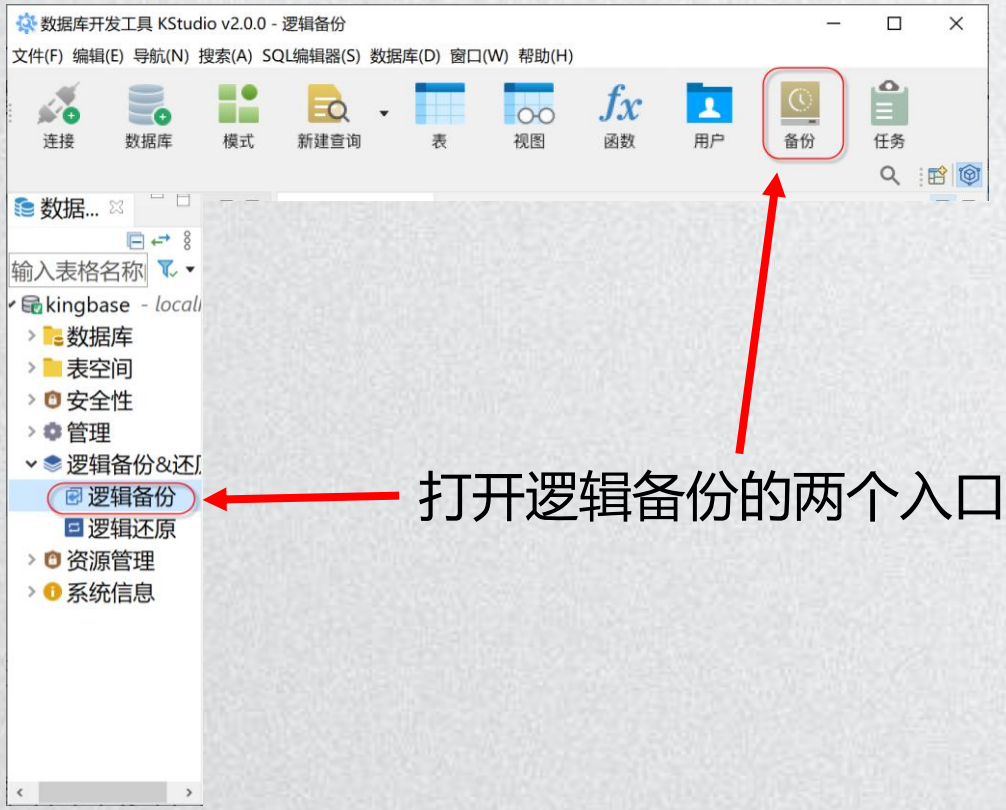
sys_dump和sys_restore导入导出工具。

KStudio进行逻辑备份和恢复时，实质是调用sys_dump和sys_restore工具。

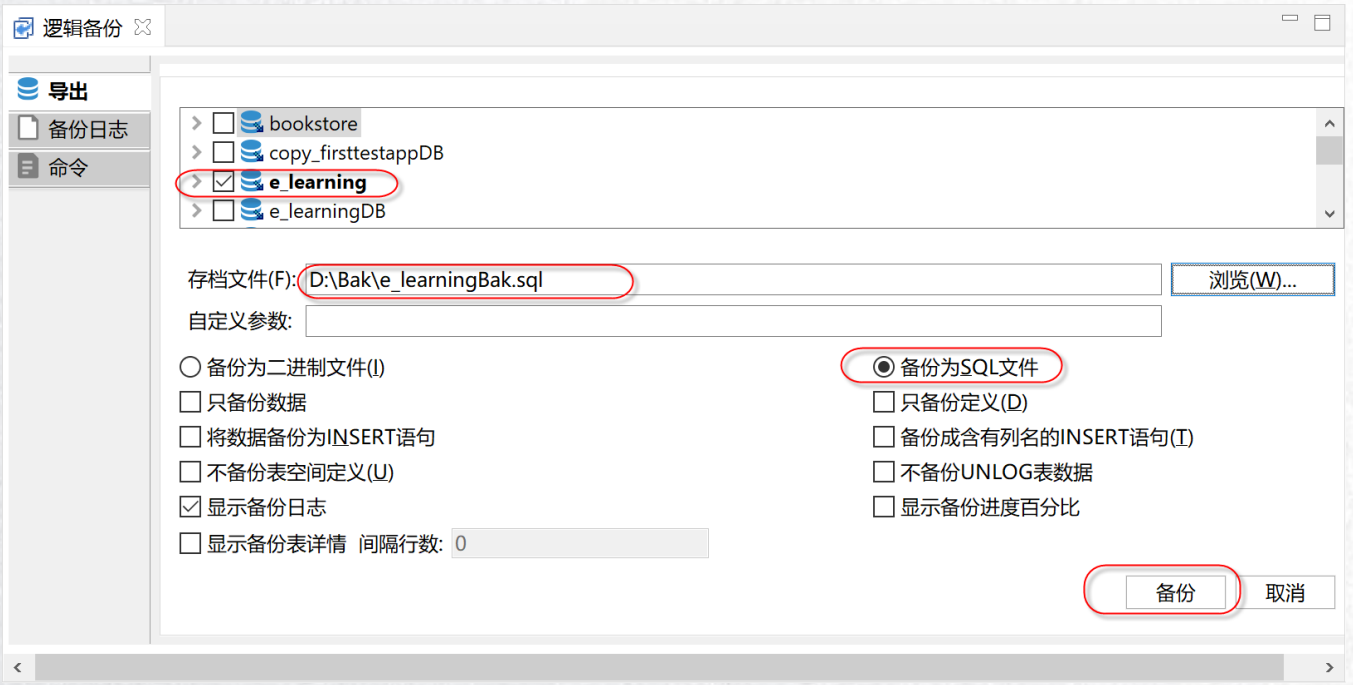


举例：数据库逻辑备份

【例】将e_learning数据库逻辑备份到文件D:\Bak\e_learningBak.sql中



打开逻辑备份的两个入口

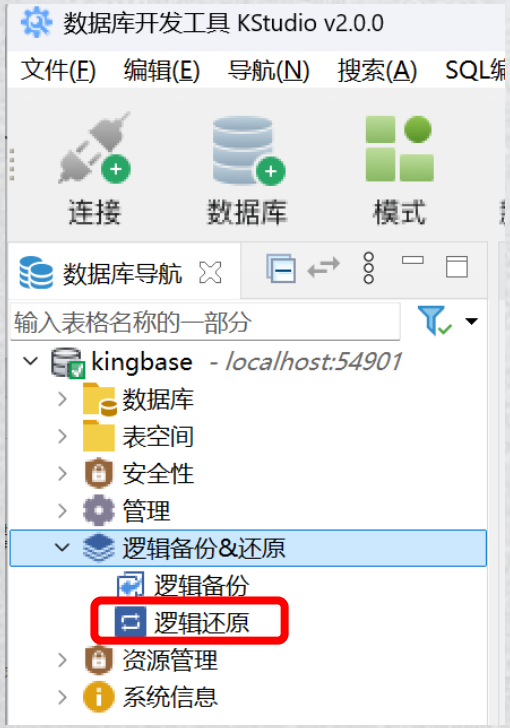


- ✓ 选择需要备份的数据库；
- ✓ 选择备份文件存放路径；
- ✓ 选择备份文件类型：二进制或SQL

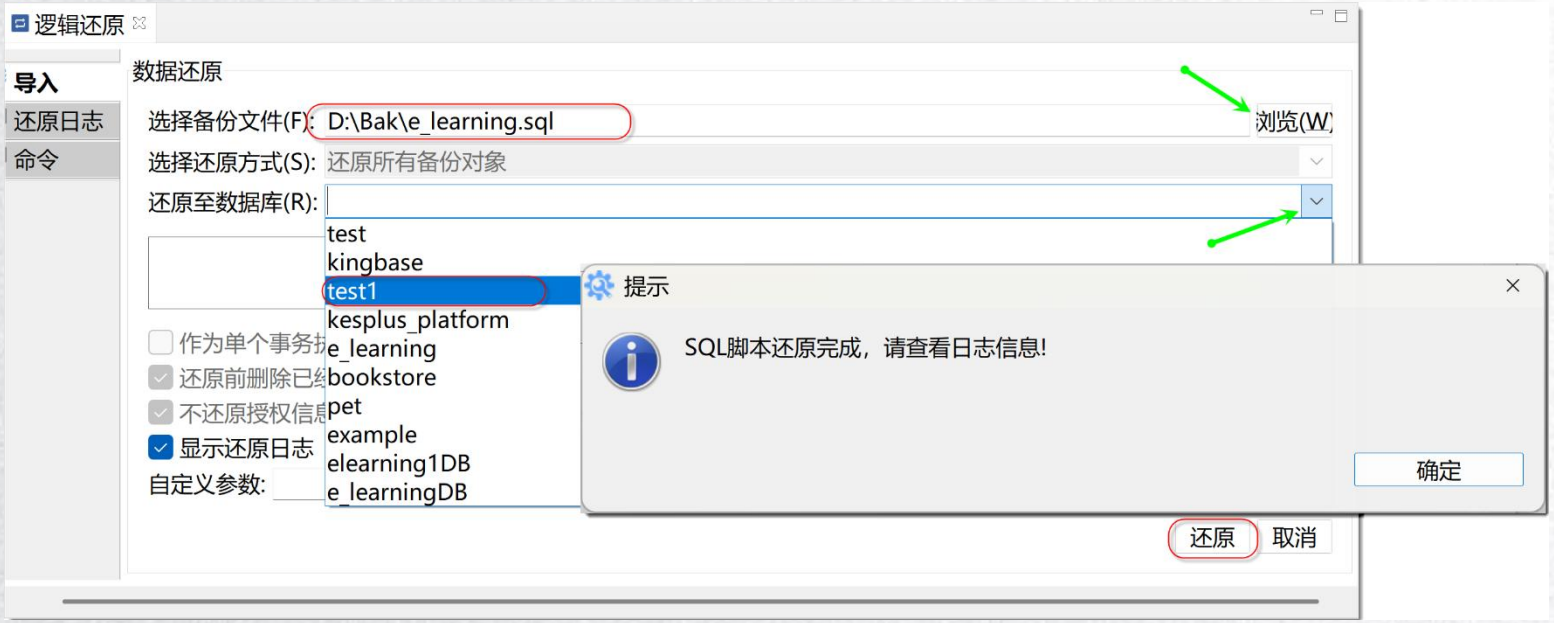
举例：用逻辑备份文件还原到新数据库

【例】使用上例中的e_learning数据库的逻辑备份文件还原数据库为一个新的test1数据库。

① 新建数据库test1。
② 打开逻辑还原页面，配置还原选项。

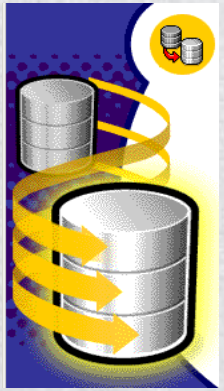


✓ 找到备份文件D:\Bak\e_learningBak.sql
✓ 还原至新创建的test1数据库



数据导入和导出

DBMS支持与其他数据源间转换数据。



导出：将数据从当前数据库复制到其他数据源。

导入：将其他数据源的数据加载到当前数据库。

其他数据源：

数据库

Excel文件

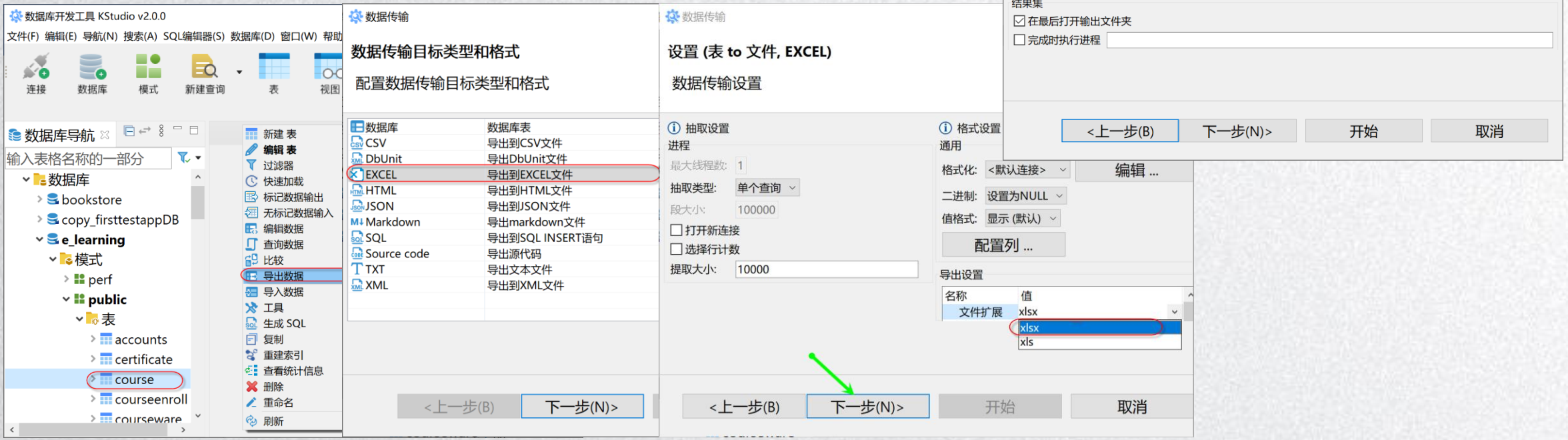
文本文件

……

举例：数据导出

【例】将course表数据导出到Excel文件，保存到D:\Bak文件夹。

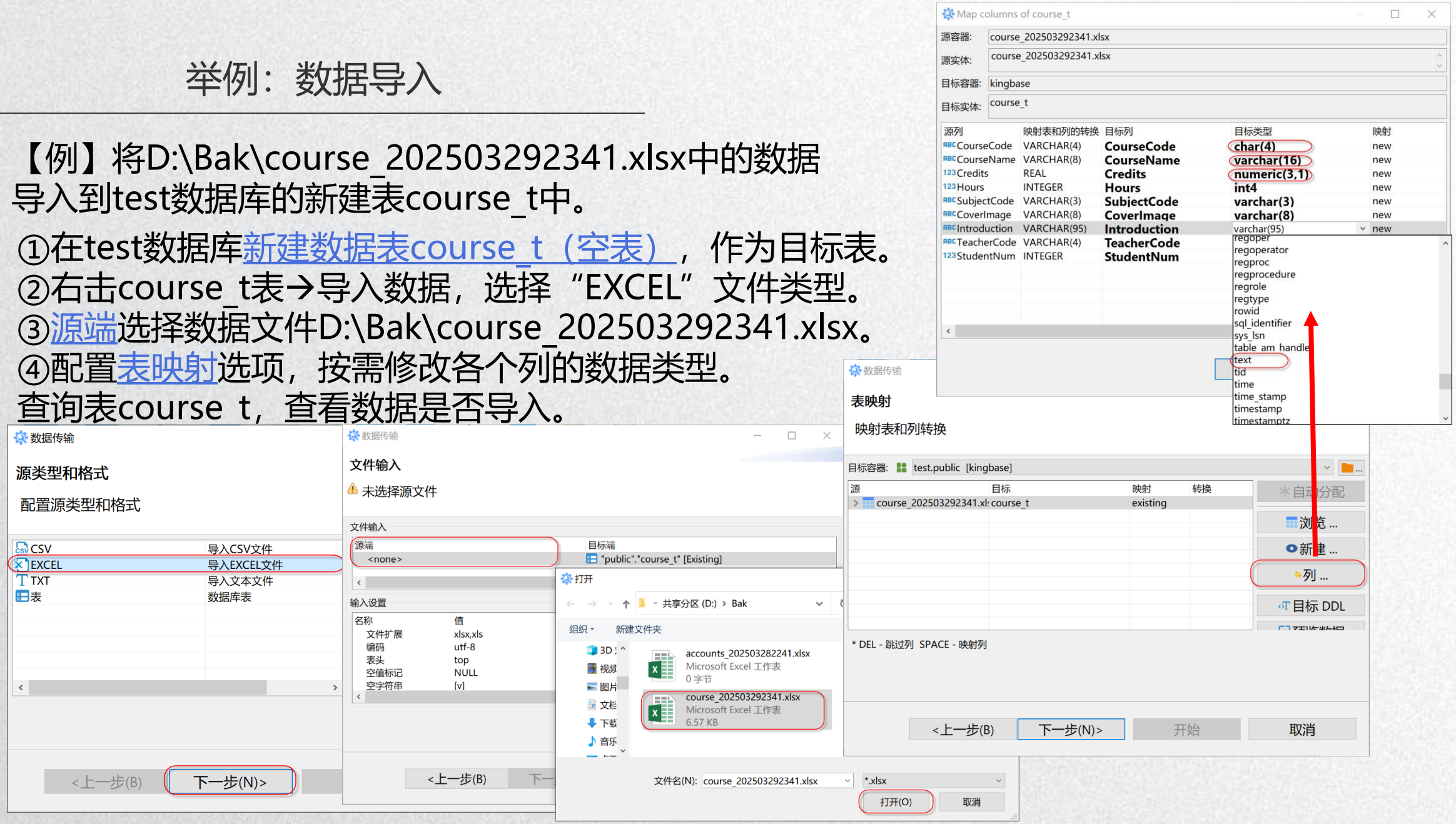
①选择需要导出数据的表，右击-导出数据。
②选择目标类型Excel文件。
③选择抽取设置和格式配置，文件后缀.xlsx。
④配置输出参数，文件存放地址D:\Bak。
打开D:\Bak，查看文件“course_时间戳.xlsx” 例如course_202503292341.xlsx



举例：数据导入

【例】将D:\Bak\course_202503292341.xlsx中的数据导入到test数据库的新建表course_t中。

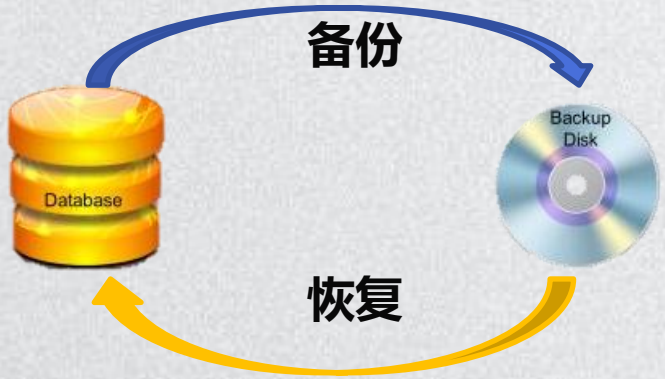
①在test数据库新建数据表course_t（空表），作为目标表。
②右击course_t表→导入数据，选择“EXCEL”文件类型。
③源端选择数据文件D:\Bak\course_202503292341.xlsx。
④配置表映射选项，按需修改各个列的数据类型。
查询表course_t，查看数据是否导入。



提示：导入数据源可以是csv文件、 txt文件、excel文件或某数据库中的已有表。导入操作前先创建目标表。

小结：数据库备份与恢复

备份是创建数据库的资料副本，可用来恢复数据库。



- ✓ 备份非常重要!
- ✓ 数据丢了无法弥补!

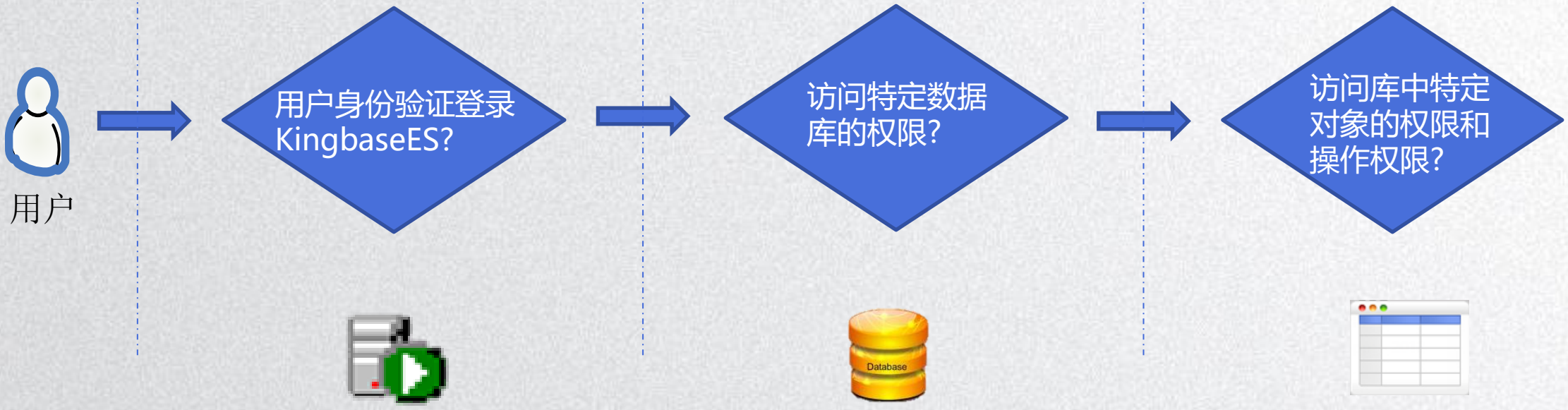
5.4 数据库安全性控制

数据库安全性控制

保护数据库，防止因用户非法使用数据库造成数据泄露、更改或破坏。

KingbaseES数据库用户访问控制

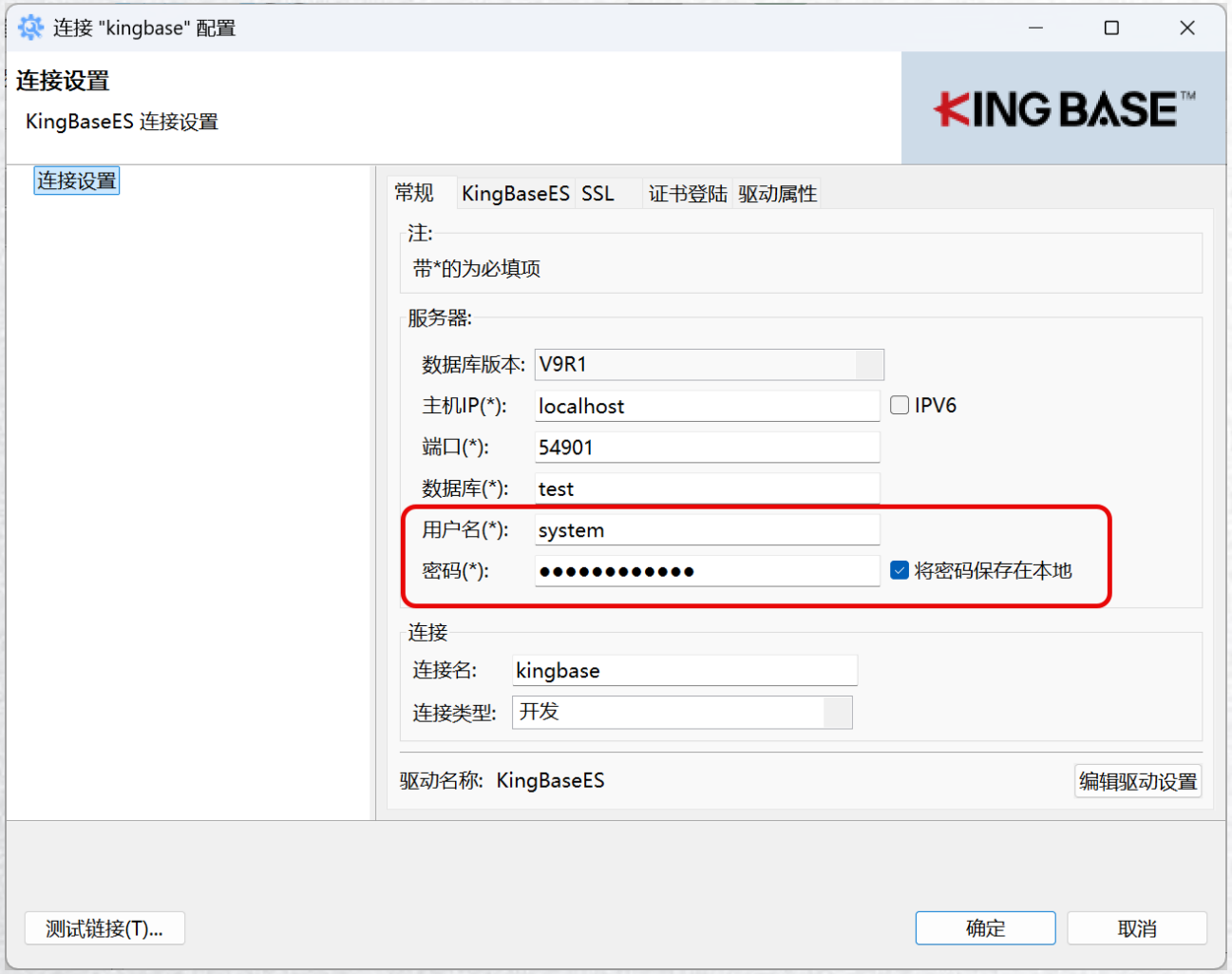
KingbaseES提供的用户访问控制机制在**服务器**、**数据库**和**数据对象**三个层次上进行安全管理。



KingbaseES的身份验证

KES通过建立与数据库服务的连接进行身份验证。

用户必须输入一个KingbaseES用户名及口令。



KingbaseES的用户类型

在数据库安装后默认创建3个管理员，三权分立，避免超级管理员权限过于集中

- ✓ 管理员用户
  - ➢ 系统管理员（system）
  - ➢ 安全管理员（sso）
  - ➢ 审计管理员（sao）
- ✓ 普通用户

默认系统管理员是system，安装时设置了默认密码kingbase123.

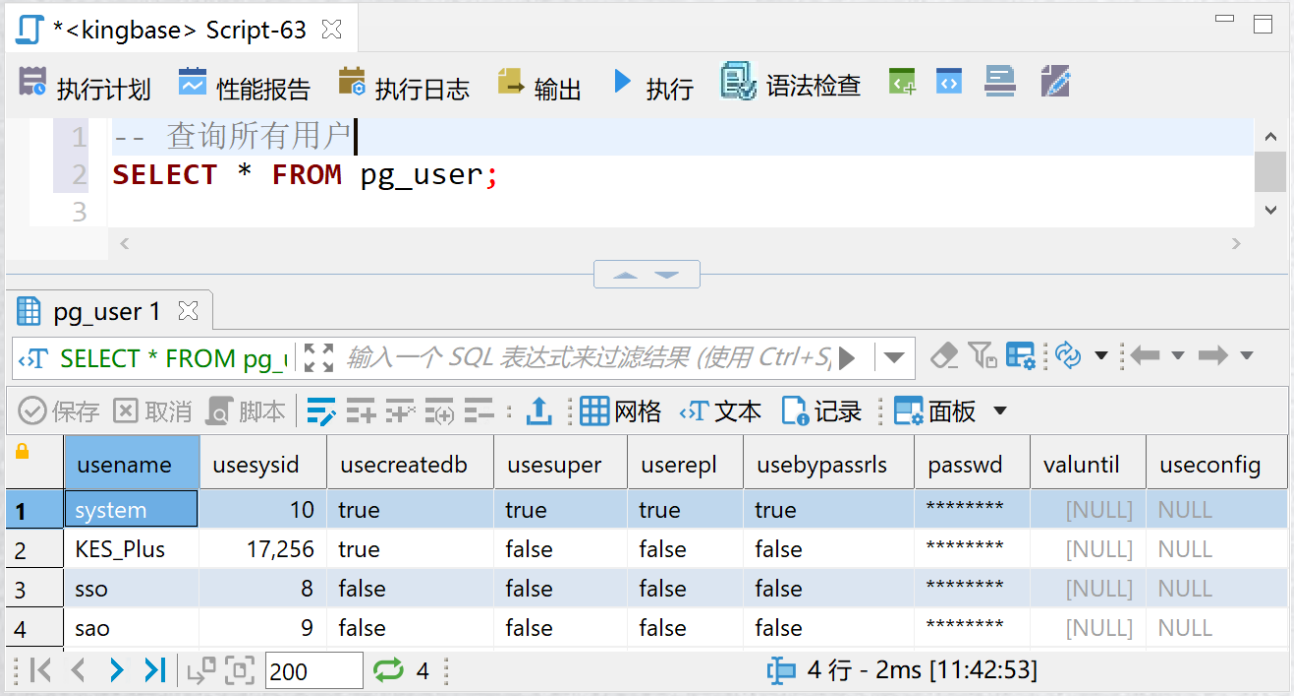
用户类别	权限范围	说明
系统管理员（数据库管理员）	管理日常操作和普通用户	主要负责执行数据库日常管理的各种操作和自主存取控制。
安全管理员	安全管理	主要负责强制访问规则的制定和管理，监督审计管理员和普通用户的操作，管理其他审计用户，但不能创建和操作普通对象。
审计管理员	审计管理	主要负责数据库的审计，监督系统管理员和安全管理员的操作，管理其他安全用户，但不能创建和操作普通对象。
普通用户	数据操作	由系统管理员创建，只具有针对特定服务器、数据库或数据对象被授予的权限。

查看数据库用户

系统表pg_user和系统视图sys_user中存放了所有数据库用户的信息。

SELECT * FROM pg_user;

SELECT * FROM sys_user;



数据库和对象的权限管理

初始状态下，只有对象所有者或系统管理员system拥有对该对象的所有特权。

用户登录后只能进行权限范围内的操作。
用户对自己所创建的对象拥有所有特权；非创建用户必须获得授权后才能访问其他用户创建的数据库对象。

授权方式

- **直接授权**：利用GRANT命令直接为用户授权；
- **间接授权**：先将权限授予角色，再将角色授予用户。角色可视为一组权限的集合。

权限类型

（1）系统权限：与数据库的整体管理和控制相关的权限

- SUPERUSER（超级用户权限，最高的系统权限）
- CREATDB（允许用户创建新的数据库）
- CREATEROLE（允许用户创建新的角色）
- REPLICATION（允许用户参与数据库的复制操作）
- LOGIN（允许角色作为数据库用户登录到 KingbaseES 数据库）

（2）对象权限：用户对特定数据库、模式、数据库对象（如表、存储过程等）的操作权限。

对象权限及涵义

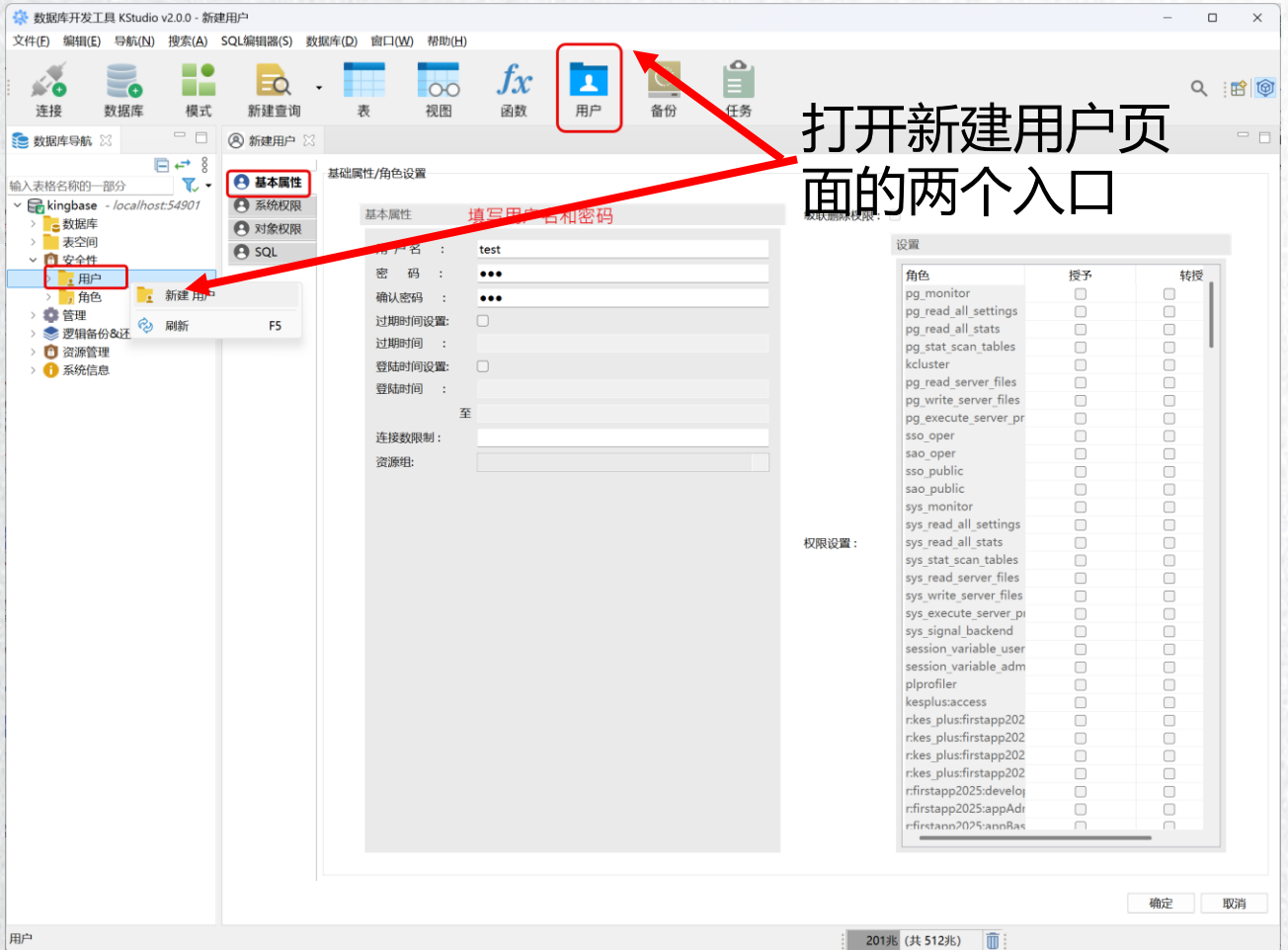
对象	权限	涵义
数据库	1.CONNECT 2.CREATE 3.TEMPORARY 或 TEMP 4. ALL	1.连接权限, 允许用户连接到指定的数据库。 2.创建权限，赋予用户在数据库中创建模式（schema）的权限。 3.临时表使用权限，允许用户在数据库会话期间使用临时表。 4.全部权限，将数据库的所有可用权限（1、2、3）授予用户。
模式	1. CREATE 2. USAGE 3. ALL	1.创建权限，允许用户在指定模式中创建新数据库对象，如表、视图等。 2.使用权限，使用户能够访问模式中的对象，如查询表、视图等，但不具备创建、修改或删除这些对象的权限。 3.全部权限，将模式的所有可用权限（1、2）授予用户。
表和视图	1. SELECT 2. INSERT 3.UPDATE 4.DELETE 5.REFERENCE 6. TRIGGER 7.ALL	1.查询权限，允许用户从表或视图中检索数据。 2.插入权限，赋予用户向表或可更新视图中插入新记录的能力。 3.更新权限，允许用户或可更新视图使用 UPDATE 语句更新表中记录。 4.删除权限，允许用户用 DELETE 语句从表或可更新视图中删除记录。 5.引用权限，允许用户在创建表时定义外键约束引用该表。 6.触发器权限，允许用户在表上创建触发器。 7.全部权限，将表的所有可用权限（1~6）或视图的所有可用权限（1~4）授予用户。
函数和存储过程	EXECUTE	执行权限，允许用户调用函数或存储过程。

举例：创建名为user1的普通用户

【例】以系统管理员system身份连接数据库，创建一个名称为user1的KingbaseES用户账号，其密码为123。先为user1授予CREATEDB和LOGIN系统权限。再为user1添加对象权限，包括对e_learning数据库public模式的全部权限，以及对student表的查询、添加、修改权限。

（1）使用system用户打开数据库连接。打开新建用户页面，在基本属性中，配置用户名user1和密码123。

第一层授权（KingbaseES数据库管理系统）



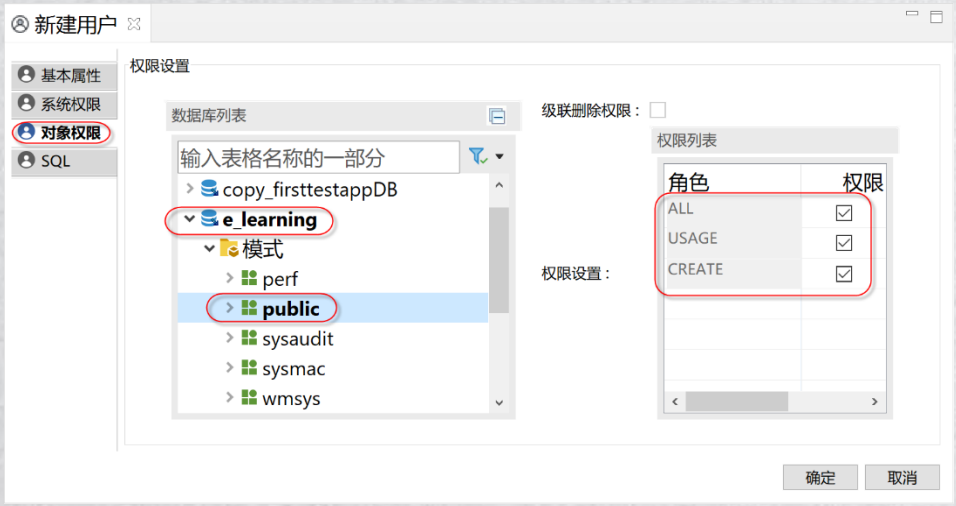
举例：创建名为user1的普通用户

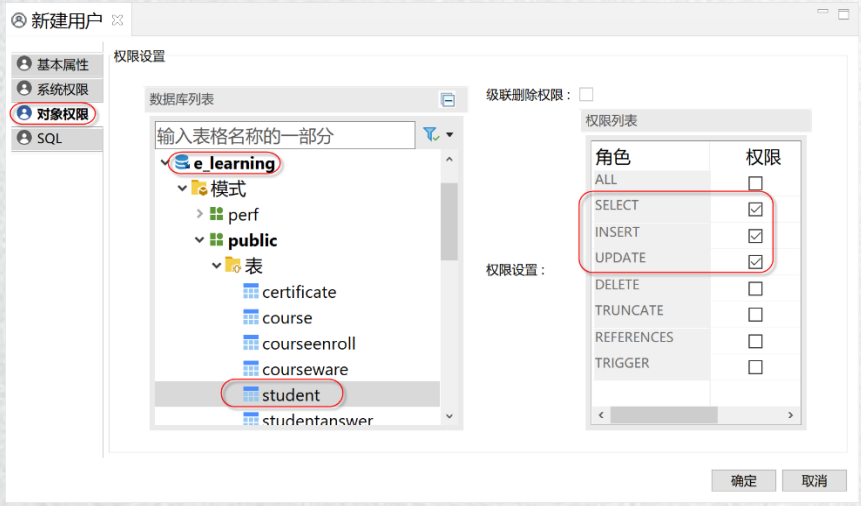
（2）添加系统权限

第二、三层授权（数据库和数据对象）

查看user1用户

（3）添加对象权限。点击确定，创建用户。



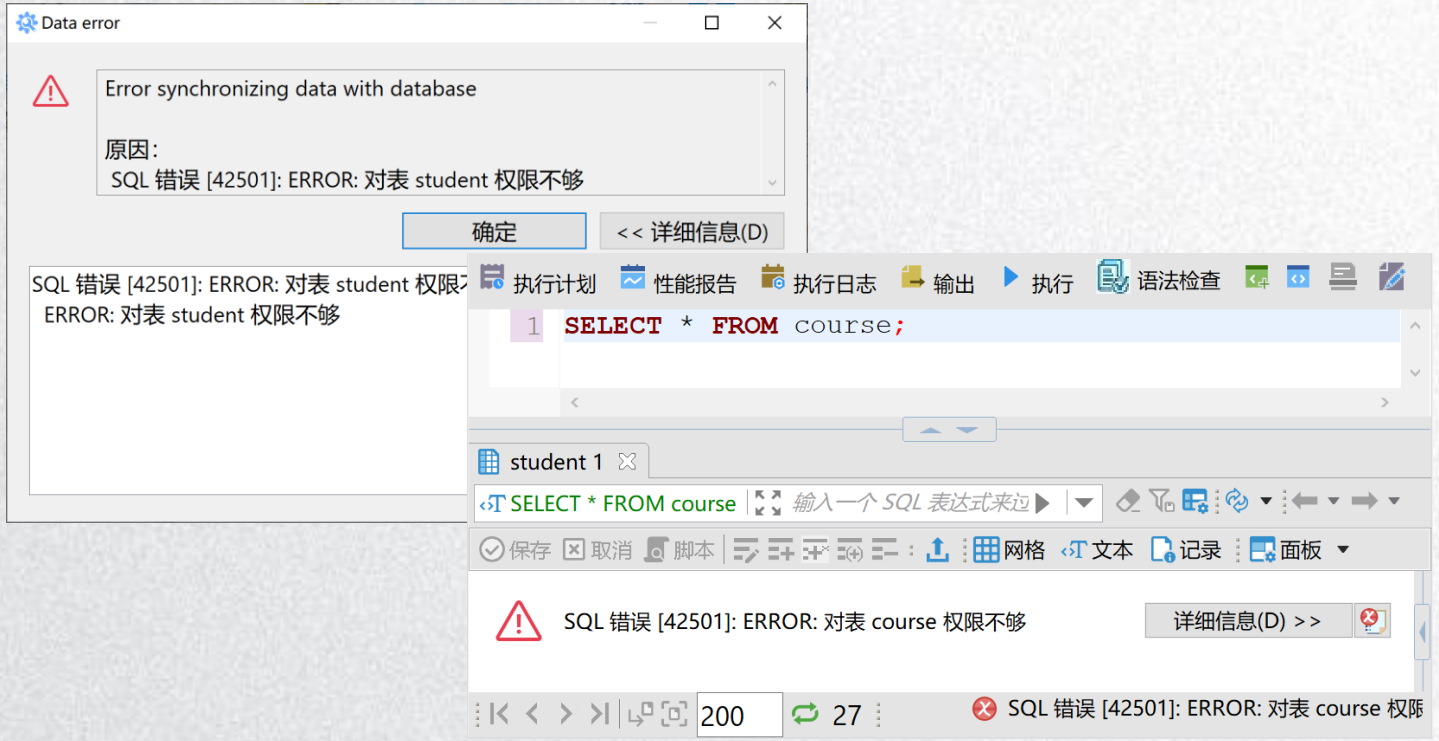


测试user1用户的权限

（4）测试user1用户的权限（**user1用户连接后测试**）
新建连接kingbase2，
用户名user1，密码123。

执行下列操作，观察执行情况。

- 查询student表【成功】
- 修改student【成功】
- 添加记录到student表【成功】
- 删除student表记录【失败】
- 查询course表【失败】



数据库角色管理

角色是权限的集合。将多个权限打包为角色，再将角色批量授予用户，可避免重复授权、简化授权管理过程。

在KingbaseES中，用户是角色的别名。

创建用户/角色可以用CREATE USER或CREATE ROLE语句。

两者区别：

- CREATE USER默认LOGIN允许用户登录数据库，
- CREATE ROLE默认NOLOGIN不允许登录数据库。

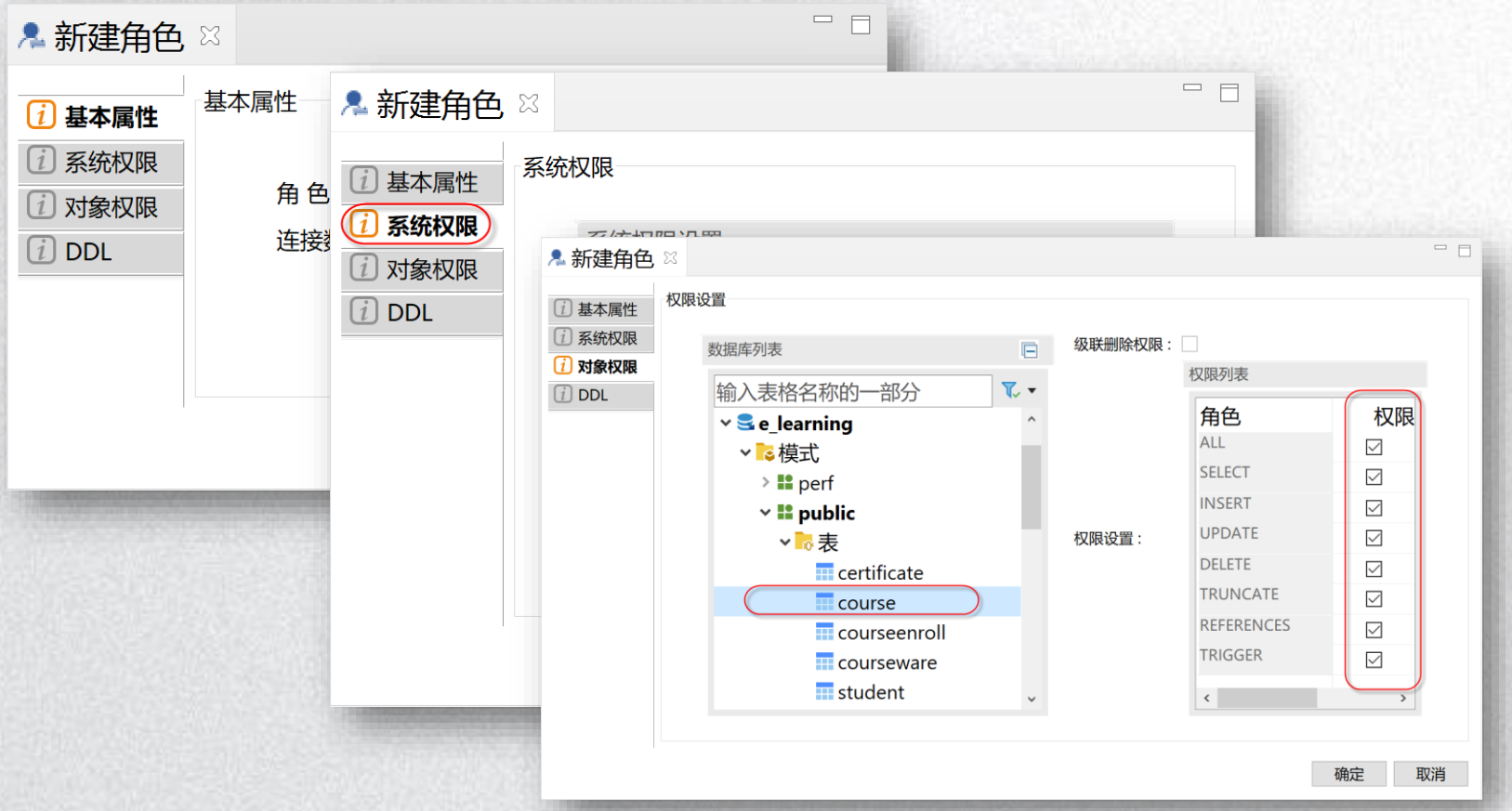
还可以通过KStudio可视化方式创建角色/用户。

举例：创建名为role1的角色

【例】创建一个名称为role1的角色，为role1授予CREATDB和LOGIN的系统权限；再为role1添加对象权限，包括：对e_learning数据库public模式的全部权限，以及对该模式中course表的ALL权限。然后，将角色role1授予用户user1和user2。

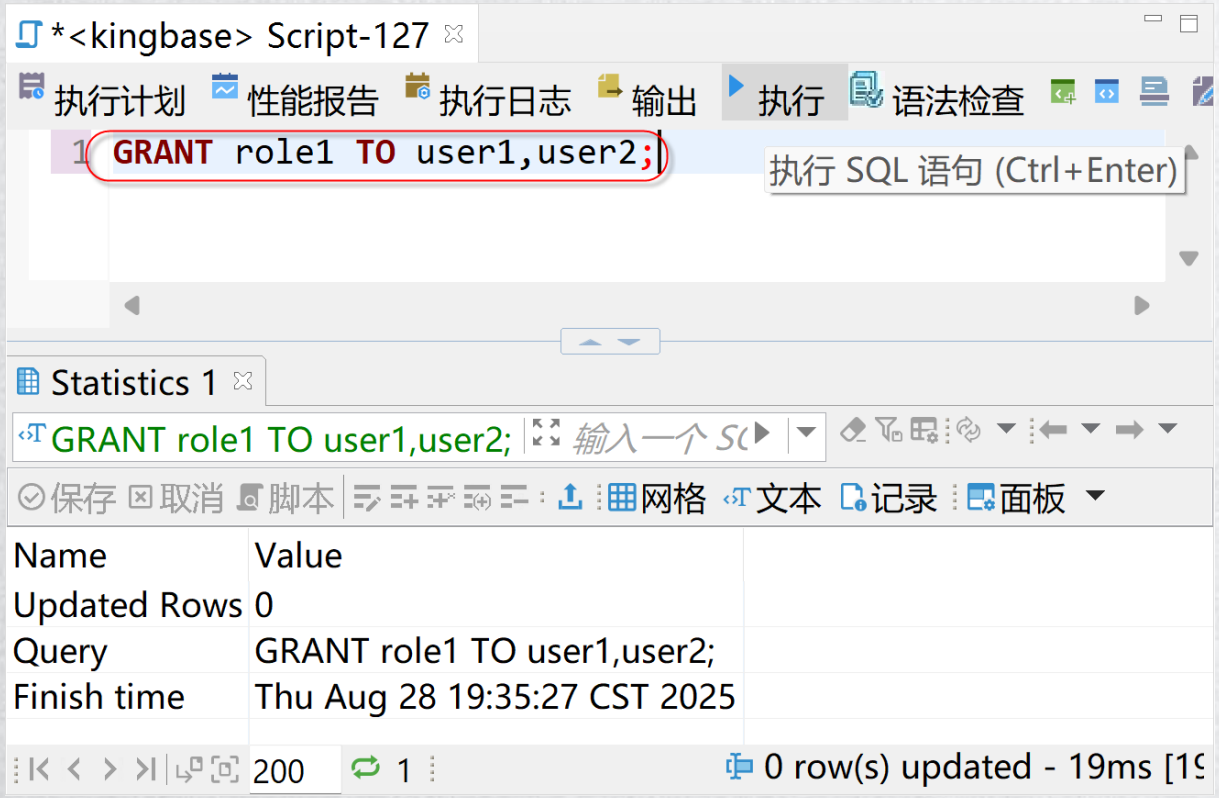
（1）使用system用户打开数据库连接。在导航树找到“安全性”节点，右击“角色”，从快捷菜单中选择“新建 角色”命令。

（2）设置系统权限和对象权限。确定



举例：将role1的授予user1和user2

（3）将role1授予user1和user2



其他数据库安全性机制

视图

为不同的用户定义不同的视图，通过视图机制把要保密的数据对无权限的用户隐藏起来。

审计

跟踪用户的全部操作，**事后检查**。把用户对数据库的所有操作自动记录下来并放入审计日志文件。

加密

对高度敏感数据进行加密，防止数据在存储和传输中失密。

THANK YOU!