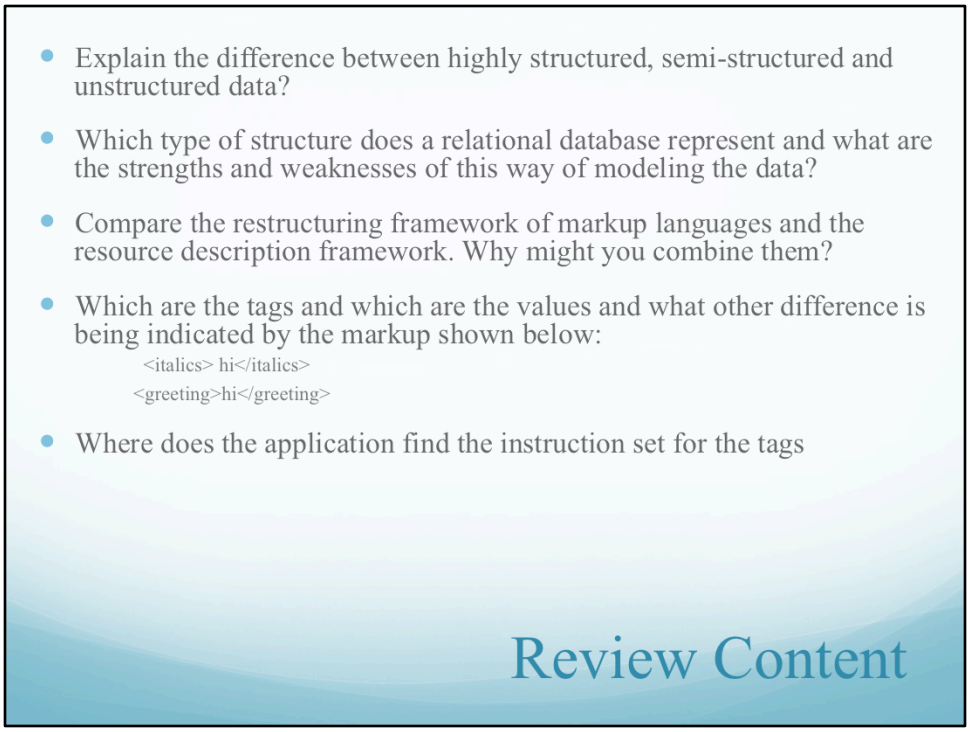
BINF 8211/6211
Design and Implementation of Bioinformatics Databases
Lecture #20

Dr. D. Andrew Carr
Dept. Bioinformatics and Genomics UNCC
Spring 2016

Review Relational Model

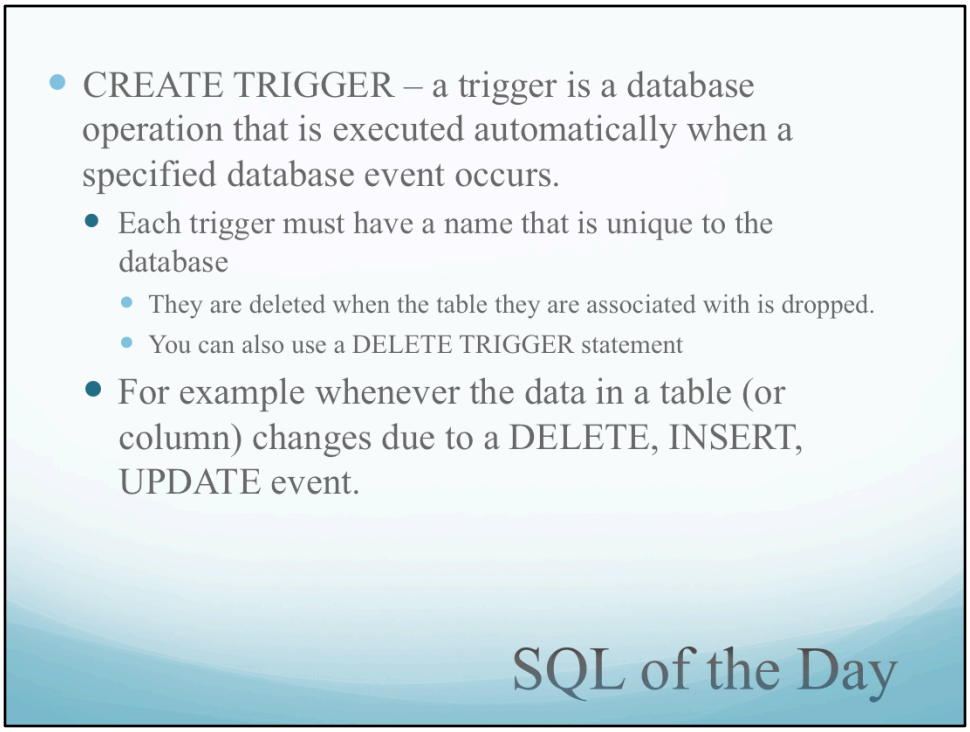
Today Topics

- SQL –Triggers
- Transaction block



Highly structured implies strongly typed informs you as to what the datatype fields are limited to (including how they are stored electronically) and how they are associated in tuples (the dimensions that are supposed to be about the same thing) and may require that very dimension have a value and that no additional dimension can be present, what operations are valid for those values and restricts the meanings that can be applied.
Semi-structured uses a looser data typing and a less constrained set of associations, generally allowing some tuples to have more attributes and some having fewer attributes with actual values in them.
Unstructured data is not organized in a pre-defined manner, usually this means a lot of free text, although dates and quantitites are often embedded. It can also be information that has a structure that seems logical to a human, like a graph of data, that a computer cannot do anything with (a computer could take x and y values and a function to create the graph but it does not know what to do with the finished product). Text mining algorithms that tag as many of the grammatical elements as possible are used to infer structure and pull out computable elements.

- A relational database is highly structured. The upside is you can infer a lot of information, the downside is that having imposed the structure you may have warped it from its true shape. You also cannot handle partial information very effectively, it will change the 'shape' of part of the data model but not all of it. The



Triggers are database callback functions, which are automatically performed/invoked when a specified database event occurs.

SQL Trigger cont.

- Here a trigger is automatically updating the inventory table by subtracting a quantity of items requisitioned from the amount that is on hand.

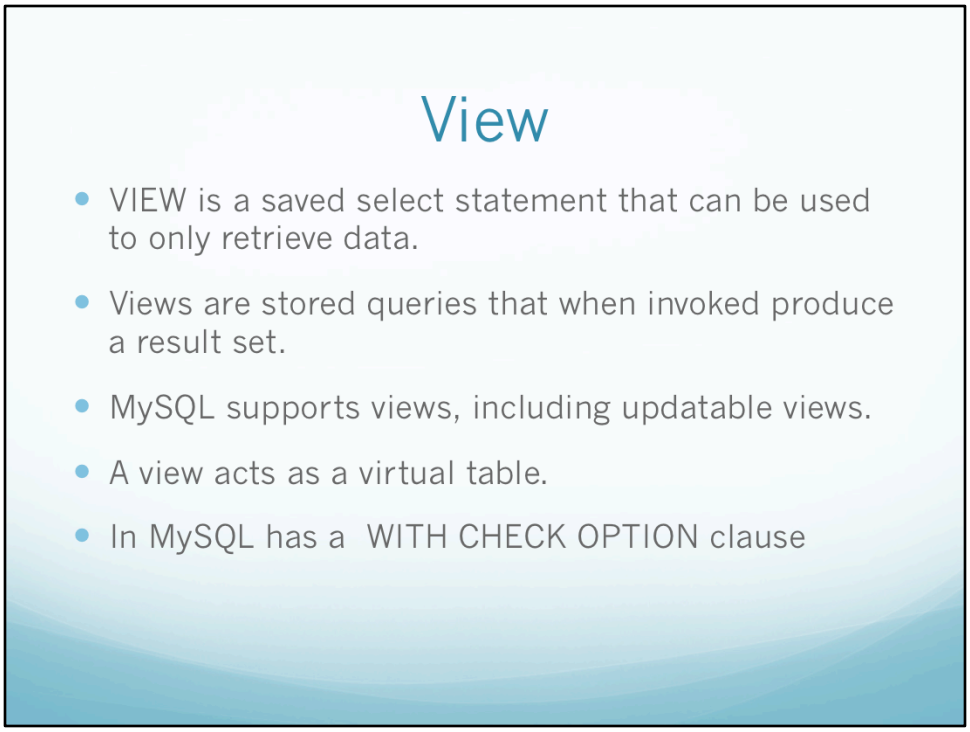
```
CREATE TRIGGER inventory_update
        AFTER INSERT ON REqDetail BEGIN
        UPDATE inventory
        SET OnHandQuant=(OnHandQuant-NEW.Quantity)
        WHERE inventory.StockNumber=NEW.StockNumber;
        END
```

```
sqlite> select StockNumber,OnHandQuan,Descrip from inventory where StockNumber =
75149;
StockNumber|OnHandQuan|Descrip
75149|92|Ball Point Pens Blue Fine tip, 12pack
sqlite> CREATE TRIGGER inventoryupdate AFTER INSERT ON ReqDetail BEGIN
   ...> UPDATE inventory SET OnHandQuan = (OnHandQuan- NEW.Quantity)
   ...> WHERE inventory.StockNumber = NEW.StockNumber;
   ...> END;
sqlite> INSERT INTO ReqDetail(ReqNumber,StockNumber,Quantity,ItemCost)
VALUES(1003,75149,3,0.77);
sqlite> select StockNumber,OnHandQuan,Descrip from inventory where StockNumber =75149;
StockNumber|OnHandQuan|Descrip
75149|89|Ball Point Pens Blue Fine tip, 12pack
sqlite>
```

- TRIGGERS:
- As of MySQL 5.7.2, it is possible to define multiple triggers for a given table that have the same trigger event and action time.
  - For example, you can have two BEFORE UPDATE triggers for a table
  - By default, triggers that have the same trigger event and action time activate in the order they were created. To affect trigger order, specify a clause after FOR EACH ROW that indicates FOLLOWS or PRECEDES and the name of an existing trigger that also has the same trigger event and action time.
  - With FOLLOWS, the new trigger activates after the existing trigger. With PRECEDES, the new trigger activates before the existing trigger.
- For example, the following trigger definition defines another BEFORE INSERT trigger for the account table:

```
mysql> CREATE TRIGGER ins_transaction BEFORE INSERT ON
account-> FOR EACH ROW PRECEDES ins_sum
    -> SET
    -> @deposits = @deposits + IF(NEW.amount>0,NEW.amount,0),
    -> @withdrawals = @withdrawals + IF(NEW.amount<0,-
NEW.amount,0);
Query OK, 0 rows affected (0.02 sec)
```

You can list all of the triggers from the sqlite_master table as follows:
Sqlite> SELECT table name FROM sqlite_master
WHERE type = ‘trigger’;
Will give you all the triggers
and
Sqlite> SELECT table name FROM sqlite_master
WHERE type = ‘trigger’ AND tbl_name=‘Gene’;
Will give just the triggers that involve the table indicated.

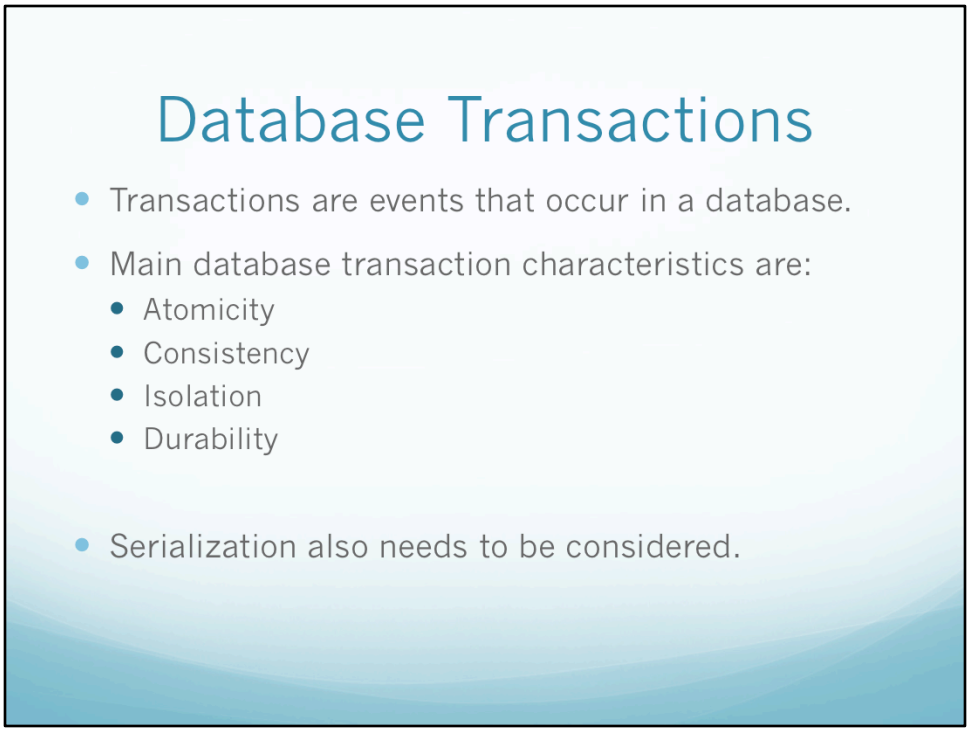


The WITH CHECK OPTION clause can be given for an updatable view to prevent inserts to rows for which the WHERE clause in the ***select_statement*** is not true. It aslo prevents updates to rows for which the WHERE clause is true but the update would cause it to be not true (in other words, it prevents visible rows from being updated to nonvisible rows).

View

- Example SQL for creating a view
- CREATE TABLE t1 (a INT);
- CREATE VIEW v1 AS SELECT * FROM t1 WHERE a < 2 WITH CHECK OPTION;
- TRIGGERS can be used on Views

From MySQL Documentation



Business databases often differ from Bioinformatics due to the number of transactions occurring. So this component is often left out of such courses. Just because the time between transactions is slower does not mean that they should be ignored.

Transaction

- A Transaction is an action that accesses the database:
  - SELECT
  - INSERT
  - UPDATE

Technically a transaction is any grouping of statements that must be entirely completed or aborted.

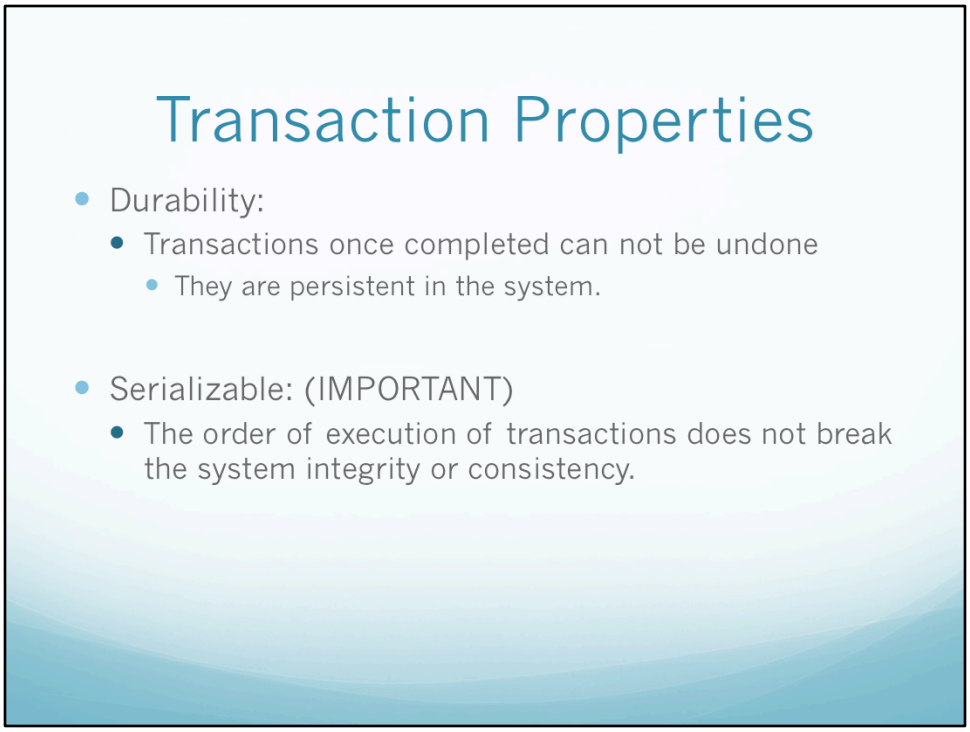
This implies no intermediate steps are acceptable.

Transactions

- All parts of the transaction have to be successful for the transaction to be successful.
  - A successful transaction is a defined as a set of statements that takes the database from one consistent state to another consistent state.
    - All constraints are met within the system
  - This statement helps to define the boundaries of the transaction.
  - Transactions can be one or many statements.
  - Most often more than two…
    - A retrieval of information
    - Followed by an update or instert of information

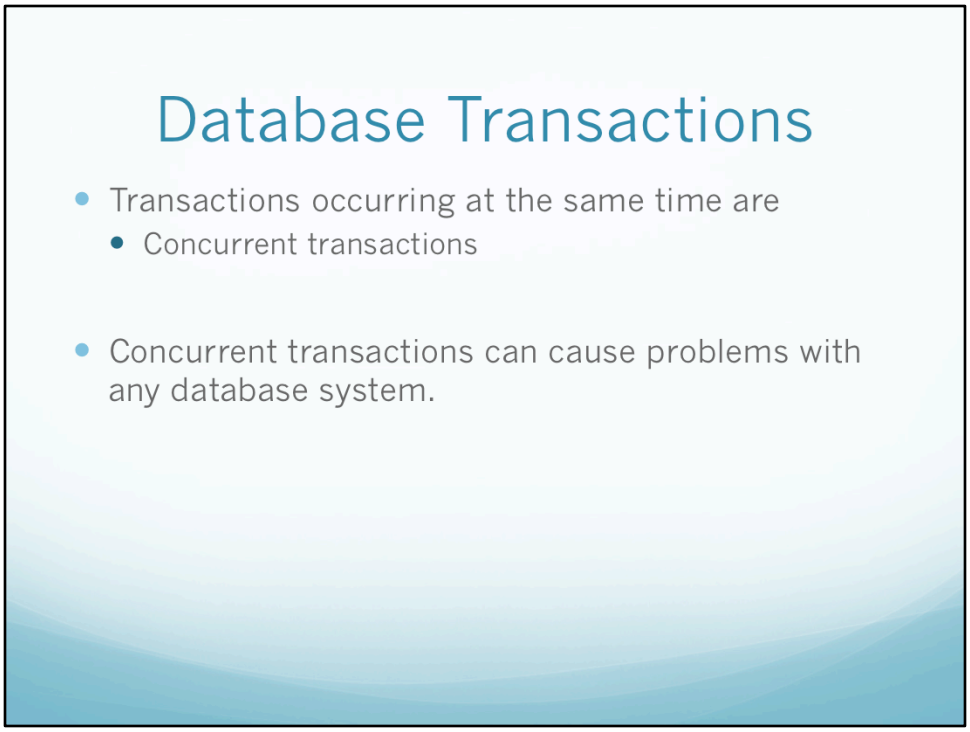
Transaction Properties

- Atomicity:
  - All operations and statements of the transaction must complete.
- Consistency:
  - The transaction can not break any of the integrity rules of the database.
- Isolation: (IMPORTANT)
  - Data can not be access and updated by more than one transaction at time.



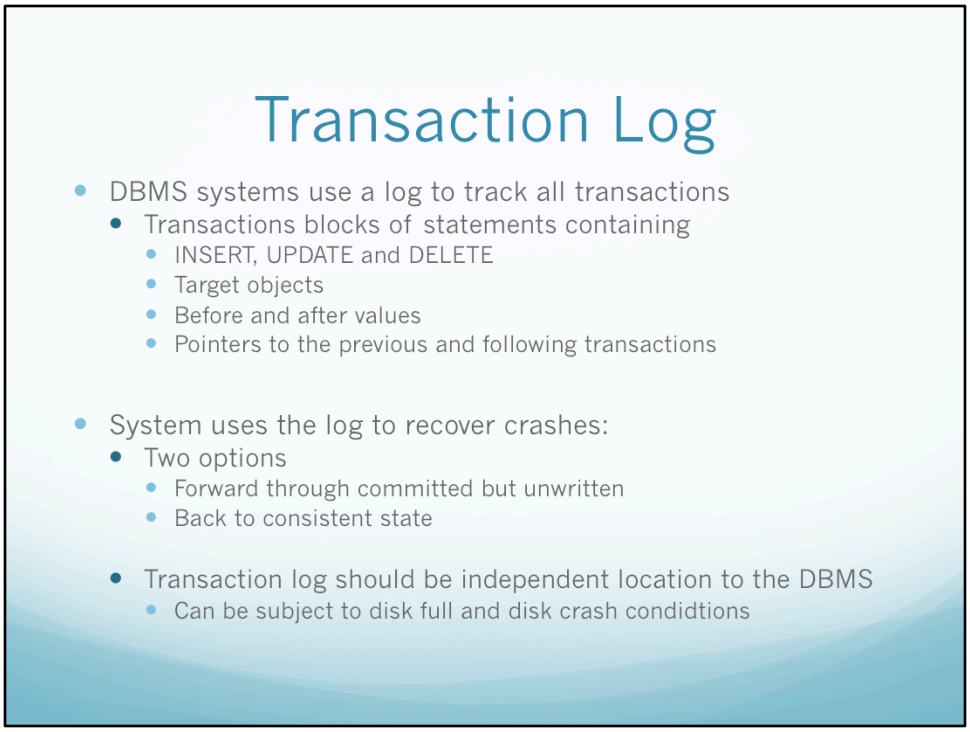
Durability means that the system is always moving forward in a manner of speaking.

Serializability is critical in multi user, multi access environments.



Programming in OpemMP or Posix will also have similar problems in management of transactions.

Best practice to hard code outcomes 3 and 4



Transaction Crashes include network and hardware.

If the log location gets corrupted then the system will fail to execute properly.

Directory permissions… can be a nightmare

Concurrency

- Management of multiple simultaneous transactions is call concurrency control
- This is non-trivial
  - Three areas of conflict that are most common
    - Lost update – when one transaction overwrites the update of another transaction.
    - Uncommitted data – one transaction is rolled back after the other has accessed the uncommitted data.
    - Inconsistent retrievals – when two transactions are reading and changing the same values at different intervals.

Lost Update Example:

Both transactions read values and update them

One adds the other subtracts from the same initial value only one of the outcomes is stored. The value of the other transaction is lost.

Scheduling transactions

- Schedulers are responsible for creating the order in which transactions are carried out.
  - Locking
  - Time stamping
  - Optimistic methods

Transaction Locking

- Locking – prevents another transaction from using the same data
- Levels:
  - Database, Table, Page, Row, Field]
- Types of locks
  - Binary Locks (on/off)
  - Shared/Exclusive
    - Shared locks allow reading of the data simultaneously provided no transaction is updating the data.
    - Exclusive are used when updating is occurring and only a single transaction is allow access.

Lock Serialization

- Serialization is carried out in two steps
  - Creation of all of the locks for a given transaction
    - Growing phase
  - Release of all of the locks
    - Shrinking phase
- DEADLOCK – when two or more transactions are waiting on each other to release data.
  - Prevention, system tests new transaction to evaluate conflict
  - Detection – Timer checks for problems rolls back the last offender
  - Avoidance -- Transaction must obtain all locks before execution.
    - Slower – because it only executes one transaction at a time.

Transaction timestamp

- Timestamps are used to decide which transaction has first access to the data.
- Conflicts are resolved by waiting one and testing the other. Rollingback if necessary

Optimistic Methods

- Presumption that minimal conflict will occur
  - Each transaction works on its own virtual copy of the data and then the two copies are merged.