This is the only documentation which is allowed during the exam.

This sheet should not have any annotation.

SQL SYNTAX

This document presents the syntax for a subset of instructions in standard SQL. The notation we are using to define the syntax is an extended version of BNF[1]. In BNF, each syntactic ter mis defined through a set of production rules. These rules define an element in terms of a formula which is composed of characters, strings and syntactic elements which can be used to construct an instance of that element. In what follows, we include the symbols and rules fo the BNF version we have used:

- *Character string in italics*: defines the name of an identifier.
- Character string in (non-italics) lowercase: it defines the name of a syntax element which is not a terminal symbol and is defined in other production rule.
- CHARACTER STRING IN UPPERCASE: it is a terminal symbol in the gramar which defines a reserved Word in SQL.
- ::= The production operator. It is used in a production rule to separate the defined element (on the left) and the form which defines it (on the right).
- [ ] Square brackets are used to indicate that the elements inside are optional.
- {} The curly brackets are used to group the elements in a formula. The portion in a formula which is included inside curly brackets must be explicitly specified.
- A vertical bar is an alternative operator which indicates that two (or more) alternative portions are possible.
- If *xyz* is a syntactic element, *list_xyz* is a list of elements of type *xyz* which are separated by any kind of separator (blank, newline, etc.).
- If *xyz* is a syntactic element, *commalist_xyz* is a list of elements of type *xyz* separated by commas.

[1] Bakus Normal Form or Bakus Naur Form

FIRST PART:

Data definition

1) schema_definition ::=
CREATE SCHEMA [*schema*] [AUTHORIZATION *user*]
[list_schema_element]

2) schema_element ::=
domain_definition | table_definition |
view_definition | constraint_definition | privilege_definition

3) domain_definition ::=
CREATE DOMAIN *domain* [AS] datatype
[default_value_definition]
[list_domain_constraint_definition]

4) default_value_definition ::= DEFAULT {*literal* | system_function | NULL }

5) domain_constraint_definition ::=
[CONSTRAINT *constraint*] CHECK (conditional_expression) [when_to_check]

6) when_to_check ::=
[[NOT] DEFERRABLE] [INITIALLY {IMMEDIATE | DEFERRED}]

7) table_definition ::=
CREATE TABLE *base_table*
commalist_column_definition [commalist_table_constraint_definition]

8) column_definition ::=
column { datatype | *domain* } [default_value_definition]
[list_column_constraint_definition]

9) column_constraint_definition ::=
[CONSTRAINT *constraint*]
{NOT NULL |
PRIMARY KEY |
UNIQUE |
REFERENCES *table* [(*commalist_column*)]
[MATCH {FULL | PARTIAL }]
[ON DELETE action_reference]
[ON UPDATE action_reference] |
CHECK (conditional_expression) }
[when_to_check]

```
10) action_reference ::=
            NO ACTION | CASCADE | SET DEFAULT | SET NULL

11) table_constraint_definition ::=
        [CONSTRAINT constraint ]
            {PRIMARY KEY                                                        |
            UNIQUE (commalist_column)                                           |
            FOREIGN KEY (commalist_column) REFERENCES table [(commalist_column)]
                                                [MATCH {FULL | PARTIAL } ]
                                                [ON DELETE action_reference]
                                                [ON UPDATE action_reference]    |
            CHECK (conditional_expression) }
        [when_to_check]

12) view_definition ::=
            CREATE VIEW view [(commalist_column)]
            AS table_expression [WITH CHECK OPTION]

13) constraint_definition ::=
            CREATE ASSERTION constraint
            CHECK (conditional_expression)    [when_to_check]

14) privilege_definition ::=
            GRANT {commalist_privilege | ALL PRIVILEGES }
            ON object TO {commalist_users | PUBLIC}
            [WITH GRANT OPTION]

15) privilege ::= SELECT                         |     INSERT [(commalist_column)]     |
                UPDATE [(commalist_column)]      |     DELETE

16) object ::= DOMAIN domain | [TABLE] table

17) domain_alteration ::=
            ALTER DOMAIN domain
            {SET DEFAULT {literal | system_function | NULL }    |
            DROP DEFAULT                                        |
            ADD domain_constraint_definition                |
            DROP CONSTRAINT constraint       }

18) table_alteration ::=
            ALTER TABLE base_table
            {ADD [COLUMN] column_definition                     |
            ALTER [COLUMN] column
            {SET DEFAULT {literal | system_function | NULL }    |
            DROP DEFAULT}                                   |
            DROP [COLUMN] column {RESTRICT | CASCADE}  }

19) schema_removal ::=      DROP SCHEMA schema {RESTRICT | CASCADE}
```

20) domain_removal ::= DROP DOMAIN *domain* {RESTRICT | CASCADE}

21) base_table_removal ::= DROP TABLE *base_table* {RESTRICT | CASCADE}

22) view_removal ::= DROP VIEW *view* {RESTRICT | CASCADE}

23) general_constraint_removal ::= DROP ASSERTION *constraint*

24) authorisation_removal ::=
REVOKE [GRANT OPTION FOR]
{ALL | SELECT | INSERT[(*commalist_column*)] |
DELETE | UPDATE [(*commalist_column*)]}
ON object TO {*commalist_users* | PUBLIC}
{RESTRICT | CASCADE}

Data Manipulation

Table expressions

25) table_expression ::=
table_join_expression | no_table_join_expression

26) table_join_expression ::=
| table_reference [NATURAL] [join_type] JOIN table_reference
[ON conditional_expression | USING (*commalist_column*)]
| (table_expression) CROSS JOIN table_reference
| (table_join_expression)

27) table_reference ::=
table [[AS] *run_variable*] |
(table_expression) [[AS] *run_variable*] |
table_join_expression

28) join_type ::=
INNER | LEFT [OUTER] | RIGHT [OUTER] |
FULL [OUTER] | UNION

29) table_set_expression ::=
table_set_term |
table_expression {UNION | EXCEPT } [ALL]
[CORRESPONDING [BY (*commalist_column*)]] table_term

30) table_set_term ::=
primary_set_table |
table_term INTERSECT [ALL]
[CORRESPONDING [BY (*commalist_column*)]] primary_table

31) table_term ::=
table_set_term | table_join_expression

32) primary_table ::=
primary_set_table | table_join_expression

33) primary_set_table ::=
TABLE *table* | table_constructor |
expression_SELECT | (table_set_expression)

34) table_constructor ::= VALUES commalist_row_constructor

35) row_constructor ::= *scalar_expression* | (*commalist_scalar_expression*) |
(table_expression)

36) expression_SELECT ::=
SELECT [ALL | DISTINCT] commalist_selected_item
FROM commalist_table_reference
[WHERE conditional_expression]
[GROUP BY *commalist_column* [HAVING conditional_expression]]

37) selected_item ::=
scalar_expression [AS] *column* | [run_variable.]*

Data modification

38) insertion ::=
INSERT INTO *table* { [(*commalist_column*)] table_expression | DEFAULT VALUES }

39) update ::=
UPDATE *table* SET commalist_assignment
[WHERE conditional_expression]

40) assignment ::= *column* = {scalar_expression | DEFAULT | NULL }

41) removal ::= DELETE FROM *table* [WHERE conditional_expression]

Conditional expressions

42) conditional_expression ::=
conditional_term | conditional_expression OR conditional_term

43) conditional_term ::=
conditional_factor | conditional_term AND conditional_factor

44) conditional_factor ::= [NOT] conditional_check

45) conditional_check ::= primary_condition [IS [NOT] {TRUE | FALSE}]

46) primary_condition ::= simple_condition | (conditional_expression)

47) simple_condition ::=
comparison_condition | between_condition | like_condition | in_condition |
null_check | match_condition | all_any_condition | exists_condition |
unique_condition

48) comparison_condition ::= row_constructor comparison_predicate row_constructor

49) comparison_predicate ::= = | < | <= | > | >= | <>

50) between_condition ::= row_constructor [NOT] BETWEEN row_constructor
AND row_constructor

51) like_condition ::= string_expression [NOT] LIKE pattern [ESCAPE escape]

52) in_condition ::=
constructor_file [NOT] IN (table_expression) |
scalar_expression [NOT] IN (commalist_scalar_expression)

53) null_check ::= row_constructor IS [NOT] NULL

54) match_condition ::=
row_constructor MATCH [UNIQUE] [PARTIAL | FULL] (table_expression)

55) all_any_condition ::=
row_constructor comparison_predicate {ALL | ANY | SOME}(table_expression)

56)) exists_condition ::= EXISTS (table_expression)

57) unique_condition ::= UNIQUE (table_expression)

Scalar expression

58) scalar_expression ::= numeric_expression |
string_expression|

59) numeric_expression ::= numeric_term |
numeric_expression {+ | -} numeric_term

60) numeric_term ::= numeric_factor |
numeric_term {* | /} numeric_factor

61) numeric_factor ::= [+ | -] primary_number

62) primary_number ::= column_reference | *literal* |
scalar_function_reference | aggregated_function_reference |
(table_expression) | (numeric_expression)

63) aggregated_function_reference ::= COUNT(*)
| {AVG | MAX | MIN | SUM | COUNT } ([ALL | DISTINCT] scalar_expression)

64) string_expression ::= string_concatenation |
primary_string

65) string_concatenation ::=
string_expression ||[2] primary_string

66) primary_string ::=
column_reference | *literal* |
user_function | scalar_function_reference |
aggregated_function_reference | (table_expression) |
(string_expression)

Miscellaneous

67) table ::= *base_table* | *view*

68) pattern ::= string_expression

69) escape ::= string_expression

70) system_function ::= user_function | time_function

71) user_function ::=
USER |
CURRENT_USER |
SESSION_USER |
SYSTEM_USER

72) time_function ::=
CURRENT_DATE |
CURRENT_TIME |
CURRENT_TIMESTAMP

[2] "||" is the concatenation operator for strings

In order to ease the handling of this document, in what follows we include all the syntactic categories in SQL99 in alphabetical order. We indicate the number which shows the order to be used to locate them.

action_reference	10)
aggregated_function_reference	63)
all_any_condition	55)
assignment	40)
authorisation_removal	24)
base_table_removal	21)
between_condition	50)
column_constraint_definition	9)
column_definition	8)
comparison_condition	48)
comparison_predicate	49)
conditional_check	45)
conditional_expression	42)
conditional_factor	44)
conditional_term	43)
constraint_definition	13)
default_value_definition	4)
domain_alteration	17)
domain_constraint_definition	5)
domain_definition	3)
domain_removal	20)
escape	69)
exists_condition	56)
expression_SELECT	36)
general_constraint_removal	23)
in_condition	52)
insertion	38)
join_type	28)
like_condition	51)
match_condition	54)
null_check	53)
numeric_expression	59)
numeric_factor	61)
numeric_term	60)
object	16)
pattern	68)
primary_condition	46)
primary_number	62)
primary_set_table	33)
primary_string	66)

primary_table	32)
privilege	15)
privilege_definition	14)
removal	41)
row_constructor	35)
scalar_expression	58)
schema_definition	1)
schema_element	2)
schema_removal	19)
selected_item	37)
simple_condition	47)
string_concatenation	65)
string_expression	64)
system_function	70)
table	67)
table_alteration	18)
table_constraint_definition	11)
table_constructor	34)
table_definition	7)
table_expression	25)
table_join_expression	26)
table_reference	27)
table_set_expression	29)
table_set_term	30)
table_term	31)
time_function	72)
unique_condition	57)
update	39)
user_function	71)
view_definition	12)
view_removal	22)
when_to_check	6)

SECOND PART: STANDARD SQL IN ORACLE (DEFINITION LANGUAGE: TABLES AND TRIGGERS)

In what follows we present the syntax of the instructions in Oracle which do not follow standar SQL:

Definition of a basic relation

```
1) basic_relation_definition ::= CREATE TABLE relation_name
                                    (commalist_basic_relation_element)
2) basic_relation_element ::= attribute_definition
                              | relation_constraint
3) attribute_definition ::= attribute_name datatypes
                            [DEFAULT (expression)]
                            [list_attribute_constraint]
4) datatypes ::=       | CHAR (length)
                       | VARCHAR (length)
                       | NUMBER [(precision[, scale])]
                       | DATE
5) attribute_constraint ::=  [CONSTRAINT constraint_name]
                       {[NOT] NULL
                        | UNIQUE
                        | PRIMARY KEY
                        | REFERENCES relation_name [(attribute_name)]
                                     [ON DELETE CASCADE]
                        | CHECK (conditional_expression) }
                       [when_to_check]

6) relation_constraint ::=
          [CONSTRAINT constraint_name]
          { UNIQUE (commalist_attribute_name)
          | PRIMARY KEY (commalist_attribute_name)
          | FOREIGN KEY (commalist_attribute_name)
                REFERENCES relation_name [(commalist_attribute_name)]
                                         [ON DELETE CASCADE]
          | CHECK (conditional_expression)}
          [when_to_check]

7) when_to_check:=
    [[NOT] DEFERRABLE] [INITIALLY {IMMEDIATE | DEFERRED}]
8) relation_alteration ::= ALTER TABLE relation_name
                             {ADD (commalist_basic_relation_element)
                              | MODIFY (commalist_attribute_definition)
                              | {DROP
                                 | [VALIDATE | NOVALIDATE] ENABLE
                                 | DISABLE }                (constraint) }
```

9) constraint ::= {PRIMARY [CASCADE]
 | UNIQUE (*commalist_attribute_name*) [CASCADE]
 | CONSTRAINT *constraint_name* }

10) view_definition ::= CREATE [OR REPLACE] VIEW *view_name*
 [(*commalist_attribute_name*)] AS expression_*SELECT*
 [WITH CHECK OPTION]

11) grant_operation_definition ::= GRANT commalist_privilege
 TO {PUBLIC | *commalist_user*}
 [WITH ADMIN OPTION]

12) rule_definition **:=**
{CREATE | REPLACE} TRIGGER *rule_name*
 {BEFORE | AFTER | INSTEAD OF} event [event_disjunction]
 ON {*relation_name* | *view_name*}
 [[REFERENCING OLD AS *reference_name* [NEW AS *reference_name*]]
 [FOR EACH ROW [WHEN (*conditional_expression)*]]
 PL/SQL block

13) event_disjunction := OR event [event_disjunction]

14) event := INSERT | DELETE | UPDATE [OF *commalist_attribute_name*]

Annex: Language PL/SQL

<u>Structure of a PL/SQL block:</u>
DECLARE Section for variable declaration;
BEGIN Block sentences; END

<u>Section for variable declaration:</u>
variable_name datatype
datatype::= {NUMBER | CHAR( ) | DATE }

<u>Sentences in a PL/SQL block:</u>
sentence_sequence::= sentence; [sentence_sequence;]
IF condition **THEN** sentence_sequence
 [**ELSE** sentence_sequence] **END IF;**
WHILE condition **LOOP** sentence_sequence; **END LOOP;**
FOR *counter* **IN** *minimum .. maximum* **LOOP** sentence_sequence ; **END LOOP**
- Assignment: *variable_name* **:=** *expression*
- Sentences SQL: INSERT, DELETE, UPDATE, SELECT... INTO...
Error handling: RAISE_APPLICATION_ERROR (*error_number,* ' *message'*)
Input-output sentences: dbms_output.put_line ('*message*').

In order to ease the handling of the syntax for Oracle, in what follows we include all the syntactic categories seen above in alphabetical order. We indicate the number which shows the order to be used to locate them.