ANSI SQL-92 CodeCount™

Counting Standard

University of Southern California

Center for Systems and Software Engineering

June, 2007

Revision Sheet

Date	Version	Revision Description	Author
6/22/07	1.0	Original Release	CSSE

1.0 CHECKLIST FOR SOURCE STATEMENT COUNTS

PHYSICAL AND LOGICAL SLOC COUNTING RULES

Measurement Unit	Order of Precedence	Physical SLOC	Logical SLOC	Comments
Executable lines	1	One per line	See table below	Defined in 2.9
Non-executable lines				
Declaration (Data) lines	2	One per line	See table below	Defined in 2.4
Compiler directives	3	One per line	See table below	Defined in 2.5
Comments				Defined in 2.8
On their own lines	4	Not included (NI)	NI	
Embedded	5	NI	NI	
Banners	6	NI	NI	
Empty comments	7	NI	NI	
Blank lines	8	NI	NI	Defined in 2.7

Table 1 Physical and Logical SLOC Counting Counts

ANSI SQL-92 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		counted once per each occurrence.	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		counted once per each occurrence.	
R03	Transactional Statements:		counted once per	

No.	Structure	Order of Precedence	Logical SLOC Rules	Comments
	COMMIT ROLLBACK		each occurrence	
R04	Conditional Statements: WHERE GROUP BY ORDER BY HAVING LIMIT JOIN UNION		counted once per each occurrence	conditional statements appearing in combination with other keywords are counted once per each occurrence

Table 2 Logical SLOC Counting Rules

2.0 DEFINITIONS

2.1 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.

2.2 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.

2.3 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), or a new line in a stored procedure or a function in SQL etc. Logical SLOC are not sensitive to format and style conventions, but they are language-dependent.

2.4 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 C/C++ 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		

NOTE: See Section 3 of this document for examples of data declaration lines.

2.5 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).

2.6 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 or more lines 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.

2.7 Executable line of code

- An executable line of code may not contain the following statements:
  - Commands which access the storage memory.
  - Keywords which perform conditional operations.
  - Data declaration (data) lines

3.0 EXAMPLES OF LOGICAL SLOC COUNTING

EXECUTABLE LINES

DATA STATEMENTS

(QUERY AND MODIFY TABLES AND COLUMNS)

ID	STATEMENT DESCRIPTION	GENERAL FORM	SPECIFIC EXAMPLE	SLOC COUNT
EDS1	SELECT	SELECT [ALL|DISTINCT] select-list		
			SELECT city FROM cities	1
		SELECT * FROM select-list	WHERE city IN	1
			(SELECT city FROM country	1
		SELECT column FROM select-list	WHERE id='1')	1
		WHERE column = <criteria>		
EDS2	UPDATE	UPDATE table SET set-list	UPDATE Customers	1
		[WHERE predicate]	SET Customer.id ='1'	1
			WHERE Customer.id='2'	1
EDS3	INSERT	INSERT INTO table [(column-list)]	INSERT INTO colors (cnum, color)	1
		VALUES (value-list)	VALUES ('C1', 'green')	0
		INSERT INTO table [(column-list)]	INSERT INTO location	1
		(query-specification)	SELECT ct.name, loc.type, 500	1
			FROM ct, loc	0
			WHERE ct.name="London" AND	1
			loc.type='Europe'	0
EDS4	DELETE	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	1
			(SELECT Customers FROM Regulars)	1
EDS5	ALTER	ALTER TABLE <TABLE NAME>	ALTER TABLE Customer	1
			ADD PRIMARY KEY (SID);	0

SCHEMA STATEMENTS

(MAINTAIN SCHEMA – CATALOG)

ID	STATEMENT DESCRIPTION	GENERAL FORM	SPECIFIC EXAMPLE	SLOC COUNT
ESS1	CREATE	CREATE TABLE table-name	CREATE TABLE locals	1
		({column-descr|constraint}	(ct VARCHAR(5) NOT NULL	
		[,{column-descr|constraint}]...)	PRIMARY KEY,	
			name VARCHAR(16),	
			city VARCHAR(16)	
			)	
		CREATE VIEW view-name	CREATE VIEW supplied_parts AS	1
		[(column-list)] AS query	SELECT * FROM parts	1
		[WITH [CASCADED|LOCAL]	WHERE pnum IN (SELECT pnum	2
		CHECK OPTION]	FROM supplier)	0

ESS2	DROP	DROP TABLE table-name {CASCADE|RESTRICT} DROP VIEW view-name {CASCADE|RESTRICT}	DROP TABLE locals DROP VIEW supplied_parts	1 1
ESS3	GRANT	GRANT privilege-list ON [TABLE] object-list TO user-list	GRANT SELECT,INSERT,UPDATE(parts) ON p TO mike	1 1 0
ESS4	REVOKE	REVOKE privilege-list ON [TABLE] object-list FROM user-list	REVOKE SELECT,INSERT,UPDATE(parts) ON p FROM mike	1 1 0

TRANSACTIONAL STATEMENTS
(MAINTAIN SCHEMA – CATALOG)

ID	STATEMENT DESCRIPTION	GENERAL FORM	SPECIFIC EXAMPLE	SLOC COUNT
ETS1	COMMIT	COMMIT [WORK]	COMMIT	1
ETS2	ROLLBACK	ROLLBACK [WORK]	ROLLBACK	1

CONDITIONAL STATEMENTS

ID	STATEMENT DESCRIPTION	GENERAL FORM	SPECIFIC EXAMPLE	SLOC COUNT

ECS1	WHERE	SELECT
		[FROM table_references]
		[WHERE where_condition]
			SELECT * FROM Table
			WHERE Table.id='1'
				1
				1

ECS2	GROUP BY	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	SELECT * FROM select_list
		ORDER BY column
		[ASC|DESC]
			SELECT * FROM
			Customers
			ORDER BY Id ASC
				1
				0
				1

ECS4	LIMIT	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	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
				1
				0
				0
				0

			LEFT OUTER JOIN leadSupplierCampaign ON leadSupplierCampaign.CampaignID = customers.Referral	1 0 0 0 0

ECS6	UNION	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 (DATA) LINES

ID	STATEMENT DESCRIPTION	GENERAL FORM	SPECIFIC EXAMPLE	SLOC COUNT
DDL1	variable declaration	< 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 1 1 1 1