

- 1、**幻想读**：事务 T1 读取一条指定 where 条件的语句，返回结果集。此时事务 T2 插入一行新记录，恰好满足 T1 的 where 条件。然后 T1 使用相同的条件再次查询，结果集中可以看到 T2 插入的记录，这条新纪录就是幻想。
- 2、**不可重复读取**：事务 T1 读取一行记录，紧接着事务 T2 修改了 T1 刚刚读取的记录，然后 T1 再次查询，发现与第一次读取的记录不同，这称为不可重复读。
- 3、**脏读**：事务 T1 更新了一行记录，还未提交所做的修改，这个 T2 读取了更新后的数据，然后 T1 执行回滚操作，取消刚才的修改，所以 T2 所读取的行就无效，也就是脏数据。

为了处理这些问题，SQL 标准定义了以下几种事务隔离级别

READ UNCOMMITTED 幻想读、不可重复读和脏读都允许。

READ COMMITTED 允许幻想读、不可重复读，不允许脏读

REPEATABLE READ 允许幻想读，不允许不可重复读和脏读

SERIALIZABLE 幻想读、不可重复读和脏读都不允许

Oracle 数据库支持 READ COMMITTED 和 SERIALIZABLE 这两种事务隔离级别。所以 Oracle 不支持脏读

SQL 标准所定义的默认事务隔离级别是 SERIALIZABLE，但是 Oracle 默认使用的是 READ COMMITTED

设置隔离级别使用 **SET TRANSACTION ISOLATION LEVEL** [READ UNCOMMITTED|READ COMMITTED|REPEATABLE READ|SERIALIZABLE]

下面是 oracle 设置 SERIALIZABLE 隔离级别一个示例：

The screenshot shows two side-by-side Oracle SQL Developer windows. The left window, representing transaction T1, has the isolation level set to READ COMMITTED. It executes the following SQL:

```
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;

select * from customers where LAST_NAME LIKE 'B%';

INSERT INTO
CUSTOMERS (CUSTOMER_ID, FIRST_NAME, LAST_NAME, DOB, PHONE)
VALUES (8, 'MA', 'BJACKAL', '', '600-342-900');

commit;

select * from customers where LAST_NAME LIKE 'B%';
```

The result table for T1 shows three rows:

	CUSTOMER_ID	FIRST_NAME	LAST_NAME	DOB
1	8	MA	BJACKAL	
2	4	Gail	Black	
3	6	jackal.ma	Brown	90/12/03

The right window, representing transaction T2, has the isolation level set to SERIALIZABLE. It executes the same query as T1:

```
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;

select * from customers where LAST_NAME LIKE 'B%';
```

The result table for T2 shows only two rows, missing the new record inserted by T1:

	CUSTOMER_ID	FIRST_NAME	LAST_NAME
1	4	Gail	Black
2	6	jackal.ma	Brown

左面是事务 T1，右面是事务 T2，因为 T2 级别为 SERIALIZABLE，所以即使事务 T1 在提交了数据之后，事务 T2 还是看不到 T1 提交的数据，幻想读和不可重复读都不允许了。

那如何能查看到 T1 新增的记录呢？上面 T1 和 T2 是并发执行，在 T1 执行 insert 的时候事务 T2 已经开始了，因为 T2 级别是 SERIALIZABLE，所以 T2 所查询的数据集是 T2 事务开始前数据库的数据。即事务 T1 在事务 T2 开始之后的 insert 和 update 操作的影响都不会影响事务 T2。现在重新开启一个事务 T3 就可以看到 T1 新增的记录了