

# CQL Specification

Draft V1.3

2002-4-25  
**TOSHIBA Corporation**

1.	<i>Scope and Definition</i> .....	5
2.	<i>Type of CQL</i> .....	5
3.	<b>SEARCH</b> .....	6
3.1.	<b>Proper Class Search</b> .....	7
3.2.	<b>General Class Search</b> .....	7
3.3.	<b>Boolean operations between classes</b> .....	8
3.4.	<b>Ancestral Search</b> .....	9
4.	<i>Predefined Class</i> .....	9
4.1.	<b>Project</b> .....	9
4.2.	<b>Library</b> .....	10
4.3.	<b>Resource</b> .....	10
4.4.	<b>User</b> .....	10
5.	<b>CML</b> .....	10
5.1.1.	create .....	10
5.1.2.	alter.....	10
5.1.3.	drop .....	10
5.1.4.	commit.....	10
5.2.	<b>dictionary</b> .....	11
5.2.1.	create .....	11
5.2.2.	alter.....	11
5.2.3.	drop .....	11
5.3.	<b>dictionary_supplier</b> .....	11
5.3.1.	create .....	11
5.3.2.	alter.....	11
5.3.3.	drop .....	11
5.4.	<b>class</b> .....	12
5.4.1.	create .....	12
5.4.2.	alter.....	12
5.4.3.	drop .....	12
5.4.4.	rename .....	12
5.5.	<b>property</b> .....	12
5.5.1.	create .....	12
5.5.2.	alter.....	13
5.5.3.	drop .....	13
5.5.4.	rename .....	13
6.	<b>DRL</b> .....	13
6.1.	<b>extension</b> .....	13
6.1.1.	create .....	13
6.1.2.	alter.....	13
6.1.3.	drop .....	14
6.2.	<b>table</b> .....	14
6.2.1.	create .....	14
6.2.2.	alter.....	14
6.2.3.	drop .....	14

<b>7.</b>	<b><i>LML</i></b> .....	<b>14</b>
<b>7.1.</b>	<b>insert</b> .....	<b>14</b>
<b>7.2.</b>	<b>update</b> .....	<b>15</b>
<b>7.3.</b>	<b>delete</b> .....	<b>15</b>
<b>8.</b>	<b><i>SET</i></b> .....	<b>15</b>
<b>9.</b>	<b><i>Security</i></b> .....	<b>15</b>
<b>9.1.</b>	<b>grant</b> .....	<b>16</b>
<b>9.2.</b>	<b>revoke</b> .....	<b>16</b>
<b>10.</b>	<b><i>Command List</i></b> .....	<b>16</b>
<b>11.</b>	<b><i>Common Definition</i></b> .....	<b>23</b>

### Documentation History

Date	Author	Note of Action	CQLVersion	Document Version
25/10/01	Yumiko MIZOGUCHI	First Draft in English	Ver. 1.1	1.0
27/10/01	Hiroshi MURAYAMA	Corrigendum & Proposal	Ver. 1.1	1.1
12/12/01	Yumiko MIZOGUCHI	Update Class_identifier (p19) Property identifier (p21) Extension_definition (p11/p21)	Ver. 1.1	1.2
04/04/02	Yumiko MIZOGUCHI	Add Boolean operation Ancestral Search Update Command List(p14)	Ver.1.2	1.3
24/04/02	Yumiko MIZOGUCHI	Add Predefined Class(p9) Security(p15) insert explanation of <reference_condition>(p15) Update Command List(p16) Command Definition (p22)	Ver.1.3	1.4
25/04/02			Ver.1.4	1.5

## 1. Scope and Definition

The CQL is an abbreviation for Class Query Language and is used to query information stored in a class-theoretic database (hereafter CDB). The aim of this language specification is to define a declarative query language for hierarchical class libraries, i.e., data are assumed to be stored as instances of a class whose collection forms an acyclic graph structure(ACGS)..

In this respect, this query language is not designed only for the retrieval of information stored in a PLIB-LMS( ISO13584 based Library Management System), but is rather designed as a general purpose data query language for CDB-like databases. Nevertheless, PLIB-LMS is one of the major applications for which this CQL approach might be especially attractive and practical. In fact, CQL is an abstraction and an extension of the basic concept described in Part42 of ISO13584. Another possible application area of this language might be ERDL( ISO15926 Part4; Epistle Reference Data Library ). However, the current version of the CQL is still a small subset of the full-scope CQL and could be considered to be at the level of “PQL”( PLIB Query language). We plan to develop this as a full scope neutral language for data modelling and manipulation for various class libraries based on different de-jure and de-facto standards, and help migration data between them.

CQL includes SQL as its subset, so it is a superset of the SQL. It does not mean, however, that the implementation of the CQL must be based on a Relational Database System. The implementation may be built on an Object-Oriented Database system, regardless of its superficial language syntax being resembling to those of SQL.

One of the major differences of the CQL with respect to SQL-99( once named SQL-3) is that CQL assumes that every record of data is classified in a node under ASGS. In other words, data must be an instance of some class which itself must form a set-theoretic subset of another class, except the case that it is the root of the class hierarchy. All the root classes are bundled together by a virtual entity named “virtual root” that corresponds to Universal Class (Universal Set) in mathematics.

#Note: A (simple) tree structure is a subset of acyclic graph structure.

## 2. Type of CQL

Type Name	Command	Definition
Class Modeling Language (CML)	create dictionary create dic_supplier create class create property alter dictionary alter dic_supplier alter class alter property drop dictionary drop dic_supplier drop class drop property rename dictionary rename dic_supplier rename class rename property	Define and Manipulate dictionary Part42, including case_of.
Data Representation Language (DRL)	create table create extension	Define and Manipulate containers for contents.

	alter table alter extension drop table drop extension	
Library Manipulation Language (LML)	insert update delete	Manipulate contents
Search	select	Search contents.
Session	connect	Create session
transaction control	commit	Control version, status
Environment Definition Language	set	Define defaults environment.

**Table 1**Type of CQL

### 3. SEARCH

To search contents, we use a select expression.  
The general form of a select expression is following.

```
"select" ["distinct"]
"*"| < displayed_property > { "," < displayed_property > }
"from" (< selected_class > { "," < selected_class > })
[ "where" < search_condition > ]
[ "order by" < order_property > [ "asc" | "desc" ] ]
```

**example)**

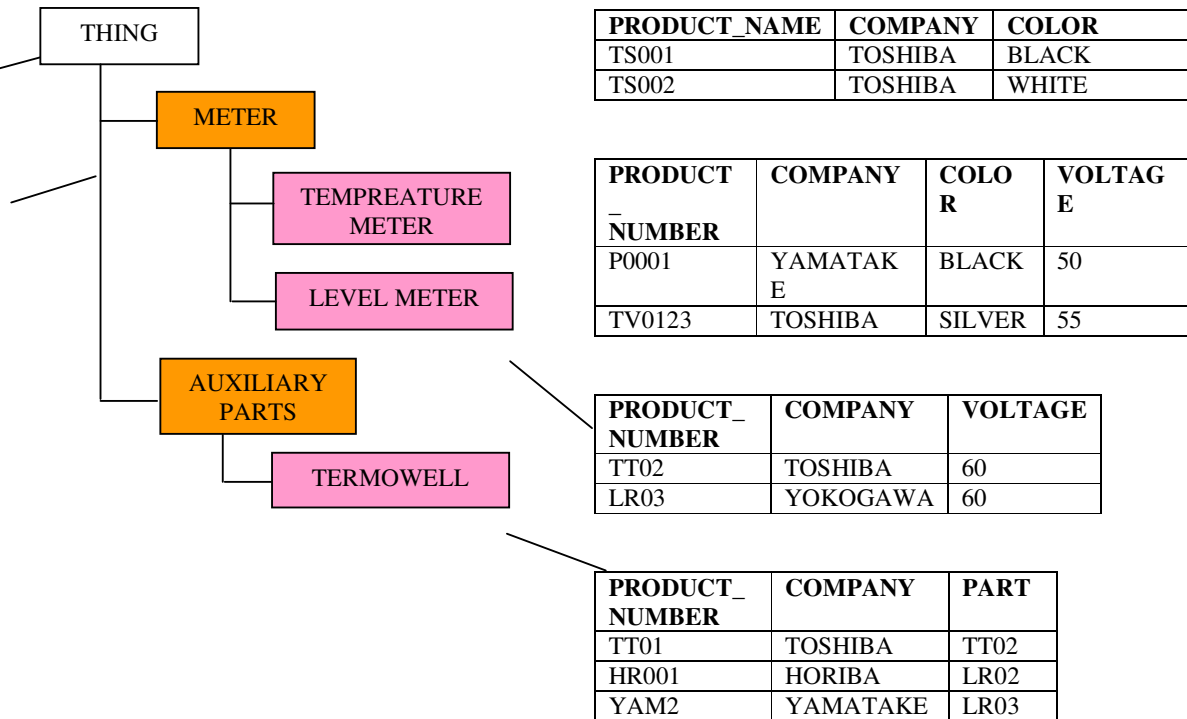


Figure 1 sample1

### 3.1. Proper Class Search

```
select PRODUCT_NUMBER, COMPANY from TEMPRATURE_METER
where COMPANY = 'TOSHIBA';
```

(We assume that PRODUCT\_NUMBER, COMPANY are property\_identifiers and TEMPRATURE\_METER is class\_identifier.)

CQL engine returns below

PRODUCT_NUMBER	COMPANY
TV0123	TOSHIBA

### 3.2. General Class Search

In Figure 1 example, it is assumed that query system will search the given class and its subclasses with given search conditions..

```
select PRODUCT_NUMBER, COMPANY from THING* where COMPANY='TOSHIBA';
```

Note that '\*' is added after THING. The asterisk denotes that any subclass of the THING shall be searched for the value of the attribute COMPANY being equal to 'TOSHIBA'

This returns the following:

PRODUCT_NUMBER	COMPANY
TV0123	TOSHIBA
TS001	TOSHIBA
TS002	TOSHIBA
TT02	TOSHIBA
TT01	TOSHIBA

If you'd like to except "AUXILIARY PARTS", you may except the class from the search scope as:

```
select PRODUCT_NUMBER, COMPANY from THING* except AUXILIARY_PARTS*
where COMPANY='TOSHIBA';
```

# It is assumed in the above that the class\_identifier of "AUXILIARY PARTS" is *AUXILIARY\_PARTS*.

```
select * from THING* where COLOR = 'BLACK';
```

This returns the following.

PRODUCT_NUMBER	COMPANY	COLOR	VOLTAGE
TS001	TOSHIBA	BLACK	
P0001	YAMATAKE	BLACK	50

# If a class doesn't have any corresponding applicable properties for the given conditions, the system removes that class from the search path. ( may be revised !)

### 3.3. Boolean operations between classes

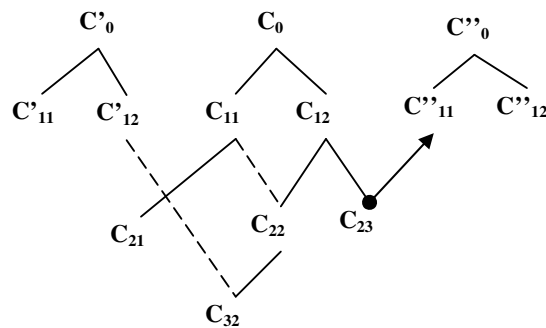


Figure 2 Sample Class Tree

- case of relationship
- class and subclass relationship
- part-whole relationship

In a class hierarchy like the one depicted in *Figure 1*, if the *from* clause is written “*from C<sub>11</sub>*”, the search scope is strictly limited to the class *C<sub>11</sub>* itself. But if the *from* clause is “*from C<sub>11</sub>\**”, the search scope is the class *C<sub>11</sub>* and its all the subclasses. Therefore, AND operation of the class *C<sub>11</sub>\** with the class *C<sub>12</sub>\** means that the search scope is the class *C<sub>22</sub>\**. Likewise, “*from C<sub>11</sub>\* and C<sub>12</sub>\**” yields the scope “*from C<sub>22</sub>\**”.

Meanwhile, subtraction operation, “*from C<sub>0</sub>\* - C<sub>22</sub>\**” means that the class *C<sub>22</sub>* and subclasses are removed from the search scope of *C<sub>0</sub>\**.

### 3.4. Ancestral Search

The Ancestral search extends the scope of search from a child to parent nodes. The expressions are given as in the following;

- select \* from A%*; (1)
- select \* from A!*; (2)

The expression (1) extends the search scope to all the superclasses of *A*, while the expression (10) limits the scope to direct ancestors of *A*. In the example given in *Figure 2*, if the *from* clause after *select* is “*from C<sub>22</sub>%*”, the search scope is extended from *C<sub>22</sub>* to all the ancestors, i.e., *C<sub>11</sub>*, *C<sub>12</sub>* and *C<sub>0</sub>*. On the other hand, if the clause is “*from C<sub>22</sub>!*”, the search scope is limited to only its direct ancestors and excepts the class *C<sub>11</sub>* that exports properties to *C<sub>22</sub>*. Consequently, the search scope is *C<sub>22</sub>*, *C<sub>12</sub>* and *C<sub>0</sub>*.

## 4. Predefined Classes

CQL provides the following four special-purpose classes to administer the LMS. They are predefined.

- 1) *Project*
- 2) *Library*
- 3) *Resource*
- 4) *User*

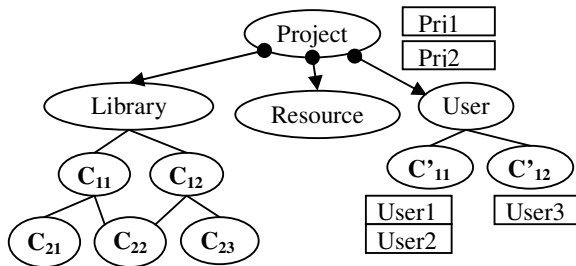


Figure 3 Predefined Class

### 4.1. Project

The *Project* is the basic unit of data management. If a user wants to know about a product catalogue through CQL, he needs to be enrolled in at least one of the projects under *Project* that contains the user in the *USER* or one of its subclasses, and the catalogue must be contained as an instance of some class in one of the library listed in *Library*.

This concept does not derive from PLIB, since the latter as a standard is for the exchange of data contained in electronic files, however, it is necessary for CQL because it assumes the existence of a certain repository of data, or a database system. The *Project* is the *whole* class for the three *Part* classes, i.e., *Library*, *Resource*, and *User*, in other words, *Project* has them .

#### 4.2. Library

The *Library* is the root class of ordinary class hierarchies modeled in a project. From time to time, several libraries may be found in a projects. A library ( in plain letter ) here means a class hierarchy(ies) with product instance data.

#### 4.3. Resource

The *Resource* is the class which stores a physical file such as image files, sound files, or programs, relating to a project.

#### 4.4. User

The *User* is the class which stores information about the users of the project. The *User* class may have subclasses, and each such a class has individual users as instances.

### 5. CML

#### 5.1.1. create

```
“create project” < project_identifier > “(“ <project_definition> “)” “;”
```

Example)

```
create project JEMIMA
( dictionary 'jemima_dictionary',
  create 'yumiko',
  active_date '2001-10-19',
  access_level 'public'
);
```

#### 5.1.2. alter

```
alter project JEMIMA
(dictionary 'jemima_dictionary2'
);
```

#### 5.1.3. drop

```
drop project JEMIMA
```

#### 5.1.4. commit

Example)

```
commit JEMIMA version '1.1.2';
```

## 5.2. dictionary

A project may contain one or more dictionaries. If it contains more than one dictionary, it may be that the “case\_of” concept of PLIB ( in other words, importation of properties from a class of another dictionary) is used. But this is not a prerequisite. Several dictionaries of different coverage may be used in a project for a certain enterprise, in order to delineate the scope of possible choices of products to be used in a business activity. In this case, dictionaries contained in a project may serve as “authorised sets of products” for the enterprise.

### 5.2.1. create

```
“create dictionary” <dictionary_identifier> “(“ <dictionary_definition> “)” “;”
```

Example)

```
create dictionary jemima_dictionary (  
    name ‘JEMIMA DICTONAEY’,  
    version ‘001’,  
    defined_by ‘9901/JEMIMA’  
);
```

### 5.2.2. alter

```
“alter dictionary” <dictionary_identifier> “(“ <dictionary_definition> “)” “;”
```

Example)

```
alter dictionary modify jemima_dictionary (  
    name ‘JEMIMA DIC’  
);
```

### 5.2.3. drop

```
”drop dictionary” <dictionary_identifier>;”
```

Example)

```
drop dictionary jemima_dictionary
```

## 5.3. dictionary\_supplier

### 5.3.1. create

```
“create dic_supplier “ < dic_supplier_identifier > “(“[ <dic_supplier_definition>] “)” “;”
```

# You can use ‘dictionary\_supplier’ in the same form, too.

Example)

```
create dic_supplier 9901/JEMIMA ();  
create dic_supplier 9901/JEMIMA (organization_name ‘JEMIMA’);
```

### 5.3.2. alter

```
“alter dic_supplier” <dic_supplier_identifier> “(“<dic_supplier_definition> “)” “;”
```

Example)

```
alter dic_supplier 9901/JEMIMA modify (oraganization_name ‘Japan Electric Measuring Instruments  
Manufacturers’ Association);
```

### 5.3.3. drop

```
“drop dic_supplier” <dic_supplier_identifier> “;”
```

Example)

```
drop dic_supplier 9901/JEMIMA;
```

## 5.4. class

### 5.4.1. create

```
“create class” <class_identifier> [“of “ < class_type >] “(“ <class_definition> “)” “;”
```

Example)

```
create class 9901/JEMIMA.LEVEL_METER (  
    super_class 'METER',  
    preferred_name.en 'Level Meter',  
    preferred_name.ja 'lebelkei'  
)
```

### 5.4.2. alter

```
“alter class ” < class_identifier > “modify” (“ <alter_class_definition> “)” “;”
```

Example)

```
alter class 9901/JEMIMA.LEVEL_METER(  
    synonymous_name(('LMET','en'),('lebelmeta','ja'))  
)
```

```
alter class 9901/JEMIMA.LEVEL_METER(  
    applicable_properties('PRODUCT_NUMBER','COLOR','COMPANY')  
)
```

### 5.4.3. drop

```
“drop class “ <class_identifier> “;”
```

Example)

```
drop 9901/JEMIMA.LEVEL_METER;
```

### 5.4.4. rename

This command is to change the class\_bsu\_code.

```
“rename class” <old_class_identifier> “to” <new_class_identifier> “;”
```

Example)

```
rename class 9901/JEMIMA.LEVEL_METER to 9901/JEMIMA.LMTR
```

## 5.5. property

### 5.5.1. create

```
“create property” < property_identifier > [ “of” <property_type> ]“(“ <property_definition> “)”  
“;”
```

Example)

```
Create property PRODUCT_NAME of non_dependent (  
    preferred_name.en 'Product Name',  
    preferred_name.ja 'seihinmei',  
    name_scope '9901/JEMIMA.THING',  
    data_type 'string_type'  
)
```

```

create property COLOR of non_dependent(
  preferred_name.en 'COLOR',
  preferred_name.ja 'iro',
  name_scope '9901/JEMIMA.THING',
  data_type enumeration_type(code, meaning.en, meaning.ja)
    (('BLACK','black','kuro'),('WHITE','white','shiro'),('SILVER','silver','gin'))
)

```

### 5.5.2. alter

“alter property” <property\_identifier> “modify” (“<alter\_property\_definition>”) “;”

Example)

```

alter property COLOR modify (
  data_type 'string_type'
)

```

### 5.5.3. drop

“drop property” <property\_identifier>

Example)

```

drop property COLOR;

```

### 5.5.4. rename

“rename property” <old\_property\_identifier> “to” <new\_property\_identifier> “;”

Example)

```

rename property COLOR to COLOR02

```

## 6. DRL

### 6.1. extension

#### 6.1.1. create

“create extension on” <class\_identifier> (“<extension\_definition>”) “;”

Example)

```

create extension on METER (
  PRODUCT_NUMBER constraint pk_pn key,
  constraint cl_pn classification 101,
  COMPANEY constraint pk_co key
  constraint cl_co classification 100
)

```

#### 6.1.2. alter

“alter extension” ”on” <class\_identifier> (“add”|“modify”|“delete”) (“<extension\_definition>”) “;”

Example)

```

alter extension on METER modify (
  PRODUCT_NUMBER,
  COMPANYY constraint pk_co key
);

```

```

alter extension on METER add (
  COLOR

```

```
);  
  
alter extension on METER delete (  
    COLOR  
);
```

### 6.1.3. drop

“drop extension” “on” <class\_identification> “;”

Example)  
drop extension on METER;

## 6.2. table

### 6.2.1. create

“create table” <table\_identifier> “on “ <class\_identifier> “(“ <extension\_definition> “)” “ ; “

Example)  
create table table01 on METER (  
 PRODUCT\_NUMBER constraint pk\_pn key,  
 COMPANEY  
)

### 6.2.2. alter

“alter table” <table\_identifier> (“add”|“modify”|“delete”) (“ <extension\_definition> “)” “ ; “

Example)  
alter table table001 on METER modify (  
 PRODUCT\_NUMBER,  
 COMPANYY constraint pk\_co key  
);

```
alter table table001 on METER add (  
    COLOR  
);
```

```
alter table table001 on METER delete (  
    COLOR  
);
```

### 6.2.3. drop

“drop table” <table\_identifier> “;”

Example)  
drop table table001;

## 7. LML

### 7.1. insert

<insert\_command > ::= “insert into” <table\_identifier>  
 [“(“<property\_bsu\_code> {“,” <property\_bsu\_code> } “)” ]

```

“values” (“ “ ” <simple_value> [ <reference_condition> ] “ ” { “ , ”
“ ” <simple_value> [ <reference_condition> ] “ ” } “ ) “ ; ”

```

```

< insert_command > ::= “insert into” <class_identifier>
[ (“ “ <property_bsu_code> { “ , ” <property_bsu_code> } “ ) ” ] “ ; ”
“values” (“ “ ” <simple_value> [ <reference_condition> ] “ ” { “ , ”
“ ” <simple_value> <condition> “ ” } “ ) ” “ ; ”

```

# First form insert data into table.  
 Second one insert data into class extension.

Example)  
 insert into METER (PRODUCT\_NAME, COMPANY,COLOR) values('TS002', 'TOSHIBA', 'WHITE');

<reference\_condition> is used for selecting appropriate instances in *Part of Part-Whole relationship*.

### 7.2. update

```

< update_command > ::= “update” (<table_identifier>| <class_identifier> )
“set” <property_identifier> “=” <simple_value> [ <reference_ondition> ]
{ “ , ” <property_identifier> “=” <simple_value> [ <reference_condition> ] }
[ “where” <search_condition> ] “ ; ”

```

# <table\_identifier> for table, <class\_identifier> for class\_extension.

Example)  
 update METER set COLOR = 'SILVER';

### 7.3. delete

```

<delete_command> ::= “delete from” <table_identifier> [ "where" < search_condition > ] “ ; ”
<delete_command> ::= “delete from” <class_identifier> [ "where" < search_condition > ] “ ; ”

```

#First form deletes data from table.  
 Second form deletes data from class\_extension.

Example)  
 delete from METER where COLOR='SILVER';

## 8. SET

This command set default environment, project, dictionary, supplier, and so on.  
 If you set default project, you don't need to specify the project in each command, except the situation that you'd like to access another project.

```

“set” <environment_variable> <user_value> “ ; ”

```

Example)  
 set project 'JEMIMA';  
 set dictionary 'JEMIMA\_DIC';  
 set dic\_supplier '9901/JEMIMA';

## 9. Security

In LMS, users are belonging to the user group based on business model that depend on each organization. LMS grants a privilege to a class under *User* to do a specific operation (“select”, “alter”, etc) to specific data. The system applies security controls to the class (user group) from the following three viewpoints.

- 1) Class

- 2) Property
- 3) Instance

### 9.1. grant

<grant\_command> ::= “grant” <privilege\_type> on <controle\_viewpoint> to <user\_group>  
 [ “where” <search\_condition> ] “;”

<privilege\_type> ::= <command\_name> | <role\_name>

<command\_name> ::= “alter”  
 | ”commit”  
 | “create”  
 | “delete”  
 | “drop”  
 | “insert”  
 | “rename”  
 | “select”  
 | “set”  
 | ”update”

<controle\_viewpoint> ::= <class\_identifier> | <property\_identifier>

note: <role\_name> is a collection of <command\_name>s. Implementers are allowed to define <role\_name> and their command set in each LMS.

The following is a sample of <role\_name>

<role\_name> ::= “ Administrator”  
 | “CLASSEditor”  
 | “OrganizationAdministrator”  
 | “Editor”  
 | ”CertifiedUser”

<user\_group> ::= <class\_identifier>

### 9.2. revoke

<revoke\_command> ::= “revoke” <privilege\_type> on <controle\_viewpoint> from <user\_group>  
 [ “where” <search\_condition> ] “;”

## 10. Command List

- alter project
- alter dictionary
- alter dic\_supplier
- alter dictionary\_supplier
- alter class
- alter extension
- alter project
- alter property
- alter table
- commit
- create project
- create dictionary
- create dic\_supplier
- create class
- create extension
- create property
- create table

delete  
 drop project  
 drop dictionary  
 drop dic\_supplier  
 drop class  
 drop extension  
 drop property  
 drop table  
 grant  
 revoke  
 insert  
 rename  
 select  
 set  
 update

### alter class

```
< alter_class_command >
 ::=
 "alter class " < class_identifier > "modify" (" <alter_class_definition> ") ";"

< alter_class_definition >
 ::=
 [ "super_class" " " <parent_class > " " ] [ "," ]
 [<preferred_name> " " < string > " " ] [ "," ]
 [<short_name> " " < string > " " ] [ "," ]
 [ "synonymous_name" " ( " < label_with_language > [<syn_condition>] "
 { "," <label_with_language> [<syn_condition>] } " ) " ] [ "," ]
 [<definition> " " < string > " " ] [ "," ]
 [<remark> " " < string > " " ] [ "," ]
 [<note> " " < string > " " ] [ "," ]
 [<graphics> " " < string > " " ] [ "," ]
 [<source_doc_of_definition> " " < string > " " ] [ "," ]
 [ "applicable_properties" " ( " < SET of class_bsu_code > " ) " ] [ "," ]
 [ "version" " " < code_type > " " ] [ "," ]
 [ "revision" " " < code_type > " " ] [ "," ]
 [ "is_case_of" " ( " <class_identifier> { "," <class_identifier> } " ) " ] [ "," ]
 [ "imported_properties" " ( " <property_identifier> { "," <property_identifier> } " ) " ] [ "," ]
 [ "imported_tables" " ( " <table_identifier> { "," <table_identifier> } " ) " ] [ "," ]
 [ "imported_documents" " ( " <document_identifier> { "," <document_identifier> }
 " ) " ] [ "," ]
 [ "imported_types" " ( " <data_type_identifier> { "," <data_type_identifier> } " ) " ] [ "," ]

<syn_condition> ::= "where 1 =" " " <label> " "

< SET of class_bsu_code > ::= <class_bsu_code> { "," <class_bsu_code> }
```

The following parameters are required only when class type is item\_class\_case\_of or component class\_case\_of or material\_class\_case\_of .

**is\_case\_of**  
**imported\_properties**  
**imported\_tables**  
**imported\_documents**  
**imported\_types**

### alter dictionary

```
<alter_dictionary_command>  
 ::= "alter dictionary" <dictionary_identifier> "modify" "(" <dictionary_definition> ")" ";"
```

#### alter dic\_supplier

```
< alter_dic_supplier_command >  
 ::=  
 "alter dic_supplier" <dic_supplier_identifier> "modify" "(" <dic_supplier_definition> ")" ";"
```

#### alter dictionary\_supplier

```
< alter_dictionary_supplier_command >  
 ::=  
 "alter dictionary_supplier" <dic_supplier_identifier> "modify" "(" <dic_supplier_definition>  
 ")" ";"
```

#### alter extension

```
< alter_extension_command >  
 ::= "alter extension" "on" <class_identifier> ("add"|"modify"|"delete")  
 "(" <extension_definition> ")" ";"
```

#### alter project

```
< alter_project_command >  
 ::=  
 "alter project" <project_identifier> "modify" "(" <project_definition> ")" ";"
```

#### alter property

```
< alter_property_command >  
 ::=  
 "alter property" <property_identifier> "modify" "(" <property_definition> ")" ";"
```

#### alter table

```
< alter_table_command >  
 ::= "alter table" <table_bsu_code> "on" <class_identifier> ("add"|"modify"|"delete")  
 "(" <extension_definition> ")" ";"
```

#### commit

```
<commit_command> ::= "commit" [<project_identifier>] ["version" <version_number>]
```

#### create dictionary

```
< create_dictionary_command >  
 ::=  
 "create dictionary" <dictionary_identifier> "(" <dictionary_definition> ")" ";"
```

#### create dic\_supplier

```
< create_dic_supplier_command >  
 ::=
```

“create dic\_supplier “ < dic\_supplier\_identifier > “[ <dic\_supplier\_definition> ]” “;”

#### create dictionary\_supplier

< create\_dic\_supplier\_command >

::=

“create dictionary\_supplier “ < dic\_supplier\_identifier > “[ <dic\_supplier\_definition> ]” “;”

#### create class

< create\_class\_command >

::=

“create class” <class\_identifier> [“of “ < class\_type >] “[ <class\_definition> “)” “;”

< class\_type > ::= “item\_class”

    | “component\_class”

    | “material\_class”

    | “item\_class\_case\_of”

    | “component\_class\_case\_of”

    | “material\_class\_case\_of”

< class\_definition >

::= <preferred\_name> “ ’ ” < string > “ ’ ” [ “;” ]

    [ “super\_class” “ ’ ” <parent\_class > “ ’ ” ] [ “;” ]

    [ <short\_name> “ ’ ” < string > “ ’ ” ] [ “;” ]

    [ “synonymous\_name “ ( ” < label\_with\_language > { “;” <label\_with\_language> } “)”

    [ “;” ]

    [ <definition> “ ’ ” < string > “ ’ ” ] [ “;” ]

    [ <remark> “ ’ ” < string > “ ’ ” ] [ “;” ]

    [ <note> “ ’ ” < string > “ ’ ” ] [ “;” ]

    [ <graphics> “ ’ ” < string > “ ’ ” ] [ “;” ]

    [ <source\_doc\_of\_definition> “ ’ ” < string > “ ’ ” ] [ “;” ]

    [ “applicable\_properties” “ ( ” < SET of class\_bsu\_code > “ ) ” ] [ “;” ]

    [ “version” “ ’ ” < code\_type > “ ’ ” ] [ “;” ]

    [ “revision” “ ’ ” < code\_type > “ ’ ” ] [ “;” ]

    [ “is\_case\_of” “[ <class\_identifier> { “;” <class\_identifier> } “)” ]

    [ “imported\_properties” “[ <property\_identifier> { “;” <property\_identifier> } “)” ]

    [ “imported\_tables” “[ <table\_identifier> { “;” <table\_identifier> } “)” ]

    [ “imported\_documents” “[ <document\_identifier> { “;” <document\_identifier> } “)” ]

    [ “imported\_types” “[ <data\_type\_identifier> { “;” <data\_type\_identifier> } “)” ]

< SET of class\_bsu\_code > ::= <class\_bsu\_code> { “;” < class\_bsu\_code> }

The following parameters are required only when class type is item\_class\_case\_of or component class\_case\_of or material\_class\_case\_of .

**is\_case\_of**

**imported\_properties**

**imported\_tables**

**imported\_documents**

**imported\_types**

#### create subclass

< create\_subclass\_command >

::=

“create subclass” <class\_identifier> [“of “ < class\_type >] “[ <subclass\_definition> “)” “;”

< subclass\_definition >

```

::= <preferred_name> “ ’ ” < string > “ ’ ” “ ,”
    “super_class” “ ’ ” <parent_class> “ ’ ” [ “ ,” ]
    [<short_name> “ ’ ” < string > “ ’ ” ] [ “ ,” ]
    [“synonymous_name “ ( ” < label_with_language > { “ ,” <label_with_language> } “ ) ”
[ “ ,” ]

    [<definition> “ ’ ” < string > “ ’ ” ] [ “ ,” ]
    [<remark> “ ’ ” < string > “ ’ ” ] [ “ ,” ]
    [<note> “ ’ ” < string > “ ’ ” ] [ “ ,” ]
    [<graphics> “ ’ ” < string > “ ’ ” ] [ “ ,” ]
    [<source_doc_of_definition> “ ’ ” < string > “ ’ ” ] [ “ ,” ]
    [“applicable_properties” “ ( ” < SET of class_bsu_code > “ ) ” ] [ “ ,” ]
    [“version” “ ’ ” < code_type > “ ’ ” ] [ “ ,” ]
    [“revision” “ ’ ” < code_type > “ ’ ” ] [ “ ,” ]
    [“is_case_of” “(“ <class_identifier> { “ ,” <class_identifier> } “)” ]
    [“imported_properties” “(“ <property_identifier> { “ ,” <property_identifier> } “)” ]
    [“imported_tables” “(“ <table_identifier> { “ ,” <table_identifier> } “)” ]
    [“imported_documents” “(“ <document_identifier> { “ ,” <document_identifier> } “)” ]
    [“imported_types” “(“ <data_type_identifier> { “ ,” <data_type_identifier> } “)” ]

```

#### create extension

```

< create_extension_command >
::=
    “ create extension on “ <class_identifier> “(“ <extension_definition> “)” “ ; “

```

#### create project

```

< create_project_command >
::=
    “create project” < project_identifier > “(“ <project_definition> “)” “ ; “

```

#### create property

```

< create_property_command >
::=
    “create property” < property_identifier > [ “of” <property_type> ] “(“ <property_definition> “)”
“ , “

```

#### < property\_type >

```

::= “condition” | “dependent” | “non_dependent”

```

#### create table

```

< create_table_command >
::=
    “ create table” <table_bsu_code> on “ <class_identifier> “(“ <table_definition> “)” “ ; “

```

#### delete

```

<delete_command> ::= “delete from” <table_identifier> [ “where” < search_condition > ] “;”
<delete_command> ::= “delete from” <class_identifier> [ “where” < search_condition > ] “;”

```

#First form deletes data from table.

Second form deletes data from class\_extension.

### drop class

```
< drop_class_command > ::= "drop class" < class_identifier > ";"
```

### drop dictionary

```
< drop_dictionary_command > ::= "drop dictionary" < dictionary_identifier > ";"
```

### drop dic\_supplier

```
< drop_dic_supplier_command > ::= "drop dic_supplier" < dic_supplier_identifier > ";"
```

### drop dictionary\_supplier

```
< drop_dictionary_supplier_command > ::= "drop dictionary_supplier" < dic_supplier_identifier > ";"
```

### drop extension

```
< drop_extension_command > ::= "drop extension on" < class_identifier >
```

### drop project

```
< drop_project_command > ::= "drop project" < project_identifier > ";"
```

### drop property

```
< drop_property_command > ::= "drop property" < property_identifier > ";"
```

### drop table

```
< drop_table_command > ::= "drop table" < table_identifier > ";"
```

### grant

```
< grant_command > ::= "grant" < privilege_type > on < controle_viewpoint > to < user_group >  
[ "where" < search_condition > ] ";"
```

### insert

```
< insert_command > ::= "insert into" < table_identifier >  
[ ("< property_bsu_code > {"," < property_bsu_code > }") ]  
"values" (" " < simple_value > [< reference_condition >] " " {","  
" " < simple_value > [< reference_condition >] " " } " ) ";"
```

```
< insert_command > ::= "insert into" < class_identifier >  
[ ("< property_bsu_code > {"," < property_bsu_code > }") ] ";"  
"values" (" " < simple_value > [< reference_condition >] " " {","  
" " < simple_value > [< reference_condition >] " " } " ) ";"
```

```
< reference_condition > ::= "condition" < string > < property_identifier > < operand > < simple_value >
```

# First form insert data into table.

Second one insert data into class extension.

### rename

```
< rename_command > ::= "rename" ( "class" < old_class_identifier > to < new_class_identifier >  
| "property" < old_property_identifier > to < new_property_identifier > ) ";"
```

## revoke

```
<revoke_command> ::= "revoke" <privilege_type> on <controle_viewpoint> from <user_group>  
    [ "where" <search_condition> ] ";"
```

## select

```
< select_contents_command >  
    ::=  
        "select" [ "distinct" ]  
        "*" | < displayed_property > { ", " < displayed_property > }  
        "from" ( < selected_class > { ( "AND" | "+" | "-" ) < selected_class > } )  
        [ "where" < search_condition > ]  
        [ "order by" < order_property > [ "asc" | "desc" ] ]  
  
< displayed_property > ::= < property_bsu_code >  
  
< selected_class > ::= < class_identifier >  
    | < class_identifier > "*" [ "except" < class_identifier > ]  
    | < class_identifier > "$" [ "except" < class_identifier > ]  
    | < class_identifier > "%" [ "except" < class_identifier > ]  
    | < class_identifier > "!" [ "except" < class_identifier > ]  
  
< order_property > ::= < property_bsu_code >  
  
< search_condition >  
    ::= [ "not" ] < logical_term >  
    { "or" < logical_term > }  
  
< logical_term > ::= < logical_factor > { "and" < logical_factor > }  
  
< logical_factor > ::= < property_identifier > < comparaision_op > < simple_value >  
    | < property_identifier > "like" < match_string >  
    | "instancenum" < limit_op > < number >  
    | "between" < simple_value > "and" < simple_value >  
    | "not between" < simple_value > "and" < simple_value >  
  
< limit_op > ::= "<" | "<="
```

```
< comparision_op > ::= "=" | ">" | "<" | ">=" | "<=" | "!="
```

```
< match_string > ::= " " { < any_character > | "_" | "%" } " " "
```

## set

```
< set_command > ::= "set" < environment_variable > " " < user_value > " " ";"
```

```
< environment_variable > ::= "project"  
    | "dictionary"  
    | "dic_supplier"  
    | "dictionary_supplier"  
    | "class"  
    | "property"
```

## update

```
< update_command > ::= "update" ( < table_identifier > | < class_identifier > )  
    "set" < property_identifier > "=" < simple_value > [ < reference_condition > ]
```

```
{“,” <property_identifier> “=” <simple_value> [<reference_condition>] }
[ “where” <search_condition> ]“;”
```

# <table\_identifier> for table, <class\_identifier> for class\_extension.

<reference\_condition> ::= “condition” <string> <property\_identifier> <operand> <simple\_value>

## 11. Common Definition

### class\_identifier

<class\_identifier>

```
::= [< project_identifier >“.”]
    [<dictionary_identifier>“.”]
    [<dic_supplier_identifier>“.”]
    <class_bsu_code>
```

< class\_bsu\_code > ::= < class\_code\_type >

<class\_code\_type> ::= STRING[ ‘\_’<version>’\_’<revision>]

If there are no version and revision, it means the latest version and revision.

### command\_name

<command\_name> ::= “alter”

```
    | “commit”
    | “create”
    | “delete”
    | “drop”
    | “insert”
    | “rename”
    | “select”
    | “set”
    | “update”
```

### controle\_viewpoint

<controle\_viewpoint> ::= <class\_identifier> | <property\_identifier>

### data\_type

```
<data_type> ::= “ ’string_type’ ”
    | “ ’boolean_type’ ”
    | “ ’int_type’ ”
    | <int_measure_type>
    | <int_currency_type>
    | “ ’real_type’ ”
    | <real_measure_type>
    | <real_currency_type>
    | “ ’number_type’ ”
    | <enumeration_type>
    | <level_type>
    | <class_instance_type>
    | “ ’aggregate_type’ ”
```

<int\_measure\_type> ::= “ ’int\_measure\_type’ ” “,” “unit” <string> “,”

<int\_currency\_type> ::= “ ’int\_currency\_type’ ” “,” “currenty” <string> “,”

<real\_measure\_type> ::= “ ’real\_measure\_type’ ” “,” “unit” <string> “,”

<real\_currency\_type> ::= “ ’real\_currency\_type’ ” “,” “currenty” <string> “,”

<enumeration\_type> ::= “enumeration\_type(“ <language\_order> “) ” (“ <enum\_members> “) “

| “non\_quantitative\_code\_type” “( “<enum\_member> “ ) “  
| “non\_quqntitqtive\_int\_type” “( “<enum\_member> “ ) “

<enum\_order> ::= “code” “,”  
“meaning” “.” <language\_code> { “,” “meaning” “.” <language\_code> }  
[ “,” “document ” ]

<enum\_members> ::= “( “<enum\_member> ” ) ” “,” “( “<enum\_member> ” ) ”  
{ “,” “( “<enum\_member> ” ) ” }

<enum\_member> ::= “ ’ ” <value\_code\_type> “ ’ ” “,” “ ’ ” <pref\_name\_type> “ ’ ”  
{ “,” “ ’ ” <pref\_name\_type> “ ’ ” }

<level\_type> ::= ( “int\_level\_type” | “real\_level\_type” )  
“ ( “ [ “min” ] [ “,” ] [ “nom” ] [ “,” ] [ “typ” ] [ “,” ] [ “max” ] “ ) ”

<class\_instance\_type> ::= “class\_instance\_type” “ ’ ” <class\_bsu\_code> “ ’ ”

#### Dictionary\_identifier

<dictionary\_identifier> ::= [ < project\_identifier > “.” ] <dictionary\_code>  
<dictionary\_code> ::= < dictionary\_code\_type >

#### dictionary\_definition

<dictionary\_definition> ::= [ “name ” <string> “,” ]  
[ “ version ” <version\_type> [ “,” ] ]  
[ “ revision ” <revision\_type> [ “,” ] ]  
[ “defined\_by” <dic\_supplier\_identification> [ “,” ] ]

#### dic\_supplier\_definition

<dic\_supplier\_definition>  
::= “organization\_id” “( “<identifier> “ ) ” “,”  
“organization\_name” “( “<label> “ ) ” “,”  
“organization\_description” “( “<text> “ ) ” “,”  
“address\_internal\_location” “( “<label> “ ) ” “,”  
“address\_street\_number” “( “<label> “ ) ” “,”  
“address\_postal\_box” “( “<label> “ ) ” “,”  
“address\_town” “( “<label> “ ) ” “,”  
“address\_region” “( “<label> “ ) ” “,”  
“address\_country” “( “<label> “ ) ” “,”  
“address\_facsimile\_number” “( “<label> “ ) ” “,”  
“address\_telephone\_number” “( “<label> “ ) ” “,”  
“address\_electronic\_mail\_number” “( “<label> “ ) ” “,”  
  
“address\_telex\_number” “( “<label> “ ) ” “,”

#### dic\_supplier\_identifier

<dic\_supplier\_identifier> ::= < supplier\_bsu\_code>  
< supplier\_bsu\_code> ::= <supplier\_code\_type >

#### label\_with\_language

< label\_with\_language > ::= “( “<label> “ , ” <language\_code> “ ) ”

#### note

<note> ::= “note” | “note.” <language\_code>

### preferred\_name

<preferred\_name> ::= "preferred\_name" | "preferred\_name." <language\_code>

### privilege\_type

<command\_name> | <role\_name>

### property\_identifier

<property\_identifier> ::= [< class\_identifier>"."] <property\_bsu\_code>

<property\_bsu\_code> ::= <property\_code\_type>

<property\_code\_type> ::= STRING[ '\_' <version> '\_' <revision> ]

If there are no version and revision, it means the latest version and revision.

### Project\_identifier

<project\_identifier> ::= <string>

### remark

<remark> ::= "remark" | "remark." <language\_code>

### short\_name

<short\_name> ::= "short\_name" | "short\_name." <language\_code>

### extension\_definition

<extension\_definition>

::= <property\_bsu\_code> ["constraint" <constraint\_name> ( "key" | "classification" <three digits> ) {"constraint" <constraint\_name> ( "key" | "classification" <three digits> )

{"", <property\_bsu\_code> ["constraint" <constraint\_name> ( "key" | "classification" <three digits> ) {"constraint" <constraint\_name> ( "key" | "classification" <three digits> ) } }

### project\_definition

<project\_definition>

::= "dictionary" " " <string> " " "

" " "creator" " " <user\_name> " " "

" " "active\_date" " " <date> " " "

" " "access\_level" " " <privileges> " " "

# More information will be required.

### Property\_definition

<property\_definition>

::= <preferred\_name> " " <string> " " " "

"name\_scope" " " <class\_bsu\_code> " " " "

"data\_type" " " <data\_type> [ " " ]

[<short\_name> " " <string> " " ] [ " " ]

["synonymous\_name" ( " " <label\_with\_language> { " " <label\_with\_language> } ) ] [ " " ]

[<definition> " " <string> " " ] [ " " ]

["value\_format" " " <value\_format\_type> " " ] [ " " ]

[<remark> " " <string> " " ] [ " " ]

```

[<note> “ ’ ” < string > “ ’ ”] [ “ , ” ]
[“graphics” “ ’ ” < string > “ ’ ”] [ “ , ” ]
[“preferred_symbol” “ ’ ” < string > “ ’ ”] [ “ , ” ]
[“ource_doc_of_definition” “ ’ ” < string > “ ’ ”] [ “ , ” ]
[“classification” “ ’ ” < DET_classification_type > “ ’ ”] [ “ , ” ]
[“formula” “ ’ ” < string > “ ’ ”] [ “ , ” ]
[“version” “ ’ ” < code_type > “ ’ ”] [ “ , ” ]
[“revision” “ ’ ” < code_type > “ ’ ”] [ “ , ” ]

```

#### search\_condition

```

< search_condition >
  ::= [ "not" ] < logical_term >
    { "or" < logical_term > }

< logical_term > ::= < logical_factor > { "and" < logical_factor > }

< logical_factor > ::= < property_identifier > < comparison_op > < simple_value >
  | < property_identifier > "like" < match_string >
  | "instancenum" < limit_op > < number >
  | "between" < simple_value > "and" < simple_value >
  | "not between" < simple_value > "and" < simple_value >

< limit_op > ::= "<" | "<="

< comparison_op > ::= "=" | ">" | "<" | ">=" | "<=" | "!="

```

#### table\_definition

```

<table_definition>
  ::= <property_bsu_code> ["constraint key"] { "," <property_bsu_code> ["constraint key"] }

```

#### table\_identifier

```

<table_identifier>
  ::= [<class_identifier> "."] <table_bsu_code>

```

memo)

Sometimes, we make model simpler than Part24, because a general dictionary designer doesn't need too complex description. For example, to create value\_domain.

#### user\_group

```

<user_group> ::= <class_identifier>

```