SQL CodeCount™

Counting Standard

University of Southern California

Center for Systems and Software Engineering

June , 2007

Revision Sheet

Date	Version	Revision Description	Author
6/22/2007	1.0	Original Release	CSSE
1/2/2013	1.1	Updated document template	CSSE
12/03/2014	1.2	Added Cyclomatic Complexity	CSSE

Table of Contents

1. Definitions

1.1. **SLOC –** Source Lines of Code is a unit used to measure the size of software program. SLOC counts the program source code based on a certain set of rules. SLOC is a key input for estimating project effort and is also used to calculate productivity and other measurements.

1.2. **Physical SLOC –** One physical SLOC is corresponding to one line starting with the first character and ending by a carriage return or an end-of-file marker of the same line, and which excludes the blank and comment line.

1.3. **Logical SLOC –** Lines of code intended to measure "statements", which normally terminate by a semicolon (C/C++, Java, C#) or a carriage return (VB, Assembly), etc. Logical SLOC are not sensitive to format and style conventions, but they are language-dependent.

1.4. **Data declaration line or data line –** A line that contains declaration of data and used by an assembler or compiler to interpret other elements of the program.

The following table lists the SQL keywords that denote data declaration lines:

Character (String)	Numeric	DateTime	Misc
CHAR (length)	SMALLINT	DATE	BOOLEAN
CHARACTER (length)	INT	TIME [(SCALE)][WITH TIME ZONE]	BLOB
VARCHAR (length)	INTEGER	TIMESTAMP [(SCALE)][WITH TIME ZONE]	
CHARACTER VARYING (length)	FLOAT	INTERVAL	
	REAL		
	DOUBLE		

Table 1 Data Declaration Types

1.5. **Compiler Directives –** A statement that tells the compiler how to compile a program, but not what to compile. SQL does not contain any compiler directives.

1.6. **Blank Line –** A physical line of code, which contains any number of white space characters (spaces, tabs, form feed, carriage return, line feed, or their derivatives).

1.7. **Comment Line –** A comment is defined as a string of zero or more characters that follow language-specific comment delimiter.

SQL comment delimiters are "/*", "--", or "{..}". A whole comment line may span one line and does not contain any compilable source code. An embedded comment can co-exist with compilable source code on the same physical line. Banners and empty comments are treated as types of comments.

1.8. **Executable Line of code –** A line that contains software instruction executed during runtime and on which a breakpoint can be set in a debugging tool. An instruction can be stated in a simple or compound form.

2. An executable line of code may contain the following statements:

2.1. Commands which access the storage memory

2.2. Keywords which perform conditional operations

2.3. Data declaration (data) lines

2. Checklist for source statement counts

PHYSICAL SLOC COUNTING RULES			
MEASUREMENT UNIT	ORDER OF PRECEDENCE	PHYSICAL SLOC	COMMENTS
Executable Lines	1	One Per line	Defined in 1.8
Non-executable Lines			
Declaration (Data) lines	2	One per line	Defined in 1.4
Compiler Directives	3	One per line	Defined in 1.5
Comments			Defined in 1.7
On their own lines	4	Not Included (NI)	
Embedded	5	NI	
Banners	6	NI	
Empty Comments	7	NI	
Blank Lines	8	NI	Defined in 1.6

LOGICAL SLOC COUNTING RULES				
NO.	STRUCTURE	ORDER OF PRECEDENCE	LOGICAL SLOC RULES	COMMENTS
R01	Data Statements: SELECT UPDATE INSERT DELETE ALTER TABLE ALTER USER DECLARE FETCH CLOSE	1	Count Once	Each statement, including nested queries, is counted once per each occurrence.
R02	Schema Statements: CREATE CREATE TRIGGER CREATE SEQUENCE CREATE INDEX CREATE SYNONYM REPLACE COMMENT TRUNCATE RENAME DROP GRANT REVOKE	2	Count Once	

R03	Transactional Statements: COMMIT ROLLBACK	3	Count Once	
R04	Conditional Statements: WHERE GROUP BY ORDER BY HAVING LIMIT JOIN UNION	4	Count Once	Conditional statements appearing in combination with other keywords are counted once per each occurrence.
R05	IF ELSIF ELSE WHEN	5	Count Once	CASE, END IF will not be counted since they must used with others.
R06	LOOP CONTINUE EXIT FOR	6	Count Once	EXIT WHEN , LOOP, END LOOP will not count.
R07	EXCEPTION	7	Count Once	Do not count basic block.
R08	CREATE [OR REPLACE] PROCEDURE AS CREATE [OR REPLACE] FUNCTION AS	8	Count Once	One function or procedure only counts once.

3. Examples

EXECUTABLE LINES

DATA Statements
(Query and Modify Tables and Columns)

EDS1 – SELECT

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
SELECT [ALL|DISTINCT] select-list	SELECT city FROM cities	1
	WHERE city IN	1
SELECT * FROM select-list	(SELECT city FROM country	1
	WHERE id='1')	1
SELECT column FROM select-list WHERE column = <criteria>		

EDS2 – UPDATE

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
UPDATE table SET set-list [WHERE predicate]	UPDATE Customers	1
	SET Customer.id ='1'	1
	WHERE Customer.id='2'	1

EDS3 – INSERT

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
INSERT INTO table [(column-list)] VALUES (value-list)	INSERT INTO colors (cnum, color) VALUES ('C1', 'green')	1 0
INSERT INTO table [(column-list)] (query-specification)	INSERT INTO location	1
	SELECT ct.name, loc.type, 500	1
	FROM ct, loc	0
	WHERE ct.name="London" AND	1
	loc.type='Europe'	0

EDS4 – DELETE

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
DELETE FROM table [WHERE predicate]	DELETE * FROM Customers	1
	WHERE Id='1'	1
DELETE FROM table WHERE column NOT IN (SELECT column FROM table)	DELETE * FROM Customers NOT IN (SELECT Customers FROM Regulars)	1 1

EDS5 – ALTER

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
ALTER TABLE <TABLE NAME>	ALTER TABLE Customer ADD PRIMARY KEY (SID);	1 0

SCHEMA Statements (Maintain Schema – Catalog)

ESS1 – CREATE

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
CREATE TABLE table-name ({column-descr|constraint} [,{column-descr|constraint}]...)	CREATE TABLE locals (ct VARCHAR(5) NOT NULL PRIMARY KEY, name VARCHAR(16), city VARCHAR(16))	1
CREATE VIEW view-name [(column-list)] AS query [WITH [CASCADED|LOCAL] CHECK OPTION]	CREATE VIEW supplied_parts AS SELECT * FROM parts WHERE pnum IN (SELECT pnum FROM supplier)	1 1 2 0

ESS2 – DROP

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
DROP TABLE table-name {CASCADE|RESTRICT}	DROP TABLE locals	1
DROP VIEW view-name {CASCADE|RESTRICT}	DROP VIEW supplied_parts	1

ESS3 – GRANT

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
GRANT privilege-list ON [TABLE] object-list TO user-list	GRANT SELECT,INSERT,UPDATE(parts) ON p TO mike	1 1 0

ESS4 – REVOKE

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
REVOKE	REVOKE privilege-list ON [TABLE] object-list FROM user-list	REVOKE SELECT,INSERT,UPDATE(parts) ON p FROM mike

TRANSACTIONAL Statements (Maintain Schema – Catalog)

ETS1 – COMMIT

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
COMMIT [WORK]	COMMIT	1

ETS2 – ROLLBACK

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
ROLLBACK [WORK]	ROLLBACK	1

CONDITIONAL Statements

ECS1 – WHERE

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
SELECT [FROM table_references] [WHERE where_condition]	SELECT * FROM Table WHERE Table.id='1'	1 1

ECS2 – GROUP BY

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
SELECT [FROM table_references] [WHERE where_condition] [GROUP BY {col_name | expr | position} [ASC | DESC	SELECT * FROM Customers GROUP BY ID	1 0 1

ECS3 – ORDER BY

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
SELECT * FROM select_list ORDER BY column [ASC|DESC]	SELECT * FROM Customers ORDER BY Id ASC	1 0 1

ECS4 – LIMIT

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
SELECT [FROM table_references] [WHERE where_condition] [LIMIT {[offset,] row_count | row_count OFFSET offset}]	SELECT * FROM Customers LIMIT 1	1 0 1

ECS5 – JOIN

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
table-1 { LEFT | RIGHT | FULL OUTER JOIN table-2 ON predicate	SELECT count(*) as totalcount, trsuser.id, trsuser.fname, trsuser.mortgage FROM customers, loanInfo,trsuser LEFT OUTER JOIN leadSupplierCampaign ON leadSupplierCampaign.CampaignID = customers.Referral	1 0 0 1 0 0 0

ECS6 – UNION

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
table_query UNION [ALL] table_query [ORDER BY column [ASC | DESC] [, ...]]	SELECT customers.name FROM customers WHERE customers.name LIKE 'T%' UNION SELECT public.name FROM public WHERE public.name LIKE 'T%'	1 0 1 1 1 0 1

DECLARATION OR DATA LINES

DDL1 – variable declaration

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
< name> < type>	userid int(10), addnewuser enum('1','0'), permission enum('1','0'), assignleadsupplier enum('1','0'), addnewtsr enum('1','0'), assigntsrls enum('1','0'), leadsquery enum('1','0'), postedleadsall enum('1','0'), postedleadsassigned enum('1','0'), leadpurchasers enum('1','0'), accountexecutives enum('1','0'), lsall enum('1','0'), lsassigned enum('1','0')	1 1 1 1 1 1 1 1 1 1 1 1 1

4. Cyclomatic Complexity

Cyclomatic complexity measures the number of linearly independent paths through a program. It is measured for each function, procedure, or method according to each specific program language. This metric indicates the risk of program complexity and also determines the number of independent test required to verify program coverage.

The cyclomatic complexity is computed by counting the number of decisions plus one for the linear path. Decisions are determined by the number of conditional statements in a function. A function without any decisions would have a cyclomatic complexity of one. Each decision such as an if condition or a for loop adds one to the cyclomatic complexity. Note that in PL/SQL , all code is treated as a single main function and hence Cyclomatic Complexity is calculated with the assumption that there is only a single function to process.

The cyclomatic complexity metric v(G) was defined by Thomas McCabe. Several variations are commonly used but are not included in the UCC. The modified cyclomatic complexity (CC3) counts the switch blocks as a single decision rather than counting each case. The strict or extended cyclomatic complexity (CC2) includes boolean operators within conditional statements as additional decisions.

Cyclomatic Complexity	Risk Evaluation
1-10	A simple program, without much risk
11-20	More complex, moderate risk
21-50	Complex, high risk program
> 50	Untestable program, very high risk

For PL/SQL, the following table lists the conditional keywords used to compute cyclomatic complexity.

Statement	CC Count	Rationale
If	+1	if adds a decision
Elsif	+1	else if adds a decision
Else	0	Decision is at the if statement
case when	+1 per when	Each when adds a decision – not the case
case default	0	Decision is at the when statements
For	+1	for adds a decision at loop start
while	+1	while adds a decision at loop start or at end of do loop