Jisy Raju
Asst.Professor CSE
CE, Cherthala

Module 3

Structured Query Language (SQL): Basic SQL Structure, examples, Set operations, Aggregate Functions, nested sub-queries, Views, assertions and triggers

3.1 Structured Query Language (SQL)

SQL is a database computer language designed for the retrieval and management of data in a relational database. SQL stands for Structured Query Language. All the Relational Database Management Systems (RDMS) like MySQL, MS Access, Oracle, Sybase, Informix, Postgres and SQL Server use SQL as their standard database language. Originally, SQL was called SEQUEL (Structured English QUEry Language). SQL uses the terms table, row, and column for the formal relational model terms relation, tuple, and attribute, respectively.

SQL uses the concept of a catalog—a named collection of schemas in an SQL environment. An SQL environment is basically an installation of an SQL-compliant RDBMS on a computer system. A catalog always contains a special schema called INFORMATION_SCHEMA , which provides information on all the schemas in the catalog and all the element descriptors in these schemas.

SQL Constraints

- **NOT NULL** - Ensures that a column cannot have a NULL value
- **UNIQUE** - Ensures that all values in a column are different
- **PRIMARY KEY** - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table
- **FOREIGN KEY** - Uniquely identifies a row/record in another table
- **CHECK** - Ensures that all values in a column satisfies a specific condition
- **DEFAULT** - Sets a default value for a column when no value is specified

SQL Commands

- DDL: Data Definition Language
- DML: Data Manipulation Language
- DCL: Data Control Language
- TCL:Transaction Control Language

DDL: The SQL DDL allows specification of not only a set of relations, but also information about each relation, including:

- The schema for each relation.
- The types of values associated with each attribute.
- The integrity constraints.
- The set of indices to be maintained for each relation.

Basic data types used are,

char	A fixed-length character string with user-specified length n. The full form, character, can be used instead.
varchar	A variable-length character string with user-specified maximum length n. The full form, character varying, is equivalent.
int	An integer (a finite subset of the integers that is machine dependent). The full form, integer, is equivalent.
float	A floating-point number, with precision of at least n digits.

Main Commands,

CREATE	Creates a new table, a view of a table, or other object in the database.
ALTER	Modifies an existing database object, such as a table.
DROP	Deletes an entire table, a view of a table or other objects in the database.
Rename	Rename a table or its attribute.
Truncate	Operation that is used to mark the extents of a table for deallocation (empty for reuse)

Create

- To create database,

  CREATE DATABASE database_name

- To create Table Statement is used to create tables to store data. Integrity Constraints can also be defined for the columns while creating the table.

  CREATE TABLE table name(Attribute1 datatype(No.),...An datatype(No));
  CREATE TABLE employee (id number(5),name char(20),dept char(10));

- To create table constraint

CREATE TABLE *table_name* (A*1 datatype constraint*, *A2 datatype constraint*, *A3 datatype constraint*, ...);

-primary key: The primary key attributes are required to be nonnull and unique.

CREATE TABLE Persons (PID int(10) NOT NULL PRIMARY KEY, LastName varchar(25), FirstName varchar(25));

Or

CREATE TABLE Persons (PID int (10) NOT NULL, LastName varchar(25),FirstName varchar(25), primary key(id));

-**Foreign Key**

CREATE TABLE Orders (OrderID int NOT NULL PRIMARY KEY, OrderNumber int NOT NULL, PersonID int, FOREIGN KEY (PID) REFERENCES Persons(PID));

ALTER:

- To add column

  ALTER TABLE table_name ADD column_name datatype;
- To delete column

  ALTER TABLE table_name Drop column column_name;

DROP:

DROP DATABASE database_name;
DROP TABLE table_name;

RENAME:

- To rename a table

  RENAME TABLE tbl_name TO new_tbl_name**;**
- To rename a column

  ALTER TABLE table_name Rename column old column name to new name;

DML: The SQL commands that deals with the manipulation of data present in database belong to DML or Data Manipulation Language and this includes most of the SQL statements.

SELECT	used to retrieve data from the a database.
UPDATE	used to update existing data
INSERT	used to insert data into a table.
DELETE	used to delete records

INSERT

INSERT INTO *tablename* (*column1*, *column2*, ..)VALUES (*value1*, *value2*,.. .);

OR

INSERT INTO *table_name* VALUES (*value1*, *value2*, *value3*, ...);

SELECT

- To select a entire table

  SELECT * FROM table_name;
- To select column

  SELECT *column1*, *column2, ...*FROM *table_name;*
- To select rows

  SELECT *column1*, *column2, ...*FROM *table_name* WHERE *condition*;

UPDATE

UPDATE table_name SET column1 = value1, column2 = value2,... WHERE condition;

DELETE

- To delete all rows

  DELETE FROM *table_name*;
- To delete specific row

  DELETE FROM table_name WHERE condition;

DCL : DCL mainly deals with the rights, permissions and other controls of the database system

GRANT	gives user's access privileges to database.
REVOKE	withdraw user's access privileges given by using the GRANT command.

GRANT

GRANT privilege_name ON Table_name TO user_name;

Eg. GRANT SELECT ON employee TO user1

REVOKE

REVOKE privilege_name ON Table_name FROM user_name;

Eg. REVOKE SELECT ON employee FROM user1;

TCL

Transaction Control Language(TCL) commands are used to manage transactions in the database. These are used to manage the changes made to the data in a table by DML statements.

COMMIT	command is used to permanently save any transaction into the database.
ROLLBACK	command to rollback changes
SAVEPOINT	command is used to temporarily save a transaction so that you can rollback to that point whenever required.

COMMIT:

Syntax- COMMIT;

ROLLBACK:

- The ROLLBACK command to rollback those changes, if they were not committed using the COMMIT command

  Rollback;

- The command restores the database to last committed state by using SAVEPOINT command.

  ROLLBACK TO savepoint_name;

SAVEPOINT

SAVEPOINT savepoint_name;

3.2 Basic SQL Structure

The basic structure of an SQL query consists of three clauses: select, from, and where. The query takes as its input the relations listed in the from clause, operates on them as specified in the where and select clauses, and then produces a relation as the result.

3.2.1 Queries on a Single Relation

Let us consider the below table Faculty and DEPT,

FID	FNAME	DEPTID	SALARY
1	JISY	1	35000
2	SANTHY	1	30000
3	SWETHA	2	25000

DEPTID	DEPTNAME	Block
1	CS	New
2	EC	New
3	EE	old

Queries on a Single Relation

Let us consider a simple query using our Faculty table, “Find the names of all instructors.

```
select FNAME from Faculty;
```

The result is a relation consisting of a single attribute. If want to force the elimination of duplicates, we insert the keyword **distinct** after select. We can rewrite the preceding query as:

FID	FNAME	DEPTNAME
1	JISY	1
2	SANTHY	1
3	SWETHA	2

```
select distinct DEPTNAME from FACULTY;
```

DEPT
CS
EC

SQL allows us to use the keyword **all** to specify explicitly that duplicates are not removed:

```
select all DEPTNAME from DEPT;
```

The select clause may also contain arithmetic expressions involving the operators +, −, ∗, and / operating on constants or attributes of tuples. For example,the query returns a relation that is the same as the Faculty relation, except that the attribute salary is multiplied by 1.1.

select FID , FNAME, SALARY * 1.1 from Faculty;

SQL allows the use of the logical connectives and, or, and not in the where clause. The operands of the logical connectives can be expressions involving the comparison operators <, <=, >, >=, =, and <>.

select FNAME from Faculty where SALARY>30000;

3.2.2 Queries on Multiple Relations

Consider two tables,

CID	**NAME**	**Addr**
1	Manju	abc
2	Jisy	cde
3	Vishnu	efg
4	Meera	hij

OID	CID	AMOUNT
1	2	100
2	1	250
3	4	300
4	3	400

- Retrieve CID, Address and amount from relation CUSTOMER and ORDER whose name= jisy

  SELECT CUSTOMER.CID, Addr, AMOUNT FROM CUSTOMER, ORDER WHERE CUSTOMERS.CID = ORDERS.CID and NAME='Jisy';

- Retrieve customer id, name, Address and amount from relation CUSTOMER and ORDER

  SELECT CUSTOMER.CID, NAME, Addr, AMOUNT FROM CUSTOMER, ORDER WHERE CUSTOMERS.CID = ORDERS.CID;

Join

Select CID, NAME, Addr, AMOUNT from CUSTOMER **Natural join** ORDER

Select CID, NAME, Addr, AMOUNT from CUSTOMER **Inner join** ORDER on CUSTOMER.CID = ORDER.CID;

SQL aliases/ correlation name/ tuple variable.

SQL aliases are used to give a table, or a column in a table, a temporary name.

- To rename column,

  Select old column name as new name from table name;

  Eg. Select CID as CustomerID , Name from CUSTOMER;

- To rename table

Select Name from Customer as Cust where CID=1;

SELECT C.CID, NAME, Addr, AMOUNT FROM CUSTOMER as C, ORDER
WHERE C.CID = ORDERS.CID;

String Operations

SQL specifies strings by enclosing them in single quotes, for example, 'Computer'. The SQL standard specifies that the equality operation on strings is case sensitive; as a result the expression " 'computer' = 'Computer' " evaluates to false.

SQL also permits a variety of functions on character strings, such as concatenating (using " ||"), extracting substrings, finding the length of strings, converting strings to uppercase (using the function upper(s) where s is a string) and lowercase (using the function lower(s)), removing spaces at the end of the string (using trim(s)). Pattern matching can be performed on strings, using the operator **like**. We describe patterns by using two special characters:

• Percent (%): multiple character
• Underscore (_): single character.

SELECT *column1, column2, ...*FROM *table_name* WHERE *columnN* LIKE *pattern*;
SELECT * FROM CUSTOMER WHERE Name LIKE 'a%';

WHERE CustomerName LIKE 'a%'	Finds any values that start with "a"
WHERE CustomerName LIKE '%a'	Finds any values that end with "a"
WHERE CustomerName LIKE '%or%'	Finds any values that have "or" in any position
WHERE CustomerName LIKE '_r%'	Finds any values that have "r" in the second position
WHERE CustomerName LIKE 'a_%_%'	Finds any values that start with "a" and are at least 3 characters in length
WHERE ContactName LIKE 'a%o'	Finds any values that start with "a" and ends with "o"

SQL ORDER BY

The ORDER BY statement in sql is used to sort the fetched data in either ascending or descending according to one or more columns. By default ORDER BY sorts the data in ascending order.

SELECT *column1*, *column2, ...*FROM *table_name* ORDER BY *column1, column2, ...* ASC/DESC;

Eg. SELECT NAME FROM CUSTOMER ORDER BY Name DESC;

NAME
Vishnu

Manju
Jisy

SQL BETWEEN Operator

The BETWEEN operator selects values within a given range. The values can be numbers, text, or dates. The BETWEEN operator is inclusive: begin and end values are included.

SELECT *column_name(s)* FROM *table_name* WHERE *column_name* BETWEEN *value1* AND *value2;*

SELECT * FROM ORDER WHERE AMOUNT BETWEEN 100 AND 350;

OID	CID	AMOUNT
1	2	100
2	1	250
3	4	300

3.3 SET OPERATIONS

The SQL operations union, intersect, and except operate on relations and correspond to the mathematical set-theory operations ∪, ∩, and −. Consider two tables First and Second,

ID	Name
1	JISY
2	SANTHY

ID	Name
3	SWETHA
2	SANTHY

UNION Operation:is used to combine the results of two or more SELECT statements. However it will eliminate duplicate rows from its resultset. In case of union, number of columns and datatype must be same in both the tables, on which UNION operation is being applied.

SELECT * FROM First UNION SELECT * FROM Second;

ID	Name
1	JISY
2	SANTHY
3	SWETHA

UNION ALL: This operation is similar to Union. But it also shows the duplicate rows.

SELECT * FROM First UNION ALL SELECT * FROM Second;

ID	Name
1	JISY
2	SANTHY
3	SWETHA
2	SANTHY

INTERSECT: Intersect operation is used to combine two SELECT statements, but it only retuns the records which are common from both SELECT statements. In case of **Intersect** the number of columns and datatype must be same

SELECT * FROM First INTERSECT SELECT * FROM Second;

ID	Name
2	SANTHY

Minus/ Except: It combines the result of two SELECT statements. Minus operator is used to display the rows which are present in the first query but absent in the second query.

SELECT * FROM First Except SELECT * FROM Second;

ID	Name
1	JISY

SELECT * FROM Second MINUS SELECT * FROM First ;

ID	Name
3	SWETHA

3.4 Aggregate Functions

Aggregate functions are functions that take a collection (a set or multiset) of values as input and return a single value.

3.4.2 Basic Aggregation

SQL offers five built-in aggregate functions:

- Average: avg
- Minimum: min
- Maximum: max
- Total: sum
- Count: count

Select Aggregate fn(column name) from table_name where condition;

Stud

RollNo.	Name	Mark	Dept

1	A	40	cs
2	B	36	cs
3	C	28	ec
4	B	30	ec
5	F	46	ee
6	G	34	cs

AVG(): SELECT AVG(*column_name*) FROM *table_name* WHERE *condition*;

Select avg(Mark) from Stud;

COUNT(): The aggregate function count used to count the number of tuples in a relation.

Select Count(*) from Stud;

Count(*)
6

Select Count(*) from Stud where Name='B';

Count(*)
2

Select Count (Distinct Name) from Stud;

Name
A
B
C
F
G

MIN(): The MIN() function returns the smallest value of the columns.

Select Min(Mark) from Stud ;

Min
28

Max(): The MAX() function returns the largest value of the selected column.

Select Max(Mark) from Stud ;

Max
46

Sum(): The SUM() function returns the total sum of a numeric column.

Select sum(Mark) from Stud ;

Select sum(M1+M2) from Stud ; (also possible)

3.4.2 Aggregation with Group by: The GROUP BY statement is often used with aggregate functions.

Select Aggregate fn(column name) from table_name **group by** column name;

Select Dept, Count(RollNo) from Stud Group By Dept;

Dept	Count
cs	3
ec	2
ee	1

Group by using the HAVING clause: Grouping data with certain condition.

SELECT column_name(s) FROM table_name WHERE condition **GROUP BY** column name(s) **HAVING** condition

Select Dept, Count(RollNo) from Stud Group By Dept Having Mark>35;

Dept	Count
cs	2
ec	0
ee	1

3.4.3 Aggregation with Null and Boolean Values

In general, aggregate functions treat nulls according to the following rule: All aggregate functions except count (*) ignore null values in their input collection. As a result of null values being ignored, the collection of values may be empty. The count of an empty collection is defined to be 0, and all other aggregate operations return a value of null when applied on an empty collection. A Boolean data type that can take values true, false, and unknown.

3.5 Nested sub-queries

A Subquery or Inner query or a Nested query is a query within another SQL query and embedded within the WHERE clause. A subquery is a SELECT statement that is nested within another SELECT statement and which return intermediate results. The result of inner query is used in execution of outer query.

Consider the relation Stud, Course and Scourse

Stud

SID	Name	Mark	Dept
1	A	40	cs
2	B	36	cs
3	C	28	ec
4	B	30	ec
5	F	46	ee
6	G	34	cs

Course

CID	Cname
c1	DBMS
c2	DS
c3	CP

Scourse

SID	CID
1	c1
1	c2
2	c3
3	c2
4	c3

- **Independent Nested Queries:** In independent nested queries, query execution starts from innermost query to outermost queries. The execution of inner query is independent of outer query, but the result of inner query is used in execution of outer query. Various operators like IN, NOT IN, ANY, ALL etc are used in writing independent nested queries.

3.5.1 IN (Set Membership)

The IN connective for set membership, where the set is a collection of values produced by a select clause. The NOT IN connective for the absence of set membership.

SELECT column-names FROM table-name1 WHERE value IN (SELECT column-name FROM table-name2 WHERE condition)

Q1. If we want to find out SID who are enrolled in Cname ‘DS’ or ‘DBMS’.

From Course table, we can find out CID for Cname ‘DS’ or DBMS’ and we can use these CIDs for finding SIDs from Scourse TABLE.

STEP 1: Finding CID for Cname =’DS’ or ‘DBMS’

Select CID from Course where Cname = ‘DS’ or Cname = ‘DBMS’;

STEP 2: Using CID of step 1 for finding SID

Select SID from Scourse where CID **IN** (Select CID from Course where Cname = ‘DS’ or Cname = ‘DBMS’);

Q2. Find out names of STUDENTs who have either enrolled in ‘DS’ or ‘DBMS’, it can be done as:

Select Name from Stud where SID **IN** (Select SID from Scourse where CID **IN** (Select CID from Course where Cname = ‘DS’ or Cname = ‘DBMS’));

Q3. If we want to find out SIDs of STUDENTs who have neither enrolled in ‘DSA’ nor in ‘DBMS’, it can be done as:

Select Name from Stud where SID **NOT IN** (Select SID from Scourse where CID **IN**

(Select CID from Course where Cname = ‘DS’ or Cname = ‘DBMS’));

3.5.2 Test for Empty Relations

SQL includes a feature for testing whether a subquery has any tuples in its result. The **exists** construct returns the value true if the argument subquery is nonempty. We can test for the nonexistence of tuples in a subquery by using the **not exists** construct

Q1. If we want to find out NAME of Student who are enrolled in CID ‘C1’

Select NAME from Stud where **EXISTS**(select * from Scourse where Stud.SID=Scourse.SID and Scourse.CID=’C1’);

Correlated Query: With a normal nested subquery, the inner SELECT query runs first and executes once, returning values to be used by the main query. A **correlated subquery** is a subquery that uses values from the outer query.
Eg.SELECT employee_number, name FROM employees emp WHERE salary > (SELECT AVG(salary) FROM employees WHERE department = emp.department);

3.5.3 Test for the Absence of Duplicate Tuples

Unique constraint in SQL is used to check whether the sub query has duplicate tuples in it’s result. Unique construct returns true only if the sub query has no duplicate tuples, else it return false. We can test for the existence of duplicate tuples in a subquery by using the not unique construct.

Q1. Find course ID who enrolled in atleast one course?

SELECT Course.CID FROM Course WHERE UNIQUE (SELECT CID FROM Scourse where Scourse.CID=Course.CID);

3.5.4 ALL

The ALL operator returns TRUE if all of the subqueries values meet the condition.

> SELECT column-names FROM table-name WHERE column-name operator ALL (SELECT column-name FROM table-name WHERE condition)

Q1. Returns the names, Rollno of students whose mark is greater than the mark of all the students in department ec:

SELECT Name , SID FROM Stud WHERE Mark > ALL (SELECT Mark FROM Stud WHERE Dept =ec);

3.6 Views(Virtual Table)

A view is a virtual table based on the result-set of an SQL statement. A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database. We can create a view by selecting fields from one or more tables present in the database. A View can either have all the rows of a table or specific rows based on certain condition. Views are a logical virtual table created by “select query” but the result is not stored

anywhere in the disk and every time we need to fire the query when we need data, so always we get updated or latest data from original tables.

SD

S_ID	NAME	ADDRESS
1	Harsh	Kolkata
2	Ashish	Durgapur
3	Pratik	Delhi
4	Dhanraj	Bihar
5	Ram	Rajasthan

SM

ID	NAME	MARKS	AGE
1	Harsh	90	19
2	Suresh	50	20
3	Pratik	80	19
4	Dhanraj	95	21
5	Ram	85	18

3.6.1 Creating Views

A View can be created from a single table or multiple tables.

CREATE VIEW view_name AS SELECT column1, column2..... FROM table_name WHERE condition;

- **Creating View from a single table:**
  CREATE VIEW Details AS SELECT NAME, ADDRESS FROM SD WHERE S_ID < 5;

To see the data in the View, we can query the view in the same manner as we query a table.

SELECT * FROM Details;

NAME	ADDRESS
Harsh	Kolkata
Ashish	Durgapur
Pratik	Delhi
Dhanraj	Bihar

- **Creating View from multiple tables**:
  CREATE VIEW Marks AS
  SELECT SD.NAME, SD.ADDRESS, SM.MARKS FROM SD, SM WHERE SD.NAME = SM.NAME;

NAME	ADDRESS	MARKS
Harsh	Kolkata	90
Pratik	Delhi	80
Dhanraj	Bihar	95
Ram	Rajasthan	85

3.6.2 UPDATING VIEWS

A view can be updated under certain conditions which are given below –

- The SELECT clause may not contain the keyword DISTINCT.
- The SELECT clause may not contain summary functions.
- The SELECT clause may not contain set functions.

- The SELECT clause may not contain set operators.
- The SELECT clause may not contain an ORDER BY clause.
- The query may not contain GROUP BY or HAVING.
- Calculated columns may not be updated.

We can use the CREATE OR REPLACE VIEW statement to add or remove fields from a view.

CREATE OR REPLACE VIEW view_name AS SELECT column1,coulmn2,..FROM table_name WHERE condition;

For example, if we want to update the view **Marks** and add the field AGE to this View from **SM** Table, we can do this as:

CREATE OR REPLACE VIEW Marks AS SELECT SD.NAME, SD.ADDRESS, SM.MARKS,SM.AGE FROM SD,SM WHERE SD.NAME = SM.NAME;

NAME	ADDRESS	MARKS	AGE
Harsh	Kolkata	90	19
Pratik	Delhi	80	19
Dhanraj	Bihar	95	21
Ram	Rajasthan	85	18

3.6.2.1 Inserting a row

INSERT INTO View name(C1,C2...Cn) VALUES(V1,V2….Vn);

INSERT INTO Details(NAME, ADDRESS) VALUES("Suresh","Gurgaon");

NAME	ADDRESS
Harsh	Kolkata
Ashish	Durgapur
Pratik	Delhi
Dhanraj	Bihar
Suresh	Gurgaon

3.6.2.2 DELETING Rows

Deleting rows from a view is also as simple as deleting rows from a table. We can use the DELETE statement of SQL to delete rows from a view. Also deleting a row from a view first delete the row from the actual table and the change is then reflected in the view.

```
DELETE FROM view_name WHERE condition;

DELETE FROM Details WHERE NAME="Suresh";
```

NAME	ADDRESS
Harsh	Kolkata
Ashish	Durgapur
Pratik	Delhi
Dhanraj	Bihar

3.6.3 DELETING VIEWS

We can delete or drop a View using the DROP statement.

```
DROP VIEW view_name;
```

3.6.4 Materialized View

Materialized views are also the logical view of our data-driven by the select query but the result of the query will get stored in the table or disk.

Materialized View vs View

View	Materialized View
Views query result is not stored in the disk or database	Materialized view allow to store the query result in disk or table.
when we create a view using any table, rowid of view is same as the original table	Materialized view rowid is different.
View we always get latest data	Materialized view we need to refresh the view for getting latest data.
Performance of View is less than Materialized view.	Performance of Materialized View is higher than view.

3.7 ASSERTION AND TRIGGER

A trigger is a *statement* or a *block of statement* which are executed automatically by the system when an event like insert, update or delete takes place on a table.

A typical trigger has three components:

1. **The event(s):** These are usually database update operations that are explicitly applied to the database.. The events are specified with two keyword BEFORE and AFTER. Before means that the trigger should be executed before the triggering operation is executed. The keyword AFTER, which specifies that the trigger should be executed after the operation specified in the event is completed.

2. **The condition** that determines whether the rule action should be executed: Once the triggering event has occurred, an optional condition may be evaluated. If no condition is specified, the action will be executed once the event occurs. If a condition is specified, it is first evaluated, and only if it evaluates to true will the rule action be executed. The condition is specified in the WHEN clause of the trigger.

3. The **action** to be taken: The action is usually a sequence of SQL statements,but it could also be a database transaction or an external program that will be automatically executed. In this example, the action is to execute the stored

procedure INFORM_SUPERVISOR.

(CREATE/ REPLACE) TRIGGER trigger_name (BEFORE/ AFTER) INSERT | UPDATE | DELETE ON table_name FOR EACH ROW WHEN(some_condition) DECLARE …..some_declarations…

CREATE TRIGGER SALARY BEFORE INSERT OR UPDATE OF SALARY ON EMPLOYEE FOR EACH ROW WHEN (NEW.SALARY > (SELECT SALARY FROM EMPLOYEE WHERE SSN = condition)

Assertions - An assertion is a piece of SQL which makes sure a condition is satisfied or it stops action being taken on a database object. It could mean locking out the whole table or even the whole database.

CREATE ASSERTION name CHECK (NOT EXISTS (SELECT column name FROM table name WHERE condition));

Trigger	Assertion
executed automatically by the system when an event like insert, update or delete takes place on a table.	a piece of SQL which makes sure a condition is satisfied or it stops action being taken on a database object.
more powerful because the can check conditions and also modify the data	do not modify the data, they only check certain conditions.
Triggers are linked to specific tables and specific events.	Assertions are not linked to specific tables in the database and not linked to specific events.