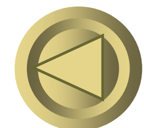
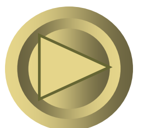
Chapter 6 The database Language SQL –as a tutorial

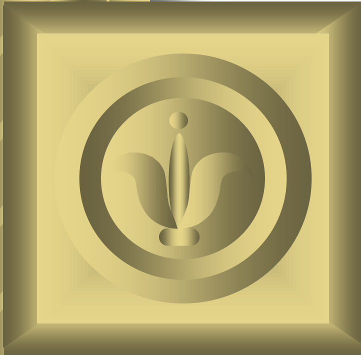
- **About SQL**

  SQL is a standard database language, adopted by many commercial systems.

  ANSI SQL, SQL-92 or SQL2, SQL99 or SQL3 extends SQL2 with object-relational features. SQL2003 is the collection of extensions to SQL3.

- How to query the database
- How to make modifications on database
- **Transactions in SQL**

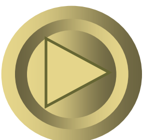


Transactions

What is transactions?

Why do we need transactions?

How to set transaction with different isolation level?



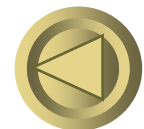
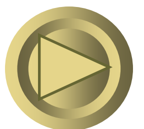
Why Transactions?

- **Concurrent database access**

  Execute *sequence of SQL statements* so they appear to be running in isolation

- **Resilience to system failures**

  Guarantee all-or-nothing execution, regardless of failures

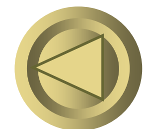
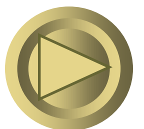



Concurrent Control

- **Accessed by many users or processes at the same time.**
  - **Both queries and modifications.**

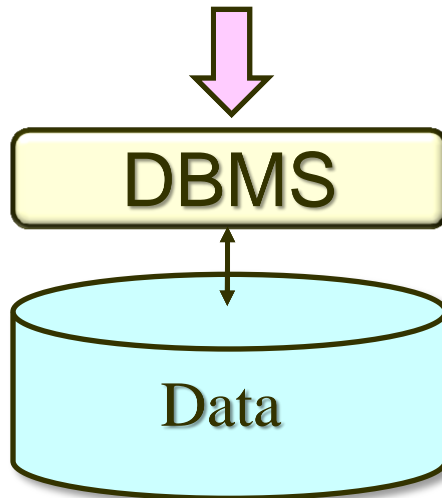
<u>Serializability</u>

Operations may be interleaved, but execution must be equivalent to *some* sequential (serial) order of all transactions

Resilience to system failures

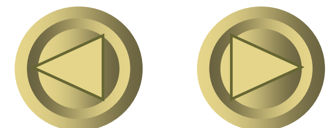
- **Failures may happen at any time.**
- **All or nothing done, never half done.**



Transfer money from one account into another account.

Update accounts set balance = balance - 1000 where accounts.number=123;

Update accounts set balance =balance +1000 where account.number= 456;

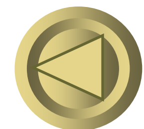
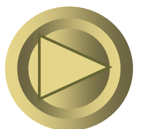


Solution for both concurrency and failures

Transactions

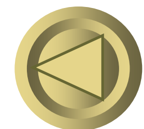
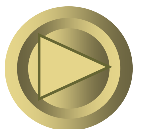
A transaction is a sequence of one or more SQL operations treated as a unit

- Transactions appear to run <u>in isolation</u>
- If the system fails, each transaction's changes are reflected <u>either entirely or not at all.</u>

Example: Interacting Processes

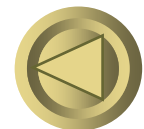
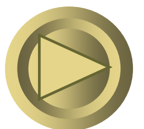
- Assume the usual Sells(bar,beer,price) relation, and suppose that Joe's Bar sells only Bud for $2.50 and Miller for $3.00.
- Sally is querying Sells for the highest and lowest price Joe charges.
- Joe decides to stop selling Bud and Miller, but to sell only Heineken at $3.50.

Sally's Program

- Sally executes the following two SQL statements called (min) and (max) to help us remember what they do.

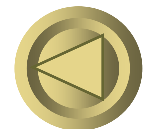
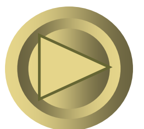
```
(max)SELECT MAX(price) FROM Sells
              WHERE bar = 'Joe''s Bar';
(min)SELECT MIN(price) FROM Sells
              WHERE bar = 'Joe''s Bar';
```

Joe's Program

- At about the same time, Joe executes the following steps: (del) and (ins).

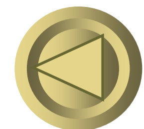
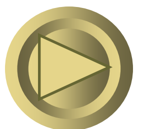
(del) DELETE FROM Sells
WHERE bar = 'Joe''s Bar';
(ins) INSERT INTO Sells
VALUES('Joe''s Bar', 'Heineken', 3.50);

Interleaving of Statements

- Sally: (max) before (min)
- Joes: (del) before (ins)
- Concurrent running:

1. (max) (del) (min) (ins)
2. (max) (del) (ins) (min)
3. (del)(max)(ins)(min)
4. …
5. (max)(min)(del)(ins)
6. (del)(ins)(max)(min)

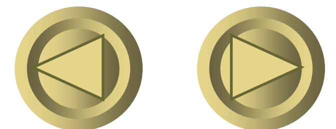
Example: Strange Interleaving

- **Suppose the steps execute in the order (max)(del)(ins)(min).**

Joe's Prices:

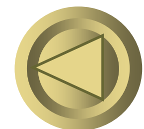
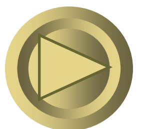
Joe's Prices:	{2.50,3.00}	{2.50,3.00}		{3.50}
Statement:	(max)	(del)	(ins)	(min)
Result:	3.00			3.50

- **Sally sees MAX < MIN!**



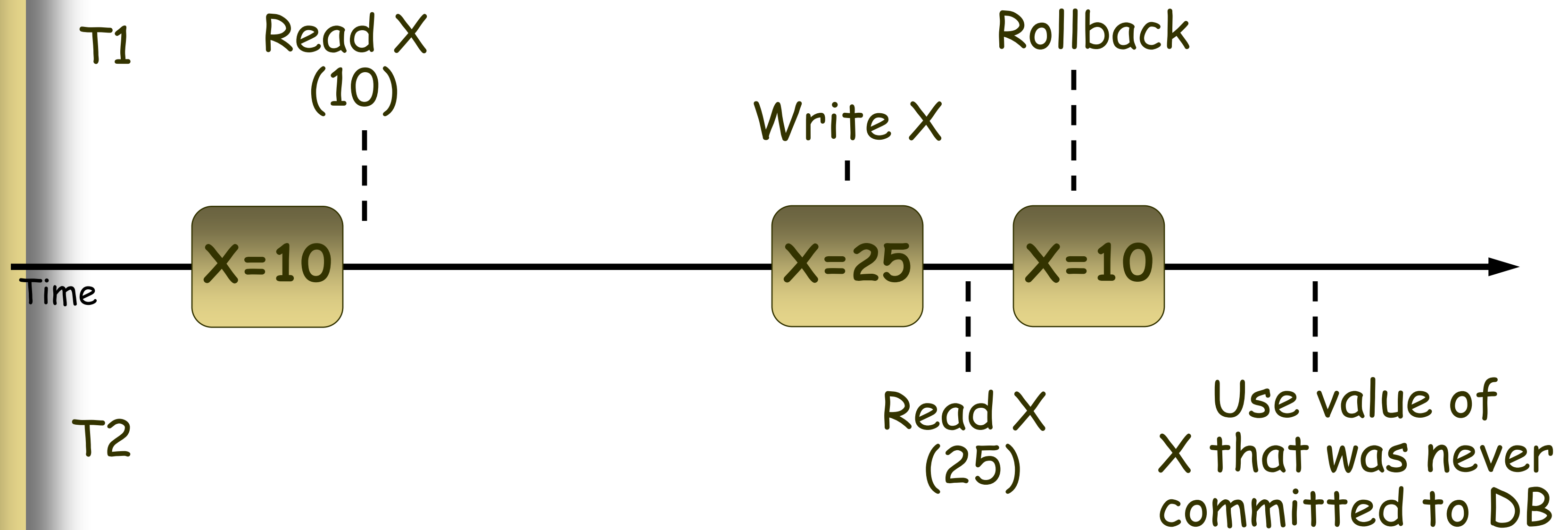
Another Problem: Rollback

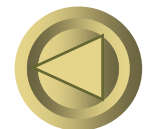
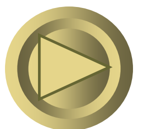
- **Suppose Joe executes (del)(ins), and then issues a ROLLBACK statement.**
- **If Sally executes her statements after (ins) but before the rollback, she sees a value, 3.50, that never existed in the database.**

Summarize of problems caused by multiple users accessing (1)

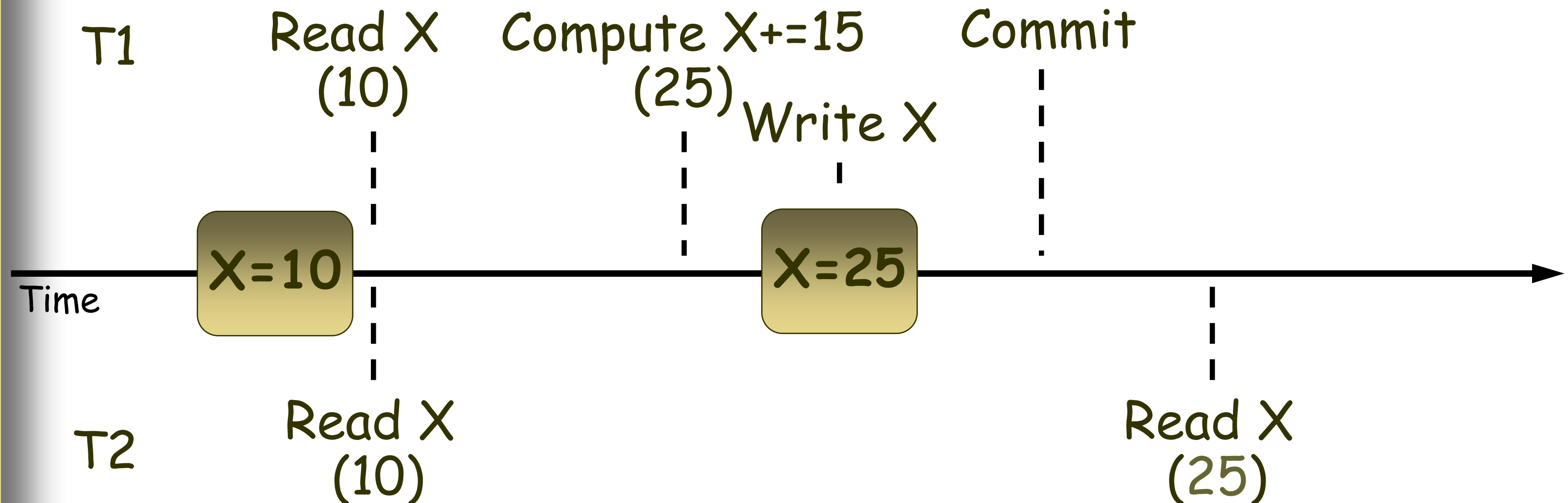
- **Dirty read**

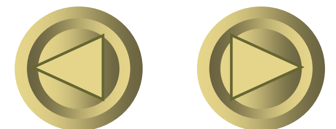


Summarize of problems caused by multiple users accessing (2)

- **Non-Repeatable Read**





Summarize of problems caused by multiple users accessing (3)

- **The "Phantom" Problem**

Jones,6

T1

Insert

Smith,4

Brewer,7

Time

Jones,6

Smith,4

Brewer,7

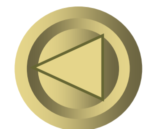
T2

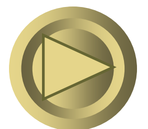
Select count (*)
where rank > 3

2 rows returned

Select count (*)
where rank > 3

3 rows returned





Solutions: Transactions

- **SQL standard:** ***i.e.***

Transaction begins automatically on first SQL statement

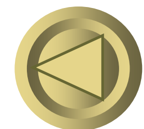
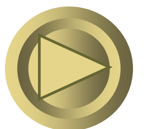
  - On “`commit`” transaction ends and new one begins.
  - Current transaction ends on session termination.
  - “`Autocommit`” turns each statement into transaction.

- **or explicit programmer control:**

  Begin Transaction

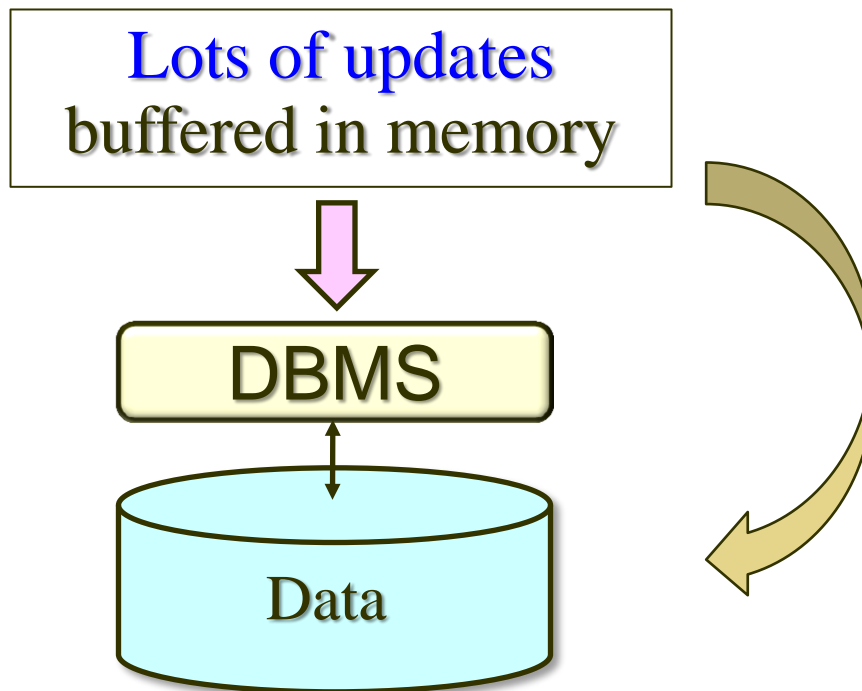
  …

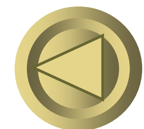
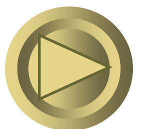
  End Transaction

Result of Transaction: COMMIT

- **The SQL statement COMMIT causes a transaction to complete.**
  - **It's database modifications are now permanent in the database.**



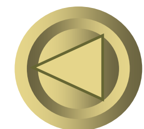
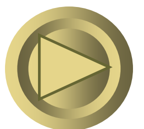



Result of Transaction: ROLLBACK

- The SQL statement ROLLBACK also causes the transaction to end, but by *aborting*.
  - **No effects on the database.**
- Application issued:

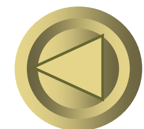
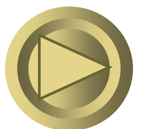
```
Begin Transaction;
<get input from user>
SQL commands based on input
<confirm results with user>
If ans='ok' Then Commit; Else Rollback;
```

- System-generated rollbacks (e.g. division by 0).

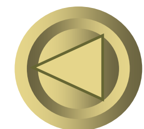
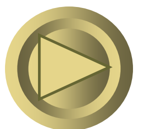
ACID Transactions

- ***ACID transactions*** **are:**
  - ***Atomic*** **: Whole transaction or none is done.**
  - ***Consistent*** **: Database constraints preserved.**
  - ***Isolated*** **: It appears to the user as if only one process executes at a time.**
  - ***Durable*** **: Effects of a process survive a crash.**

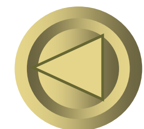
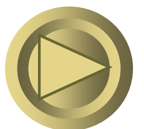
Consistency and isolation

- **Application** **defines consistency.**
- **Application** **requires isolation to achieve consistent results, there are** four ***isolation levels.***
- **Locking** **typically used to achieve isolation.**

Isolation Levels

- SQL defines four *isolation levels* = choices about what interactions are allowed by transactions that execute at about the same time.
- Only one level ("serializable") = ACID transactions.
- Each DBMS implements transactions in its own way.

Choosing the Isolation Level

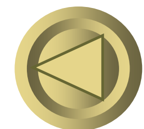
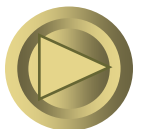
- **Within a transaction, we can say:**

SET TRANSACTION ISOLATION LEVEL *X*

where *X* =

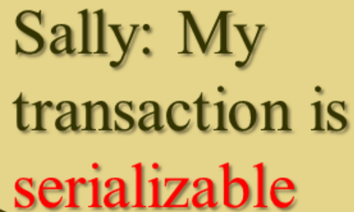
1. **SERIALIZABLE** ⇒ Overhead ⇒ Reduction in concurrency
2. **REPEATABLE READ**
3. **READ COMMITTED**
4. **READ UNCOMMITTED**

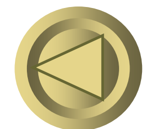
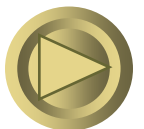
(Items 2–4:) ↓ Overhead ↑ Concurrency ↓ Consistency Guarantees

Isolation Level Is Personal Choice

- **Your choice, e.g., run serializable, affects only how *you* see the database, not how others see it.**



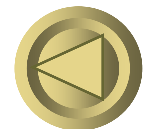
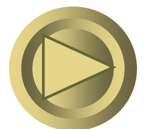
Serializable Transactions

- **Sally runs with isolation level SERIALIZABLE, then she will see the database either before or after Joe runs, but not in the middle.**

Set transaction isolation level serializable (default)

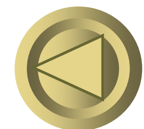
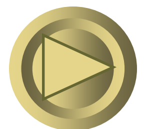
SELECT MAX(price) FROM Sells WHERE bar = 'Joe''s Bar';

SELECT MIN(price) FROM Sells WHERE bar = 'Joe''s Bar';

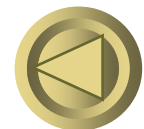
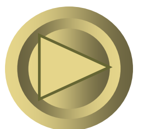
Read-Commited Transactions

- Sally runs with isolation level READ COMMITTED, then she can see only committed data, but not necessarily the same data each time.
- Example: Under READ COMMITTED, the interleaving (max)(del)(ins)(min) is allowed, as long as Joe commits.
  - Sally sees MAX < MIN.

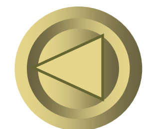
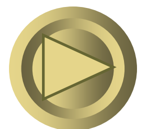
Repeatable-Read Transactions

- **Requirement is like read-committed, plus: if data is read again, then everything seen the first time will be seen the second time.**
  - **But the second and subsequent reads may see *more* tuples as well.**

Example: Repeatable Read

- **Suppose Sally runs under REPEATABLE READ, and the order of execution is (max)(del)(ins)(min).**
  - **(max) sees prices 2.50 and 3.00.**
  - **(min) can see 3.50, but must also see 2.50 and 3.00, because they were seen on the earlier read by (max).**

Example: Repeatable Read (cont.)

Sally:

Set transaction isolation level Repeatable read;

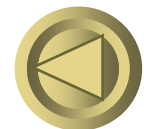
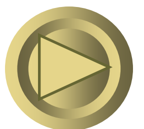
SELECT avg(price) FROM Sells;

SELECT avg(price) FROM Sells;

Joe:

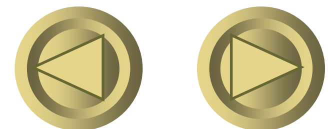
Insert into Sells values [100 tuples];

What are the result of Sally's query?

Read Uncommitted

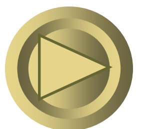
- A transaction running under READ UNCOMMITTED can see data in the database, even if it was written by a transaction that has not committed (and may never).
- Example: If Sally runs under READ UNCOMMITTED, she could see a price 3.50 even if Joe later aborts.



From weakest to strongest and the read behaviors they permit:

isolation level	dirty reads	nonrepeatable reads	phantoms
READ UNCOMMITTED	Y	Y	Y
READ COMMITTED	N	Y	Y
REPEATABLE READ	N	N	Y
SERIALIZABLE	N	N	N

- True isolation is expensive in terms of concurrency
  - Many systems allow application to choose the phenomena they will live with
  - Trade off between correctness and concurrency



Read only transactions

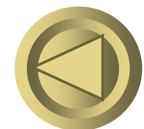
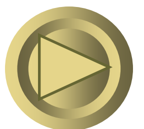
- **Help system optimize performance**
- **Independent of isolation level**

Set transaction read only;

Set transaction isolation level Repeatable read;

SELECT avg(price) FROM Sells;

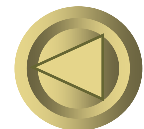
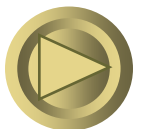
SELECT avg(price) FROM Sells;

Homework

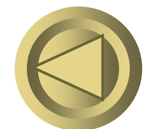
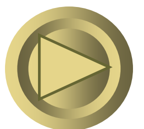
- Exercise 6.2.2 e)
- Exercise 6.3.1 c)
- Exercise 6.4.6 i)
- Exercise 6.6.4

Upload your homework until April.7

Summary

- SQL: The language is the principal query language for relational database systems. (SQL2, SQL3)
- Select-From-Where Queries
- Subqueries: The operators EXISTS, IN,ALL and ANY may be used to express boolean-valued conditions about the relations that are the result of a subquery
- Set Operations on Relations: UNION, INTERSECT, EXCEPT

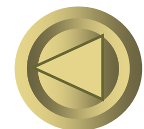
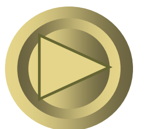



Summary(cont.)

- The bag model for SQL, DISTINCT elimination of duplicate tuples; ALL allows the result to be a bag.
- Aggregations: SUM,AVG,MIN,MAX,COUNT

GROUP BY, HAVING

- Modification Statements: INSERT, DELETE, UPDATE

SUMMARY(cont.)

- Transactions: ACID
- Isolation levels :

1. Serializable: the transaction must appear to run either completely before or completely after each other transaction
2. Repeatable read: every tuple read in response to a query will reappear if the query is repeated.
3. read-committed: only tuples written by transactions that have already committed may be seen by the transaction.
4. Read-uncommitted: no constraint.