Appendix I

A Grammar for Tutorial D

In this appendix we present the **Tutorial D** production rules in alphabetical order, for ease of reference. Extensions from Part IV to deal with inheritance are included. However, the following are not:

- Features defined (or merely hinted at, in some cases) in Chapter 5 in prose form instead of by means of formal production rules
- Features merely "suggested" in Chapter 10
- Rules regarding the use of an *<attribute ref>* in the inheritance extensions (e.g., in a *<type test>*) wherever a *<selector inv>* is permitted
- Rules regarding the use of a TREAT invocation as a pseudovariable reference

In connection with the last of these, here for reference is a complete list of syntactic categories not explicitly defined in this appendix:

<array var name>
<attribute assign>
<attribute name>
<bool exp>
<character string literal>
<constraint name>
<identifier>
<integer exp>
<introduced name>
<parameter name>
<possrep component assign>
<possrep component name>
<possrep name>
<relation var name>
<scalar selector inv>
<scalar var name>
<statement name>
<THE_ op name>
<THE_ pv name>
<tuple var name>
<user op name>
<user scalar type name>
<version name>

For an explanation of the conventions used in the metalanguage (e.g., its use of syntactic categories of the form *<... list>* and *<... commalist>*), see the introductory section in Chapter 5.

——— ♦♦♦♦♦ ———

<additional constraint def>
::= CONSTRAINT *<bool exp>*

<agg op inv>
::= *<agg op name>* ([*<integer exp>*,] *<relation exp>* [, *<attribute ref>*])

<agg op name>
::= COUNT | SUM | AVG | MAX | MIN | AND | OR | XOR | EXACTLY | UNION | D_UNION | INTERSECT

Copyright © 2005 C. J. Date and Hugh Darwen

Date and Darwen, DATABASES TYPES & THE RELATIONAL MODEL, pp. 499-511,
© 2007 Pearson Education, Inc. Reprinted by permission of Pearson Education, Inc. All rights reserved.

```
<application relation var def>
     ::=   VAR <relation var name> <private or public>
               <relation type or init value>
              <candidate key def list>

<argument>
    ::=   <exp>

<array cardinality>
    ::=   COUNT ( <array var ref> )

<array target>
    ::=   <array var ref>

<array var def>
    ::=   VAR <array var name> ARRAY <tuple type>

<array var ref>
    ::=   <array var name>

<assign>
    ::=   <scalar assign>
        | <tuple assign>
        | <relation assign>

<assignment>
    ::=   <assign commalist>

<attribute>
    ::=   <attribute name> <type>

<attribute extractor inv>
    ::=   <attribute ref> FROM <tuple exp>

<attribute ref>
    ::=   <attribute name>

<attribute target>
    ::=   <attribute ref>
        | <attribute THE_ pv ref>

<attribute test>
    ::=   IS_<scalar type name> ( <attribute ref> )
        | IS_SAME_TYPE_AS ( <exp> , <attribute ref> )

<attribute THE_ pv ref>
    ::=   <THE_ pv name> ( <attribute target> )

<attribute treat>
    ::=   TREAT_AS_<scalar type name> ( <attribute ref> )
        | TREAT_AS_SAME_TYPE_AS ( <exp> , <attribute ref> )

<begin transaction>
    ::=   BEGIN TRANSACTION
```

Copyright © 2005 C. J. Date and Hugh Darwen

Date and Darwen, DATABASES TYPES & THE RELATIONAL MODEL, pp. 499-511,
© 2007 Pearson Education, Inc. Reprinted by permission of Pearson Education, Inc. All rights reserved.

```
<built-in relation op inv>
    ::=   <relation selector inv>
        | <THE_ op inv>
        | <attribute extractor inv>
        | <project>
        | <n-adic other built-in relation op inv>
        | <monadic or dyadic other built-in relation op inv>

<built-in scalar op inv>
    ::=   <scalar selector inv>
        | <THE_ op inv>
        | <attribute extractor inv>
        | <agg op inv>
        | <type test>
        | <scalar treat>
        | ... plus the usual possibilities

<built-in scalar type name>
    ::=   INTEGER | RATIONAL | CHARACTER | CHAR | BOOLEAN

<built-in tuple op inv>
    ::=   <tuple selector inv>
        | <THE_ op inv>
        | <attribute extractor inv>
        | <tuple extractor inv>
        | <tuple project>
        | <n-adic other built-in tuple op inv>
        | <monadic or dyadic other built-in tuple op inv>

<call>
    ::=   CALL <user op inv>

<candidate key def>
    ::=   KEY { <attribute ref commalist> }

<case>
    ::=   CASE ;
             <when def list>
           [ ELSE <statement> ]
          END CASE

<commit>
    ::=   COMMIT

<compose>
    ::=   <relation exp> COMPOSE <relation exp>

<compound statement body>
    ::=   BEGIN ; <statement list> END

<constraint def>
    ::=   CONSTRAINT <constraint name> <bool exp>

<constraint drop>
    ::=   DROP CONSTRAINT <constraint name>
```


Date and Darwen, DATABASES TYPES & THE RELATIONAL MODEL, pp. 499-511,
© 2007 Pearson Education, Inc. Reprinted by permission of Pearson Education, Inc. All rights reserved.

```
<database relation var def>
    ::=   <real relation var def>
        | <virtual relation var def>

<derived possrep component def>
    ::=   <possrep component name> = <exp>

<derived possrep def>
    ::=   POSSREP [ <possrep name> ]
                  { <derived possrep component def commalist> }

<direction>
    ::=   ASC | DESC

<divide>
    ::=   <relation exp> DIVIDEBY <relation exp> <per>

<do>
    ::=   [ <statement name> : ]
          DO <scalar var ref> :=
              <integer exp> TO <integer exp> ;
             <statement>
          END DO

<dyadic disjoint union>
    ::=   <relation exp> D_UNION <relation exp>

<dyadic intersect>
    ::=   <relation exp> INTERSECT <relation exp>

<dyadic join>
    ::=   <relation exp> JOIN <relation exp>

<dyadic other built-in relation op inv>
    ::=   <dyadic union> | <dyadic disjoint union>
        | <dyadic intersect> | <minus> | <dyadic join>
        | <compose> | <semijoin> | <semiminus>
        | <divide> | <summarize>

<dyadic other built-in tuple op inv>
    ::=   <dyadic tuple union> | <tuple compose>

<dyadic tuple union>
    ::=   <tuple exp> UNION <tuple exp>

<dyadic union>
    ::=   <relation exp> UNION <relation exp>

<exp>
    ::=   <scalar exp>
        | <nonscalar exp>

<extend>
    ::=   EXTEND <relation exp>
                 ADD ( <extend add commalist> )

<extend add>
    ::=   <exp> AS <introduced name>
```

Copyright © 2005 C. J. Date and Hugh Darwen

Date and Darwen, DATABASES TYPES & THE RELATIONAL MODEL, pp. 499-511,
© 2007 Pearson Education, Inc. Reprinted by permission of Pearson Education, Inc. All rights reserved.

```
<filter and cast>
    ::=   <relation exp> : <attribute test>

<group>
    ::=   <relation exp> GROUP ( <grouping commalist> )

<grouping>
    ::=   { [ ALL BUT ] <attribute ref commalist> }
                                        AS <introduced name>

<heading>
    ::=   { <attribute commalist> }

<if>
    ::=   IF <bool exp> THEN <statement>
                     [ ELSE <statement> ]
          END IF

<is def>
    ::=   <single inheritance is def>
        | <multiple inheritance is def>

<leave>
    ::=   LEAVE <statement name>

<minus>
    ::=   <relation exp> MINUS <relation exp>

<monadic or dyadic other built-in relation op inv>
    ::=   <monadic other built-in relation op inv>
        | <dyadic other built-in relation op inv>

<monadic or dyadic other built-in tuple op inv>
    ::=   <monadic other built-in tuple op inv>
        | <dyadic other built-in tuple op inv>

<monadic other built-in relation op inv>
    ::=   <rename> | <where> | <extend> | <wrap> | <unwrap>
        | <group> | <ungroup> | <substitute> | <tclose>
        | <relation treat> | <filter and cast>

<monadic other built-in tuple op inv>
    ::=   <tuple rename> | <tuple extend> | <tuple wrap>
        | <tuple unwrap> | <tuple substitute> | <tuple treat>

<multiple inheritance is def>
    ::=   IS { <scalar type name commalist>
                                <derived possrep def list> }

<n-adic disjoint union>
    ::=   D_UNION [ <heading> ] { <relation exp commalist> }

<n-adic intersect>
    ::=   INTERSECT [ <heading> ] { <relation exp commalist> }

<n-adic join>
    ::=   JOIN { <relation exp commalist> }
```

Copyright © 2005 C. J. Date and Hugh Darwen

Date and Darwen, DATABASES TYPES & THE RELATIONAL MODEL, pp. 499-511,
© 2007 Pearson Education, Inc. Reprinted by permission of Pearson Education, Inc. All rights reserved.

```
<n-adic other built-in relation op inv>
    ::=   <n-adic union> | <n-adic disjoint union>
        | <n-adic intersect> | <n-adic join>

<n-adic other built-in tuple op inv>
    ::=   <n-adic tuple union>

<n-adic tuple union>
    ::=   UNION { <tuple exp commalist> }

<n-adic union>
    ::=   UNION [ <heading> ] { <relation exp commalist> }

<name intro>
    ::=   <exp> AS <introduced name>

<no op>
    ::=   ... an empty string

<nonscalar exp>
    ::=   <tuple exp>
        | <relation exp>

<order item>
    ::=   <direction> <attribute ref>

<parameter def>
    ::=   <parameter name> <type>

<per>
    ::=   PER ( <relation exp> [, <relation exp> ] )

<per or by>
    ::=   <per>
        | BY { [ ALL BUT ] <attribute ref commalist> }

<possrep component def>
    ::=   <possrep component name> <type>

<possrep component ref>
    ::=   <possrep component name>

<possrep component target>
    ::=   <possrep component ref>
        | <possrep THE_ pv ref>

<possrep constraint def>
    ::=   CONSTRAINT <bool exp>

<possrep def>
    ::=   POSSREP [ <possrep name> ]
                  { <possrep component def commalist>
                        [ <possrep constraint def> ] }

<possrep or specialization details>
    ::=   <possrep def list>
        | <additional constraint def>
            [ <derived possrep def list> ]
```


Date and Darwen, DATABASES TYPES & THE RELATIONAL MODEL, pp. 499-511,
© 2007 Pearson Education, Inc. Reprinted by permission of Pearson Education, Inc. All rights reserved.

```
<possrep THE_ pv ref>
    ::=   <THE_ pv name> ( <possrep component target> )

<previously defined statement body>
    ::=   <assignment>
        | <user op def> | <user op drop>
        | <user scalar type def> | <user scalar type drop>
        | <scalar var def> | <tuple var def>
        | <relation var def> | <relation var drop>
        | <constraint def> | <constraint drop>
        | <array var def> | <relation get> | <relation set>

<private or public>
    ::=   PRIVATE | PUBLIC

<project>
    ::=   <relation exp>
                { [ ALL BUT ] <attribute ref commalist> }

<real or base>
    ::=   REAL | BASE

<real relation var def>
    ::=   VAR <relation var name>
              <real or base> <relation type or init value>
                   <candidate key def list>

<relation assign>
    ::=   <relation target> := <relation exp>
        | <relation insert>
        | <relation delete>
        | <relation update>

<relation comp>
    ::=   <relation exp> <relation comp op> <relation exp>

<relation comp op>
    ::=   = | ≠ | ⊂ | ⊆ | ⊃ | ⊇

<relation delete>
    ::=   DELETE <relation target> [ WHERE <bool exp> ]

<relation exp>
    ::=   <relation with exp>
        | <relation nonwith exp>

<relation get>
    ::=   LOAD <array target> FROM <relation exp>
                          ORDER ( <order item commalist> )

<relation insert>
    ::=   INSERT <relation target> <relation exp>

<relation nonwith exp>
    ::=   <relation var ref>
        | <relation op inv>
        | ( <relation exp> )
```


Date and Darwen, DATABASES TYPES & THE RELATIONAL MODEL, pp. 499-511,
© 2007 Pearson Education, Inc. Reprinted by permission of Pearson Education, Inc. All rights reserved.

```
<relation op inv>
    ::=   <user op inv>
        | <built-in relation op inv>

<relation selector inv>
    ::=   RELATION [ <heading> ] { <tuple exp commalist> }
        | TABLE_DEE
        | TABLE_DUM

<relation set>
     ::=   LOAD <relation target> FROM <array var ref>

<relation target>
    ::=   <relation var ref>
        | <relation THE_ pv ref>

<relation THE_ pv ref>
    ::=   <THE_ pv name> ( <scalar target> )

<relation treat>
    ::=   TREAT_AS_SAME_TYPE_AS
                      ( <relation exp> , <relation exp> )
        | <relation exp> <attribute treat>

<relation type>
    ::=   <relation type name>
        | SAME_TYPE_AS ( <relation exp> )
        | RELATION SAME_HEADING_AS ( <nonscalar exp> )

<relation type name>
    ::=   RELATION <heading>

<relation type or init value>
    ::=   <relation type> | INIT ( <relation exp> )
        | <relation type> INIT ( <relation exp> )

<relation update>
    ::=   UPDATE <relation target> [ WHERE <bool exp> ]
                       ( <attribute assign commalist> )

<relation var def>
    ::=   <database relation var def>
        | <application relation var def>

<relation var drop>
    ::=   DROP VAR <relation var ref>

<relation var ref>
    ::=   <relation var name>

<relation with exp>
    ::=   WITH <name intro commalist> : <relation exp>

<rename>
    ::=   <relation exp> RENAME ( <renaming commalist> )
```

Copyright © 2005 C. J. Date and Hugh Darwen

Date and Darwen, DATABASES TYPES & THE RELATIONAL MODEL, pp. 499-511,
© 2007 Pearson Education, Inc. Reprinted by permission of Pearson Education, Inc. All rights reserved.

```
<renaming>
    ::=   <attribute ref> AS <introduced name>
        | PREFIX <character string literal>
               AS <character string literal>
        | SUFFIX <character string literal>
               AS <character string literal>

<return>
    ::=   RETURN [ <exp> ]

<rollback>
    ::=   ROLLBACK

<scalar assign>
    ::=   <scalar target> := <scalar exp>
        | <scalar update>

<scalar comp>
    ::=   <scalar exp> <scalar comp op> <scalar exp>

<scalar comp op>
    ::=   = | ≠ | < | ≤ | > | ≥

<scalar exp>
    ::=   <scalar with exp>
        | <scalar nonwith exp>

<scalar nonwith exp>
    ::=   <scalar var ref>
        | <scalar op inv>
        | ( <scalar exp> )

<scalar op inv>
    ::=   <user op inv>
        | <built-in scalar op inv>

<scalar target>
    ::=   <scalar var ref>
        | <scalar THE_ pv ref>

<scalar THE_ pv ref>
    ::=   <THE_ pv name> ( <scalar target> )

<scalar treat>
    ::=   TREAT_AS_<scalar type name> ( <scalar exp> )
        | TREAT_AS_SAME_TYPE_AS ( <scalar exp> , <scalar exp> )

<scalar type>
    ::=   <scalar type name>
        | SAME_TYPE_AS ( <scalar exp> )

<scalar type name>
    ::=   <user scalar type name>
        | <built-in scalar type name>
```


Date and Darwen, DATABASES TYPES & THE RELATIONAL MODEL, pp. 499-511,
© 2007 Pearson Education, Inc. Reprinted by permission of Pearson Education, Inc. All rights reserved.

```
<scalar type or init value>
    ::=   <scalar type> | INIT ( <scalar exp> )
        | <scalar type> INIT ( <scalar exp> )

<scalar update>
    ::=   UPDATE <scalar target>
              ( <possrep component assign commalist> )

<scalar var def>
    ::=   VAR <scalar var name> <scalar type or init value>

<scalar var ref>
    ::=   <scalar var name>

<scalar with exp>
    ::=   WITH <name intro commalist> : <scalar exp>

<selector inv>
    ::=   <scalar selector inv>
        | <tuple selector inv>
        | <relation selector inv>

<semijoin>
    ::=   <relation exp> SEMIJOIN <relation exp>
        | <relation exp> MATCHING <relation exp>

<semiminus>
    ::=   <relation exp> SEMIMINUS <relation exp>
        | <relation exp> NOT MATCHING <relation exp>

<single inheritance is def>
    ::=   IS { <scalar type name>
                   <possrep or specialization details> }

<statement>
    ::=   <statement body> ;

<statement body>
    ::=   <previously defined statement body>
        | <begin transaction> | <commit> | <rollback>
        | <call> | <return> | <case> | <if> | <do> | <while>
        | <leave> | <no op> | <compound statement body>

<subscript>
    ::=   <integer exp>

<substitute>
    ::=   UPDATE <relation exp>
              ( <attribute assign commalist> )

<summarize>
    ::=   SUMMARIZE <relation exp> [ <per or by> ]
                    ADD ( <summarize add commalist> )

<summarize add>
    ::=   <summary> AS <introduced name>
```

Copyright © 2005 C. J. Date and Hugh Darwen

Date and Darwen, DATABASES TYPES & THE RELATIONAL MODEL, pp. 499-511,
© 2007 Pearson Education, Inc. Reprinted by permission of Pearson Education, Inc. All rights reserved.

```
<summary>
    ::=   <summary spec> ( [ <integer exp>, ]
                                    [ <scalar exp> ] )

<summary spec>
    ::=   COUNT | COUNTD | SUM | SUMD | AVG | AVGD | MAX | MIN
        | AND | OR | ALL | ANY | XOR | EXACTLY | EXACTLYD
        | UNION | D_UNION | INTERSECT

<synonym def>
    ::=   SYNONYMS { <user op name commalist> }

<tclose>
    ::=    TCLOSE <relation exp>

<THE_ op inv>
    ::=   <THE_ op name> ( <scalar exp> )

<tuple assign>
    ::=   <tuple target> := <tuple exp>
        | <tuple update>

<tuple comp>
    ::=   <tuple exp> <tuple comp op> <tuple exp>
        | <tuple exp> ∈ <relation exp>
        | <tuple exp> ∉ <relation exp>

<tuple comp op>
    ::=   = | ≠

<tuple component>
    ::=   <attribute ref> <exp>

<tuple compose>
    ::=   <tuple exp> COMPOSE <tuple exp>

<tuple exp>
    ::=   <tuple with exp>
        | <tuple nonwith exp>
        | <array var ref> ( <subscript> )

<tuple extend>
    ::=   EXTEND <tuple exp> ADD ( <extend add commalist> )

<tuple extractor inv>
    ::=   TUPLE FROM <relation exp>

<tuple nonwith exp>
    ::=   <tuple var ref>
        | <tuple op inv>
        | ( <tuple exp> )

<tuple op inv>
    ::=   <user op inv>
        | <built-in tuple op inv>
```

Copyright © 2005 C. J. Date and Hugh Darwen

Date and Darwen, DATABASES TYPES & THE RELATIONAL MODEL, pp. 499-511,
© 2007 Pearson Education, Inc. Reprinted by permission of Pearson Education, Inc. All rights reserved.

```
<tuple project>
    ::=   <tuple exp>
                { [ ALL BUT ] <attribute ref commalist> }

<tuple rename>
     ::=   <tuple exp> RENAME ( <renaming commalist> )

<tuple selector inv>
    ::=   TUPLE { <tuple component commalist> }

<tuple substitute>
    ::=   UPDATE <tuple exp> ( <attribute assign commalist> )

<tuple target>
    ::=   <tuple var ref>
        | <tuple THE_ pv ref>

<tuple THE_ pv ref>
    ::=   <THE_ pv name> ( <scalar target> )

<tuple treat>
    ::=   TREAT_AS_SAME_TYPE_AS ( <tuple exp> , <tuple exp> )
        | <tuple exp> <attribute treat>

<tuple type>
    ::=   <tuple type name>
        | SAME_TYPE_AS ( <tuple exp> )
        | TUPLE SAME_HEADING_AS ( <nonscalar exp> )

<tuple type name>
    ::=   TUPLE <heading>

<tuple type or init value>
    ::=   <tuple type> | INIT ( <tuple exp> )
        | <tuple type> INIT ( <tuple exp> )

<tuple unwrap>
    ::=   <tuple exp> UNWRAP ( <unwrapping commalist> )

<tuple update>
    ::=   UPDATE <tuple target>
               ( <attribute assign commalist> )

<tuple var def>
    ::=   VAR <tuple var name> <tuple type or init value>

<tuple var ref>
    ::=   <tuple var name>

<tuple with exp>
    ::=   WITH <name intro commalist> : <tuple exp>

<tuple wrap>
    ::=   <tuple exp> WRAP ( <wrapping commalist> )
```


Date and Darwen, DATABASES TYPES & THE RELATIONAL MODEL, pp. 499-511,
© 2007 Pearson Education, Inc. Reprinted by permission of Pearson Education, Inc. All rights reserved.

```
<type>
    ::=   <scalar type>
        | <tuple type>
        | <relation type>

<type test>
    ::=   IS_<scalar type name> ( <scalar exp> )
        | IS_SAME_TYPE_AS ( <exp> , <exp> )

<ungroup>
    ::=    <relation exp> UNGROUP ( <ungrouping commalist> )

<ungrouping>
    ::=    <attribute ref>

<unwrap>
    ::=   <relation exp> UNWRAP ( <unwrapping commalist> )

<unwrapping>
    ::=    <attribute ref>

<user op def>
    ::=   <user update op def>
        | <user read-only op def>

<user op drop>
    ::=   DROP OPERATOR <user op name>

<user op inv>
    ::=   <user op name> ( <argument commalist> )

<user read-only op def>
    ::=   OPERATOR <user op name> ( <parameter def commalist> )
              RETURNS <type>
            [ <synonym def> ] [ VERSION <version name> ] ;
            [ <statement> ]
          END OPERATOR

<user scalar nonroot type def>
    ::=   TYPE <user scalar type name>
              [ ORDINAL ] [ UNION ] <is def>

<user scalar root type def>
    ::=   TYPE <user scalar type name>
              [ ORDINAL ] [ UNION ] <possrep def list>

<user scalar type def>
    ::=   <user scalar root type def>
        | <user scalar nonroot type def>

<user scalar type drop>
    ::=   DROP TYPE <user scalar type name>
```


Date and Darwen, DATABASES TYPES & THE RELATIONAL MODEL, pp. 499-511,
© 2007 Pearson Education, Inc. Reprinted by permission of Pearson Education, Inc. All rights reserved.

```
<user update op def>
    ::=   OPERATOR <user op name> ( <parameter def commalist> )
               UPDATES { <parameter name commalist> }
            [ <synonym def> ] [ VERSION <version name> ] ;
            [ <statement> ]
          END OPERATOR

<virtual relation var def>
    ::=   VAR <relation var name> VIRTUAL ( <relation exp> )
                                  <candidate key def list>

<when def>
    ::=   WHEN <bool exp> THEN <statement>

<where>
    ::=   <relation exp> WHERE <bool exp>

<while>
    ::=   [ <statement name> : ]
          WHILE <bool exp> ;
             <statement>
          END WHILE

<wrap>
    ::=   <relation exp> WRAP ( <wrapping commalist> )

<wrapping>
    ::=   { [ ALL BUT ] <attribute ref commalist> }
                                   AS <introduced name>
```

***** End of Appendix I *****

Copyright © 2005 C. J. Date and Hugh Darwen

Date and Darwen, DATABASES TYPES & THE RELATIONAL MODEL, pp. 499-511,
© 2007 Pearson Education, Inc. Reprinted by permission of Pearson Education, Inc. All rights reserved.