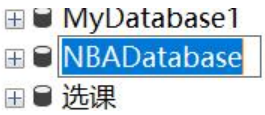
数据库实验-交互式 SQL

一、实验环境

使用交互式 SQL 实验中已经搭建好的实验环境，以及交互式 SQL 实验操作完成后的数据库 NBADatabase



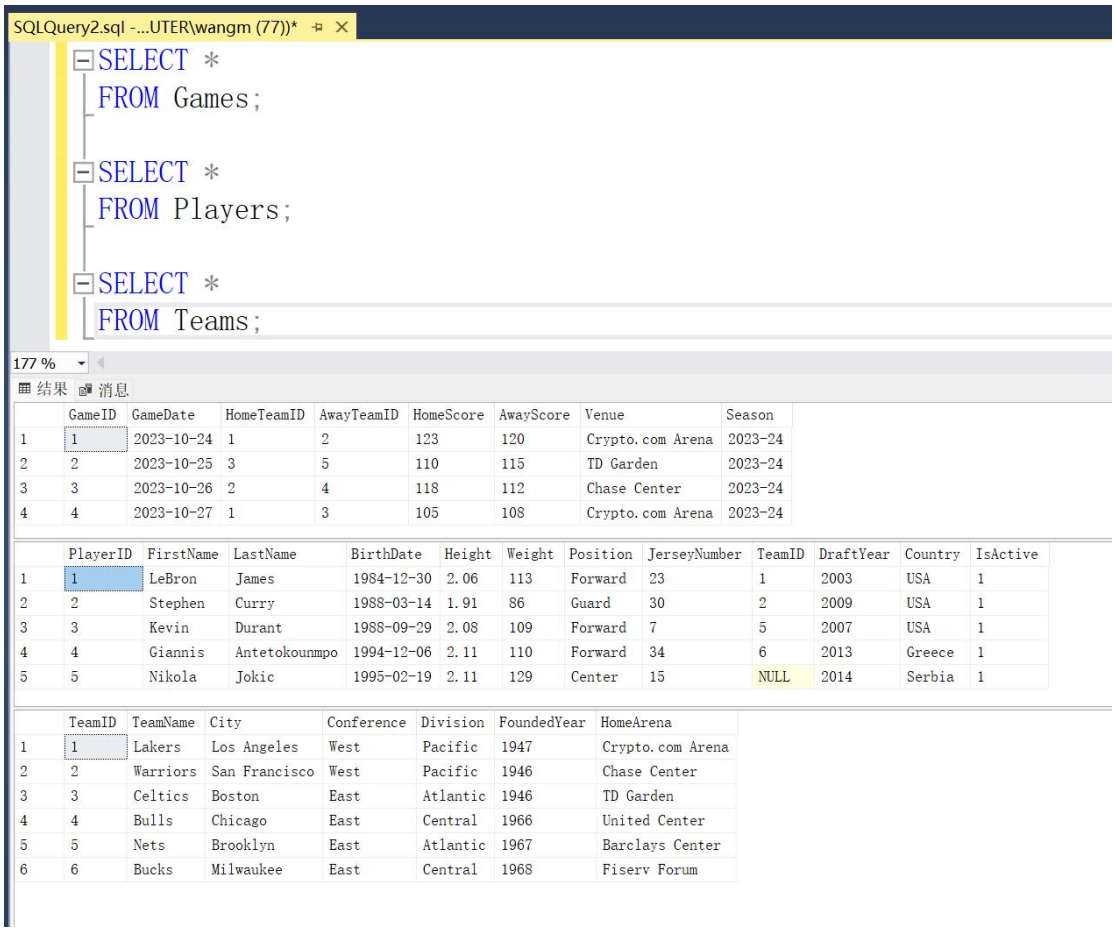
NBADatabase 数据库具有

Teams（球队表），

Players（球员表），

Games（比赛表）， 三个表

每个表的内容分别是



	GameID	GameDate	HomeTeamID	AwayTeamID	HomeScore	AwayScore	Venue	Season
1	1	2023-10-24	1	2	123	120	Crypto.com Arena	2023-24
2	2	2023-10-25	3	5	110	115	TD Garden	2023-24
3	3	2023-10-26	2	4	118	112	Chase Center	2023-24
4	4	2023-10-27	1	3	105	108	Crypto.com Arena	2023-24

	PlayerID	FirstName	LastName	BirthDate	Height	Weight	Position	JerseyNumber	TeamID	DraftYear	Country	IsActive
1	1	LeBron	James	1984-12-30	2.06	113	Forward	23	1	2003	USA	1
2	2	Stephen	Curry	1988-03-14	1.91	86	Guard	30	2	2009	USA	1
3	3	Kevin	Durant	1988-09-29	2.08	109	Forward	7	5	2007	USA	1
4	4	Giannis	Antetokounmpo	1994-12-06	2.11	110	Forward	34	6	2013	Greece	1
5	5	Nikola	Jokic	1995-02-19	2.11	129	Center	15	NULL	2014	Serbia	1

	TeamID	TeamName	City	Conference	Division	FoundedYear	HomeArena
1	1	Lakers	Los Angeles	West	Pacific	1947	Crypto.com Arena
2	2	Warriors	San Francisco	West	Pacific	1946	Chase Center
3	3	Celtics	Boston	East	Atlantic	1946	TD Garden
4	4	Bulls	Chicago	East	Central	1966	United Center
5	5	Nets	Brooklyn	East	Atlantic	1967	Barclays Center
6	6	Bucks	Milwaukee	East	Central	1968	Fiserv Forum

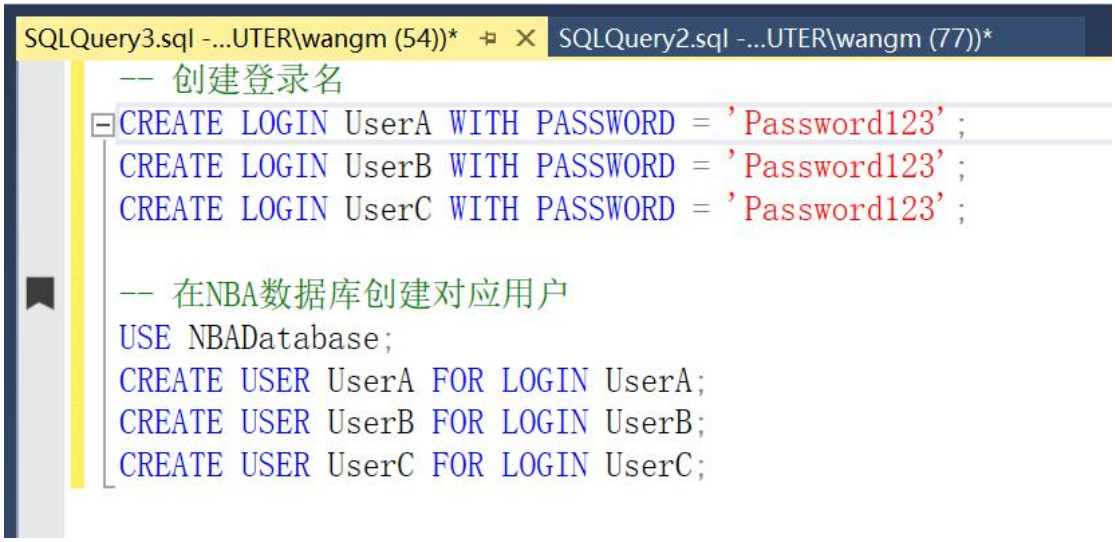
至此，实验环境已准备好。

二、实验内容与完成情况

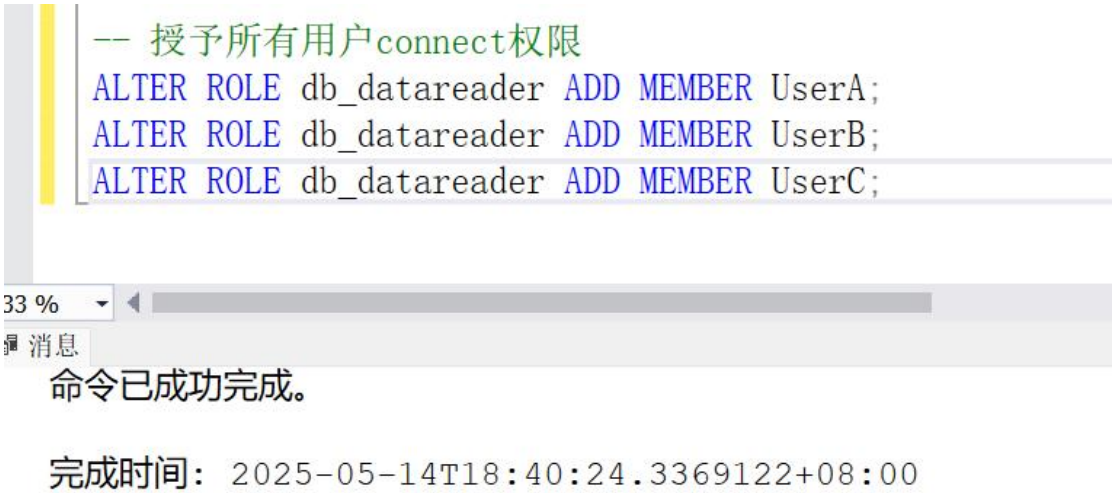
1）数据安全部分：

1、用户创建和权限选择

首先，新创建登录名，以及 NBADatabase 数据库用户：



然后，将所有用户的初始权限设置为 CONNECT：



2、用户间授权与回收

用例 1：UserA 对 Players 表的 SELECT 权限的授权与回收

授予 UserA Players 表的 SELECT 权限

```sql
--权限授予
GRANT SELECT ON Players TO UserA;
```

验证 UserA 的权限：

```sql
-- 1. 验证SELECT权限
EXECUTE AS USER = 'UserA';
SELECT * FROM Players; -- 测试是否可查询
REVERT;
```

133 %

结果 消息

	PlayerID	FirstName	LastName	BirthDate	Height	Weight	Position	JerseyNumber	Tear
1	1	LeBron	James	1984-12-30	2.06	113	Forward	23	1
2	2	Stephen	Curry	1988-03-14	1.91	86	Guard	30	2
3	3	Kevin	Durant	1988-09-29	2.08	109	Forward	7	5
4	4	Giannis	Antetokounmpo	1994-12-06	2.11	110	Forward	34	6
5	5	Nikola	Jokic	1995-02-19	2.11	129	Center	15	NUL

UserA 查询 Players 表成功，若使用 UserA 查询 Games 表，则失败报错：

```sql
-- 1. 验证SELECT权限
EXECUTE AS USER = 'UserA';
SELECT * FROM Games; -- 测试是否可查询
REVERT;
```

121 %

消息

```
消息 229，级别 14，状态 5，第 22 行
拒绝了对对象 'Games' (数据库 'NBADatabase'，架构 'dbo')的 SELECT 权限。

完成时间：2025-05-14T20:25:26.8933918+08:00
```

再回收 UserA 对 Players 表的 SELECT 权限

```sql
REVOKE SELECT ON Players FROM UserA;

-- 1. 验证SELECT权限
EXECUTE AS USER = 'UserA';
SELECT * FROM Players; -- 测试是否可查询
REVERT;
```

121 %

消息

```
消息 229，级别 14，状态 5，第 29 行
拒绝了对对象 'Players' (数据库 'NBADatabase'，架构 'dbo')的 SELECT 权限。

完成时间：2025-05-14T20:27:21.0876829+08:00
```

回收后再次查询会报错，回收成功

用例 2：UserB 对 Teams 表的 SELECT 和 UPDATE 权限的授权与回收

授予 UserB Teams 表的 SELECT 权限

```
GRANT SELECT, UPDATE ON Teams TO UserB;
```

121 %

消息

命令已成功完成。

完成时间：2025-05-14T20:30:10.6554276+08:00

验证 UserB 的权限：

```
-- 验证SELECT权限
EXECUTE AS USER = 'UserB';
SELECT * FROM Teams; -- 应该成功
REVERT;

-- 验证UPDATE权限
EXECUTE AS USER = 'UserB';
BEGIN TRY
    UPDATE Teams SET City = '洛杉矶' WHERE TeamID = 1;
    PRINT 'UPDATE成功';
    ROLLBACK; -- 回滚测试数据
END TRY
BEGIN CATCH
    PRINT 'UPDATE失败: ' + ERROR_MESSAGE();
END CATCH
REVERT;
```

1 %

结果 消息

TeamID	TeamName	City	Conference	Division	FoundedYear	HomeArena
1	Lakers	洛杉矶	West	Pacific	1947	Crypto.com Arena
2	Warriors	San Francisco	West	Pacific	1946	Chase Center
3	Celtics	Boston	East	Atlantic	1946	TD Garden
4	Bulls	Chicago	East	Central	1966	United Center
5	Nets	Brooklyn	East	Atlantic	1967	Barclays Center
6	Bucks	Milwaukee	East	Central	1968	Fiserv Forum

这个检测语句用 UserB 成功地把湖人队的 city 从英文改成了“洛杉矶”，并且 UserB 可以成功查询。

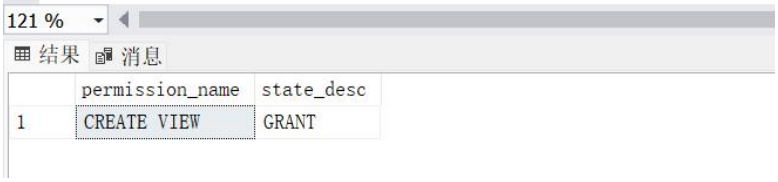
用例 3：UserC 创建视图权限的授权与回收

运行：

```
-- 授予 UserC 创建视图的权限
GRANT CREATE VIEW TO UserC;
```

验证：检查 UserC 的权限，发现 UserC 确实有创建视图权限

```
-- 查看UserC的权限
SELECT
    permission_name,
    state_desc
FROM sys.database_permissions
WHERE grantee_principal_id = USER_ID('UserC')
AND permission_name = 'CREATE VIEW';
```




```
-- 回收 UserC 创建视图的权限
REVOKE CREATE VIEW FROM UserC;
```

收回 UserC 创建视图的权限。

用例 4：UserA 对 Games 表的 INSERT 权限的授权与回收

```
-- 授予 UserA 对 Games 表的 INSERT 权限
GRANT INSERT ON Games TO UserA;
```

验证成功：UserA 成功创建了(5,'2023-11-01', 1, 2, 100, 98, '测试场馆','2023-24')的比赛实例。

```sql
-- 验证INSERT权限
EXECUTE AS USER = 'UserA';
BEGIN TRY
    INSERT INTO Games(GameID, GameDate,HomeTeamID, AwayTeamID, HomeScore, Aw
    VALUES (5,'2023-11-01', 1, 2, 100, 98, '测试场馆','2023-24');
    PRINT 'INSERT成功';
    SELECT * FROM Games WHERE Venue = '测试场馆'; -- 验证插入
    DELETE FROM Games WHERE Venue = '测试场馆'; -- 清理测试数据
END TRY
BEGIN CATCH
    PRINT 'INSERT失败: ' + ERROR_MESSAGE();
END CATCH
REVERT;
```

```
(1 行受影响)
INSERT成功
INSERT失败: 拒绝了对对象 'Games' (数据库 'NBADatabase', 架构 'dbo')的 SELEC

完成时间: 2025-05-14T20:50:38.9885510+08:00
```

```sql
-- 回收 UserA 对 Games 表的 INSERT 权限

REVOKE INSERT ON Games FROM UserA;
```

2）完整性部分：

1、实体完整性：

实做一个主键约束测试，在 Players 表中，再次插入 PlayerID 为 1 的实例，结果报错：“违反 PRIMARY KEY 约束，插入失败”。

```sql
-- 尝试插入重复PlayerID
INSERT INTO Players (PlayerID, FirstName, LastName)
VALUES (1, 'Test', 'Player');
```

```
消息 2627, 级别 14, 状态 1, 第 115 行
违反了 PRIMARY KEY 约束"PK__Players__4A4E74A80FAF3E5E"。不能在对象"dbo.Pla
语句已终止。

完成时间: 2025-05-14T20:59:13.2300497+08:00
```

2、参照完整性：

1）插入不存在的 TeamID 的球员

在 Players 表中，插入 TeamID 为 999 的实例，结果报错：“违反 FOREIGN KEY 约束，插入失败”。

2）更新 Teams 表中被引用的 TeamID

在 Teams 表中，更新被引用过的球队号‘1’，结果报错：“违反 FOREIGN KEY 约束，更新失败”。

3）删除被引用过的球队

在 Teams 表中，删除球队号为‘1’的球队，结果报错：“违反 FOREIGN KEY 约束，删除失败”。

4）插入不存在球队的比赛

在 Gamess 表中，插入球队号为‘100’和‘99’的球队，结果报错：“INSERT 语句与 FOREIGN KEY 约束‘FK__Games__HomeTeamI__5629CD9C’冲突”插入失败。

3、用户定义完整性：

在 Players 表中，首先增加限制球员身高在 1.5 到 2.5 之间的约束

```
ALTER TABLE Players
ADD CONSTRAINT CHK_PlayerHeight
CHECK (Height BETWEEN 1.5 AND 2.5); -- 限制身高在 1.5 米到 2.5 米之间
```

然后尝试插入身高不符合的球员，报错“与 CHECK 约束‘CHK_PlayerHeight’冲突”插入失败

4、CHECK 短语：

在 Games 表中，首先增加主客队不能相同的约束

```
-- 添加检查主客队不能相同的约束

ALTER TABLE Games

ADD CONSTRAINT CHK_TeamsDifferent

CHECK (HomeTeamID <> AwayTeamID);
```

尝试插入主客队相同的比赛，报错“与 CHECK 约束‘CHK_TeamsDifferent’冲突”插入失败

5、CONSTRAINT 子句：

在 Games 表中添加主客球队得分不能为负数的 CONSTRAINT 子句

```
-- 为 Games 表添加分数非负约束

ALTER TABLE Games

ADD CONSTRAINT CHK_ScoreNonNegative
```

```
CHECK (HomeScore >= 0 AND AwayScore >= 0);
```

尝试插入负分数的比赛，报错“与 CHECK 约束‘CHK_ScoreNonNegative’冲突”插入失败

三、出现的问题及解决方法

1、用户有没有被授予的权力

问题：数据安全实验里第一个用例是 UserA 对 Players 表的 SELECT 权限的授权与回收，只授予了 UserA 对 Players 表的 SELECT 权限，然后我使用

```
EXECUTE AS USER = 'UserA';
SELECT * FROM Games; -- 测试是否可查询
REVERT;
```

验证 UserA 不具有查询 Games 表的权限，但是查询成功了。

```sql
-- 1. 验证SELECT权限
EXECUTE AS USER = 'UserA';
SELECT * FROM Games; -- 测试是否可查询
REVERT;
```

结果 消息

GameID	GameDate	HomeTeamID	AwayTeamID	HomeScore	AwayScore	Venue	Season
1	2023-10-24	1	2	123	120	Crypto.com Arena	2023-24
2	2023-10-25	3	5	110	115	TD Garden	2023-24
3	2023-10-26	2	4	118	112	Chase Center	2023-24
4	2023-10-27	1	3	105	108	Crypto.com Arena	2023-24

调查原因：首先检查 UserA 直接对 Games 表的权限

```sql
-- 1. 查看UserA直接对Games表的权限
SELECT
    permission_name,
    state_desc
FROM sys.database_permissions
WHERE grantee_principal_id = USER_ID('UserA')
AND major_id = OBJECT_ID('Games');
```

结果 消息

permission_name	state_desc

发现 UserA 确实没有对 Games 表的任何权限

所以应该是其他地方授予的权限，检查 UserA 是不是 db_datareader 角色成员身份：

```sql
-- 检查UserA是否属于db_datareader角色
SELECT IS_ROLEMEMBER('db_datareader', 'UserA') AS IsDbDataReader;
```

结果 消息

	IsDbDataReader
1	1

结果显示 UserA 是 db_datareader 数据库角色成员，而 db_datareader 是 SQL Server 内置的数据库角色，该角色的成员自动拥有对数据库中所有表和视图的 SELECT 权限。

错误原因：UserA 是 db_datareader 数据库角色成员

解决方法：首先将 UserA 从 db_datareader 数据库角色成员中删除，然后再次验证 UserA 是不是 db_datareader 数据库角色成员

确定 UserA 不是 db_datareader 数据库角色成员了，问题解决；