APPENDIX

CREATING TABLES

```
CREATE TABLE <TableName> ( <AttributeName> Type [NOT
NULL],...,
PRIMARY KEY(<AttributeName1>,<AttributeName2>,...),),
FOREIGN KEY(<AttributeName3,<AttributeName4>,…) REFERENCES
<TableNameReferenced>(<AttributeName5,<AttributeName6>,…)
<delete rule> <modification rule>,...,
UNIQUE (<AttributeName7,<AttributeName8>,…)
)
```

where:

```
<delete rule>::= ON DELETE <referential action>
<modification rule>::= ON UPDATE <referential action>
<referential action>::= CASCADE | SET NULL |
SET DEFAULT | NO ACTION
```

MODIFYING TABLES

- Adding an attribute:

```
ALTER TABLE <TableName> ADD <AttributeName>
<TypeAttribute>;
```

- Modifying an attribute:

```
ALTER TABLE <TableName> RENAME COLUMN <AttributeName> TO
<NuevoAttributeName>;
ALTER TABLE <TableName> MODIFY <AttributeName>
<NuevoTypeAttribute>;
```

- Deleting an attribute:

```
ALTER TABLE <TableName> DROP COLUMN <AttributeName>;
```

- Adding a restriction:

```
ALTER TABLE <TableName> ADD {PRIMARY KEY
(<Field1>,<Field2>,...) | UNIQUE (<Field1>,<Field2>,...) |
FOREIGN KEY(<Field>) REFERENCES <Table>(<Field>) | CHECK
(<Expression booleana>)};
```

DELETING TABLES

```
DROP TABLE <TableName>;
```

MANAGING DATA

```
INSERT INTO <TableName> VALUES (<Value1, Value2, ...>);
INSERT INTO <TableName> <query>;

UPDATE <TableName>
SET AttributeName1=<Expression1>,
    AttributeName2=<Expression2>, ...
WHERE <Predicate>;

DELETE FROM <TableName> WHERE <Predicate>;
```

QUERIES

```
SELECT ALL | DISTINCT A1,A2,...,An
FROM r1,r2,...,rm
WHERE <predicate>;
```

donde:

```
     <predicate>::= <predicate> OR <predicate> |
<predicate> AND <predicate> | NOT <predicate>
```

WHERE

```
WHERE <attribute> LIKE | NOT LIKE <pattern>;
```

- %: match with any substring.
- _ : match with any character.

```
WHERE <attribute> IS NULL
```

ORDER BY

```
ORDER BY <attribute1> [Desc|Asc]?, <attribute2>
[Desc|Asc]?, ...
```

UNION, INTERSECT Y MINUS

```
<query> UNION <query>
<query> INTERSECT <query>
<query> MINUS <query>
```

IN/NOT IN

```
WHERE <attribute> IN | NOT IN <subquery>
```

EXISTS/NOT EXISTS

```
WHERE <attribute> EXISTS | NOT EXISTS <subquery>
```

SOME/ALL

```
WHERE <attribute> <comparison operator> SOME | ALL
<subquery>
```

INNER/LEFT/RIGHT JOIN

```
FROM <table1> INNER | LEFT | RIGHT JOIN <table2> ON
<AttributeCommonTable1 = AttributeCommonTable2>
```

GROUP BY/HAVING

```
SELECT <Attribute1>,<Attribute2>,... COUNT(*), MIN(At),
MAX(At), SUM(At), AVG(At)
FROM r1,r2,...,rm
WHERE <predicate>
GROUP BY <Attribute1>,<Attribute2>,...
HAVING <condition>
```

Note:

i. It is possible that At would be different for each operation.
ii. It can be more than one operation in the same query.

VIEWS

```
CREATE VIEW <nombre> AS <query>
```

FUNCTIONS

create or replace function *<nombre_función>*(*<parámetros>)*
returns *<tipo_retorno>*
as
$$
declare
<declaración de variables>
begin
<sentencias>
end;
$$ language plpgsql;

Si la función devuelve el resultado de un SELECT:
<tipo_retorno> es "table (<nombreColumna1> <tipoColumna1>, <nombreColumna2> <tipoColumna2>, ...)"
El return de la función es "return query <Consulta>"

PROCEDURES

```
create or replace procedure <nombre_procedimiento>(<parámetros>)
as $$
declare
<declaración de variables>
begin
<sentencias>
end;
$$ language plpgsql;
```

TRIGGERS

```
create trigger <nombre_trigger>  {before | after}
{insert | or update | or delete or…}
on <nombre_tabla> for each {statement | row}
[ when (<condicion>) ]
execute procedure <nombre_función>(<parámetros>);
```

CONTROL STRUCTURES

```
for <target> in <consulta> loop
<sentencias>
end loop;

for <var> in <expr>..<expr> loop
<sentencias>
end loop;

while <condicion> loop
<sentencias>
end loop;

foreach <target> in array <expr> loop
<sentencias>
end loop

if <expr> then
<sentencias>
[elseif <expr> then <sentencias>]
[else <sentencias>]
end if;
```