1 Structure of the Benchmark XML Definition File

1.1 Conceptual model

In Figure 1, there is a simplified conceptual model of the SQL benchmark. The exact structure of the XML definition file will be described later. The benchmark consists of one *init script* and one *clean-up script*. The init script prepares the database before the benchmark starts. The clean-up script cleans-up the database after the benchmark finishes. The both scripts have the same structure; they contain a *default statement list*, which is independent of any particular DBMS, and zero or more *DBMS-specific statement lists*, which can be utilized, if a particular DBMS does not support the syntax of the default statement list. Naturally, each statement list (default or DBMS-specific) consists of zero or more SQL *statements*.

The benchmark contains one or more *test groups*, where each test group includes one or more *tests* and each test consists of at least two *variants*. All variants within a particular test represent semantically equivalent SQL queries. Since the SQL syntax supported by each DBMS can be a bit different, a variant is associated with a *default statement* and, if a DBMS does not support the syntax of the default statement, a *specific statement* for the DBMS can be specified. It can also happen that a variant is not supported by a particular DBMS (e.g., some DBMSs does not support lateral joins).

Some of the test are parametrized. If a test is parametrized, one or more *parameters* for the test are specified. Certain settings of the parameters are encapsulated in *templates*. For each parameter and template, there is a *parameter value*. The parameters are then referenced in the SQL statements (default or specific) by writing the `$name` of the parameter. When the benchmark runs, a parametrized test is executed for each of the specified templates such that the names of the parameters are substituted by the values of the template. In such a way, it is possible to, for example, run the same test with different constants causing a different query selectivity.

Apart from tests, a test group also contains one or more *configurations*. A configuration represents a physical design of the database, i.e., it specifies available indexes. Usually, two configurations are specified – one including only indexes on the primary keys and the other including more appropriate indexes on foreign keys, search attributes etc. When the benchmark runs, all tests in a group are repeated for each configuration. In such a way, we can examine how the selection of a query plan is influenced by a physical design of the database. Each configuration consists of an *init script* and a *clean-up script* working the same as the init and clean-up scripts of the whole benchmark described above.

Finally, the benchmark includes zero or more *annotations*. An annotation represents a feature which can be associated with tests, variants or templates. For example, by an annotation, we can distinguish variants utilizing window functions, or we can distinguish high- and low- selective templates.

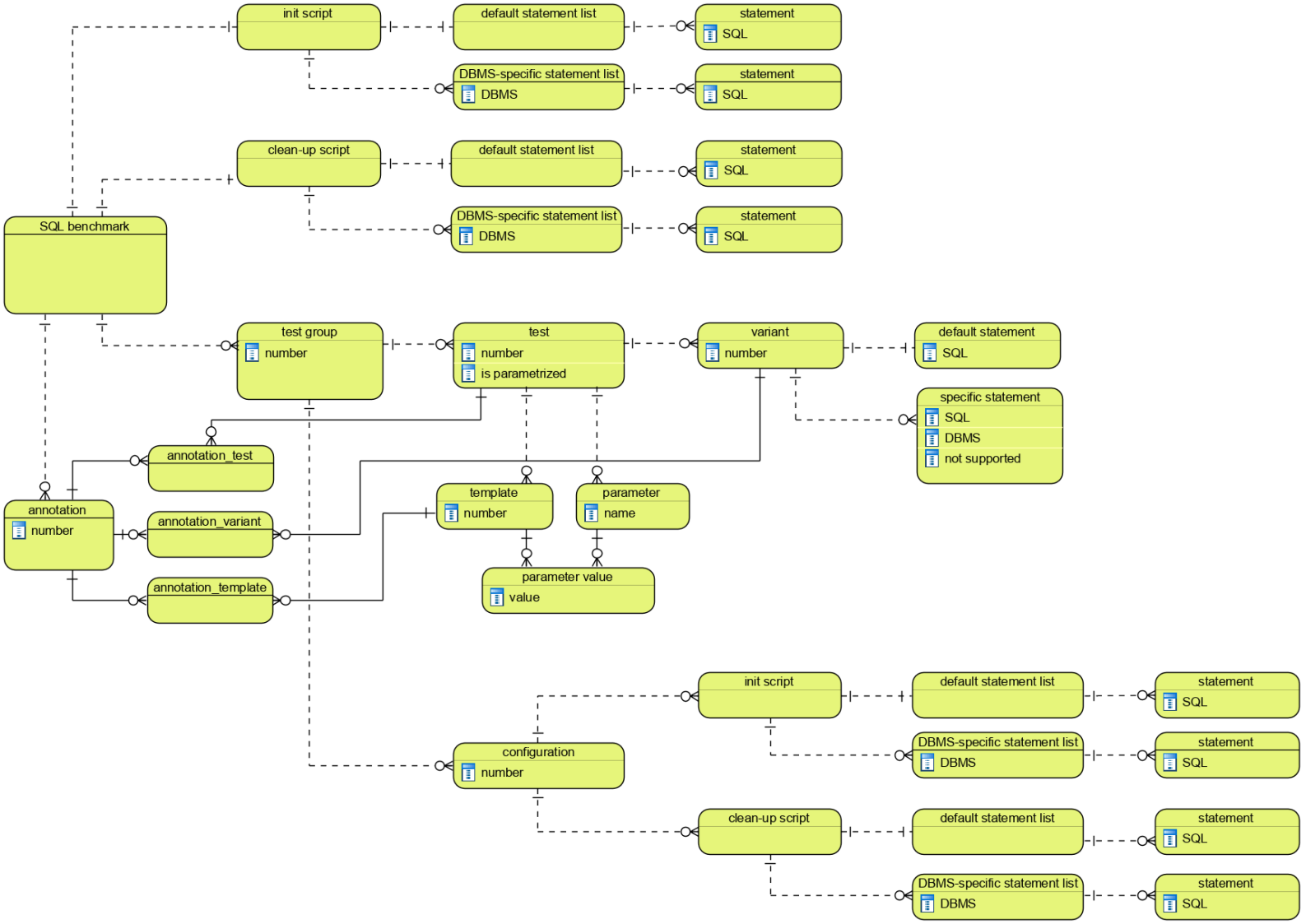

Figure 1: Conceptual model of the SQL benchmark

1.2 Workflow

The flowchart of the whole benchmark is depicted in Figure 2. The benchmarks starts with the execution of the init script. After that it iterates through all test groups. For each test group, it iterates through all configurations, i.e., physical index designs of the database. In each configuration, the init script of the configuration is executed. After that, the benchmark loops through all tests in the current test group. If a test is parametrized, all templates are iterated. For each template, the benchmark iterates through all the variants such that it first substitutes parameter references to the values of the template and, subsequently, the variant is executed. If a test is not parametrized, then all the variants are iterated and executed. At the end of each configuration, a clean-up script of the configuration is executed. The benchmark finishes with the execution of the benchmark's clean-up script.

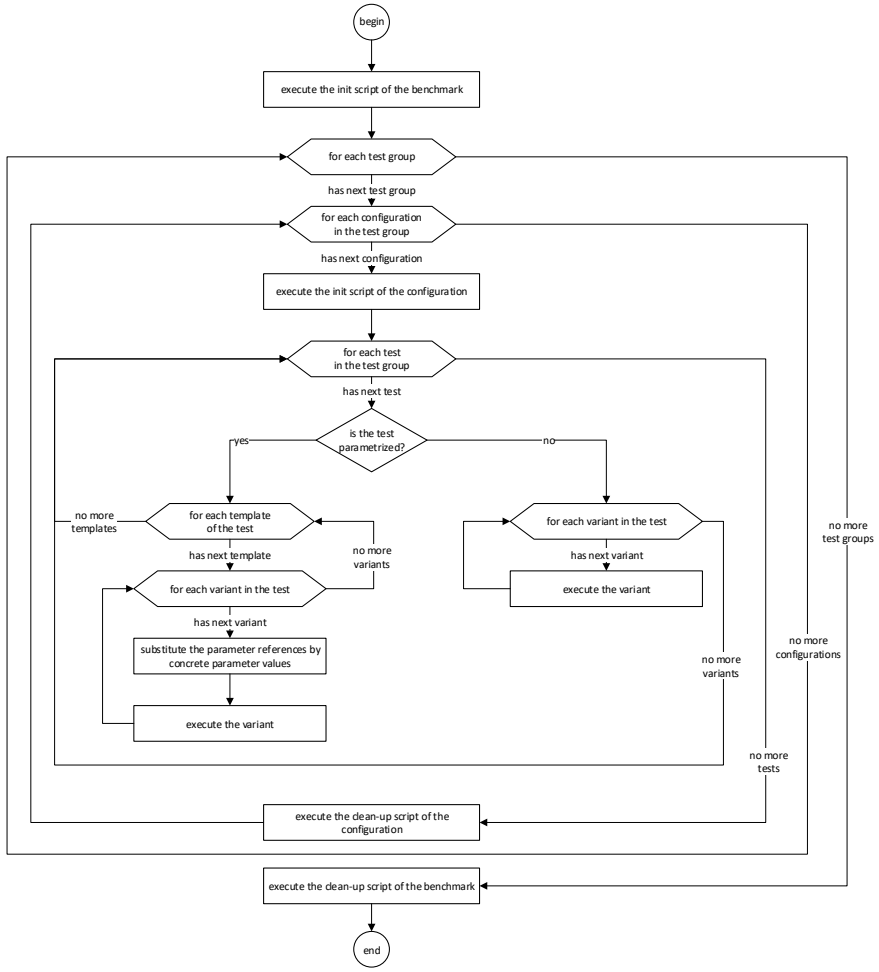

Figure 2: Flowchart of the SQL benchmark

1.3 XML file

The structure of the benchmark XML definition file is described in the following table.

Line	Tree structure of the XML	Cardinality	Description
1	<sql_benchmark>		The root element.
2	├ <name>		Name of the benchmark.
3	├ <author>		Author of the benchmark.
4	├ <description>		Description of the benchmark.
5	├ <init_script>		Script to initialize the database for the benchmark.
6	│ ├ <default_statement_list>		Default DBMS-independent list of SQL statements.
7	│ │ └ <statements>		List of SQL statements.
8	│ │ └ <statement>	[0...*]	SQL statement.
9	│ │ └ <command_text>		SQL statement command.
10	│ └ <specific_statement_lists>		DBMS-dependent lists of SQL statement.
11	│ └ <specific_statement_list>	[0...*]	List of SQL statements.
12	│ ├ <provider_name>		Name of the DBMS.
13	│ └ <statements>		List of SQL statements.
14	│ └ <statement>	[0...*]	SQL statement.
15	│ └ <command_text>		SQL statement command.
16	├ <clean_up_script>		Script to clean the database when the benchmark is finished.
17	│ ├ <default_statement_list>		Default DBMS-independent list of SQL statements.
18	│ │ └ <statements>		List of SQL statements.
19	│ │ └ <statement>	[0...*]	SQL statement.
20	│ │ └ <command_text>		SQL statement command.
21	│ └ <specific_statement_lists>		DBMS-dependent lists of SQL statement.
22	│ └ <specific_statement_list>	[0...*]	List of SQL statements.
23	│ ├ <provider_name>		Name of the DBMS.
24	│ └ <statements>		List of SQL statements.
25	│ └ <statement>	[0...*]	SQL statement.
26	│ └ <command_text>		SQL statement command.
27	├ <test_groups>		Test groups.

28	└ <test_group>	[0...*]	Test group.
29	├ <id>		Unique identifier.
30	├ <number>		Number of the test group.
31	├ <name>		Name of the test group.
32	├ <description>		Description of the test group.
33	├ <tests>		Tests in the test group.
34	│ └ <test>	[0...*]	Test in the test group.
35	│ ├ <id>		Unique identifier.
36	│ ├ <number>		Number of the test.
37	│ ├ <name>		Name of the test.
38	│ ├ <description>		Description of the test.
39	│ ├ <active>		Whether the test should be processed.
40	│ └ <variants>		Variants of the test.
41	│ ├ <variant>	[0...*]	Variant of the test.
42	│ │ ├ <id>		Unique identifier.
43	│ │ ├ <number>		Number of the variant.
44	│ │ ├ <name>		Name of the variant.
45	│ │ ├ <description>		Description of the variant.
46	│ │ ├ <default_statement>		Default DBMS-independent SQL statement.
47	│ │ │ └ <command_text>		SQL statement command.
48	│ │ ├ <specific_statements>		DBMS-dependent SQL statements.
49	│ │ │ └ <specific_statement>	[0...*]	DBMS-dependent SQL statement.
50	│ │ │ ├ <provider_name>		Name of the DBMS.
51	│ │ │ └ <not_supported>		Whether the variant is not supported by the DBMS.
52	│ │ │ └ <command_text>		SQL statement command.
53	│ │ └ <selected_annotations>		Annotations selected for the variant.
54	│ │ └ <selected_annotation>	[0...*]	Annotation selected for the variant.
55	│ │ └ <annotation_id>		ID of the annotation (line 114)
56	│ ├ <selected_annotations>		Annotations selected for the test.
57	│ │ └ <selected_annotation>	[0...*]	Annotation selected for the test.
58	│ │ └ <annotation_id>		ID of the annotation (line 114)

59	<expected_result_size>		Expected number of rows returned by the SQL statement.
60	<parametrized>		Whether the test is parametrized by templates.
61	<parameters>	[0...1]	Parameters of the test.
62	<parameter>	[0...*]	Parameter of the test.
63	<id>		Unique identifier.
64	<name>		Name of the parameter.
65	<templates>	[0...1]	Templates of the test.
66	<template>	[0...*]	Template of the test (particular settings of parameters in line 61).
67	<id>		Unique identifier.
68	<number>		Number of the template.
69	<expected_result_size>		Expected number of rows returned by the SQL statement for the template.
70	<selected_annotations>		Annotations selected for the template.
71	<selected_annotation>	[0...*]	Annotation selected for the template.
72	<annotation_id>		ID of the annotation (line 114)
73	<parameter_values>		Values for parameters and templates.
74	<parameter_value>		Value for parameters and templates.
75	<template_id>		ID of the template (line 67)
76	<parameter_id>		ID of the parameter (line 63)
77	<value>		Value of the parameter for the template.
78	<configurations>		Index configurations for the test group.
79	<configuration>	[1...*]	Index configuration for the test group.
80	<id>		Unique identifier.
81	<number>		Number of the configuration.
82	<name>		Name of the configuration.
83	<description>		Description of the configuration.
84	<init_script>		Script to initialize the configuration.
85	<default_statement_list>		Default DBMS-independent list of SQL statements.
86	<statements>		List of SQL statements.
87	<statement>	[0...*]	SQL statement.
88	<command_text>		SQL statement command.
89	<specific_statement_lists>		DBMS-dependent lists of SQL statement.

90	└ <specific_statement_list>	[0...*]	List of SQL statements.
91	└ <provider_name>		Name of the DBMS.
92	└ <statements>		List of SQL statements.
93	└ <statement>	[0...*]	SQL statement.
94	└ <command_text>		SQL statement command.
95	└ <clean_up_script>		Script to clean-up the configuration.
96	├ <default_statement_list>		Default DBMS-independent list of SQL statements.
97	└ <statements>		List of SQL statements.
98	└ <statement>	[0...*]	SQL statement.
99	└ <command_text>		SQL statement command.
100	└ <specific_statement_lists>		DBMS-dependent lists of SQL statement.
101	└ <specific_statement_list>	[0...*]	List of SQL statements.
102	└ <provider_name>		Name of the DBMS.
103	└ <statements>		List of SQL statements.
104	└ <statement>	[0...*]	SQL statement.
105	└ <command_text>		SQL statement command.
106	├ <connection_settings>		Connection settings.
107	├ <current_provider>		Currently selected DBMS.
108	└ <providers>		DBMSs connection settings.
109	└ <provider>	[0...*]	DBMS connection settings.
110	├ @name		Name of the DBMS.
111	└ *DBMS-specific attributes*		
112	└ <annotations>		Annotations applicable to tests (line 34), variants (line 41) and templates (line 66).
113	└ <annotation>	[0...*]	Annotation
114	├ <id>		Unique identifier.
115	├ <number>		Number of the annotation.
116	├ <name>		Name of the annotation.
117	└ <description>		Description of the annotation.

2 Structure of the Database of Results

2.1 Relational Data Model

The entity-relationship diagram of the database is depicted in Figure 3.

Figure 3: Entity-relationship diagram of the database of results

2.2 Data Dictionary

TestRun

The records in the `TestRun` table represent particular runs of the whole benchmark. All the records in all other tables are always related to one record in `TestRun`, either by a direct foreign key or via other tables.

column	data type	PK	FK	description
test_run_id	int	yes	no	Unique identifier of the benchmark run.
benchmark_id	int	no	no	Reserved for further purposes.
name	varchar, max. 50 chars.	no	no	Name of the benchmark.
start_date	date	no	no	Time stamp when the benchmark started.
end_date	date	no	no	Time stamp when the benchmark finished.
settings_info	varchar, max. 300 chars.	no	no	Settings of the DBMS connection.
executor_info	varchar, max. 300 chars.	no	no	Settings of the benchmark execution engine.

ConfigurationResult

The records in the `ConfigurationResult` table represent configurations (i.e., physical database designs) that were specified when the benchmark was executed.

column	data type	PK	FK	description
test_run_id	int	yes	yes	Foreign key to the `TestRun` table.
configuration_id	int	yes	no	Forms the primary key together with the `test_run_id` attribute.
configuration_number	varchar, max. 20 chars.	no	no	User-friendly number of the configuration.
configuration_name	varchar, max. 50 chars.	no	no	User-friendly name of the configuration.
init_script_started	bit	no	no	Determines whether the init script of the configuration was started.
init_script_completed	bit	no	no	Determines whether the init script of the configuration was completed.
init_error_message	varchar, max. 1000 chars.	no	no	Message of the exception if the init script was not completed successfully.
clean_up_script_started	bit	no	no	Determines whether the clean-up script of the configuration was started.
clean_up_script_completed	bit	no	no	Determines whether the clean-up script of the configuration was completed.
clean_up_error_message	varchar, max. 1000 chars.	no	no	Message of the exception if the clean-up script was not completed successfully.

TestGroupResult

The records in the `TestGroupResult` table represent test groups defined when the benchmark was executed.

column	data type	PK	FK	description
test_run_id	int	yes	yes	Foreign key to the `TestRun` table.
test_group_id	int	yes	no	Forms the primary key together with the `test_run_id` attribute.
test_group_number	varchar, max. 20 chars.	no	no	User-friendly number of the test group.
test_group_name	varchar, max. 50 chars.	no	no	User-friendly name of the test group.

AnnotationResult

Records in the `AnnotationResult` table represent a set of defined annotations when the benchmark was executed.

column	data type	PK	FK	description
test_run_id	int	yes	yes	Foreign key to the `TestRun` table.
annotation_id	int	yes	no	This attribute together with `test_run_id` form the primary key.
annotation_number	varchar, max. 20 chars.	no	no	User-friendly number of the annotation.
annotation_name	varchar, max. 50 chars.	no	no	User-friendly name of the annotation.

TestResult

The records in the `TestResult` table represent benchmark tests that were executed in particular configurations and with particular templates (in the case of parametrized tests).

column	data type	PK	FK	description
test_result_id	int	yes	no	Suggorate primary key.
test_run_id	int	no	yes	Foreign key to the `TestRun` table.
test_id	int	no	no	Unique identifier of the test in the XML definition file. Note that one test in the benchmark definition is repeated for each configuration of the test group and, if it is a parametrized test, then also for each template.
test_number	varchar, max. 20 chars.	no	no	User-friendly number of the test.
test_name	varchar, max. 50 chars.	no	no	User-friendly name of the test.
error_message	varchar, max. 1000 chars.	no	no	Message of an exception if the test was not completed.
test_group_id	int	no	yes	Foreign key to the `TestGroupResult` table.
configuration_id	int	no	yes	Foreign key to the `ConfigurationResult` table.
distinct_query_plans	int	no	no	Number of distinct query plans. Applicable only to DBMSs supporting the extraction of a query plan.
successfully_completed_variants	int	no	no	Number of variants within the test that were successfully completed.
started	bit	no	no	Determines whether the test was started.
completed	bit	no	no	Determines whether all the variants in the test were completed.
template_number	varchar, max. 20 chars.	no	no	For parametrized tests, this attribute represents the template number.

QueryVariantResult

Each record in the `QueryVariantResult` table represents an execution of a test variant under certain configuration and, for parametrized test, with specific parameter values. In other words, each record in this table represents a concrete SQL query which was executed under a concrete physical database design.

column	data type	PK	FK	description
query_variant_result_id	int	yes	no	Suggorate primary key.
test_result_id	int	no	yes	Foreign key to the `TestResult` table.
query	varchar, max. 1000 chars.	no	no	The SQL query.
token_count	int	no	no	Number of tokens in the SQL query.
query_variant_number	varchar, max. 20 chars.	no	no	User-friendly number of the variant.
query_variant_name	varchar, max. 50 chars.	no	no	User-friendly name of the variant.
query_processing_time	float	no	no	Query processing time in milliseconds.
expected_result_size	int	no	no	Expected number of records in the result of the SQL query.
result_size	int	no	no	Actual number of records in the result of the SQL query.
started	bit	no	no	Determines whether the variant was started.
completed	bit	no	no	Determines whether the variant was completed.
error_message	varchar, max. 1000 chars.	no	no	Message of an exception if the variant was not completed successfully.
query_variant_id	int	no	no	Unique identifier of the variant in the XML definition file. Note that one variant in the benchmark definition is repeated for each configuration of the test group and, if it belongs to a parametrized test, then also for each template.
query_plan	varchar, max. 2282 chars.	no	no	String representation of a query plan. Applicable only to DBMSs supporting the extraction of a query plan.

SelectedAnnotationResult

This table associates annotations specified in the `AnnotationResult` table with the records in `TestResult` or `QueryVariantResult` table.

column	data type	PK	FK	description
selected_annotation_result_id	int	yes	no	Suggorate primary key.
test_run_id	int	no	yes	Foreign key to the `TestRun` table.
test_result_id	int	no	yes	Foreign key to the `TestResult` table.
query_variant_result_id	int	no	yes	Foreign key to the `QueryVariantResult` table.
annotation_id	int	no	yes	Foreign key to the `AnnotationResult` table.
is_template_annotation	bit	no	no	Determines whether the annotation is associated with the template of the record in the `TestResult` table.