12 Access control

12.1 <grant statement>

Function

Define privileges and role authorizations.

Format

```
<grant statement> ::=
        <grant privilege statement>
      | <grant role statement>
```


Syntax Rules

None.

Access Rules

None.

General Rules

1) For every involved grantee *G* and for every domain *D1* owned by *G*, if all of the following are true:

   a) The applicable privileges of *G* include the grantable REFERENCES privilege on every column referenced in the <search condition> *SC* included in a domain constraint descriptor included in the domain descriptor of *D1*.

   b) The applicable privileges of *G* include the grantable EXECUTE privileges on all SQL-invoked routines that are subject routines of <routine invocation>s contained in *SC*.

   c) The applicable privileges of *G* include the grantable SELECT privilege on every table *T1* and every method *M* such that there is a <method reference> *MR* contained in *SC* such that *T1* is in the scope of the <value expression primary> of *MR* and *M* is the method identified by the <method name> of *MR* included in a domain constraint descriptor included in the domain descriptor of *D1*.

   d) The applicable privileges of *G* include the grantable SELECT privilege WITH HIERARCHY OPTION on at least one supertable of the scoped table of every <reference resolution> contained in *SC*.

   e) The applicable privileges of *G* include the grantable USAGE privilege on all domains, character sets, collations, and translations whose <domain name>s, <character set name>s, <collation name>s, and <translation name>s, respectively, are included in the domain descriptor of *D1*.

then for every privilege descriptor with <action> USAGE, a grantor of "_SYSTEM", object *D1*, and grantee *G* that is not grantable, the following <grant statement> is effectively executed with a current user identifier of "_SYSTEM" and without further Access Rule checking:

```
GRANT USAGE ON DOMAIN D1 TO G WITH GRANT OPTION
```

2) For every involved grantee *G* and for every collation *C1* owned by *G*, if the applicable privileges of *G* include a grantable USAGE privilege for the character set name included in the collation descriptor of *C1* and a grantable USAGE privilege for the translation name, if any, included in the collation descriptor of *C1*, then for every privilege descriptor with <action> USAGE, a grantor of "_SYSTEM", object of *C1*, and grantee *G* that is not grantable, the following <grant statement> is effectively executed with a current user identifier of "_SYSTEM" and without further Access Rule checking:

```
GRANT USAGE ON COLLATION C1 TO G WITH GRANT OPTION
```

3) For every involved grantee *G* and for every translation *T1* owned by *G*, if the applicable privileges of *G* contain a grantable USAGE privilege for every character set identified by a <character set specification> contained in the <translation definition> of *T1*, then for every privilege descriptor with <action> *P*, a grantor of "_SYSTEM", object of *T1*, and grantee *G* that is not grantable, the following <grant statement> is effectively executed as though the current user identifier were "_SYSTEM" and without further Access Rule checking:


```
GRANT P ON TRANSLATION T1 TO G WITH GRANT OPTION
```

4) For every table *T* specified by some involved privilege descriptor and for each view *V* owned by some involved grantee *G* such that *T* or some column *CT* of *T* is referenced in the <query expression> *QE* of *V*, or *T* is a supertable of the scoped table of a <reference resolution> contained in *QE*, let RT_i, for *i* ranging from 1 (one) to the number of tables identified by the <table reference>s contained in *QE*, be the <table name>s of those tables. For every column *CV* of *V*:

a) Let CRT_{ij}, for *j* ranging from 1 (one) to the number of columns of RT_i that are underlying columns of *CV*, be the <column name>s of those columns.

b) If, following successful execution of the <grant statement>, all of the following are true:

i) The applicable privileges of *G* include grantable SELECT privileges on all of the columns CRT_{ij}.

ii) The applicable privileges of *G* include grantable EXECUTE privileges on all SQL-invoked routines that are subject routines of <routine invocation>s contained in *QE*.

iii) The applicable privileges of *G* include grantable SELECT privilege on every table *T1* and every method *M* such that there is a <method reference. *MR* contained in *QE* such that *T1* is in the scope of the <value expression primary> of *MR* and *M* is the method identified by the <method name> of *MR*.

iv) The applicable privileges of *G* include grantable SELECT privilege WITH HIERARCHY OPTION on at least one supertable of the scoped table of every <reference resolution> that is contained in *QE*.

then the following <grant statement> is effectively executed as though the current user identifier were "_SYSTEM" and without further Access Rule checking:

```
GRANT SELECT (CV) ON V TO G WITH GRANT OPTION
```

c) If, following successful execution of the <grant statement>, the applicable privileges of *G* will include REFERENCES(CRT_{ij}) for all *i* and for all *j*, and will include a REFERENCES privilege on some column of RT_i for all *i*, then:

i) Case:

1) If all of the following are true:

A) The applicable privileges of *G* will include grantable REFERENCES(CRT_{ij}) for all *i* and for all *j*, and will include a grantable REFERENCES privilege on some column of RT_i for all *i*.

B) The applicable privileges of *G* include grantable EXECUTE privileges on all SQL-invoked routines that are subject routines of <routine invocation>s contained in *QE*.

C) The applicable privileges of *G* include grantable SELECT privilege on every table *T1* and every method *M* such that there is a <method reference. *MR* contained in *QE* such that *T1* is in the scope of the <value expression primary> of *MR* and *M* is the method identified by the <method name> of *MR*.

D) The applicable privileges of *G* include grantable SELECT privilege WITH HIERARCHY OPTION on at least one supertable of the scoped table of every <reference resolution> that is contained in *QE*.

then let *WGO* be "WITH GRANT OPTION".

2) Otherwise, let *WGO* be a zero-length string.

ii) The following <grant statement> is effectively executed as though the current user identifier were "_SYSTEM" and without further Access Rule checking:

```
GRANT REFERENCES (CV) ON V TO G WGO
```

d) If, following successful execution of the <grant statement>, the applicable privileges of *G* include grantable SELECT privilege on every column of *V*, then the following <grant statement> is effectively executed as though the current user identifier were "_SYSTEM" and without further Access Rule checking:

```
GRANT SELECT ON V TO G WITH GRANT OPTION
```

e) Following successful execution of the <grant statement>,

Case:

i) If the applicable privileges of *G* include REFERENCES privilege on every column of *V*, then let *WGO* be a zero-length string.

ii) If the applicable privileges of *G* include grantable REFERENCES privilege on every column of *V*, then let *WGO* be "WITH GRANT OPTION".

iii) The following <grant statement> is effectively executed as though the current user identifier were "_SYSTEM" and without further Access Rule checking:

```
GRANT REFERENCES ON V TO G WITH GRANT OPTION
```

5) Following the successful execution of the <grant statement>, for every table *T* specified by some involved privilege descriptor and for every updatable view *V* owned by some grantee *G* such that *T* is some leaf underlying table of the <query expression> of *V*:

   a) Let *VN* be the <table name> of *V*.

   b) If *QE* is fully updatable with respect to *T*, and the applicable privileges of *G* include *PA*, where *PA* is either INSERT, UPDATE, or DELETE, then the following <grant statement> is effectively executed as though the current user identifier were "_SYSTEM" and without further Access Rule checking:

      GRANT *PA* ON *VN* TO *G*

   c) If *QE* is fully updatable with respect to *T*, and the applicable privileges of *G* include grantable *PA* privilege on *T*, where *PA* is either INSERT, UPDATE, or DELETE, then the following <grant statement> is effectively executed as though the current user identifier were "_SYSTEM" and without further Access Rule checking:

      GRANT *PA* ON *VN* TO *G* WITH GRANT OPTION

   d) For each column *CV* of *V*, named *CVN*, that has a counterpart *CT* in *T*, named *CTN*, if *QE* is fully or partially updatable with respect to *T*, and the applicable privileges of *G* include *PA*(*CTN*) privilege on *T*, where *PA* is INSERT or UPDATE, then the following <grant statement> is effectively executed as though the current user identifier were "_SYSTEM" and without further Access Rule checking:

      GRANT *PA*(*CVN*) ON *VN* TO *G*

   e) For each column *CV* of *V*, named *CVN*, that has a counterpart *CT* in *T*, named *CTN*, if *QE* is fully or partially updatable with respect to *T*, and the applicable privileges of *G* include grantable *PA*(*CTN*) privilege on *T*, where *PA* is INSERT or UPDATE, then the following <grant statement> is effectively executed as though the current user identifier were "_SYSTEM" and without further Access Rule checking:

      GRANT *PA*(*CVN*) ON *VN* TO *G* WITH GRANT OPTION

6) For every involved grantee *G* and for every referenceable view *V*, named *VN*, owned by *G*, if following the successful execution of the <grant statement>, the applicable privileges of *G* include grantable UNDER privilege on the direct supertable of *V*, then the following <grant statement> is effectively executed with a current authorization identifier of "_SYSTEM" and without further Access Rule checking:

   GRANT UNDER ON *VN* TO *G* WITH GRANT OPTION

7) For every involved grantee *G* and for every schema-level SQL-invoked routine *R1* owned by *G*, if the applicable privileges of *G* contain all of the privileges necessary to successfully execute every <SQL procedure statement> contained in the <routine body> of *R1* are grantable, then for every privilege descriptor with <action> EXECUTE, a grantor of "_SYSTEM", object of *R1*, and grantee *G* that is not grantable, the following <grant statement> is effectively executed with a current authorization identifier of "_SYSTEM" and without further Access Rule checking:

   GRANT EXECUTE ON *R1* TO *G* WITH GRANT OPTION

   NOTE 268 – The privileges necessary include the EXECUTE privilege on every subject routine of every <routine invocation> contained in the <SQL procedure statement>.

8) If two privilege descriptors are identical except that one indicates that the privilege is grantable and the other indicates that the privilege is not grantable, then both privilege descriptors are set to indicate that the privilege is grantable.

9) If two privilege descriptors are identical except that one indicates WITH HIERARCHY OPTION and the other does not, then both privilege descriptors are set to indicate that the privilege has the WITH HIERARCHY OPTION.

10) Redundant duplicate privilege descriptors are removed from the multiset of all privilege descriptors.

Conformance Rules

None.

12.2 <grant privilege statement>

Function

Define privileges.

Format

```
<grant privilege statement> ::=
       GRANT <privileges>
         TO <grantee> [ { <comma> <grantee> }... ]
           [ WITH HIERARCHY OPTION ]
           [ WITH GRANT OPTION ]
           [ GRANTED BY <grantor> ]
```

Syntax Rules

1) Let *O* be the object identified by the <object name> contained in <privileges>.

2) Let *U* be the current user identifier and let *R* be the current role name.

3) Case:

   a) If GRANTED BY <grantor> is not specified, then

      Case:

      i) If *U* is not the null value, then let *A* be *U*.

      ii) Otherwise, let *A* be *R*.

   b) If GRANTED BY CURRENT_USER is specified, then let *A* be *U*.

   c) If GRANTED BY CURRENT_ROLE is specified, then let *A* be *R*.

4) A set of privilege descriptors is identified. The privilege descriptors identified are those defining, for each <action> explicitly or implicitly in <privileges>, that <action> on *O* held by *A* with grant option.

5) If the <grant statement> is not contained in a <schema definition>, then the schema identified by the explicit or implicit qualifier of the <object name> shall include the descriptor of *O*. If the <grant statement> is contained in a <schema definition> *S*, then the schema identified by the explicit or implicit qualifier of the <object name> shall include the descriptor of *O* or *S* shall include a <schema element> that creates the descriptor of *O*.

6) If WITH HIERARCHY OPTION is specified, then:

   a) <privileges> shall specify an <action> of SELECT without a <privilege column list> and without a <privilege method list>.

   b) *O* shall be a table of a structured type.

Access Rules

1) The applicable privileges shall include a privilege identifying *O*.
   NOTE 269 – "applicable privileges" are defined in Subclause 10.5, "<privileges>".

General Rules

1) The <object privileges> specify one or more privileges on the object identified by the <object name>.

2) For every identified privilege descriptor *IPD*, a privilege descriptor is created for each <grantee>, that specifies grantee <grantee>, action <action>, object *O*, and grantor *A*. Let *CPD* be the set of privilege descriptors created.

3) For every privilege descriptor in *CPD* whose action is INSERT, UPDATE, or REFERENCES without a column name, privilege descriptors are also created and added to *CPD* for each column *C* in *O* for which *A* holds the corresponding privilege with grant option. For each such column, a privilege descriptor is created that specifies the identical <grantee>, the identical <action>, object *C*, and grantor *A*.

4) For every privilege descriptor in *CPD* whose action is SELECT without a column name or method name, privilege descriptors are also created and added to *CPD* for each column *C* in *O* for which *A* holds the corresponding privilege with grant option. For each such column, a privilege descriptor is created that specifies the identical <grantee>, the identical <action>, object *C*, and grantor *A*.

5) For every privilege descriptor in *CPD* whose action is SELECT without a column name or method name, if the table *T* identified by the object of the privilege descriptor is a table of a structured type *TY*, then table/method privilege descriptors are also created and added to *CPD* for each method *M* of *TY* for which *A* holds the corresponding privilege with grant option. For each such method, a table/method privilege descriptor is created that specifies the identical <grantee>, the identical <action>, object consisting of the pair of table *T* and method *M*, and grantor *A*.

6) If WITH GRANT OPTION was specified, then each privilege descriptor also indicates that the privilege is grantable.

7) Let *SWH* be the set of privilege descriptors in *CPD* whose action is SELECT WITH HIERARCHY OPTION, and let *ST* be the set of subtables of *O*, then for every grantee *G* in *SWH* and for every table *T in* ST, the following <grant statement> is effectively executed without further Access Rule checking:

```
GRANT SELECT ON T TO G GRANTED BY A
```

8) For every combination of <grantee> and <action> on *O* specified in <privileges>, if there is no corresponding privilege descriptor in *CPD*, then a completion condition is raised: *warning — privilege not granted*.

9) If ALL PRIVILEGES was specified, then for each grantee *G*, if there is no privilege descriptor in *CPD* specifying grantee *G*, then a completion condition is raised: *warning — privilege not granted*.

10) The *set of involved privilege descriptors* is defined to be *CPD*.

11) The *set of involved grantees* is defined as the set of specified <grantee>s.

Conformance Rules

1) Without Feature S024, "Enhanced structured types", a <specific routine designator> contained in a <grant statement> shall not identify a method.

2) Without Feature S081, "Subtables", conforming SQL language shall not specify WITH HIERARCHY OPTION.

12.3 <role definition>

Function

Define a role.

Format

```
<role definition> ::=
       CREATE ROLE <role name>
         [ WITH ADMIN <grantor> ]
```

Syntax Rules

1) The specified <role name> shall not be equivalent to any other <authorization identifier> in the SQL-environment.

Access Rules

1) The privileges necessary to execute the <role definition> are implementation-defined.

General Rules

1) A <role definition> defines a role.

2) Let *U* be the current user identifier and *R* be the current role name.

3) Case:

   a) If WITH ADMIN <grantor> is not specified, then

      Case:

      i) If *U* is not the null value, then let *A* be *U*.

      ii) Otherwise, let *A* be *R*.

   b) If WITH ADMIN CURRENT_USER is specified, then let *A* be *U*.

   c) If WITH ADMIN CURRENT_ROLE is specified, then let *A* be *R*.

4) A role authorization descriptor is created that identifies that the role identified by <role name> has been granted to *A* WITH ADMIN OPTION, with a grantor of "_SYSTEM".

Conformance Rules

1) Without Feature T331, "Basic roles", conforming SQL language shall contain no <role definition>.

2) Without Feature T332, "Extended roles", conforming SQL language shall not specify WITH ADMIN.

12.4 <grant role statement>

Function

Define role authorizations.

Format

```
<grant role statement> ::=
      GRANT <role granted> [ { <comma> <role granted> }... ]
        TO <grantee> [ { <comma> <grantee> }... ]
          [ WITH ADMIN OPTION ]
          [ GRANTED BY <grantor> ]

<role granted> ::= <role name>
```

Syntax Rules

1) No role identified by a specified <grantee> shall be contained in any role identified by a specified <role granted>; that is, no cycles of role grants are allowed.

2) Let *U* be the current user identifier and *R* be the current role name.

3) Case:

   a) If GRANTED BY <grantor> is not specified, then

      Case:

      i) If *U* is not the null value, then let *A* be *U*.

      ii) Otherwise, let *A* be *R*.

   b) If GRANTED BY CURRENT_USER is specified, then let *A* be *U*.

   c) If GRANTED BY CURRENT_ROLE is specified, then let *A* be *R*.

Access Rules

1) Every role identified by <role granted> shall be contained in the applicable roles for *A* and the corresponding role authorization descriptors shall specify WITH ADMIN OPTION.

General Rules

1) For every <grantee> specified, a set of role authorization descriptors is created that defines the grant of each role identified by a <role granted> to the <grantee> with a grantor of *A*.

2) If WITH ADMIN OPTION is specified, then each role authorization descriptor also indicates that the role is grantable with the WITH ADMIN OPTION.

3) If two role authorization descriptors are identical except that one indicates that the role is grantable WITH ADMIN OPTION and the other indicates that the role is not, then both role authorization descriptors are set to indicate that the role is grantable with the WITH ADMIN OPTION.

4) Redundant duplicate role authorization descriptors are removed from the multiset of all role authorization descriptors.

5) The *set of involved privilege descriptors* is the union of the sets of privilege descriptors corresponding to the applicable privileges of every <role granted> specified.

6) The *set of involved grantees* is the union of the set of <grantee>s and the set of <role name>s that contain at least one of the <role name>s that is possibly specified as a <grantee>.

Conformance Rules

1) Without Feature T331, "Basic roles", conforming SQL language shall contain no <grant role statement>.

2) Without Feature T332, "Extended roles", conforming SQL language shall not specify <grantor>.

12.5 <drop role statement>

Function

Destroy a role.

Format

```
<drop role statement> ::= DROP ROLE <role name>
```

Syntax Rules

1) Let *R* be the role identified by the specified <role name>.

Access Rules

1) At least one of the enabled authorization identifiers shall have a role authorization identifier that authorizes *R* with the WITH ADMIN OPTION.

General Rules

1) Let *A* be any <authorization identifier> identified by a role authorization descriptor as having been granted to *R*.

2) The following <revoke role statement> is effectively executed without further Access Rule checking:

```
REVOKE R FROM A
```

3) The descriptor of *R* is destroyed.

Conformance Rules

1) Without Feature T331, "Basic roles", conforming SQL language shall contain no <drop role statement>.

12.6 <revoke statement>

Function

Destroy privileges and role authorizations.

Format

```
<revoke statement> ::=
        <revoke privilege statement>
      | <revoke role statement>

<revoke privilege statement> ::=
      REVOKE [ <revoke option extension> ] <privileges>
        FROM <grantee> [ { <comma> <grantee> }... ]
          [ GRANTED BY <grantor> ]
          <drop behavior>

<revoke option extension> ::=
        GRANT OPTION FOR
      | HIERARCHY OPTION FOR

<revoke role statement> ::=
      REVOKE [ ADMIN OPTION FOR ]
        <role revoked> [ { <comma> <role revoked> }... ]
        FROM <grantee> [ { <comma> <grantee> }... ]
          [ GRANTED BY <grantor> ]
          <drop behavior>

<role revoked> ::= <role name>
```


Syntax Rules

1) Let *O* be the object identified by the <object name> contained in <privileges>. If *O* is a table *T*, then let *S* be the set of subtables of *O*. If *T* is a table of a structured type, then let *TY* be that type.

2) If WITH HIERARCHY OPTION is specified, the <privileges> shall specify an <action> of SELECT without a <privilege column list> and without a <privilege method list> and *O* shall be a table of a structured type.

3) Let *U* be the current user identifier and *R* be the current role name.

4) Case:

   a) If GRANTED BY <grantor> is not specified, then

      Case:

      i) If *U* is not the null value, then let *A* be *U*.

      ii) Otherwise, let *A* be *R*.

   b) If GRANTED BY CURRENT_USER is specified, then let *A* be *U*.

   c) If GRANTED BY CURRENT_ROLE is specified, then let *A* be *R*.

5) SELECT is equivalent to specifying both the SELECT table privilege and SELECT (<privilege column list>) for all columns of <table name>. If *T* is a table of a structured type *TY*, then SELECT also specifies SELECT (<privilege column list>) for all columns inherited from *T* in each of the subtables of *T*, and SELECT (<privilege method list>) for all methods of *TY* in each of the subtables of *T*.

6) INSERT is equivalent to specifying both the INSERT table privilege and INSERT (<privilege column list>) for all columns of <table name>.

7) UPDATE is equivalent to specifying both the UPDATE table privilege and UPDATE (<privilege column list>) for all columns of <table name>, as well as UPDATE (<privilege column list>) for all columns inherited from *T* in each of the subtables of *T*.

8) REFERENCES is equivalent to specifying both the REFERENCES table privilege and REFERENCES (<privilege column list>) for all columns of <table name>, as well as REFERENCES (<privilege column list>) for all columns inherited from *T* in each of the subtables of *T*.

9) Case:

   a) If the <revoke statement> is a <revoke privileges statement>, then for every <grantee> specified, a set of privilege descriptors is identified. A privilege descriptor *P* is said to be *identified* if it belongs to the set of privilege descriptors that defined, for any <action> explicitly or implicitly in <privileges>, that <action> on *O*, or any of the objects in *S*, granted by *A* to <grantee>.

      NOTE 270 – Column privilege descriptors become identified when <action> explicitly or implicitly contains a <privilege column list>. Table/method descriptors become identified when <action> explicitly or implicitly contains a <privilege method list>.

   b) If the <revoke statement> is a <revoke role statement>, then for every <grantee> specified, a set of role authorization descriptors is identified. A role authorization descriptor is said to be *identified* if it defines the grant of any of the specified <role revoked>s to <grantee> with grantor *A*.

10) A privilege descriptor *D* is said to be *directly dependent on* another privilege descriptor *P* if either:

   a) All of the following conditions hold:

      i) *P* indicates that the privilege that it represents is grantable.

      ii) The grantee of *P* is the same as the grantor of *D* or the grantee of *P* is PUBLIC, or, if the grantor of *D* is a <role name>, the grantee of *P* belongs to the set of applicable roles of the grantor of *D*.

      iii) Case:

         1) *P* and *D* are both column privilege descriptors. The action and the identified column of *P* are the same as the action and identified column of *D*, respectively.

         2) *P* and *D* are both table privilege descriptors. The action and the identified table of *P* are the same as the action and the identified table of *D*, respectively.

         3) *P* and *D* are both execute privilege descriptors. The action and the identified SQL-invoked routine of *P* are the same as the action and the identified SQL-invoked routine of *D*, respectively.

4) *P* and *D* are both usage privilege descriptors. The action and the identified domain, character set, collation, translation, or user-defined type of *P* are the same as the action and the identified domain, character set, collation, translation, or user-defined type of *D*, respectively.

5) *P* and *D* are both under privilege descriptors. The action and the identified user-defined type or table of *P* are the same as the action and the identified user-defined type or table of *D*, respectively.

6) *P* and *D* are both table/method privilege descriptors. The action and the identified method and table of *P* are the same as the action and the identified method and table of *D*, respectively.

b) All of the following conditions hold:

i) The privilege descriptor for *D* indicates that its grantor is the special grantor value "_SYSTEM".

ii) The action of *P* is the same as the action of *D*.

iii) The grantee of *P* is the owner of the table, collation, or translation identified by *D* or the grantee of *P* is PUBLIC.

iv) One of the following conditions hold:

1) *P* and *D* are both table privilege descriptors, the privilege descriptor for *D* identifies the <table name> of an updatable view *V*, and the identified table of *P* is the underlying table of the <query expression> of *V*.

2) *P* and *D* are both column privilege descriptors, the privilege descriptor *D* identifies a <column name> *CVN* explicitly or implicitly contained in the <view column list> of a <view definition> *V*, and one of the following is true:

A) *V* is an updatable view. For every column *CV* identified by a <column name> *CVN*, there is a corresponding column in the underlying table of the <query expression> *TN*. Let *CTN* be the <column name> of the column of the <query expression> from which *CV* is derived. The action for *P* is UPDATE or INSERT and the identified column of *P* is *TN.CTN*.

B) For every table *T* identified by a <table reference> contained in the <query expression> of *V* and for every column *CT* that is a column of *T* and an underlying column of *CV*, the action for *P* is REFERENCES and either the identified column of *P* is *CT* or the identified table of *P* is *T*.

C) For every table *T* identified by a <table reference> contained in the <query expression> of *V* and for every column *CT* that is a column of *T* and an underlying column of *CV*, the action for *P* is SELECT and either the identified column of *P* is *CT* or the identified table of *P* is *T*.

3) The privilege descriptor *D* identifies the <collation name> of a <collation definition> *CO* and the identified character set name of *P* is included in the collation descriptor for *CO*, or the identified translation name of *P* is included in the collation descriptor for *CO*.