大型数据库应用技术

闪回、课程回顾

授课教师：王欢

内容提纲

- 闪回技术

- 课程回顾

闪回技术

闪回技术概述

闪回技术（Flashback）是 Oracle 强大数据库备份恢复机制的一部分，在数据库发生逻辑错误的时候，闪回技术能提供快速且最小损失的恢复（多数闪回功能都能在数据库联机状态下完成）。

需要注意的是，闪回技术旨在快速恢复逻辑错误，对于物理损坏或是介质丢失的错误，闪回技术就回天乏术了，还是得借助于 Oracle 一些高级的备份恢复工具如 RMAN 去完成。

闪回技术

□撤销段

大部分闪回技术都需要依赖撤销段（UNDO SEGMENT）中的撤销数据。撤销数据是反转 DML 语句结果所需的信息，只要某个事务修改了数据，那么更新前的原有数据就会被写入一个撤销段。

事务启动时，Oracle 会为其分配一个撤销段，事务和撤销段存在多对一的关系，即一个事务只能对应一个撤销段，多个事务可以共享一个撤销段（不过在数据库正常运行时一般不会发生这种情况）。

闪回技术

闪回技术

- 闪回数据库（Flashback Databse）
- 闪回丢弃（Flashback Drop）
- 闪回表（Flashback Table）
- 闪回查询（Flashback Query）
- 闪回事务（Flashback Transaction）
- 闪回数据归档（Flashback Data Archive）

闪回技术

7种闪回技术

闪回级别	应用场景	依赖对象	影响数据
数据库	表截断、逻辑错误、其他多表意外事件	闪回日志、undo	是
DROP	删除表	回收站	是
表	更新、删除、插入记录	还原数据、undo	是
查询	当前数据和历史数据对比		否
版本查询	比较行版本		否
事务查询	比较		否
归档	DDL、DML	归档日志	是

闪回技术

闪回基本操作

开启闪回的必要条件：

1. 开启归档日志。

```
archive log list;  --查看当前模式
alter database archivelog;  --修改数据库为归档模式
```

2. 设置合理的闪回区

db_recovery_file_dest：指定闪回恢复区的位置

db_recovery_file_dest_size：指定闪回恢复区的可用空间大小

db_flashback_retention_target：指定数据库可以回退的时间，单位为分钟，默认1440分钟(1天),实际取决于闪回区大小

闪回技术

闪回基本操作

检查是否开启闪回：

```
select flashback_on from v$database;
```

开启/关闭闪回：

```
alter database flashback on/off;
```

闪回技术

闪回数据库

闪回数据库是指将整个数据库回退到指定的一个时间点，实际上是数据库不完全恢复的另一种方式。

真正的不完全恢复需要消耗的时间与数据库的大小有密切联系，数据库越庞大，需要的时间就越多，而闪回数据库技术改变了这个窘境，其处理问题方式着重于如何将相对较短的时间内发生的变更去除，而不是如何还原足够旧的备份。

闪回数据库需要使用两种日志：闪回日志和归档日志。闪回日志的记载正好与归档日志的记载相反。

闪回技术

□闪回数据库——基本原理

设当前时间点为 TC，数据库闪回的目标时间点为 T1（TC 大于 T1）。

方式一：

- 第一阶段：利用闪回日志将数据库从 TC“倒推”至较 T1 更旧的某一时刻 T2。
- 第二阶段：利用归档日志将数据库从 T2“前滚”至 T1。

闪回技术

□闪回数据库——基本原理

设当前时间点为 TC，数据库闪回的目标时间点为 T1

方式二：

- 第一阶段：确定一个比 T1 更旧的某一时刻 T2。对于闪回日志无法恢复的对象或数据（执行了truncate、drop等），设此类对象创建时间为 T3，首先利用 T3 到 T2 的归档日志将其重新产生并恢复。
- 第二阶段：利用闪回日志将数据库从 TC“倒推”至 T2。
- 第三阶段：利用归档日志将数据库从T2“前滚”至 T1（途中经过 T3）。

闪回技术

闪回数据库——基本步骤

startup force mount;　--进入数据库的MOUNT状态

flashback database to scn 1041440;　--回到SCN号1041440

alter database open read;　-- 以只读形式打开数据库，查看SCN在1041440时所有数据，以便判断闪回操作是否达到目的

....

startup force mount;

alter database open resetlogs;　--获得满意结果后，如果是主库，最后以resetlogs方式打开数据库，效果和不完全恢复一模一样

alter database recover managed standby database using current logfile disconnect from session parallel 4;　--如果是从库，通过dblink或dump将需要数据取出，然后恢复主从同步

闪回技术

闪回数据库——相关操作

```
--指定数据库保留两天的闪回日志
alter system set db_flashback_retention_target=2880;
--查看闪回日志是否已启用
select flashback_on from v$database;
--检查最远可以闪回到何时
select oldest_flashback_scn,to_char(oldest_flashback_time,'YYYY-
MM-DD HH24:MI:SS') from v$flashback_database_log;
```

闪回技术

闪回丢弃

闪回丢弃指的是撤销“DROP TABLE”的效果，需要启用回收站。执行 drop table 时，表和索引并没有被真正删除，其所占空间（称为段）只是分配给了回收站对象，并且这种分配没有使数据和数据块发生任何移动，还是存在原来的数据文件及表空间中。

```
show recyclebin    --查看回收站信息
 alter system set recyclebin='off' scope=spfile;    --禁用回收站功能
purge user_recyclebin;    --删除用户回收站中的所有对象
purge dba_recyclebin;    --删除数据库中所有回收站中的所有对象
```

闪回技术

闪回丢弃——相关操作

```
--示例
create table emp1 tablespace users as select * from scott.emp;
drop table emp1;
flashback table emp1 to before drop;
select count(*) from emp1;
--其它相关操作
flashback table emp1 to before drop rename to emp1old;
select owner,object_name,original_name from dba_recyclebin where
can_undrop='NO';   --确认哪些回收站对象不能回到被DROP之前
drop table emp1 purge;   --drop时跳过回收站
```

闪回技术

口闪回表

闪回表命令能够以表为单位将数据恢复为“以前”的样子。被闪回的表必须启用行移动功能

- 闪回表命令执行者必须有 FLASHBACK ANY TABLE 权限或者在被闪回的表上有 FLASHBACK 对象权限。
- FLASHBACK TABLE 属于 DDL 命令，自动提交。
- SYS 用户的表无法使用此功能。

```
--启用行移动功能
alter table scott.emp enable row movement;
```

闪回技术

闪回表——相关操作

```
--将scott.emp闪回到10分钟之前
flashback table scott.emp to timestamp (systimestamp - interval '10' minute);
--将scott.emp闪回到SCN为1080381的时候
flashback table scott.emp to scn 1080381;
--将scott.emp和scott.dept两张表同时闪回到SCN为1080381的时候（有外键关系的表可能需要一起闪回）
flashback table scott.emp, scott.dept to scn 1080381;
```

闪回技术

闪回查询

以表为单位查询其过去的数据称为闪回查询，闪回查询能够在 undo 段内搜索“旧”数据，数据库能够保留多少 undo 数据决定了闪回查询的时间窗口的大小。

- 闪回时间点查询：在过去的一个时间点上的查询（as of子句与dbms_flashback包）
- 闪回版本查询：在过去的一段时间范围内的查询（versions between子句）

闪回技术

闪回查询——闪回时间点查询

```
--查询scott.emp在2019-12-25 06:00:00时所有的行
select * from scott.emp as of timestamp to_timestamp('2019-12-25
06:00:00','YYYY-MM-DD HH24:MI:SS');
--查询7369号员工5分钟前的薪水
select sal from scott.emp as of timestamp (systimestamp - interval '5'
minute) where empno=7369;
--将7369号员工的薪水修改为5分钟前的值
update scott.emp set sal =(select sal from scott.emp as of
timestamp  (systimestamp - interval '5' minute) where empno=7369)
where empno=7369
```

闪回技术

闪回查询——闪回版本查询

--闪回版本查询用于查询一段时间范围内表的数据，只使用一条查询就能返回该时间窗口内的不同时间点上的数据

--比如，首先通过3个事务将7934号员工的薪水进行修改。其值原先是sal1，然后是sal2和sal3，最后为sal4。对7934号员工执行闪回版本查询的结果将返回多个7934号员工查询值。

```
select empno,sal from scott.emp
versions between timestamp
(systimestamp - interval '10' minute)
and maxvalue where employee_id=7934;
```

EMPNO	SAL
-7934	sal4的值
7934	sal3的值
7934	sal2的值
7934	sal1的值

接口技术

JDBC

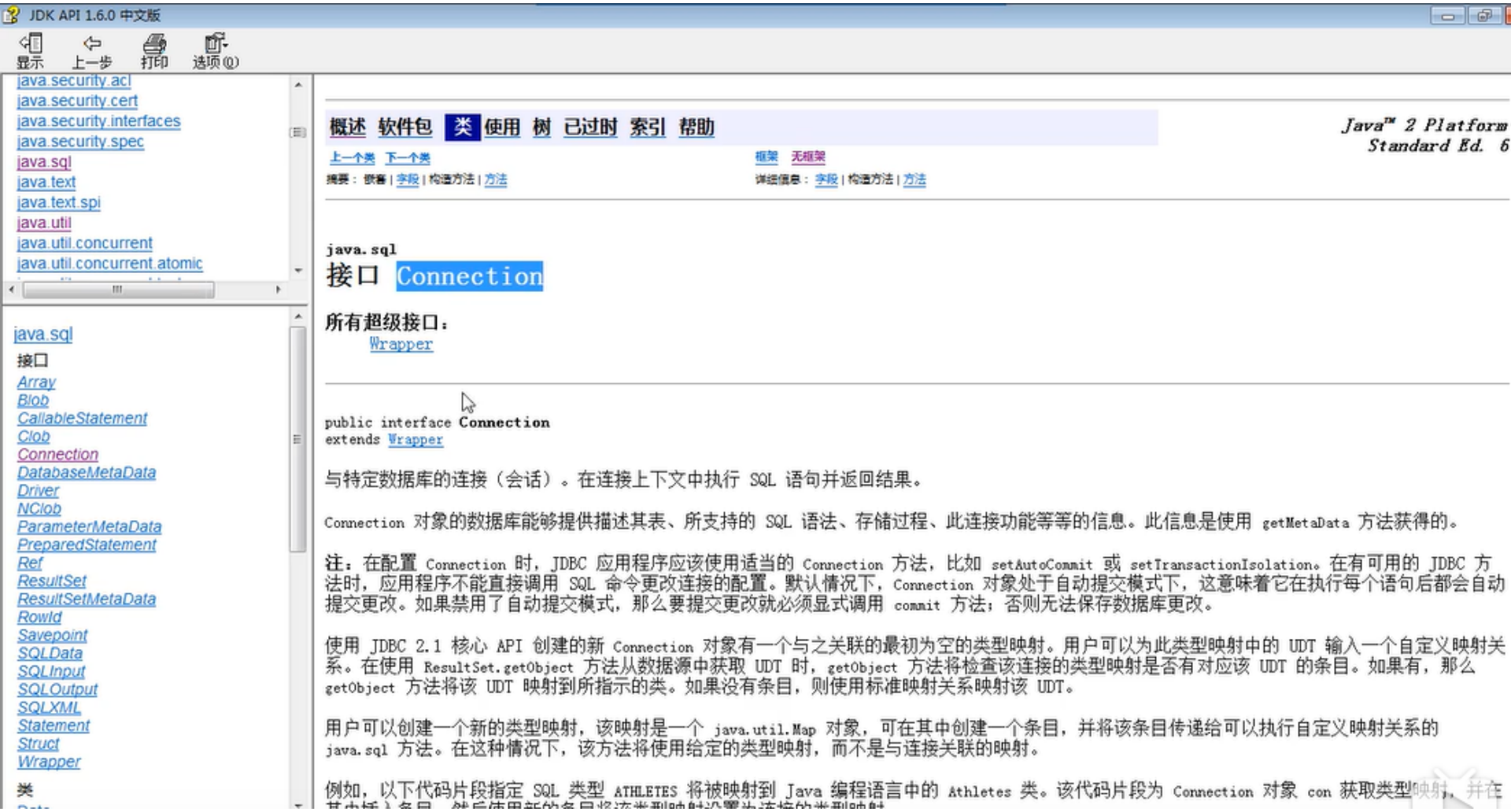
- Java Database Connectivity是SUN公司制定的一套接口(interface)

  JDK帮助文档，在 Java.sql包下面

- 为什么要面向接口编程?

  降低程序耦合度，提高程序扩展力

  面向抽象编程的思想


JDK API 1.6.0 中文版
显示
上一步
打印
选项(O)
java.security.acl
java.security.cert
java.security.interfaces
java.security.spec
java.sql
java.text
java.text.spi
java.util
java.util.concurrent
java.util.concurrent.atomic
概述 软件包 类 使用 树 已过时 索引 帮助
Java™ 2 Platform
Standard Ed. 6
上一个类 下一个类
框架 无框架
摘要： 嵌套 | 字段 | 构造方法 | 方法
详细信息： 字段 | 构造方法 | 方法
java.sql
接口 Connection
所有超级接口：
Wrapper
java.sql
接口
Array
Blob
CallableStatement
Clob
Connection
DatabaseMetaData
Driver
NClob
ParameterMetaData
PreparedStatement
Ref
ResultSet
ResultSetMetaData
RowId
Savepoint
SQLData
SQLInput
SQLOutput
SQLXML
Statement
Struct
Wrapper
类
public interface Connection
extends Wrapper
与特定数据库的连接（会话）。在连接上下文中执行 SQL 语句并返回结果。
Connection 对象的数据库能够提供描述其表、所支持的 SQL 语法、存储过程、此连接功能等等的信息。此信息是使用 getMetaData 方法获得的。
注：在配置 Connection 时，JDBC 应用程序应该使用适当的 Connection 方法，比如 setAutoCommit 或 setTransactionIsolation。在有可用的 JDBC 方法时，应用程序不能直接调用 SQL 命令更改连接的配置。默认情况下，Connection 对象处于自动提交模式下，这意味着它在执行每个语句后都会自动提交更改。如果禁用了自动提交模式，那么要提交更改就必须显式调用 commit 方法；否则无法保存数据库更改。
使用 JDBC 2.1 核心 API 创建的新 Connection 对象有一个与之关联的最初为空的类型映射。用户可以为此类型映射中的 UDT 输入一个自定义映射关系。在使用 ResultSet.getObject 方法从数据源中获取 UDT 时，getObject 方法将检查该连接的类型映射是否有对应该 UDT 的条目。如果有，那么 getObject 方法将该 UDT 映射到所指示的类。如果没有条目，则使用标准映射关系映射该 UDT。
用户可以创建一个新的类型映射，该映射是一个 java.util.Map 对象，可在其中创建一个条目，并将该条目传递给可以执行自定义映射关系的 java.sql 方法。在这种情况下，该方法将使用给定的类型映射，而不是与连接关联的映射。
例如，以下代码片段指定 SQL 类型 ATHLETES 将被映射到 Java 编程语言中的 Athletes 类。该代码片段为 Connection 对象 con 获取类型映射，并在

接口技术

JDBC

JDBC编程五步：

- 1 加载驱动（告诉Java程序，即将连接的是什么数据库）
- 2 获取连接
- 3 获取数据库操作对象（构建sql语句）
- 4 执行SQL语句（DML、DQL。。。）
- 5 处理查新结果集（DQL）
- 6 释放资源（java和数据库属于进程间的通信，开启之后要记得关闭）

接口技术

JDBC

- **Java jdbc连接Oracle参考：**

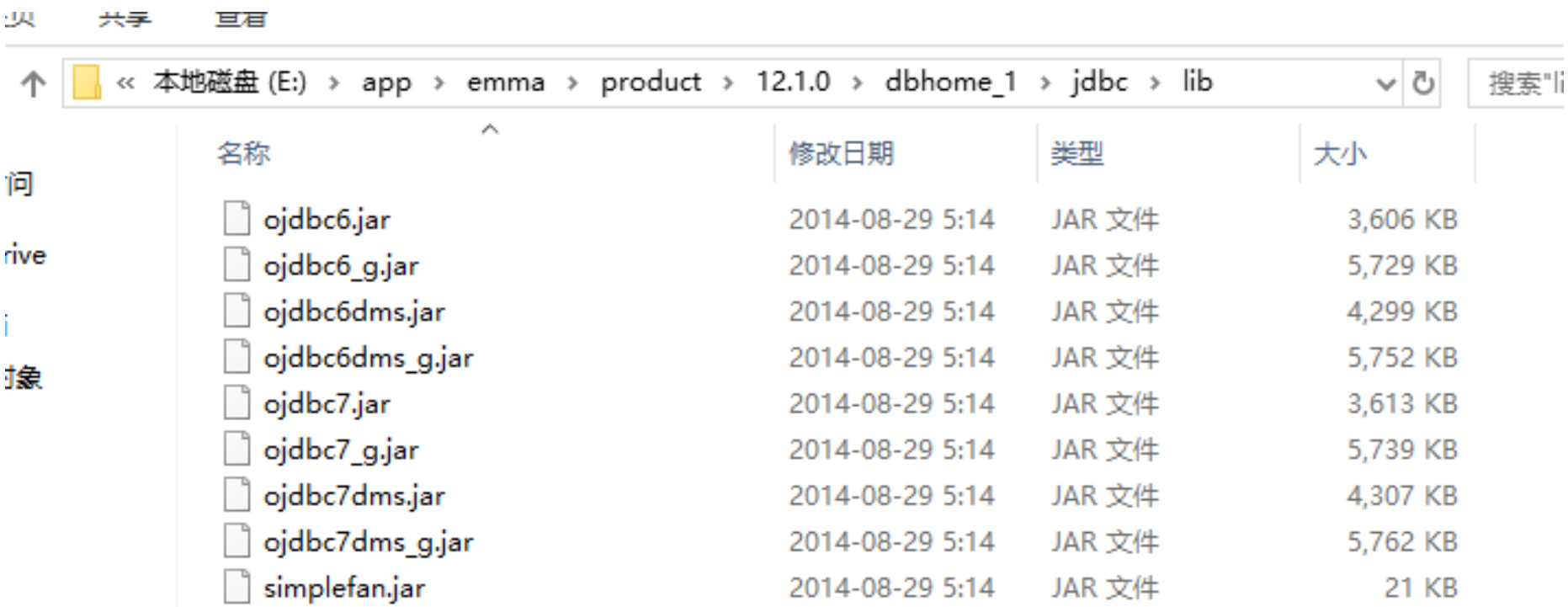
https://blog.csdn.net/LovecraftXu/article/details/92424308?utm_medium=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-2.add_param_isCf&depth_1-utm_source=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-2.add_param_isCf
https://www.cnblogs.com/linlf03/p/8215677.html

- **Ojdbc7.jar下载：**

https://www.oracle.com/database/technologies/jdbc-drivers-12c-downloads.html

- **学习视频：**

https://www.bilibili.com/video/BV1S441137SL?p=46


« 本地磁盘 (E:) › app › emma › product › 12.1.0 › dbhome_1 › jdbc › lib
名称
修改日期
类型
大小
ojdbc6.jar 2014-08-29 5:14 JAR 文件 3,606 KB
ojdbc6_g.jar 2014-08-29 5:14 JAR 文件 5,729 KB
ojdbc6dms.jar 2014-08-29 5:14 JAR 文件 4,299 KB
ojdbc6dms_g.jar 2014-08-29 5:14 JAR 文件 5,752 KB
ojdbc7.jar 2014-08-29 5:14 JAR 文件 3,613 KB
ojdbc7_g.jar 2014-08-29 5:14 JAR 文件 5,739 KB
ojdbc7dms.jar 2014-08-29 5:14 JAR 文件 4,307 KB
ojdbc7dms_g.jar 2014-08-29 5:14 JAR 文件 5,762 KB
simplefan.jar 2014-08-29 5:14 JAR 文件 21 KB

内容提纲

- 闪回技术

- 课程回顾

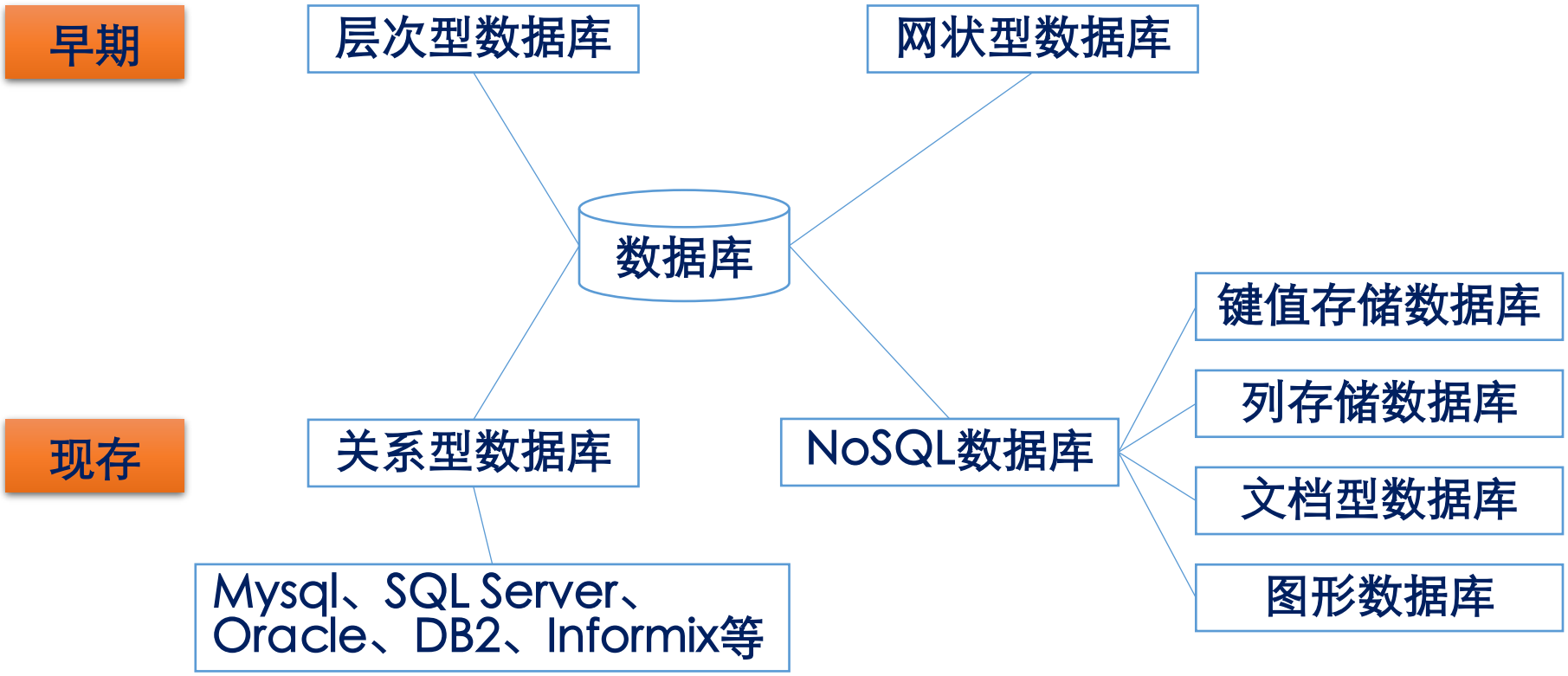
课程回顾

□基本概念

- 数据（Data）
- 数据库（DB）
- 数据库系统（DBS）
- 数据库管理系统（DBMS）
- 数据库管理员（DBA）

课程回顾

□数据库分类


早期
层次型数据库
网状型数据库
数据库
现存
关系型数据库
NoSQL数据库
键值存储数据库
列存储数据库
文档型数据库
图形数据库
Mysql、SQL Server、
Oracle、DB2、Informix等

课程回顾

Oracle12c 工具

- Net Manager
- Net Configuration Assistant
- Enterprise Manager Express
- SQL Developer
- SQL*Plus

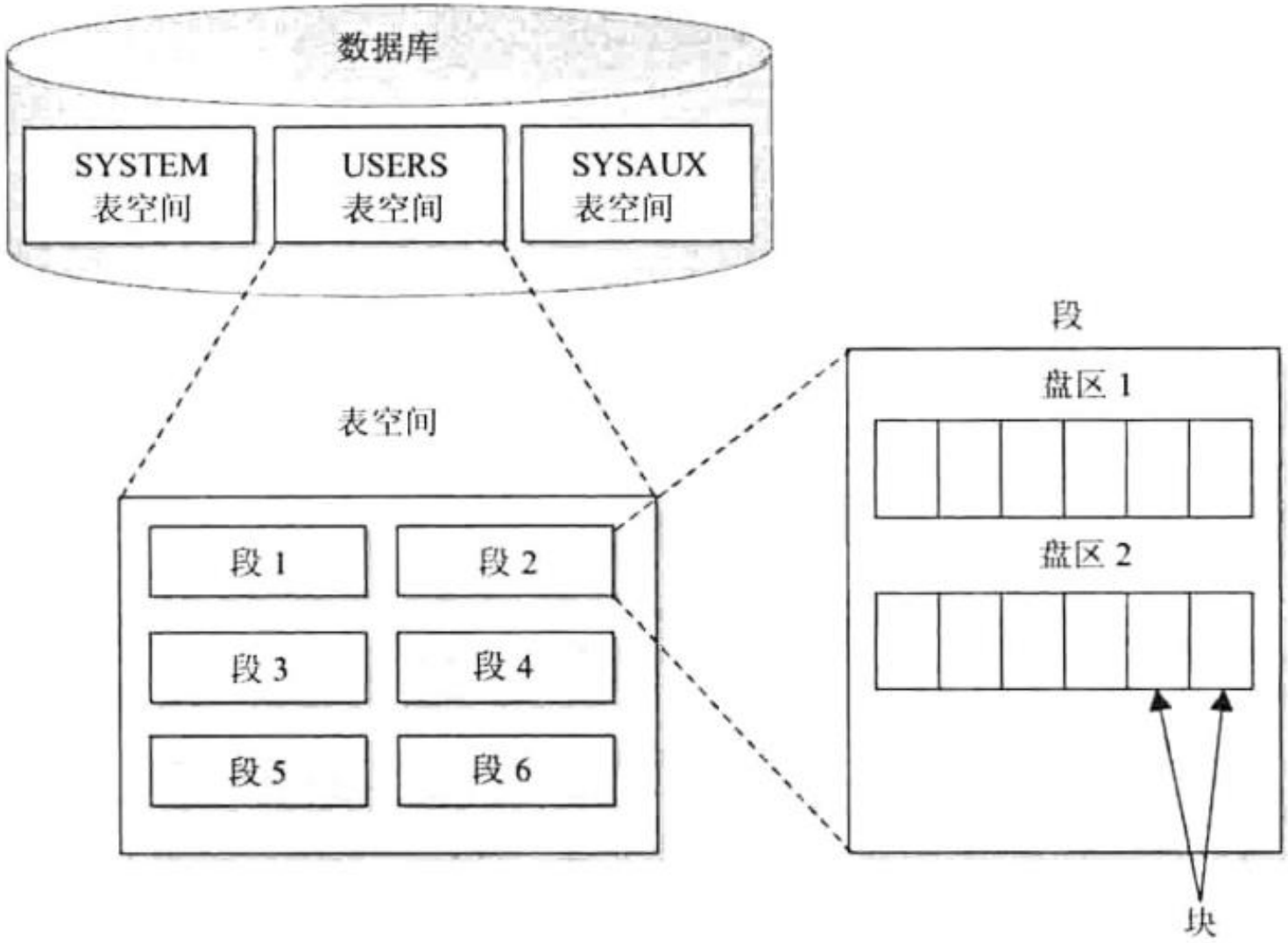
课程回顾

体系结构

- 体系结构是从某一个角度，来分析与考察 Oracle 数据库的组成、工作原理、工作过程、包括数据在数据库中的组织与管理机制、进程的分工协作机制，以及各个组成部分的必要性、功能、和它们之间的联系。
- 从体系结构上讲，完整的 Oracle 数据库系统包括数据库DB、数据库管理系统DBMS两大部分。这两个部分分别对应的是存储结构和软件结构。
- 存储结构核心是“数据如何存放”。包括逻辑存储结构和物理存储结构。
- 软件结构核心是“软件如何运行”。包括内存结构和进程结构。

课程回顾

□逻辑存储结构


数据库
SYSTEM
表空间
USERS
表空间
SYSAUX
表空间
表空间
段 1
段 2
段 3
段 4
段 5
段 6
段
盘区 1
盘区 2
块

体系结构

□ 存储结构 —— 逻辑存储结构

- 表空间(Tablespaces)：数据库被分成称为表空间的逻辑存储单元。 表空间是段的逻辑容器。 每个表空间包含一个或多个数据文件，单个数据文件是且只能是一个表空间的一部分。
- 段(Segments)：分配用于存储用户对象（例如表或索引）的一组范围。段一般是数据库终端用户要处理的最小存储单位。分为数据段、索引段、临时段和回滚段。
- 盘区(Extents)：用于存储特定类型信息的逻辑连续数据块。
- 数据块(Data Blocks)：Oracle 数据库最小的存储单位，块的大小是数据库内给定表空间中特点数量的存储字节。Oracle 初始参数。DB_BLOCK_SIZE 指定默认的块大小，默认为 8KB。

体系结构

物理存储结构

- **数据文件**
  - **系统数据文件**（SYSTEM01.DBF和SYSAUX01.DBF）
  - **回滚数据文件**（UNDOTBS01.DBF）
  - **用户数据文件**（USERS01.DBF、TBSP_1.DBF）
  - **临时数据文件**（TEMP02.DBF）
- **控制文件**

最重要的文件，建库之前就要在初始化参数文件中把路径设置好

- **日志文件**
  - **重做日志文件**
    记录所有数据变化，提供恢复机制
  - **归档日志文件**
    重做日志的历史备份

其他文件：服务器参数文件、密码文件、警告文件（运行+错误）、跟踪文件

体系结构

□ 存储结构 —— 物理存储结构

物理结构由构成数据库的操作系统文件所决定。

每个Oracle数据库都由3种类型的文件组成：

数据文件、日志文件和控制文件。

- 数据文件 Oracle数据库有一个或多个物理的数据文件。数据库的数据文件包含全部数据库数据。逻辑数据物理地存储在数据文件中。
  - 每一个数据库有一个或多个物理的数据文件(data file)，一个数据文件仅与一个数据库直接相关；
  - 逻辑数据库对象的数据也存储在数据文件中；
  - 一旦建立，数据文件不能改变大小；
  - 数据文件中的数据在需要时可以读取并存储在内存中。

体系结构

□ 存储结构 —— 物理存储结构

- 日志文件　每个数据库有两个或多个日志文件组，日志文件组用于收集数据库日志。日志的主要功能是记录对数据所作的修改，所以对数据库作的全部修改记录在日志中。
  - 每一个数据库有两个或多个日志文件（redo log file），在系统或介质故障恢复数据库时使用；
  - 联机日志文件：用来循环记录数据库改变的操作系统文件；
  - 归档日志文件：为避免联机日志文件重写时丢失重复数据而对联机日志文件所做的备份。

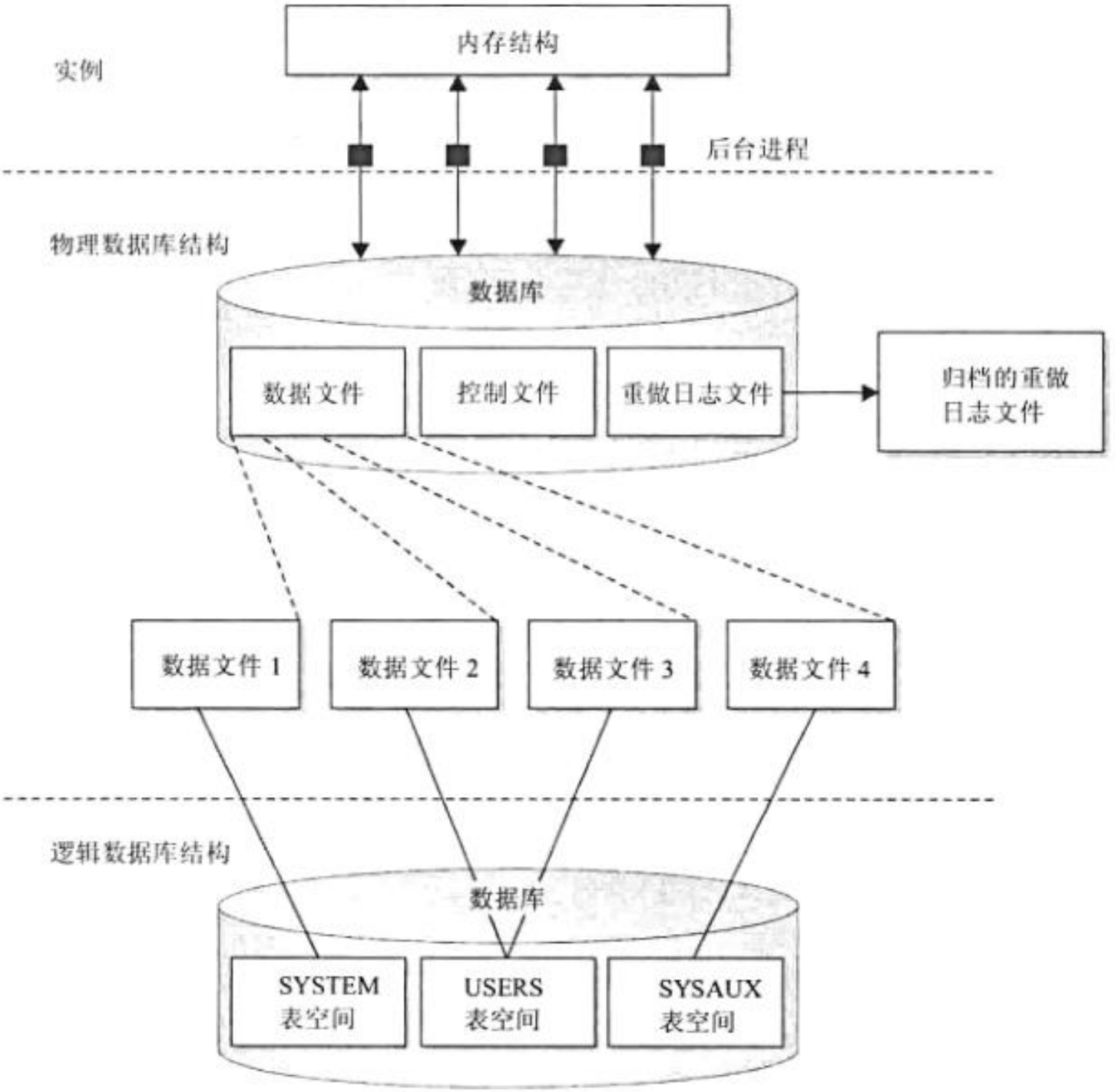
体系结构

□ 存储结构 —— 物理存储结构

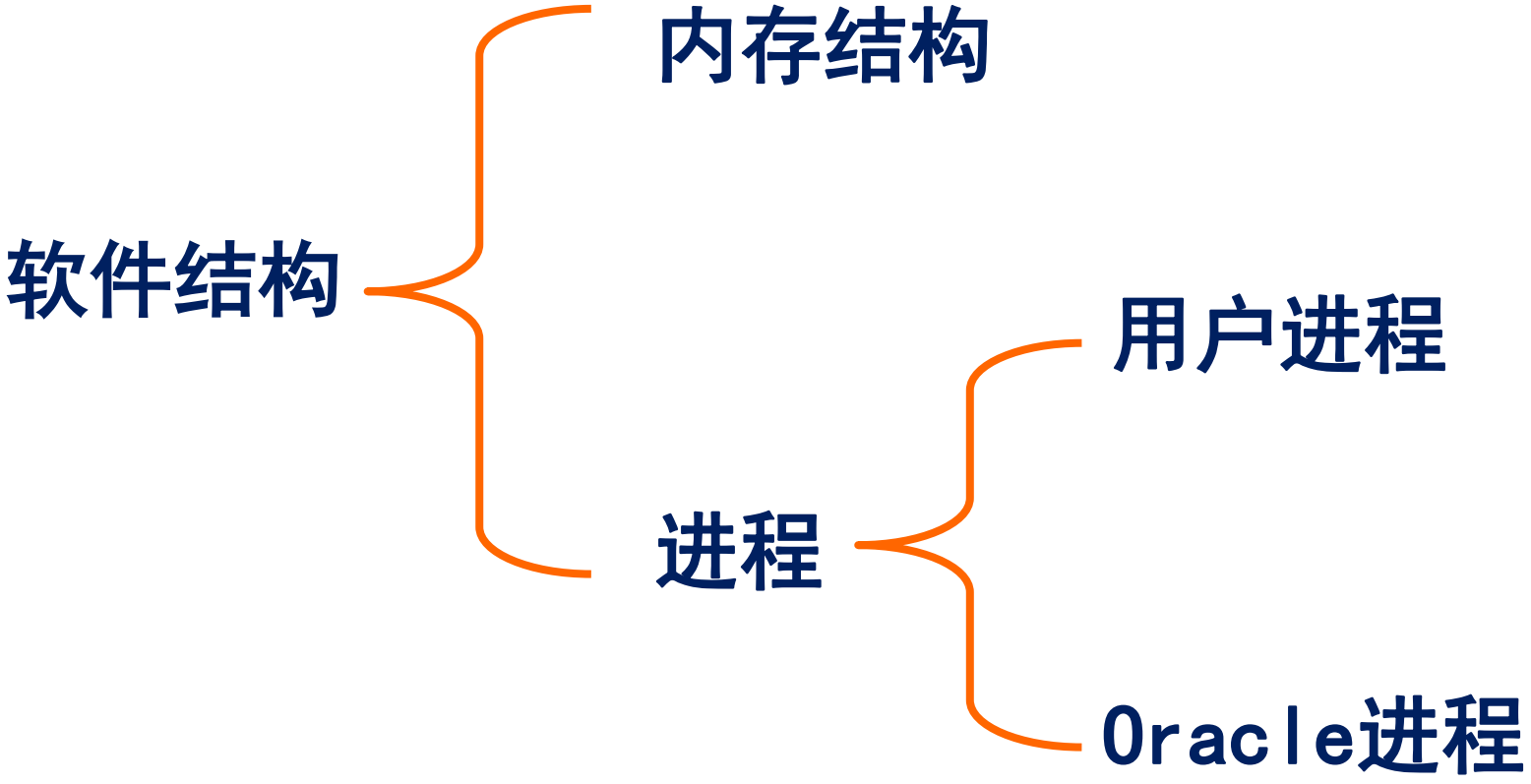
- 控制文件
  - 每一个数据库有一个控制文件(control file)，记录数据库的物理结构，包含下列信息类型： 数据库名、 数据文件和日志文件的名字和位置；数据库建立日期；
  - 当数据库的物理组成更改时，ORACLE 自动更改该数据库的控制文件。数据恢复时，也要使用控制文件。
- 参数文件
  - 记录了Oracle数据库的基本参数信息，主要包括数据库名、控制文件所在路径、进程等。
- 备份文件和密码文件

课程回顾

- 物理存储结构


实例
内存结构
后台进程
物理数据库结构
数据库
数据文件
控制文件
重做日志文件
归档的重做
日志文件
数据文件 1
数据文件 2
数据文件 3
数据文件 4
逻辑数据库结构
数据库
SYSTEM
表空间
USERS
表空间
SYSAUX
表空间

课程回顾



内存结构

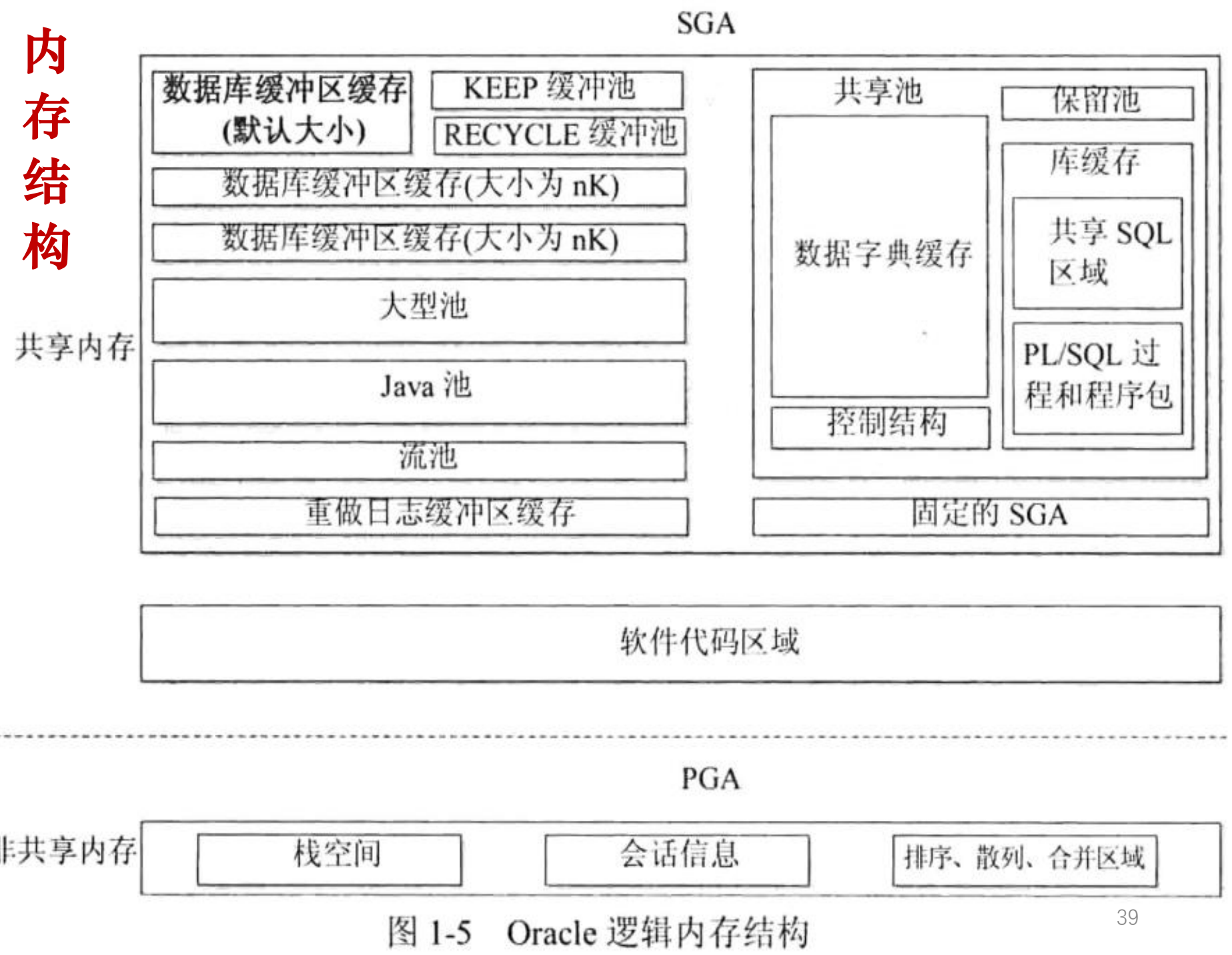
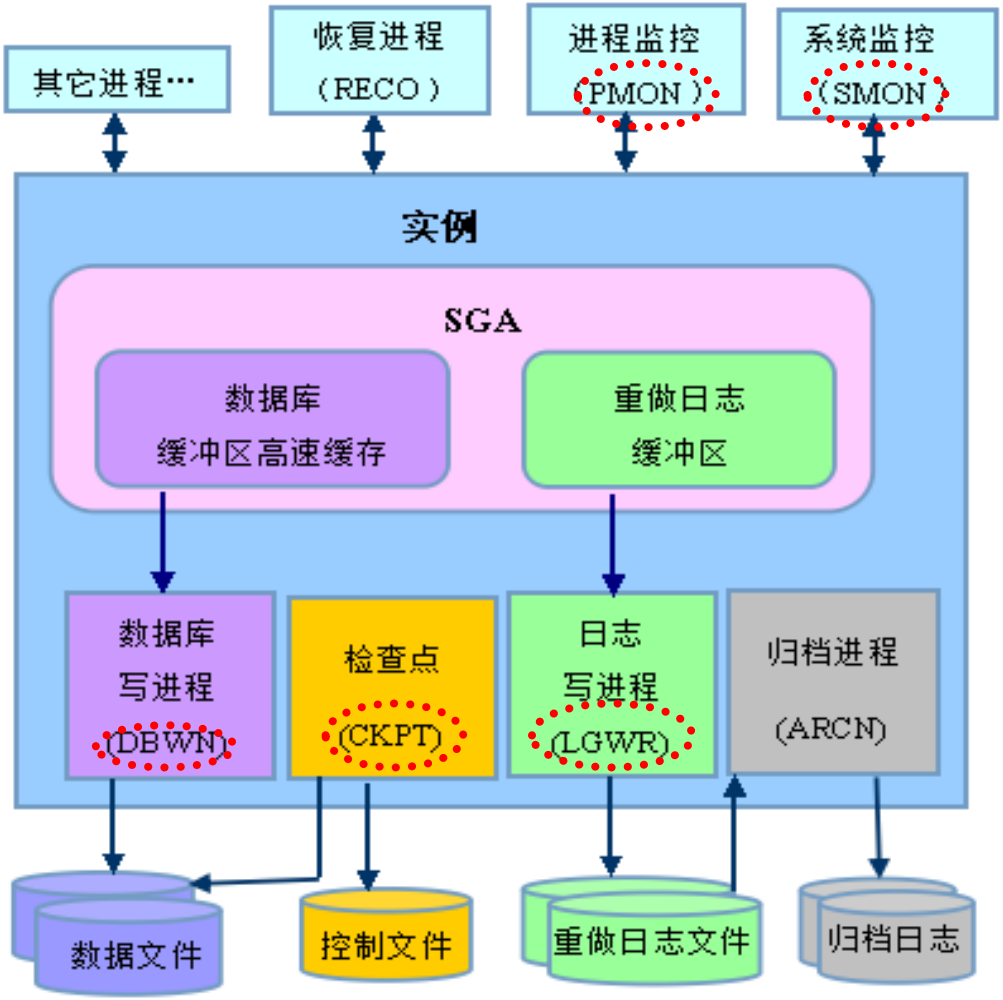
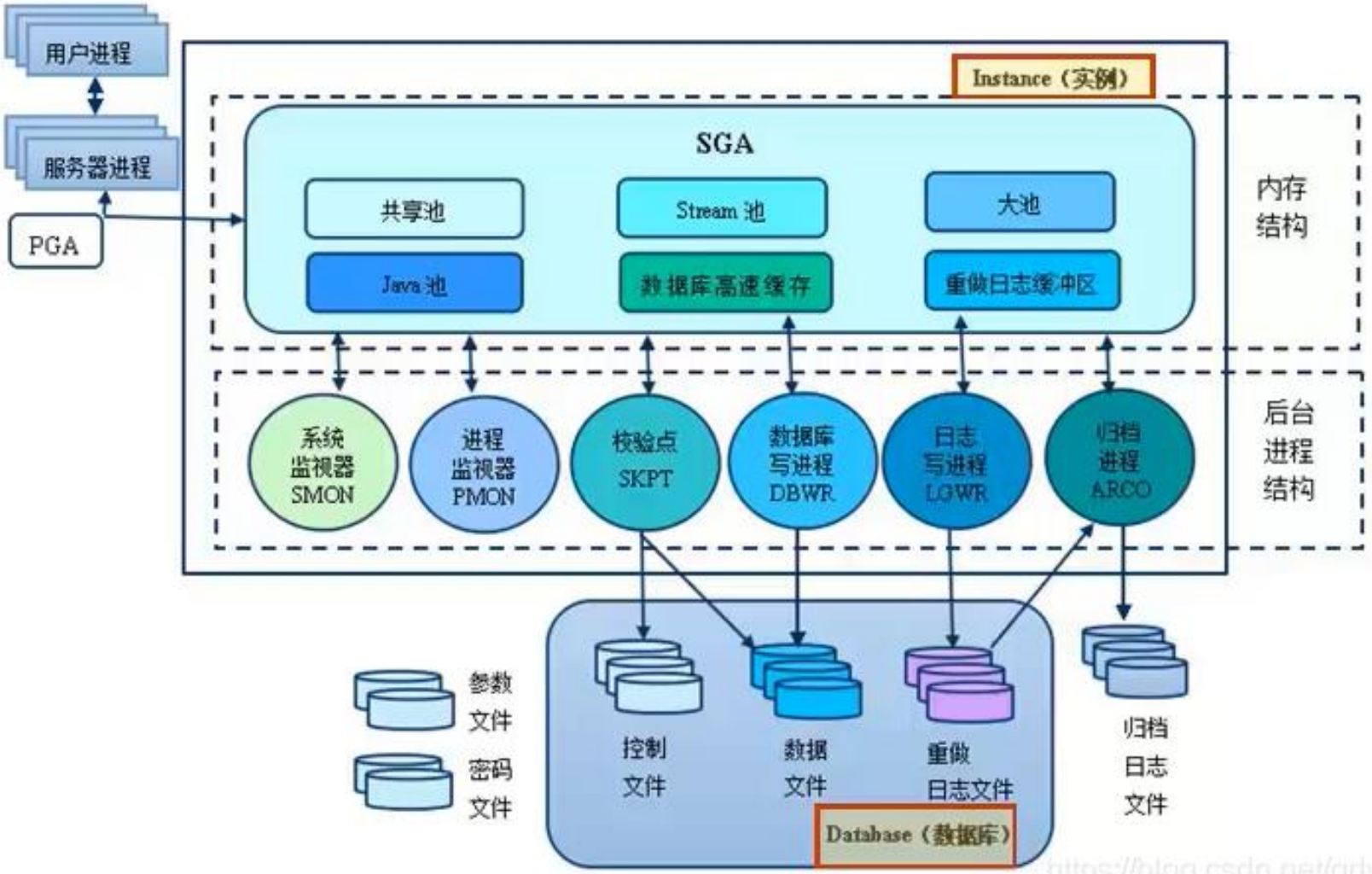


图 1-5　Oracle 逻辑内存结构

课程回顾

后台进程

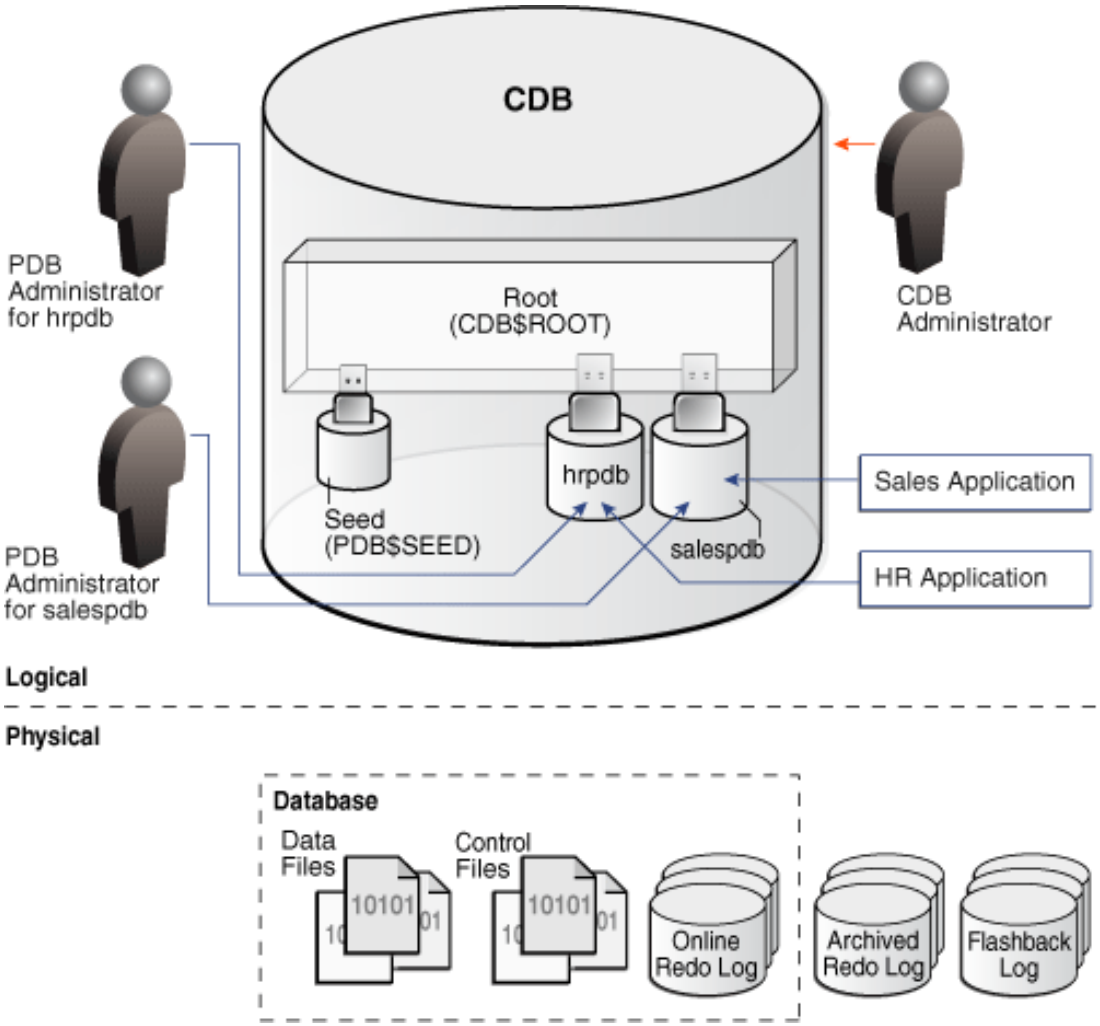

其它进程…
恢复进程
（RECO）
进程监控
（PMON）
系统监控
（SMON）
实例
SGA
数据库
缓冲区高速缓存
重做日志
缓冲区
数据库
写进程
(DBWN)
检查点
(CKPT)
日志
写进程
(LGWR)
归档进程
(ARCN)
数据文件
控制文件
重做日志文件
归档日志


用户进程
服务器进程
PGA
Instance（实例）
SGA
共享池
Stream 池
大池
Java 池
数据库高速缓存
重做日志缓冲区
内存
结构
后台
进程
结构
系统
监视器
SMON
进程
监视器
PMON
校验点
SKPT
数据库
写进程
DBWR
日志
写进程
LGWR
归档
进程
ARCO
参数
文件
密码
文件
控制
文件
数据
文件
重做
日志文件
归档
日志
文件
Database（数据库）

总体结构

课程回顾

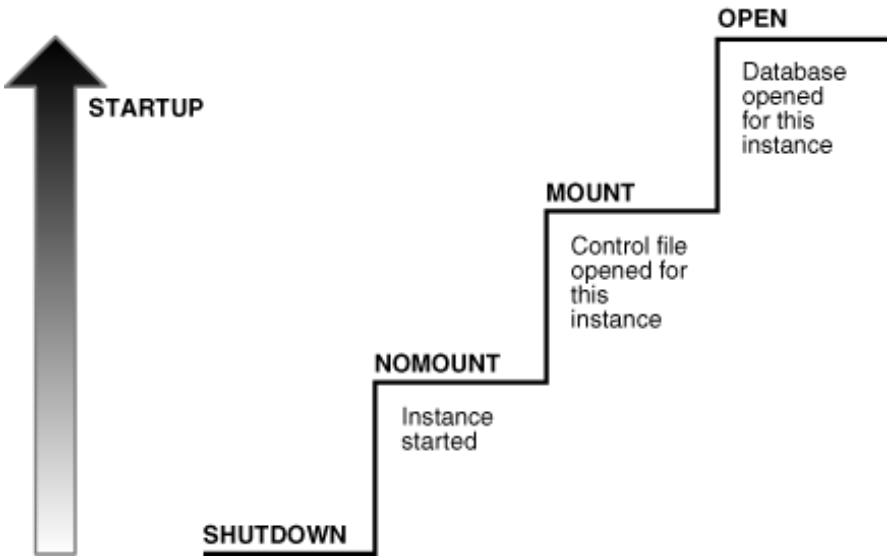
□多租户体系

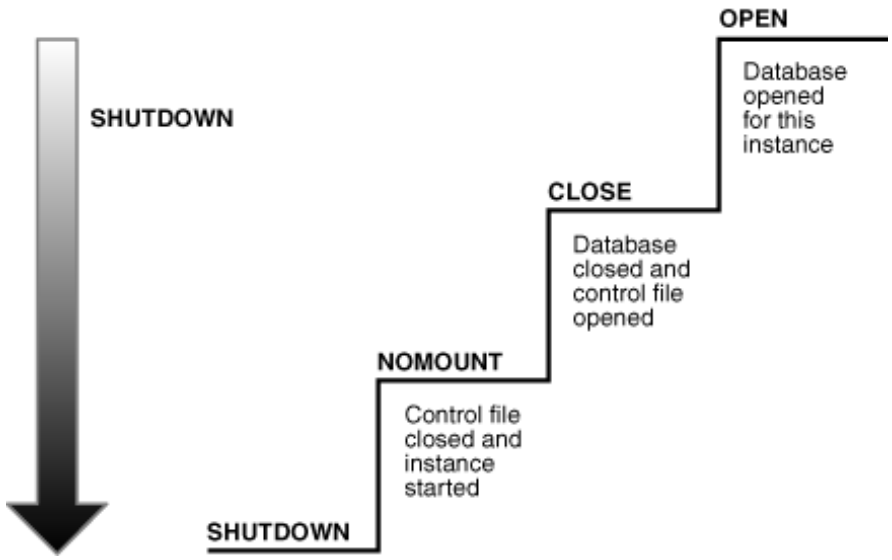


- CDB$ROOT
- PDB$SEED
- PDBs

课程回顾

□启动和关闭数据库





课程回顾

SQL 分类——根据SQL指令完成的数据库操作不同

- 数据定义：（Data Definition Language，DDL），用于完成对数据库对象（数据库、表、视图、索引等）的创建、修改和删除操作，分别对应 CREATE 、ALTER 和 DROP 三条语句 。DDL语句执行后自动提交。
- 数据查询：（Data Query Language，DQL)，用于对数据库中的各种数据对象进行查询。查询语句（SELECT）可以由许多子句组成，用以进行查询、统计、分组、排序等操作 。
- 数据操纵：（Data Manipulation Language，DML），用于改变数据库中的数据，数据操纵包括插入、删除和修改三种操作，分别对应 INSERT、DELETE 和 UPDATE 三条语句 。DCL语句执行后不会自动提交。
- 数据控制：（Data Control Language，DCL），用于对基表和视图的授权、完整性规则的描述以及事务开始和结束等控制语句，对应的语句有 GRANT、REVOKE、COMMIT和ROLLBACK等。DCL执行后自动提交。

课程回顾

PL/SQL编程

- PL/SQL概述
- 字符集和数据类型
- 程序结构和语句
- 过程和函数
- 游标
- 程序包和触发器

PL/SQL编程

PL/SQL概述

PL/SQL（Procedural Language / SQL）是 Oracle 在标准 SQL 上扩展后的程序设计语言，是 Oracle 数据库特有的、支持应用开发的语言。

PL/SQL程序块的结构：

[DECLARE]——声明部分，可选

声明一些变量、常量、用户定义的数据类型以及游标等

BEGIN——执行部分，必须

主程序体，可以加入各种合法语句

[EXCEPTION]——异常处理部分，可选

异常处理程序，当程序中出现错误时执行这 一部分

END ;——主程序体结束

数据类型

类型概览

- 系统预定义类型
  - 标量型：标量数据类型的变量只有一个值，且内部没有分量。包括数值型，字符型，日期/时间型和布尔型。
  - 复合型：含有能够被单独操作的内部组件，包括 record，table 和 cursor 型。
  - LOB 型：专门用于存储大对象的数据，包括大文本、图像图像、视频剪辑等。分为内部 LOB 和外部 LOB。
  - 引用类型：引用数据类型是 PL/SQL 程序语言特有的数据类型，是用来引用数据库当中的某一行或者某个字段作为数据类型的声明。
- 用户自定义子类型：

```
SUBTYPE subtype_name IS base_type[ (constraint) ] [ NOT NULL ];
```

数据类型

特殊数据类型

- **%TYPE类型**：使用 %TYPE 关键字可以声明一个与指定列名称相同的数据类型，它通常紧跟在指定列名的后面。`pename emp.ename%type;`
- **RECORD类型**：“记录类型”，使用该类型的变量可以存储由多个列值组成的一行数据。在声明记录类型变量之前，首先需要定义记录类型，然后才可以声明记录类型的变量。

```
type emp_type is record
(
    var_ename varchar2(20),
    var_job varchar2(20),
    var_sal number
);
```

- **%ROWTYPE 类型**：可以根据数据表中行的结构定义一种特殊的数据类型，用来存储从数据表中检索到的一行数据。

```
row Var_emp emp%rowtype;
```

程序结构和语句

结构控制语句

控制语句	意义说明
if...then	判断 if 正确则执行 then
if...then...else	判断 if 正确则执行 then，否则执行 else
if...then...elsif	嵌套式判断
case	有逻辑地从数值中做出选择
loop...exit...end	循环控制，用判断语句执行 exit
loop...exit when...end	同上，当 when 为真时执行 exit
while...loop...end	当 while 为真时循环
for...in...loop...end	已知循环次数的循环
goto	无条件转向控制

程序结构和语句

异常

语句执行过程中，会因为各种原因导致不能正常执行，并可能造成更大错误甚至系统崩溃，所以必须采取措施防止这种情况，PL/SQL 提供了异常（Exception）处理方法。

- **预定义异常**：Oracle 能够检测出并确定类型的异常，用户可以使用预定义的异常名称进行异常处理。

预定义异常的处理，语句写在程序块的 EXCEPTION 部分。

WHEN *exception_name* THEN 处理语句

- **自定义异常**：Oracle 无法检测出或确定类型的异常，需要用户在程序中自行定义，由 Oracle 根据定义引发。

预定义异常处理示例

【例】在 SCOTT 模式中，对于某个查询的异常处理。

匿名块已完成
返回记录超过一行

deptno=10改为100

匿名块已完成
无数据记录

```
SET serveroutput ON
DECLARE
  var_empno NUMBER;          --定义变量，存储雇员编号
  var_ename VARCHAR2(50);    --定义变量，存储雇员名称
BEGIN
  /* 检索部门编号为10的雇员信息 */
  SELECT empno,ename INTO var_empno,var_ename FROM emp WHERE
deptno=10;
EXCEPTION                                   --捕获异常
WHEN too_many_rows THEN  --若SELECT INTO语句的返回记录超一行
  dbms_output.put_line('返回记录超过一行');
WHEN no_data_found THEN --若SELECT INTO语句的返回记录为0行
  dbms_output.put_line('无数据记录');
END;
```

自定义异常处理示例

【例】在 SCOTT 模式中，自定义一个异常变量，在向 dept 表插入数据时，若 loc 字段的值为空，则引发异常。

```
DECLARE
  null_exception EXCEPTION;   --声明一个exception类型的异常变量
  dept_row dept%rowtype;      --声明rowtype类型的变量dept_row
BEGIN
  dept_row.deptno := 66;                  --给部门编号变量赋值
  dept_row.dname  := '公关部';            --给部门名称变量赋值
  /* 向dept表中插入一条记录 */
  INSERT INTO dept VALUES(dept_row.deptno,dept_row.dname,dept_row.loc);
  IF dept_row.loc IS NULL THEN  raise null_exception;  --引发null异常，转入
exception部分
  END IF;
EXCEPTION
WHEN null_exception THEN   --当raise引发的异常是null_exception时
  dbms_output.put_line('loc字段的值不许为null'); --输出异常提示信息
  ROLLBACK;                                      --回滚插入的数据记录
END;
```

匿名块已完成
loc字段的值不许为null

游标

游标分类

静态游标：在使用前，游标的定义已经完成，不能再更改。静态游标在打开时会将数据集存储在tempdb中，因此显示的数据与游标打开时的数据集保持一致，在游标打开以后对数据库的更新不会显示在游标中。

- 显示游标：用户声明和操作的一种游标，通常用于操作查询结果集。
- 隐式游标：不用明确声明的游标，执行 DML 语句、FOR语句、SELECT...INTO 语句会使用隐式游标。

动态游标：声明时没有设定，在打开时可以对其进行修改。动态游标在打开后会反映对数据库的更改。

- 自定义类型 ref
- 系统类型 SYS_REFCURSOR

游标

游标分类

静态游标：在使用前，游标的定义已经完成，不能再更改。静态游标在打开时会将数据集存储在tempdb中，因此显示的数据与游标打开时的数据集保持一致，在游标打开以后对数据库的更新不会显示在游标中。

- 显式游标：用户声明和操作的一种游标，通常用于操作查询结果集。
- 隐式游标：不用明确声明的游标，执行 DML 语句、SELECT...INTO 语句会使用隐式游标。

动态游标：声明时没有设定，在打开时可以对其进行修改。动态游标在打开后会反映对数据库的更改。可实现在程序间传递结果集的功能。

- 自定义类型 ref
- 系统类型 SYS_REFCURSOR

游标

游标属性

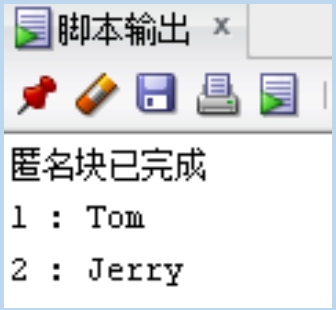
- 通过游标的属性可以获取 SQL 的执行结果以及游标的状态信息
- 游标属性只能用在PL/SQL的流程控制语句内，而不能用在SQL语句内

属性	返回值	描述
游标变量名%ISOPEN	布尔型	游标是否开启， true：开启，false：关闭。
游标变量名%FOUND	布尔型	游标发现数据，前一个 fetch 语句是否有值，true：有，false：没有。
游标变量名%NOTFOUND	布尔型	游标没有发现数据，常被用于退出循环，true：没有，false：有，null：空。
游标变量名%ROWCOUNT	布尔型	当前成功执行的数据行数（非总记录数）。

- 隐式游标的属性：*游标变量名* 部分为“SQL”，即四个属性分别为：SQL%ISOPEN、SQL%FOUND、SQL%NOTFOUND 和 SQL%ROWCOUNT。

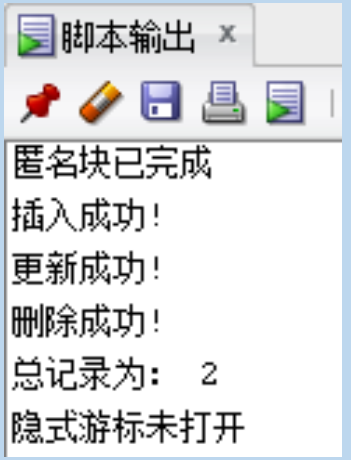
游标 □ 显式游标使用示例

```
DECLARE
  CURSOR cur_stu IS SELECT * FROM stu;   -- 步骤1: 声明游标
  v_stu  cur_stu%ROWTYPE; -- 定义一个变量存放游标指示的内容
BEGIN
  OPEN cur_stu;                                  -- 步骤2: 打开游标
  LOOP
    FETCH cur_stu INTO v_stu;                    -- 步骤3: 提取数据
    EXIT WHEN cur_stu%NOTFOUND;
     dbms_output.put_line(v_stu.s_id ||' : '||v_stu.s_xm);
  END LOOP;
  CLOSE cur_stu;                                 -- 步骤4: 关闭游标
EXCEPTION
  WHEN OTHERS THEN
    dbms_output.put_line(SQLCODE || ' : ' || SQLERRM);
    dbms_output.put_line(dbms_utility.format_error_backtrace);
END;
```



□ 隐式游标使用示例

```
DECLARE
  v_count NUMBER;
BEGIN
  INSERT INTO stu(s_id, s_xm) VALUES(3, 'Micky');
  IF SQL%FOUND THEN dbms_output.put_line('插入成功!');  END IF;
  UPDATE stu SET stu.s_xm = 'Donald' WHERE stu.s_id = 3;
  IF SQL%FOUND THEN dbms_output.put_line('更新成功!');  END IF;
  DELETE FROM stu t WHERE t.s_id = 3;
  IF SQL%FOUND THEN dbms_output.put_line('删除成功!');  END IF;
  SELECT COUNT(*) INTO v_count FROM stu t;
  IF SQL%FOUND THEN dbms_output.put_line('总记录为：  '||v_count);
END IF;
  IF SQL%ISOPEN THEN    --对于隐式游标而言永远为FALSE
    dbms_output.put_line('隐式游标已打开');
  ELSE
    dbms_output.put_line('隐式游标未打开');
  END IF;
EXCEPTION   略
END;
```



课程回顾

- 函数
- 存储过程
- 触发器

课程回顾

口事务的四个特性

- 原子性（Atomicity）：一个事务里面所有包含的 SQL 语句是一个执行整体，不可分割，要么都做，要么都不做。
- 一致性（Consistency）：是指事务应该正确的转换系统状态。事务开始时，数据库中的数据是一致的，事务结束时，数据库的数据也必须是一致的。
- 隔离性（Isolation）：是指数据库允许多个并发事务同时对数据进行读写和修改的能力，而不会导致数据不一致状态，即不同事务的执行不能互相干扰。
- 持久性（Durability）：事务一旦被提交，它对数据库中数据的改变就是永久性的。

课程回顾

口事务控制

事务的提交方式：

- 显示提交：使用 commit 命令使当前事务生效
- 自动提交：SQL Plus中执行 set autocommit on；语句，只针对当前连接有效。
- 隐式提交：
  - 正常执行完成 DDL/DCL 语句。
  - 正常退出 SQLPlus 或者 SQL Developer 等客户端。
  - 正常关闭数据库。

课程回顾

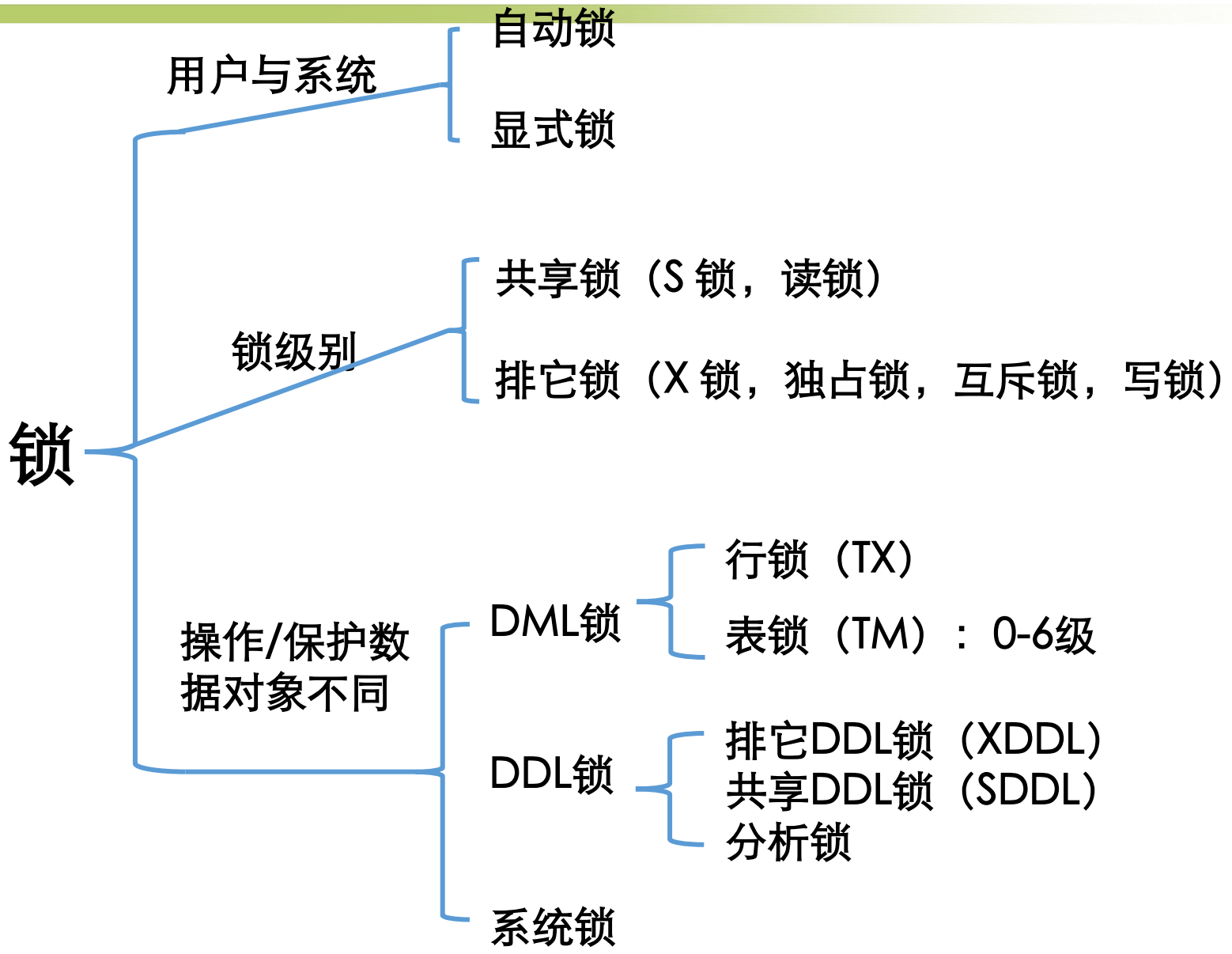
□数据的隔离级别

事务4个隔离等级与数据异常的关系

隔离级别	脏读	不可重复读	幻读
Read uncommitted（读未提交）	是	是	是
Read committed（读已提交）	否	是	是
Repeatable read（可重复读）	否	否	是
Serializable（串行读）	否	否	否

避免不可重复读需要锁行就行，避免幻影读则需要锁表

锁总结


锁
用户与系统
自动锁
显式锁
锁级别
共享锁（S 锁，读锁）
排它锁（X 锁，独占锁，互斥锁，写锁）
操作/保护数据对象不同
DML锁
行锁（TX）
表锁（TM）：0-6级
DDL锁
排它DDL锁（XDDL）
共享DDL锁（SDDL）
分析锁
系统锁

锁

□锁的类型——按操作和保护对象的不同

- DML 锁（DML Locks）：用于控制并发事务中的数据操纵，保证数据的一致性和完整性。

  例如，DML锁可防止两个客户从同一在线书店购买某一实体书所剩的最后一本。

- DDL 锁（Dictionary Locks)：用于保护数据库对象的结构，如表、索引等的结构定义。

- 系统锁（System Locks）：Oracle 数据库使用各种类型的系统锁，来保护数据库内部和内存结构。包括闩锁、互斥体、和内部锁。

锁

DML 锁

- 行锁（Row Locks，TX）：防止两个事务同时修改相同的数据行。当一个事务需要修改一行数据时，就需对此行数据加锁。
- 表锁（Table Locks，TM）：对 DML 执行时并发的 DDL 操作进行访问控制。例如防止在 DML 语句执行期间相关的表被移除。
- 当 Oracle 执行 DML 语句时，首先自动在所要操作的表上申请 TM 类型的锁。当 TM 锁获得后，再自动申请 TX 类型的锁，并将实际锁定的数据行的锁标志位进行置位。

课程回顾

DML 锁

模式	所模式别名	锁类型	对应SQL操作
0、1	None、NULL		select
2	行共享表锁 Row Share (RS)	TM	select for update、lock for update、lock in row share
3	行排他表锁 Row Exclusive(RX)	TM	insert、update、delete、lock in row exclusive mode
4	共享表锁 Share(S)	TM	lock in share mode
5	共享行排他表锁 Share Row Exclusive (SRX)	TM	lock in share row exclusive mode
6	排他锁 Exclusive(X)	TM/TX	lock in exclusive mode

课程回顾

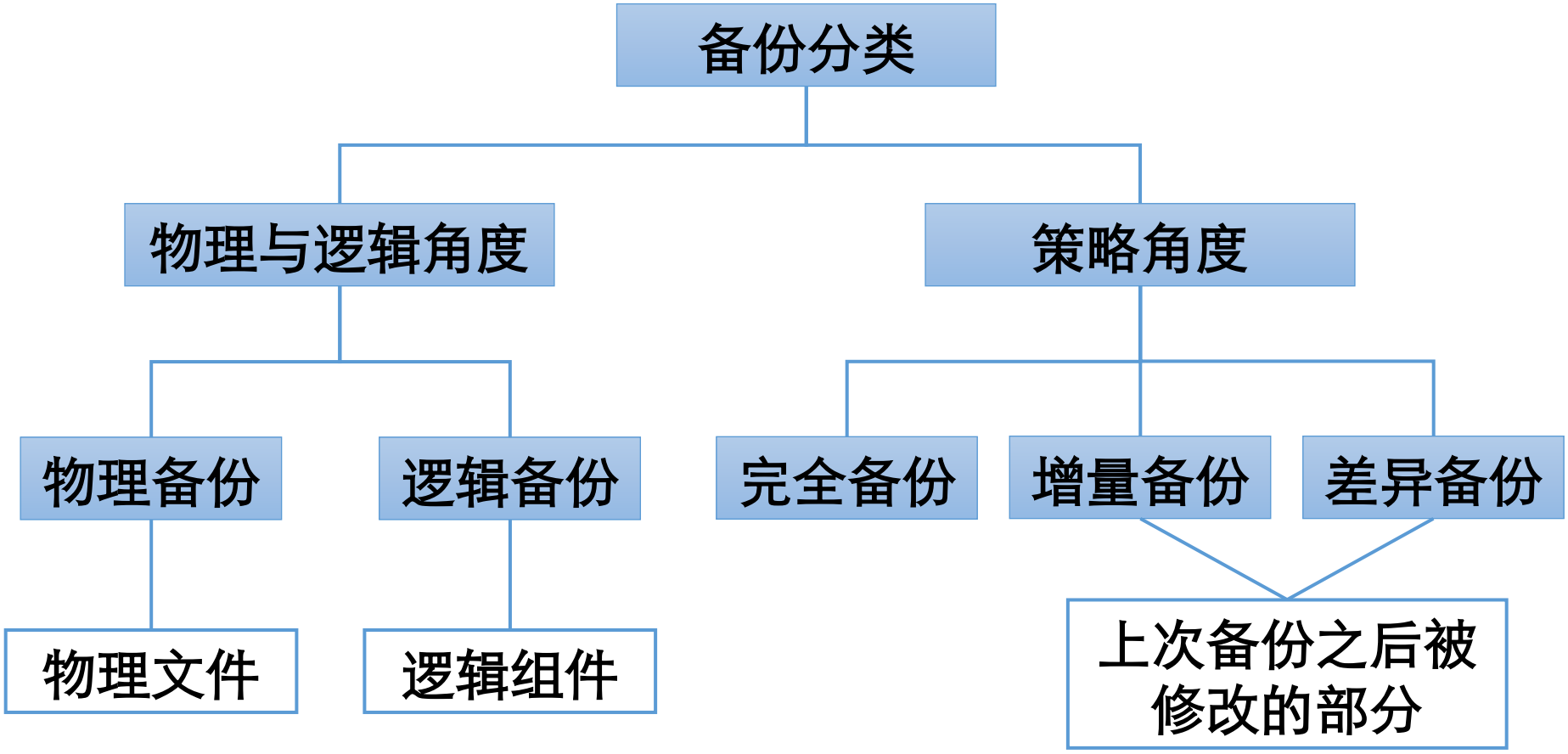
□安全控制策略

Oracle 数据库从以下几个方面进行安全管理：

- 用户账户和认证方式管理。
- 权限和角色管理：限制用户对数据库的访问和操作。
- 数据加密管理：保证网络传输的安全性。
- 表空间设置和配额：控制用户对数据库存储空间的使用。
- 用户资源限制：过概要文件设置，可以限制用户对数据库资源的使用。
- 数据库审计：监视和记录数据库中的活动。

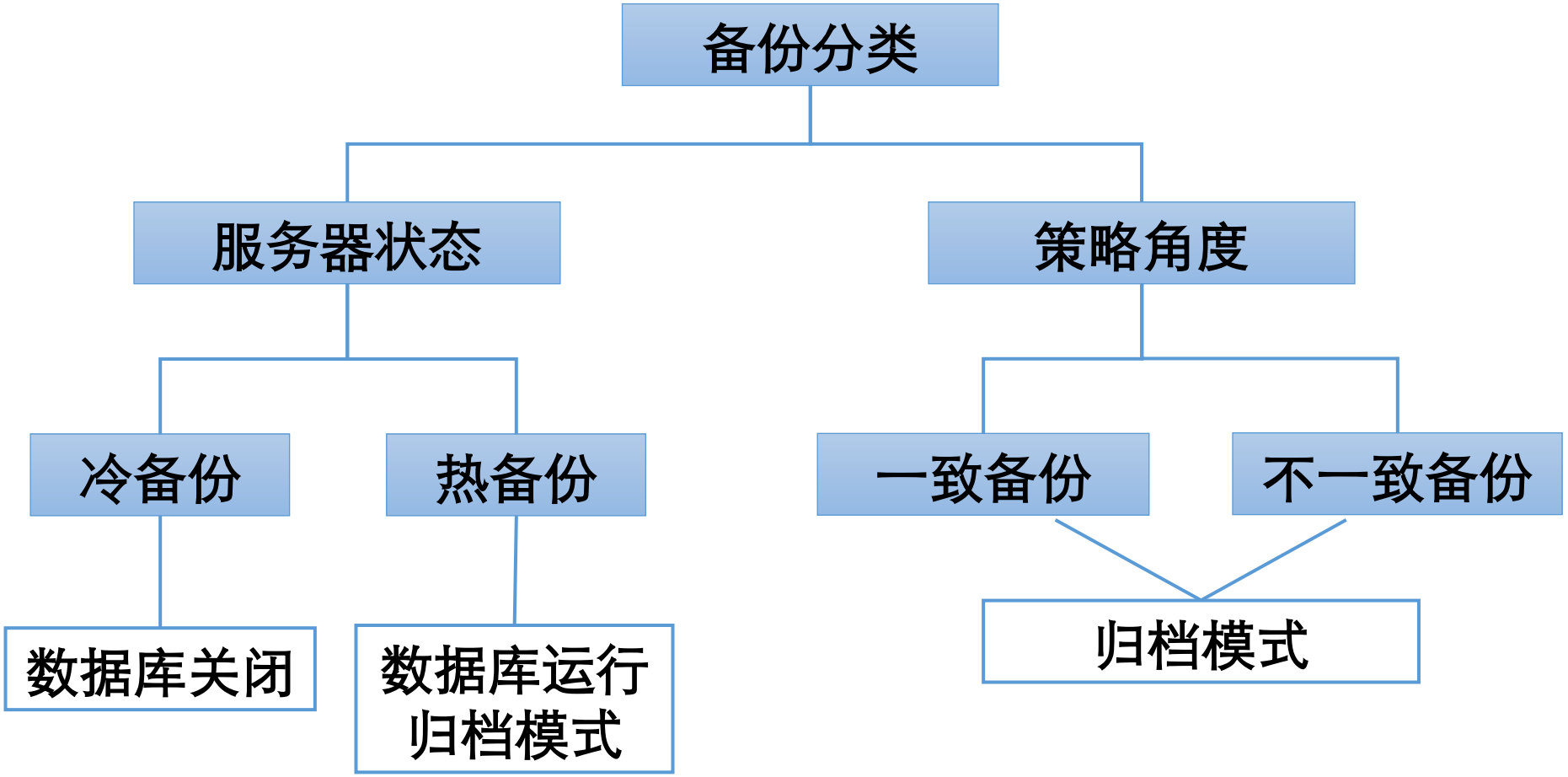
课程回顾

□备份概述

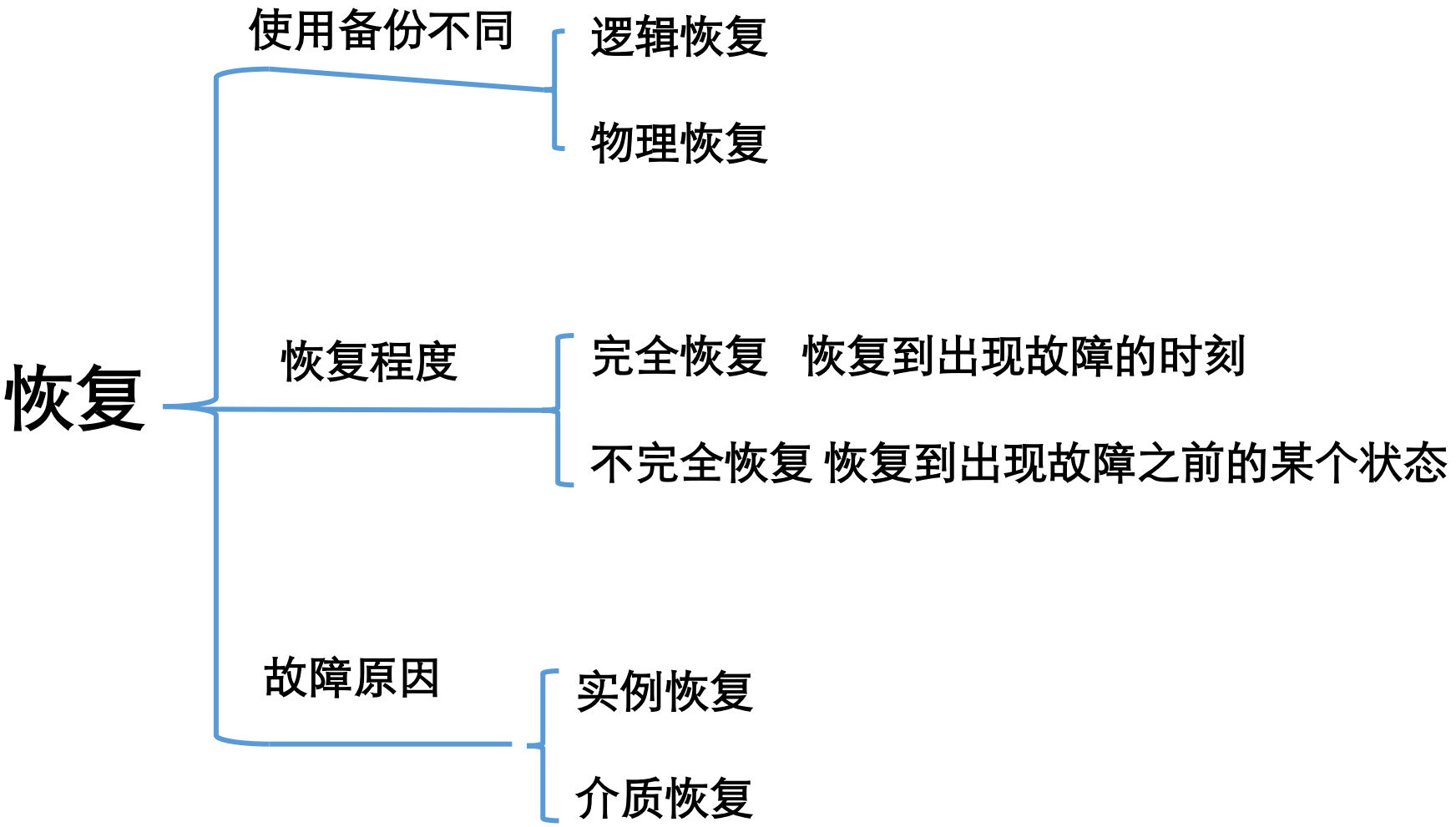

备份分类
物理与逻辑角度
策略角度
物理备份
逻辑备份
完全备份
增量备份
差异备份
物理文件
逻辑组件
上次备份之后被修改的部分

备份与恢复

备份概述



恢复


恢复
使用备份不同
逻辑恢复
物理恢复
恢复程度
完全恢复 恢复到出现故障的时刻
不完全恢复 恢复到出现故障之前的某个状态
故障原因
实例恢复
介质恢复

备份与恢复

RMAN 恢复——restore 和 recover

restore 是还原，文件级的恢复。就是物理文件还原。

recover 是恢复，数据级的恢复。逻辑上恢复，比如应用归档日志、重做日志，全部同步，保持一致。

简单讲，用restore先把备份文件拷贝到数据库目录下进行替换，再用recover经过一些处理，数据库就恢复正常了。