รายการอ้างอิง

1. Bowen, J., Formal Specification and Documentation Using Z. International Thomson Computer Press, 1996.
2. Fuchs, N. E., Specifications are (Preferably) Executable, Software Engineering Journal Volume: 7 5 September 1992 Page(s): 323-333.
3. Bowen, J. P., Formal Specification in Z as A Design and Document Tool, Software Engineering, 1988 Software Engineering 88, Second IEE/BCS Conference: 1988, Page(s): 164-168
4. Sriratanalai, Vatcharawan, A Tool for Translating The Entity Relationship Model to Formal Specification in Z notation, Chulalongkorn University, 1999.
5. Vatanawood, Wiwat and Rivepiboon W., Formal Specification Synthesis for Relational Data Model, Technical Report, Department of Computer Engineering, Faculty of Engineering, Chulalongkorn University, 2000.
6. Ratcliff, Bryan, Introducing Specification Using Z : A Pracical Case Study Approach, The McGraw-Hill International (UK) Limited, 1994.
7. Kolman, B. and Busby, R. C., Discrete Mathematical Structures for Computer Science. Second Edition Englewood Cliffs N.J. : Prentice Hall, 1987.
8. Date, C. J., An Introduction to Database Systems, Volume I, Seventh Edition, Addison-Wesley, 2000.
9. Graeme C. Simsion, Data Modeling Essentials : Analysis, Design, and Innovation, Van Nostrand Reinhold, 1994.
10. Hawryszkiewycz, I. T., Database Analysis and Design, Second Edition, Maxwel Macmillan International, 1991 : 111-142.
11. Schmuller, Joseph, Teach Yourself UML in 24 Hours, Sams Publishing. 1999.
12. Larman, Craig, Applying UML and Patterns: An Introduction to Object Oriented Analysis and Design, Prentice Hall. 1998.

ภาคผนวก

ภาคผนวก ก

ข้อกำหนดเซตของระบบให้เช่ารถแวน

(Van System)

แฟ้มข้อความพจนานุกรมข้อมูลของระบบให้เช่ารถแวน มีรูปแบบ คือ

```
Entity Name:Attribute Name:Data Type:Key:<FK Entity-Attribute>:<CS Constraint>
```

โดยมีรายละเอียดแฟ้มข้อความพจนานุกรมข้อมูลของระบบให้เช่ารถแวน ดังนี้

```
Vanclass:className:CLASSNAME:PK
Vanclass:fuelType:FUELTYPE:NK
Van:reg:REG:PK
Van:model:MODEL:NK
Van:class:CLASSNAME:NK:FK Vanclass-className
Customer:custId:CUSTID:PK
Customer:custName:CUSTNAME:NK
Customer:custAddr:CUSTADDR:NK
Booking:bookId:BOOKID:PK
Booking:startDate:DATE:NK
Booking:endDate:DATE:NK
Booking:custId:CUSTID:NK:FK Customer-custId
Booking:class:CLASSNAME:NK:FK Vanclass-className
Booking:vanNo:REG:NK:FK Van-reg
```

แฟ้มข้อความพจนานุกรมความสัมพันธ์ของระบบให้เช่ารถแวน มีรูปแบบ คือ

```
Relation Name:Entity 1:Entity 2:Relation Type
```

โดยมีรายละเอียดแฟ้มข้อความพจนานุกรมความสัมพันธ์ของระบบให้เช่ารถแวน ดังนี้

```
IsIn:Van:Vanclass:m-1
Reserve:Booking:Vanclass:m-1
Makes:Customer:Booking:1-m
Booked:Booking:Van:1-1
```

ข้อกำหนดเซตของระบบให้เช่ารถแวน มีดังนี้

```
\begin{zed}
        [CLASSNAME, FUELTYPE, REG, MODEL, \\
        CUSTID, CUSTNAME, CUSTADDR, BOOKID, DATE] \\
\end{zed}

\begin{zed}
```

```
        BOOLEAN ::= True | False \\
\end{zed}

\begin{schema} {Vanclass}
        className : CLASSNAME \\
        fuelType : FUELTYPE \\
\end{schema}

\begin{schema} {Van}
        reg : REG \\
        model : MODEL \\
        class : CLASSNAME \\
\end{schema}

\begin{schema} {Customer}
        custId : CUSTID \\
        custName : CUSTNAME \\
        custAddr : CUSTADDR \\
\end{schema}

\begin{schema} {Booking}
        bookId : BOOKID \\
        startDate : DATE \\
        endDate : DATE \\
        custId : CUSTID \\
        class : CLASSNAME \\
        vanNo : REG \\
\end{schema}

\begin{schema} {VanclassExt}
        vanclassSet : \finset Vanclass \\
        vanclassKey : \finset CLASSNAME \\
        vanclassAtt : CLASSNAME \pfun Vanclass \\
\where
        \forall vanclass1,vanclass2 : Vanclass \\
        | vanclass1 \in vanclassSet \land vanclass2 \in vanclassSet \land \\
          vanclass1 \neq vanclass2 @ vanclass1.className \neq vanclass2.className \\
        \dom vanclassAtt \subseteq vanclassKey \\
\end{schema}

\begin{schema} {VanExt}
        vanSet : \finset Van \\
        vanKey : \finset REG \\
        vanAtt : REG \pfun Van \\
\where
        \forall van1,van2 : Van \\
        | van1 \in vanSet \land van2 \in vanSet \land van1 \neq van2 \\
        @ van1.reg \neq van2.reg \\
        \forall vanVar2 : Van \\
        @ ( \exists vanclassVar1 : Vanclass @ vanVar2.class = vanclassVar1.className) \\
        \dom vanAtt \subseteq vanKey \\
\end{schema}

\begin{schema} {CustomerExt}
        customerSet : \finset Customer \\
        customerKey : \finset CUSTID \\
        customerAtt : CUSTID \pfun Customer \\
```

```
\where
        \forall customer1, customer2 : Customer \\
        | customer1 \in customerSet \land customer2 \in customerSet \land \\
          customer1 \neq customer2 @ customer1.custId \neq customer2.custId \\
        \dom customerAtt \subseteq customerKey \\
\end{schema}

\begin{schema} {BookingExt}
        bookingSet : \finset Booking \\
        bookingKey : \finset BOOKID \\
        bookingAtt : BOOKID \pfun Booking \\
\where
        \forall booking1,booking2 : Booking \\
        | booking1 \in bookingSet \land booking2 \in bookingSet \land \\
          booking1 \neq booking2 @ booking1.bookId \neq booking2.bookId \\
        \forall bookingVar2 : Booking
        @ (\exists vanVar1 : Van @ bookingVar2.vanNo = vanVar1.reg) \\
        \forall bookingVar2 : Booking \\
        @ (\exists vanclassVar1 : Vanclass @ bookingVar2.class = vanclassVar1.className) \\
        \forall bookingVar2 : Booking \\
        @ (\exists customerVar1 : Customer @ bookingVar2.custId = customerVar1.custId) \\
        \dom bookingAtt \subseteq bookingKey \\
\end{schema}

\begin{schema} {RelationshipIsin}
        isin : \finset (Van \cross Vanclass)
\where
        \forall isin1,isin2 : isin
        @ first~isin1 = first~isin2 \implies second~isin1 = second~isin2 \also
\end{schema}

\begin{schema} {RelationshipReserve}
        reserve : \finset (Booking \cross Vanclass)
\where
        \forall reserve1,reserve2 : reserve \\
        @ first~reserve1 = first~reserve2 \implies second~reserve1 = second~reserve2 \also
\end{schema}

\begin{schema} {RelationshipMakes}
        makes : \finset (Customer \cross Booking)
\where
        \forall makes1,makes2 : makes \\
        @ second~makes1 = second~makes2 \implies first~makes1 = first~makes2 \also
\end{schema}

\begin{schema} {RelationshipBooked}
        booked : \finset (Booking \cross Van)
\where
        \forall booked1, booked2 : booked \\
        @ second~booked1 = second~booked2 \implies first~booked1 = first~booked2 \land \\
            first~booked1 = first~booked2 \implies second~booked1 = second~booked2 \also
\end{schema}

\begin{zed}
```

```
        Response ::= Success | NotFound \\
\end{zed}

\begin{schema} {Success}
        r! : Response \\
\where
        r! = Success \\
\end{schema}

\begin{schema} {NotFound}
        r! : Response \\
\where
        r! = NotFound \\
\end{schema}
```

กรณีทดสอบที่ 1 มีรายละเอียดแฟ้มข้อความของแผนภาพเชิงลำดับชั้นของภาษาเอสคิวแอล ดังนี้

```
%BDDBEGIN
1:Join:Booking:Customer:Booking,custId = Customer,custId:2:NotFound
2:Select:1:-:class = "Minivan"_\land_custAddr = "Bangkok":3:NotFound
3:Project:2:-:vanNo custName:Success:NotFound
%BDDEND
```

ข้อกำหนดเซตของแผนภาพเชิงลำดับชั้นของภาษาเอสคิวแอล มีดังนี้

```
\begin{schema} {BDD1R1}
        bookId : BOOKID \\
        startDate : DATE \\
        endDate : DATE \\
        custId : CUSTID \\
        class : CLASSNAME \\
        vanNo : REG \\
        custName : CUSTNAME \\
        custAddr : CUSTADDR \\
\end{schema}

\begin{schema} {BDD1Op1Join}
        input1? : \finset Booking \\
        input2? : \finset Customer \\
        output! : \finset BDD1R1 \\
\where
        input1? \neq \emptyset \\
        input1? \neq \emptyset \\
        \forall out : output!; in1 : input1?; in2 : input2? | in1.custId = in2.custId \\
        @ out.bookId = in1.bookId  \land out.startDate = in1.startDate \land \\
          out.endDate = in1.endDate \land out.custId = in1.custId \land \\
          out.class = in1.class \land out.vanNo = in1.vanNo \land \\
          out.custName = in2.custName \land out.custAddr = in2.custAddr
\end{schema}

\begin{schema} {BDD1R2}
        bookId : BOOKID \\
        startDate : DATE \\
        endDate : DATE \\
        custId : CUSTID \\
```

```
        class : CLASSNAME \\
        vanNo : REG \\
        custName : CUSTNAME \\
        custAddr : CUSTADDR \\
\end{schema}

\begin{schema} {BDD1Op2Select}
        input1? : \finset BDD1R1 \\
        classValue? : CLASSNAME \\
        custAddrValue? : CUSTADDR \\
        output! : \finset BDD1R2 \\
\where
        input1? \neq \emptyset \\
        \forall out : output!; in1 : input1? \\
        | in1.class = classValue? \land in1.custAddr = custAddrValue? @ out = in1 \\
\end{schema}

\begin{schema} {BDD1R3}
        vanNo : REG \\
        custName : CUSTNAME \\
\end{schema}

\begin{schema} {BDD1Op3Project}
        input1? : \finset BDD1R2 \\
        output! : \finset BDD1R3 \\
\where
        input1? \neq \emptyset \\
        \forall out : output!; in1 : input1? \\
        @ out.vanNo = in1.vanNo \land out.custName = in1.custName
\end{schema}

\begin{zed}
        BDD1 \defs \\
        BDD1Op1Join [op1input1?/input1?, op1input2?/input2?, op1output!/output!] \semi \\
        BDD1Op2Select [op2input1?/input1?, op2output!/output!] \semi \\
        BDD1Op3Project [op3input1?/input1?, op3output!/output!] \\
\end{zed}
```

การทดสอบโดยการสร้างเงื่อนไขก่อน ของ BDD1Op1Join ด้วยคำสั่ง

try ∀ input1? : 𝔽 Booking; input2? : 𝔽 Customer • pre BDD1Op1Join;

เมื่อใช้คำสั่ง prove by reduce ได้ผลดังนี้

```
input1? \in \finset Booking \land input2? \in \finset Customer \\
\implies
(\exists output!: \power \lblot bookId: BOOKID, class: CLASSNAME, custAddr: CUSTADDR,
        custId: CUSTID, custName: CUSTNAME,
        endDate: DATE, startDate: DATE, vanNo: REG \rblot
@       output! \in \finset BDD1R1 \land \lnot input1? = \{\} \land \\
        \lnot input2? = \{\} \land
        (\forall out: output!; in1: input1?; in2: input2? | in1.custId = in2.custId
```

@ (out.bookId = in1.bookId \land \\
out.startDate = in1.startDate \land \\
out.endDate = in1.endDate \land out.custId = in1.custId \\
\land out.class = in1.class \land out.vanNo = in1.vanNo \\
\land out.custName = in2.custName \\
\land out.custAddr = in2.custAddr)))

การทดสอบโดยการสร้างเงื่อนไขก่อน ของ BDD1Op2Select ด้วยคำสั่ง

try ∀ input1? : 𝔽 BDD1R1; classValue? : CLASSNAME;
custAddrValue? : CUSTADDR • pre BDD1Op2Select;

เมื่อใช้คำสั่ง prove by reduce ได้ผลดังนี้

input1? \in \finset BDD1R1 \land classValue? \in CLASSNAME \\
\land custAddrValue? \in CUSTADDR \\
\implies
(\exists output!: \power \lblot bookId: BOOKID, class: CLASSNAME, custAddr: CUSTADDR,
custId: CUSTID, custName: CUSTNAME, endDate: DATE, startDate: DATE,
vanNo: REG \rblot
@ output! \in \finset BDD1R2 \land \lnot input1? = \{\} \land \\
(\forall out: output!; in1: input1? \\
| in1.class = classValue? \land custAddrValue? = in1.custAddr
@ out = in1))

การทดสอบโดยการสร้างเงื่อนไขก่อน ของ BDD1Op3Project ด้วยคำสั่ง

try ∀ input1? : 𝔽 BDD1R2 • pre BDD1Op3Project;

เมื่อใช้คำสั่ง prove by reduce ได้ผลดังนี้

input1? \in \finset BDD1R2 \\
\implies
(\exists output!: \power \lblot custName: CUSTNAME, vanNo: REG \rblot
@ output! \in \finset BDD1R3 \land \lnot input1? = \{\} \land \\
(\forall out: output!; in1: input1?
@ (out.vanNo = in1.vanNo \land out.custName = in1.custName)))

การทดสอบ BDD1 ด้วยคำสั่ง

try BDD1 [op1input1? := BookingData, op1input2? := CustomerData,
op2classValue? := Minivan, op2custAddrValue? := Bangkok];

เมื่อใช้คำสั่ง prove by reduce ได้ผลดังนี้

BookingData \in \power \lblot bookId: BOOKID, class: CLASSNAME, custId: CUSTID, \\
endDate: DATE, startDate: DATE, vanNo: REG \rblot \\

```
        \land
CustomerData \in \power \lblot custAddr: CUSTADDR, custId: CUSTID,
                custName: CUSTNAME \rblot \\
\land op1output! \in \power \lblot bookId: BOOKID, class: CLASSNAME,
        custAddr: CUSTADDR, custId : CUSTID, custName: CUSTNAME,
        endDate: DATE, startDate: DATE, vanNo: REG \rblot \\
\land Minivan \in CLASSNAME \\
\land Bangkok \in CUSTADDR \\
\land op2input1? \in \power \lblot bookId: BOOKID, class: CLASSNAME,
        custAddr: CUSTADDR, custId: CUSTID, custName: CUSTNAME,
        endDate: DATE, startDate: DATE,  vanNo: REG \rblot \\
\land op2output! \in \power \lblot bookId: BOOKID, class: CLASSNAME,
        custAddr: CUSTADDR, custId: CUSTID, custName: CUSTNAME,
        endDate: DATE, startDate: DATE,  vanNo: REG \rblot \\
\land op3input1? \in \power \lblot bookId: BOOKID, class: CLASSNAME,
        custAddr: CUSTADDR, custId: CUSTID, custName: CUSTNAME,
        endDate: DATE, startDate: DATE, vanNo: REG \rblot \\
\land op3output! \in \power \lblot custName: CUSTNAME, vanNo: REG \rblot \\
\land BookingData \in \finset Booking \\
\land CustomerData \in \finset Customer \\
\land op1output! \in \finset BDD1R1 \\
\land op2input1? \in \finset BDD1R1 \\
\land op2output! \in \finset BDD1R2 \\
\land op3input1? \in \finset BDD1R2 \\
\land op3output! \in \finset BDD1R3 \\
        \land \lnot BookingData = \{\} \\
        \land \lnot CustomerData = \{\} \\
        \land \lnot op2input1? = \{\} \\
        \land \lnot op3input1? = \{\} \\
        \land (         out \in op1output! \\
        \land in1 \in BookingData \\
        \land in2 \in CustomerData \\
        \land in1.custId = in2.custId \\
\implies out.bookId = in1.bookId \\
```

```
        \land out.startDate = in1.startDate \\
        \land out.endDate = in1.endDate \\
        \land out.custId = in1.custId \\
        \land out.class = in1.class \\
        \land out.vanNo = in1.vanNo \\
        \land out.custName = in2.custName \\
        \land out.custAddr = in2.custAddr) \\
        \land (out\_\_0 \in op2output! \\
        \land in1\_\_0 \in op2input1? \\
        \land in1\_\_0.class = Minivan \\
        \land in1\_\_0.custAddr = Bangkok \\
\implies out\_\_0 = in1\_\_0) \\
        \land (out\_\_1 \in op3output! \\
        \land in1\_\_1 \in op3input1? \\
\implies out\_\_1.vanNo = in1\_\_1.vanNo \\
        \land out\_\_1.custName = in1\_\_1.custName)
```

กรณีทดสอบที่ 2 มีรายละเอียดแฟ้มข้อความของแผนภาพเชิงลำดับชั้นของภาษาเอสคิวแอล ดังนี้

```
%BDDBEGIN
1:Select:Booking:-:class = "Minivan":Success:NotFound
2:Select:Booking:-:class = "Wagon":Success:NotFound
3:Union:1:2:-:4:NotFound
4:Project:3:-:vanNo:Success:NotFound
%BDDEND
```

ข้อกำหนดเซตของแผนภาพเชิงลำดับชั้นของภาษาเอสคิวแอล มีดังนี้

```
\begin{schema} {BDD2R1}
        bookId : BOOKID \\
        startDate : DATE \\
        endDate : DATE \\
        custId : CUSTID \\
        class : CLASSNAME \\
        vanNo : REG \\
\end{schema}

\begin{schema} {BDD2Op1Select}
        input1? : \finset Booking \\
        classValue? : CLASSNAME \\
        output! : \finset BDD2R1 \\
\where
        input1? \neq \emptyset \\
        \forall out : output!; in1 : input1? | in1.class = classValue? @ out = in1 \\
\end{schema}
```

```
\begin{schema} {BDD2R2}
        bookId : BOOKID \\
        startDate : DATE \\
        endDate : DATE \\
        custId : CUSTID \\
        class : CLASSNAME \\
        vanNo : REG \\
\end{schema}

\begin{schema} {BDD2Op2Select}
        input1? : \finset Booking \\
        classValue? : CLASSNAME \\
        output! : \finset BDD2R2 \\
\where
        input1? \neq \emptyset \\
        \forall out : output!; in1 : input1? | in1.class = classValue? @ out = in1 \\
\end{schema}

\begin{schema} {BDD2R3}
        bookId : BOOKID \\
        startDate : DATE \\
        endDate : DATE \\
        custId : CUSTID \\
        class : CLASSNAME \\
        vanNo : REG \\
\end{schema}

\begin{schema} {BDD2Op3Union}
        \Xi BDD2R1 \\
        \Xi BDD2R2 \\
        input1? : \finset BDD2R1 \\
        input2? : \finset BDD2R2 \\
        output! : \finset BDD2R3 \\
\where
        \theta BDD2R1 = \theta BDD2R2 \\
        \forall out : \finset output!; in1: input1?; in2 : input2? \\
        @ out = \{ in1 \} \cup \{ in2 \}
\end{schema}

\begin{schema} {BDD2R4}
        vanNo : REG \\
\end{schema}

\begin{schema} {BDD2Op4Project}
        input1? : \finset BDD2R3 \\
        output! : \finset BDD2R4 \\
\where
        input1? \neq \emptyset \\
        \forall out : output!; in1 : input1? @ out.vanNo = in1.vanNo
\end{schema}

\begin{zed}
        BDD2 \defs \\
        BDD2Op1Select [op1input1?/input1?, op1classValue?/classValue?,
                op1output!/output!] \semi \\
        BDD2Op2Select [op2input1?/input1?, op2classValue?/classValue?,
                op2output!/output!] \semi \\
        BDD2Op3Union [op3input1?/input1?, op3input2?/input2?,
```

```
                op3output!/output!] \semi \\
        BDD2Op4Project [op4input1?/input1?, op4output! / output! ]
\end{zed}
```

การทดสอบโดยการสร้างเงื่อนไขก่อน ของ BDD2Op1Select ด้วยคำสั่ง

try $\forall$ *input1? :* $\mathbb{F}$ *Booking; classValue? : CLASSNAME • pre BDD2Op1Select;*

เมื่อใช้คำสั่ง prove by reduce ได้ผลดังนี้

```
input1? \in \finset Booking \land classValue? \in CLASSNAME \\
\implies
(\exists output!: \power \lblot bookId: BOOKID, class: CLASSNAME, custId: CUSTID,
        endDate: DATE, startDate: DATE, vanNo: REG \rblot
@       output! \in \finset BDD2R1 \land \lnot input1? = \{\} \land \\
        (\forall out: output!; in1: input1? | in1.class = classValue? @ out = in1))
```

การทดสอบโดยการสร้างเงื่อนไขก่อน ของ BDD1Op2Select ด้วยคำสั่ง

try $\forall$ *input1? :* $\mathbb{F}$ *Booking; classValue? : CLASSNAME • pre BDD2Op2Select;*

เมื่อใช้คำสั่ง prove by reduce ได้ผลดังนี้

```
input1? \in \finset Booking \land classValue? \in CLASSNAME \\
\implies
(\exists output!: \power \lblot bookId: BOOKID, class: CLASSNAME, custId: CUSTID,
        endDate: DATE, startDate: DATE, vanNo: REG \rblot
@       output! \in \finset BDD2R2 \land \lnot input1? = \{\} \land \\
        (\forall out: output!; in1: input1? | in1.class = classValue? @ out = in1))
```

การทดสอบโดยการสร้างเงื่อนไขก่อน ของ BDD1Op3Unionn ด้วยคำสั่ง

try $\forall$ *BDD2R1; BDD2R2; input1? :* $\mathbb{F}$ *BDD2R1; input2? :* $\mathbb{F}$ *BDD2R2;*
• pre BDD2Op3Union;

เมื่อใช้คำสั่ง prove by reduce ได้ผลดังนี้

```
bookId \in BOOKID \land startDate \in DATE \land endDate \in DATE \\
        \land custId \in CUSTID \land class \in CLASSNAME \\
        \land vanNo \in REG \land input1? \in \finset BDD2R1 \\
        \land input2 \in \finset BDD2R2 \\
\implies
(\exists output!: \power \lblot bookId: BOOKID, class: CLASSNAME,
        custId: CUSTID, endDate: DATE, startDate: DATE, vanNo: REG \rblot
@       input2? \in \finset BDD2R2 \land output! \in \finset BDD2R3 \land \\
        (\forall out: \finset output!; in1: input1?; in2: input2?
        |       in1.custId \in CUSTID \land in2.custId \in CUSTID \\
                \land \lnot in1.custId = in2.custId @ out = \{in1\} \cup \{in2\}))
```

การทดสอบโดยการสร้างเงื่อนไขก่อน ของ BDD1Op4Project ด้วยคำสั่ง

try ∀ input1? : 𝔽 BDD2R3 • pre BDD2Op4Project;

เมื่อใช้คำสั่ง prove by reduce ได้ผลดังนี้

input1? \in \finset BDD2R3 \\

\implies

(\exists output!: \power \lblot custName: CUSTNAME, vanNo: REG \rblot

@ output! \in \finset BDD2R4 \land \lnot input1? = \{\} \\

\land (\forall out: output!; in1: input1? @ out.vanNo = in1.vanNo))

การทดสอบ BDD2 ด้วยคำสั่ง

try BDD2 [op1input1? := BookingData, op1classValue? := Minivan, op2input1? := BookingData, op2classValue? := Wagon];

เมื่อใช้คำสั่ง prove by reduce ได้ผลดังนี้

bookId' \in BOOKID \\

\land bookId = bookId' \\

\land class' \in CLASSNAME \\

\land class = class' \\

\land custId' \in CUSTID \\

\land custId = custId' \\

\land endDate' \in DATE \\

\land endDate = endDate' \\

\land Minivan \in CLASSNAME \\

\land BookingData \in \power \lblot bookId: BOOKID, class: CLASSNAME,

custId: CUSTID, endDate: DATE, startDate: DATE, vanNo: REG \rblot \\

\land op1output! \in \power \lblot bookId: BOOKID, class: CLASSNAME,

custId: CUSTID, endDate: DATE, startDate: DATE, vanNo: REG \rblot \\

\land Wagon \in CLASSNAME \\

\land BookingData \in \power \lblot bookId: BOOKID, class: CLASSNAME,

custId: CUSTID, endDate: DATE, startDate: DATE, vanNo: REG \rblot \\

\land op2output! \in \power \lblot bookId: BOOKID, class: CLASSNAME, custId: CUSTID,

endDate: DATE, startDate: DATE, vanNo: REG \rblot \\

\land op3input1? \in \power \lblot bookId: BOOKID, class: CLASSNAME, custId: CUSTID,

endDate: DATE, startDate: DATE, vanNo: REG \rblot \\

\land op3input2? \in \power \lblot bookId: BOOKID, class: CLASSNAME, custId: CUSTID,

endDate: DATE, startDate: DATE, vanNo: REG \rblot \\

\land op3output! \in \power \lblot bookId: BOOKID, class: CLASSNAME, custId: CUSTID,

```
        endDate: DATE, startDate: DATE, vanNo: REG \rblot \\
\land op4input1? \in \power \lblot bookId: BOOKID, class: CLASSNAME, custId: CUSTID,
        endDate: DATE, startDate: DATE, vanNo: REG \rblot \\
\land op4output! \in \power \lblot vanNo: REG \rblot \\
\land startDate' \in DATE \\
\land startDate = startDate' \\
\land vanNo' \in REG \\
\land vanNo = vanNo' \\
\land BookingData \in \finset Booking \\
\land op1output! \in \finset BDD2R1 \\
\land BookingData \in \finset Booking \\
\land op2output! \in \finset BDD2R2 \\
\land op3input1? \in \finset BDD2R1 \\
\land op3input2? \in \finset BDD2R2 \\
\land op3output! \in \finset BDD2R3 \\
\land op4input1? \in \finset BDD2R3 \\
\land op4output! \in \finset BDD2R4 \\
        \land \lnot BookingData = \{\} \\
        \land \lnot BookingData = \{\} \\
        \land \lnot op4input1? = \{\} \\
        \land (out \in op1output! \\
        \land in1 \in BookingData \\
        \land in1.class = Minivan \\
\implies  out = in1) \\
        \land (out\_\_0 \in op2output! \\
        \land in1\_\_0 \in BookingData \\
        \land in1\_\_0.class = Wagon \\
\implies out\_\_0 = in1\_\_0) \\
        \land (out\_\_1 \in \finset op3output! \\
        \land in1\_\_1 \in op3input1? \\
        \land in2 \in op3input2? \\
        \land in1\_\_1.custId \in CUSTID \\
        \land in2.custId \in CUSTID \\
        \land \lnot in1\_\_1.custId = in2.custId \\
```

```
\implies out\_\_1 = \{in1\_\_1\} \cup \{in2\}) \\
        \land (out\_\_2 \in op4output! \\
        \land in1\_\_2 \in op4input1? \\
\implies out\_\_2.vanNo = in1\_\_2.vanNo)
```

กรณีทดสอบที่ 3 มีรายละเอียดแฟ้มข้อความของแผนภาพเชิงลำดับชั้นของภาษาเอสคิวแอล ดังนี้

```
%BDDBEGIN
1:Join:Customer:Booking:Customer,custId = Booking,custId:2:NotFound
2:Select:1:-:class = "Minivan":Success:NotFound
3:Join:Customer:Booking:Customer,custId = Booking,custId:4:NotFound
4:Select:3:-:class = "Wagon":Success:NotFound
5:Except:2:4:-:6:NotFound
6:Project:5:-:custName:Success:NotFound
%BDDEND
```

ข้อกำหนดเซตของแผนภาพเชิงลำดับชั้นของภาษาเอสคิวแอล มีดังนี้

```
\begin{schema} {BDD3R1}
        custId : CUSTID \\
        custName : CUSTNAME \\
        custAddr : CUSTADDR \\
        bookId : BOOKID \\
        startDate : DATE \\
        endDate : DATE \\
        custId : CUSTID \\
        class : CLASSNAME \\
        vanNo : REG \\
\end{schema}

\begin{schema} {BDD3Op1Join}
        input1? : \finset Customer \\
        input2? : \finset Booking \\
        output! : \finset BDD3R1 \\
\where
        input1? \neq \emptyset \\
        input2? \neq \emptyset \\
        \forall out : output!; in1 : input1?; in2 : input2? \\
        | in1.custId = in2.custId \\
        @ out.custId = in1.custId \land out.custName = in1.custName \land \\
            out.custAddr = in1.custAddr \land out.bookId = in2.bookId  \land \\
            out.startDate = in2.startDate \land out.endDate = in2.endDate \land \\
            out.class = in2.class \land out.vanNo = in2.vanNo \\
\end{schema}

\begin{schema} {BDD3R2}
        custId : CUSTID \\
        custName : CUSTNAME \\
        custAddr : CUSTADDR \\
        bookId : BOOKID \\
        startDate : DATE \\
        endDate : DATE \\
        custId : CUSTID \\
        class : CLASSNAME \\
```

```
        vanNo : REG \\
\end{schema}

\begin{schema} {BDD3Op2Select}
        input1? : \finset BDD3R1 \\
        classValue? : CLASSNAME \\
        output! : \finset BDD3R2 \\
\where
        input1? \neq \emptyset \\
        \forall out : output!; in1 : input1? | in1.class = classValue? @ out = in1 \\
\end{schema}

\begin{schema} {BDD3R3}
        custId : CUSTID \\
        custName : CUSTNAME \\
        custAddr : CUSTADDR \\
        bookId : BOOKID \\
        startDate : DATE \\
        endDate : DATE \\
        custId : CUSTID \\
        class : CLASSNAME \\
        vanNo : REG \\
\end{schema}

\begin{schema} {BDD3Op3Join}
        input1? : \finset Customer \\
        input2? : \finset Booking \\
        output! : \finset BDD3R3 \\
\where
        input1? \neq \emptyset \\
        input2? \neq \emptyset \\
        \forall out : output!; in1 : input1?; in2 : input2? \\
        | in1.custId = in2.custId \\
        @ out.custId = in1.custId \land out.custName = in1.custName \land \\
            out.custAddr = in1.custAddr \land out.bookId = in2.bookId  \land \\
            out.startDate = in2.startDate \land out.endDate = in2.endDate \land \\
            out.class = in2.class \land out.vanNo = in2.vanNo \\
\end{schema}

\begin{schema} {BDD3R4}
        custId : CUSTID \\
        custName : CUSTNAME \\
        custAddr : CUSTADDR \\
        bookId : BOOKID \\
        startDate : DATE \\
        endDate : DATE \\
        custId : CUSTID \\
        class : CLASSNAME \\
        vanNo : REG \\
\end{schema}

\begin{schema} {BDD3Op4Select}
        input1? : \finset BDD3R3 \\
        classValue? : CLASSNAME \\
        output! : \finset BDD3R4 \\
\where
        input1? \neq \emptyset \\
        \forall out : output!; in1 : input1? | in1.class = classValue? @ out = in1 \\
```

```
\end{schema}

\begin{schema} {BDD3R5}
        custId : CUSTID \\
        custName : CUSTNAME \\
        custAddr : CUSTADDR \\
        bookId : BOOKID \\
        startDate : DATE \\
        endDate : DATE \\
        custId : CUSTID \\
        class : CLASSNAME \\
        vanNo : REG \\
\end{schema}

\begin{schema} {BDD3Op5Except}
        \Xi BDD3R2 \\
        \Xi BDD3R4 \\
        input1? : \finset BDD3R2 \\
        input2? : \finset BDD3R4 \\
        output! : \finset BDD3R5 \\
\where
        \theta BDD3R2 = \theta BDD3R4 \\
        \forall out : \finset output!; in1: input1?; in2 : input2? \\
                @ out = \{ in1 \} \setminus \{ in2 \} \\
\end{schema}

\begin{schema} {BDD3R6}
        custName : CUSTNAME \\
\end{schema}

\begin{schema} {BDD3Op6Project}
        input1? : \finset BDD3R5 \\
        output! : \finset BDD3R6 \\
\where
        input1? \neq \emptyset \\
        \forall out : output!; in1 : input1? @ out.custName = in1.custName \\
\end{schema}

\begin{zed}
        BDD3 \defs \\
        BDD3Op1Join [op1input1?/input1?, op1input2?/input2?, op1output!/output!] \semi \\
        BDD3Op2Select [op2input1?/input1?, op2classValue?/classValue?,
                op2output!/output!] \semi \\
        BDD3Op3Join [op3input1?/input1?, op3input2?/input2?, op3output!/output!] \semi \\
        BDD3Op4Select [op4input1?/input1?, op4classValue?/classValue?,
                op4output!/output!] \semi \\
        BDD3Op5Except [op5input1?/input1?, op5input2?/input2?,
                op5output!/output!] \semi \\
        BDD3Op6Project [op6input1?/input1?, op6output!/output!] \\
\end{zed}
```

การทดสอบโดยการสร้างเงื่อนไขก่อน ของ BDD3Op1Join ด้วยคำสั่ง

try $\forall$ *input1? :* $\mathbb{F}$ *Booking; input2? :* $\mathbb{F}$ *Customer • pre BDD3Op1Join;*

เมื่อใช้คำสั่ง prove by reduce ได้ผลดังนี้

input1? \in \finset Booking \land input2? \in \finset Customer \\

\implies

(\exists output!: \power \lblot bookId: BOOKID, class: CLASSNAME, custAddr: CUSTADDR,

custId: CUSTID, custName: CUSTNAME, endDate: DATE, startDate: DATE,

vanNo: REG \rblot

@ output! \in \finset BDD3R1 \land \lnot input1? = \{\} \land \lnot input2? = \{\} \land \\

(\forall out: output!; in1: input1?; in2: input2? | in1.custId = in2.custId

@ (out.bookId = in1.bookId \land out.startDate = in1.startDate \\

\land out.endDate = in1.endDate \land out.custId = in1.custId \\

\land out.class = in1.class \land out.vanNo = in1.vanNo \\

\land out.custName = in2.custName \land out.custAddr = in2.custAddr)))

การทดสอบโดยการสร้างเงื่อนไขก่อน ของ BDD3Op2Select ด้วยคำสั่ง

try $\forall$ *input1? :* $\mathbb{F}$ *BDD3R1; classValue? : CLASSNAME • pre BDD3Op2Select;*

เมื่อใช้คำสั่ง prove by reduce ได้ผลดังนี้

input1? \in \finset BDD3R1 \land classValue? \in CLASSNAME \\

\implies

(\exists output!: \power \lblot bookId: BOOKID, class: CLASSNAME, custAddr: CUSTADDR,

custId: CUSTID, custName: CUSTNAME, endDate: DATE, startDate: DATE,

vanNo: REG \rblot

@ output! \in \finset BDD3R2 \land \lnot input1? = \{\} \land \\

(\forall out: output!; in1: input1? | in1.class = classValue? @ out = in1))

การทดสอบโดยการสร้างเงื่อนไขก่อน ของ BDD3Op3Join ด้วยคำสั่ง

try $\forall$ *input1? :* $\mathbb{F}$ *Booking; input2? :* $\mathbb{F}$ *Customer • pre BDD3Op3Join;*

เมื่อใช้คำสั่ง prove by reduce ได้ผลดังนี้

input1? \in \finset Booking \land input2? \in \finset Customer \\

\implies

(\exists output!: \power \lblot bookId: BOOKID, class: CLASSNAME, custAddr: CUSTADDR,

custId: CUSTID, custName: CUSTNAME, endDate: DATE, startDate: DATE,

vanNo: REG \rblot

@ output! \in \finset BDD3R3 \land \lnot input1? = \{\} \land \lnot input2? = \{\} \land \\

(\forall out: output!; in1: input1?; in2: input2? | in1.custId = in2.custId

@ (out.bookId = in1.bookId \land out.startDate = in1.startDate \\

\land out.endDate = in1.endDate \land out.custId = in1.custId \\

\land out.class = in1.class \land out.vanNo = in1.vanNo \\

\land out.custName = in2.custName \land out.custAddr = in2.custAddr)))

การทดสอบโดยการสร้างเงื่อนไขก่อน ของ BDD3Op4Select ด้วยคำสั่ง

try $\forall$ *input1? :* $\mathbb{F}$ *BDD3R3; classValue? : CLASSNAME • pre BDD3Op4Select;*
เมื่อใช้คำสั่ง prove by reduce ได้ผลดังนี้

```
input1? \in \finset BDD3R3 \land classValue? \in CLASSNAME \\
\implies
(\exists output!: \power \lblot bookId: BOOKID, class: CLASSNAME, custAddr: CUSTADDR,
        custId: CUSTID, custName: CUSTNAME, endDate: DATE, startDate: DATE,
        vanNo: REG \rblot
@       output! \in \finset BDD3R4 \land \lnot input1? = \{\} \land \\
        (\forall out: output!; in1: input1? | in1.class \neq classValue? @ out = in1))
```

การทดสอบโดยการสร้างเงื่อนไขก่อน ของ BDD3Op5Except ด้วยคำสั่ง

try $\forall$ *BDD3R2; BDD3R4; input1? :* $\mathbb{F}$ *BDD3R2; input2? :* $\mathbb{F}$ *BDD3R5*
• pre BDD3Op5Except;
เมื่อใช้คำสั่ง prove by reduce ได้ผลดังนี้

```
bookId \in BOOKID \land startDate \in DATE \land endDate \in DATE \land \\
        custId \in CUSTID \land class \in CLASSNAME \land vanNo \in REG \land \\
        input1? \in \finset BDD3R2 \land input2 \in \finset BDD3R4 \\
\implies
(\exists output!: \power \lblot bookId: BOOKID, class: CLASSNAME, custId: CUSTID,
        endDate: DATE, startDate: DATE, vanNo: REG \rblot
@       input2? \in \finset BDD3R2 \land output! \in \finset BDD3R4 \land \\
       (\forall out: \finset output!; in1: input1?; in2: input2?
        | in1.custId \in CUSTID \land in2.custId \in CUSTID \land \\
          \lnot in1.custId = in2.custId @ out = \{in1\} \setminus \{in2\}))
```

การทดสอบโดยการสร้างเงื่อนไขก่อน ของ BDD3Op6Project ด้วยคำสั่ง

try $\forall$ *input1? :* $\mathbb{F}$ *BDD3R5 • pre BDD3Op6Project;*
เมื่อใช้คำสั่ง prove by reduce ได้ผลดังนี้

```
input1? \in \finset BDD3R5 \\
\implies
(\exists output!: \power \lblot custName: CUSTNAME, vanNo: REG \rblot
@       output! \in \finset BDD3R6 \land \lnot input1? = \{\} \land \\
        (\forall out: output!; in1: input1? @ out.custName = in1.custName))
```

การทดสอบ BDD3 ด้วยคำสั่ง

```
try BDD3 [op1input1? := CustomerData, op1input2? := BookingData,
        op2classValue? := minivan, op3input1? := CustomerData,
        op3input2? := BookingData, op4classValue? := Wagon];
```

เมื่อใช้คำสั่ง prove by reduce ได้ผลดังนี้

```
bookId' \in BOOKID \\
\land bookId = bookId' \\
\land class' \in CLASSNAME \\
\land class = class' \\
\land custAddr' \in CUSTADDR \\
\land custAddr = custAddr' \\
\land custId' \in CUSTID \\
\land custId = custId' \\
\land custName' \in CUSTNAME \\
\land custName = custName' \\
\land endDate' \in DATE \\
\land endDate = endDate' \\
\land CustomerData \in \power \lblot custAddr: CUSTADDR, custId: CUSTID,
        custName: CUSTNAME \rblot \\
\land BookingData \in \power \lblot bookId: BOOKID, class: CLASSNAME,
        custId: CUSTID, endDate: DATE, startDate: DATE, vanNo: REG \rblot \\
\land op1output! \in \power \lblot bookId: BOOKID, class: CLASSNAME,
        custAddr: CUSTADDR, custId: CUSTID, custName: CUSTNAME,
        endDate: DATE, startDate: DATE, vanNo: REG \rblot \\
\land minivan \in CLASSNAME \\
\land op2input1? \in \power \lblot bookId: BOOKID, class: CLASSNAME,
        custAddr: CUSTADDR, custId: CUSTID, custName: CUSTNAME,
        endDate: DATE, startDate: DATE, vanNo: REG \rblot \\
\land op2output! \in \power \lblot bookId: BOOKID, class: CLASSNAME,
        custAddr: CUSTADDR, custId: CUSTID, custName: CUSTNAME,
        endDate: DATE, startDate: DATE, vanNo: REG \rblot \\
\land op3output! \in \power \lblot bookId: BOOKID, class: CLASSNAME,
        custAddr: CUSTADDR, custId: CUSTID, custName: CUSTNAME,
        endDate: DATE, startDate: DATE, vanNo: REG \rblot \\
\land Wagon \in CLASSNAME \\
\land op4input1? \in \power \lblot bookId: BOOKID, class: CLASSNAME,
        custAddr: CUSTADDR, custId: CUSTID, custName: CUSTNAME,
        endDate: DATE, startDate: DATE, vanNo: REG \rblot \\
```

```
\land op4output! \in \power \lblot bookId: BOOKID, class: CLASSNAME,
        custAddr: CUSTADDR, custId: CUSTID, custName: CUSTNAME,
        endDate: DATE, startDate: DATE, vanNo: REG \rblot \\
\land op5input1? \in \power \lblot bookId: BOOKID, class: CLASSNAME,
        custAddr: CUSTADDR, custId: CUSTID, custName: CUSTNAME,
        endDate: DATE, startDate: DATE, vanNo: REG \rblot \\
\land op5input2? \in \power \lblot bookId: BOOKID, class: CLASSNAME,
        custAddr: CUSTADDR, custId: CUSTID, custName: CUSTNAME,
        endDate: DATE, startDate: DATE, vanNo: REG \rblot \\
\land op5output! \in \power \lblot bookId: BOOKID, class: CLASSNAME,
        custAddr: CUSTADDR, custId: CUSTID, custName: CUSTNAME,
        endDate: DATE, startDate: DATE, vanNo: REG \rblot \\
\land op6input1? \in \power \lblot bookId: BOOKID, class: CLASSNAME,
        custAddr: CUSTADDR, custId: CUSTID, custName: CUSTNAME,
        endDate: DATE, startDate: DATE, vanNo: REG \rblot \\
\land op6output! \in \power \lblot custName: CUSTNAME \rblot \\
\land startDate' \in DATE \\
\land startDate = startDate' \\
\land vanNo' \in REG \\
\land vanNo = vanNo' \\
\land CustomerData \in \finset Customer \\
\land BookingData \in \finset Booking \\
\land op1output! \in \finset BDD3R1 \\
\land op2input1? \in \finset BDD3R1 \\
\land op2output! \in \finset BDD3R2 \\
\land op3output! \in \finset BDD3R3 \\
\land op4input1? \in \finset BDD3R3 \\
\land op4output! \in \finset BDD3R4 \\
\land op5input1? \in \finset BDD3R2 \\
\land op5input2? \in \finset BDD3R4 \\
\land op5output! \in \finset BDD3R5 \\
\land op6input1? \in \finset BDD3R5 \\
\land op6output! \in \finset BDD3R6 \\
\land \lnot CustomerData = \{\} \\
```

```
\land \lnot BookingData = \{\} \\
\land \lnot op2input1? = \{\} \\
\land \lnot op4input1? = \{\} \\
\land \lnot op6input1? = \{\} \\
\land
(out \in op1output! \land in1 \in CustomerData \land in2 \in BookingData \\
        \land in1.custId = in2.custId \\
\implies out.custId = in1.custId \\
        \land out.custName = in1.custName \\
        \land out.custAddr = in1.custAddr \\
        \land out.bookId = in2.bookId \\
        \land out.startDate = in2.startDate \\
        \land out.endDate = in2.endDate \\
        \land out.class = in2.class \\
        \land out.vanNo = in2.vanNo) \\
        \land (out\_\_0 \in op2output! \\
        \land in1\_\_0 \in op2input1? \\
        \land in1\_\_0.class = minivan \\
\implies out\_\_0 = in1\_\_0) \\
        \land (out\_\_1 \in op3output! \\
        \land in1\_\_1 \in CustomerData \\
        \land in2\_\_0 \in BookingData \\
        \land in1\_\_1.custId = in2\_\_0.custId \\
\implies out\_\_1.custId = in1\_\_1.custId \\
        \land out\_\_1.custName = in1\_\_1.custName \\
        \land out\_\_1.custAddr = in1\_\_1.custAddr \\
        \land out\_\_1.bookId = in2\_\_0.bookId \\
        \land out\_\_1.startDate = in2\_\_0.startDate \\
        \land out\_\_1.endDate = in2\_\_0.endDate \\
        \land out\_\_1.class = in2\_\_0.class \\
        \land out\_\_1.vanNo = in2\_\_0.vanNo) \\
        \land (out\_\_2 \in op4output! \\
        \land in1\_\_2 \in op4input1? \\
        \land in1\_\_2.class \in CLASSNAME \\
```

```
        \land \lnot in1\_\_2.class = Wagon \\
\implies out\_\_2 = in1\_\_2) \\
        \land (out\_\_3 \in \finset op5output! \\
        \land in1\_\_3 \in op5input1? \\
        \land in2\_\_1 \in op5input2? \\
        \land in1\_\_3.custName \in CUSTNAME \\
        \land in2\_\_1.custName \in CUSTNAME \\
        \land \lnot in1\_\_3.custName = in2\_\_1.custName \\
\implies out\_\_3 = \{in1\_\_3\}) \\
        \land (out\_\_4 \in op6output! \\
        \land in1\_\_4 \in op6input1? \\
\implies out\_\_4.custName = in1\_\_4.custName)
```

กรณีทดสอบที่ 4 มีรายละเอียดแฟ้มข้อความของแผนภาพเชิงลำดับชั้นของภาษาเอสคิวแอล ดังนี้

```
%BDDBEGIN
1:Select:Van:-:Van.class = "Minivan":2:NotFound
2:Count:1:-:-:Success:NotFound
%BDDEND
```

ข้อกำหนดเซตของแผนภาพเชิงลำดับชั้นของภาษาเอสคิวแอล มีดังนี้

```
\begin{schema} {BDD4R1}
        reg : REG \\
        model : MODEL \\
        class : CLASSNAME \\
\end{schema}

\begin{schema} {BDD4Op1Select}
        input1? : \finset Van \\
        classValue? : CLASSNAME \\
        output! : \finset BDD4R1 \\
\where
        input1? \neq \emptyset \\
        \forall out : output!; in1 : input1? | in1.class = classValue? @ out = in1 \\
\end{schema}

\begin{schema} {BDD4Op2Count}
        input1? : \finset BDD4R1 \\
        output! : \nat \\
\where
        input1? \neq \emptyset \\
        output! = \# input1? \\
\end{schema}

\begin{zed}
        BDD4 \defs \\
        BDD4Op1Select  [op1input1?/input1?, op1classValue?/classValue?,
```

```
                op1output!/output!] \semi \\
        BDD4Op2Count [op2input1?/input1?, op2output!/output!] \\
\end{zed}
```

การทดสอบโดยการสร้างเงื่อนไขก่อน ของ BDD4Op1Select ด้วยคำสั่ง

try $\forall$ *input1? :* $\mathbb{F}$ *Van; classValue? : CLASSNAME • pre BDD3Op2Select;*

เมื่อใช้คำสั่ง prove by reduce ได้ผลดังนี้

```
input1? \in \finset Van \land classValue? \in CLASSNAME \\
\implies
(\exists output!: \power \lblot class: CLASSNAME, model: MODEL, reg: REG \rblot
@       output! \in \finset BDD4R1 \land \lnot input1? = \{\} \land \\
        (\forall out: output!; in1: input1? | in1.class = classValue? @ out = in1))
```

การทดสอบโดยการสร้างเงื่อนไขก่อน ของ BDD4Op2Count ด้วยคำสั่ง

try $\forall$ *input1? :* $\mathbb{F}$ *BDD4R1 • pre BDD4Op2Count;*

เมื่อใช้คำสั่ง prove by reduce ได้ผลดังนี้

```
input1? \in \finset BDD4R1 \\
\implies
\# input1? \geq 0 \land \lnot input1? = \{\}
```

การทดสอบ BDD4 ด้วยคำสั่ง

```
try BDD4 [op1input1? := VanData, op1classValue? := Minivan];
```

เมื่อใช้คำสั่ง prove by reduce ได้ผลดังนี้

```
Minivan \in CLASSNAME \\
\land VanData \in \power \lblot class: CLASSNAME, model: MODEL, reg: REG \rblot \\
\land op1output! \in \power \lblot class: CLASSNAME, model: MODEL, reg: REG \rblot \\
\land op2input1? \in \power \lblot class: CLASSNAME, model: MODEL, reg: REG \rblot \\
\land op2input1? \in \finset BDD4R1 \land op2output! = \# op2input1? \\
\land \# op2input1? \in \num \land VanData \in \finset Van \\
\land op1output! \in \finset BDD4R1 \land \lnot VanData = \{\} \land \# op2input1? \geq 0 \\
\land \lnot op2input1? = \{\} \land
(out \in op1output! \land in1 \in VanData \land in1.class = Minivan \\
\implies
out = in1)
```

กรณีทดสอบที่ 5 มีรายละเอียดแฟ้มข้อความของแผนภาพเชิงลำดับชั้นของภาษาเอสคิวแอล ดังนี้

```
%BDDBEGIN
1:Insert:Customer:-:-:Success:NotFound
%BDDEND
```

ข้อกำหนดเซตของแผนภาพการเชิงลำดับชั้นของภาษาเอสคิวแอล มีดังนี้

```
\begin{schema} {BDD5Op1Init}
        \Delta CustomerExt \\
\where
        customerSet' = \emptyset \\
        customerKey' = \emptyset \\
        customerAtt' = \emptyset \\
\end{schema}

\begin{schema} {BDD5Op2Insert}
        \Delta CustomerExt \\
        customerKey? : CUSTID \\
        newValue? : Customer \\
\where
        customerKey? \notin \dom customerAtt \\
        customerSet' = customerSet \cup \{ newValue? \}
\end{schema}

\begin{zed}
        BDD5 \defs \\
        BDD5Op1Init  \semi \\
        BDD5Op2Insert [op2customerKey?/customerKey?, op2newValue?/newValue?] \\
\end{zed}
```

การทดสอบโดยการสร้างเงื่อนไขก่อน ของ BDD5Op1Init ด้วยคำสั่ง

try $\forall$ *CustomerExt • pre BDD5Op1Init;*

เมื่อใช้คำสั่ง prove by reduce ได้ผลลัพธ์เป็น True

การทดสอบโดยการสร้างเงื่อนไขก่อน ของ BDD5Op2Insert ด้วยคำสั่ง

try $\forall$ *CustomerExt; customerKey? : CUSTID; newValue? : • pre BDD5Op2Insert;*

เมื่อใช้คำสั่ง prove by reduce ได้ผลดังนี้

```
customerSet \in \finset Customer \land customerKey \in \finset CUSTID \\
\land customerAtt \in CUSTID \pfun Customer \\
\land customerKey? \in CUSTID \land newValue? \in Customer \land \\
(\forall customer1: Customer; customer2: Customer
|       customer1 \in customerSet \land customer2 \in customerSet \\
        \land \lnot customer1 = customer2
@ (customer1.custId \in CUSTID \land customer2.custId \in CUSTID \land
        \lnot customer1.custId = customer2.custId)) \land \\
        \land \dom customerAtt \in \power customerKey \\
\implies
(\exists customerAtt': \power (CUSTID \cross \lblot custAddr: CUSTADDR,
        custId: CUSTID, custName: CUSTNAME \rblot)
```

```
@       (\exists customerKey': \power CUSTID
        @       customerKey' \in \finset CUSTID \land \\
                customerAtt' \in CUSTID \pfun Customer \land \\
                (\forall customer1\_\_0: Customer; customer2\_\_0: Customer
                |       \lnot customer1\_\_0 = customer2\_\_0 \\
                        \land (customer1\_\_0 \in customerSet \\
                        \lor customer1\_\_0 = newValue?) \\
                        \land (customer2\_\_0 \in customerSet \\
                        \lor customer2\_\_0 = newValue?)
                @       (customer1\_\_0.custId \in CUSTID \land \\
                        customer2\_\_0.custId \in CUSTID \land \\
                        \lnot customer1\_\_0.custId = customer2\_\_0.custId))
        \land \dom customerAtt' \in \power customerKey' \\
        \land \lnot customerKey? \in \dom customerAtt))
```

การทดสอบ BDD5 ด้วยคำสั่ง

```
try BDD5 [op2customerKey? := CustomerKeyData, op2newValue? := CustomerData];
```

เมื่อใช้คำสั่ง prove by reduce ได้ผลดังนี้

```
customerAtt \in \power (CUSTID \cross \lblot custAddr: CUSTADDR, custId: CUSTID,
        custName: CUSTNAME \rblot) \\
        \land customerAtt' \in \power (CUSTID \cross \lblot custAddr: CUSTADDR,
                custId: CUSTID, custName: CUSTNAME \rblot) \\
        \land customerKey \in \power CUSTID \\
        \land customerKey' \in \power CUSTID \\
        \land customerSet \in \power \lblot custAddr: CUSTADDR, custId: CUSTID,
                custName: CUSTNAME \rblot \\
        \land CustomerData \in \lblot custAddr: CUSTADDR, custId: CUSTID,
                custName: CUSTNAME \rblot \\
        \land CustomerData \in Customer \\
        \land customerSet' = \{CustomerData\} \\
        \land CustomerKeyData \in CUSTID \\
        \land customerSet \in \finset Customer \\
        \land customerKey \in \finset CUSTID \\
        \land customerAtt \in CUSTID \pfun Customer \\
        \land customerKey' \in \finset CUSTID \\
```

```
        \land customerAtt' \in CUSTID \pfun Customer \\
        \land \dom customerAtt \in \power customerKey \\
        \land \dom customerAtt' \in \power customerKey' \\
        \land (   customer1 \in Customer \\
                  \land customer2 \in Customer \\
                  \land customer1 \in customerSet \\
                  \land customer2 \in customerSet \\
                  \land \lnot customer1 = customer2 \\
\implies
customer1.custId \in CUSTID \land customer2.custId \in CUSTID \land \\
        \lnot customer1.custId = customer2.custId)
```

ภาคผนวก ข

ข้อกำหนดเซตของระบบขายสินค้า

(Selling System)

แฟ้มข้อความพจนานุกรมข้อมูลของระบบขายสินค้า มีรูปแบบ คือ

Entity Name:Attribute Name:Data Type:Key:<FK Entity-Attribute>:<CS Constraint>

โดยมีรายละเอียดแฟ้มข้อความพจนานุกรมข้อมูลของระบบขายสินค้า ดังนี้

Customer:creditNo:CREDITNO:PK
Customer:custName:NAME:NK
Customer:custAddr:ADDR:NK
Customer:creditExpire:DATE:NK
Customer:custPoint:\num:NK:CS \geq 0
Customer:custPaid:\num:NK:CS \geq 0
Customer:custTel:TEL:NK
Customer:custTax:TAXID:NK
Supplier:supplierId:SUPPID:PK
Supplier:supplierName:SUPPNAME:NK
Supplier:supplierAddr:ADDR:NK
Supplier:supplierTel:TEL:NK
Supplier:supplierFax:TEL:NK
Cargo:goodsId:GOODSID:PK
Cargo:goodsName:GOODSNAME:NK
Cargo:supplierId:SUPPID:NK:FK Supplier-supplierId
Cargo:safetyStock:\num:NK:CS \geq 0
Cargo:goodsStatus:GOODSSTAUTS:NK
Cargo:goodsCmment:COMMENT:NK
Cargo:onHandGoods:\num:NK:CS \geq 0
Catalog:catalogId:CATID:PK
Catalog:goodsId:GOODSID:PK
Catalog:goodsPoint:\num:NK:CS \geq 0
Catalog:goodsPrice:\num:NK
Catalog:pointRef:\num:NK:CS \geq 0
Catalog:priceRef:\num:NK
Officer:officerId:OFFID:PK
Officer:officerName:NAME:NK
Officer:officerAddr:ADDR:NK
Officer:officerTel:TEL:NK
Selling:sellId:SELLID:PK
Selling:credit:CREDITNO:NK:FK Customer-creditNo
Selling:receiveName:NAME:NK
Selling:receiveAddr:ADDR:NK
Selling:orderDate:DATE:NK
Selling:officerId:OFFID:NK:FK Officer-officerId
Selling:togetherFlag:BOOLEAN:NK
Selling:catalogId:CATID:NK:FK Catalog-catalogId

```
GoodsInSelling:sellId:SELLID:PK:FK Selling-sellId
GoodsInSelling:goodsId:GOODSID:PK:FK Cargo-goodsId
GoodsInSelling:sellType:SELLTYPE:NK
GoodsInSelling:quantity:\nat:NK
GoodsInSelling:sellPoint:\num:NK:CS \geq 0
GoodsInSelling:sellMoney:\num:NK:CS \geq 0
GoodsInSelling:catalogId:CATID:NK:FK Catalog-catalogId
GoodsInSelling:sentFlag:BOOLEAN:NK
GoodsInSelling:sentDate:DATE:NK
BackOrder:sellId:SELLID:PK:FK Selling-sellId
BackOrder:goodsId:GOODSID:PK:FK Cargo-goodsId
BackOrder:sellType:SELLTYPE:NK
BackOrder:quantity:\nat:NK
BackOrder:sellPoint:\num:NK:CS \geq 0
BackOrder:sellMoney:\num:NK:CS \geq 0
BackOrder:catalogId:CATID:NK
Buying:orderId:ORDERID:PK
Buying:goodsId:GOODSID:PK:FK Cargo-goodsId
Buying:buyQuantity:\nat:NK
Buying:pricePerUnit:\num:NK:CS \geq 0
Buying:orderDate:DATE:NK
Buying:sendDate:DATE:NK
Buying:orderOfficer:OFFID:NK:FK Officer-officerId
Buying:receiverOfficer:OFFID:NK:FK Officer-officerId
Buying:orderComment:COMMENT:NK
Buying:receiveFlag:BOOLEAN:NK
```

แฟ้มข้อความพจนานุกรมความสัมพันธ์ของระบบขายสินค้าร มีรูปแบบ คือ

```
Relation Name:Entity 1:Entity 2:Relation Type
```

โดยมีรายละเอียดแฟ้มข้อความพจนานุกรมความสัมพันธ์ของระบบขายสินค้า ดังนี้

```
CusSel:Customer:Selling:1-m
OffSel:Officer:Selling:1-m
OffBuy:Officer:Buying:1-m
CatSel:Catalog:Selling:1-m
CatGood:Catalog:GoodsInSelling:1-m
CatBack:Catalog:BackOrder:1-m
SelGood:GoodsInSelling:Selling:m-1
SelBack:BackOrder:Selling:m-1
CarBack:BackOrder:Cargo:m-1
CatCar:Catalog:Cargo:m-1
SupCar:Cargo:Supplier:m-1
CarGood:Cargo:GoodsInSelling:1-m
CarBuy:Cargo:Buying:1-m
```

ข้อกำหนดเซตของระบบขายสินค้า มีดังนี้

```
\begin{zed}
        [CREDITNO, NAME, ADDR, DATE, TEL, TAXID, \\
        SUPPID, SUPPNAME, GOODSID, GOODSNAME, GOODSSTATUS, \\
        COMMENT, CATID, OFFID, SELLID, SELLTYPE, ORDERID ] \\
```

```
\end{zed}

\begin{zed}
        BOOLEAN ::= True | False \\
\end{zed}

\begin{schema} {Customer}
        creditNo : CREDITNO \\
        custName : NAME \\
        custAddr : ADDR \\
        creditExpire : DATE \\
        custPoint : \num \\
        custPaid : \num \\
        custTel : TEL \\
        custTax : TAXID \\
\where
        custPoint \geq 0 \\
        custPaid \geq 0 \\
\end{schema}

\begin{schema} {Supplier}
        supplierId : SUPPID \\
        supplierName : SUPPNAME \\
        supplierAddr : ADDR \\
        supplierTel : TEL \\
        supplierFax : TEL \\
\end{schema}

\begin{schema} {Cargo}
        goodsId : GOODSID \\
        goodsName : GOODSNAME \\
        supplierId : SUPPID \\
        safetyStock : \num \\
        goodsStatus : GOODSSTATUS \\
        goodsComment : COMMENT \\
        onHandGoods : \num \\
\where
        safetyStock \geq 0 \\
        onHandGoods \geq 0 \\
\end{schema}

\begin{schema} {Catalog}
        catalogId : CATID \\
        goodsId : GOODSID \\
        goodsPoint : \num \\
        goodsPrice : \num \\
        pointRef : \num \\
        priceRef : \num \\
\where
        goodsPoint \geq 0 \\
        pointRef \geq 0 \\
\end{schema}

\begin{schema} {Officer}
        officerId : OFFID \\
        officerName : NAME \\
        officerAddr : ADDR \\
```

```
        officerTel : TEL \\
\end{schema}

\begin{schema} {Selling}
        sellId : SELLID \\
        credit : CREDITNO \\
        receiveName : NAME \\
        receiveAddr : ADDR \\
        receiveTel : TEL \\
        orderDate : DATE \\
        officerId : OFFID \\
        togetherFlag : BOOLEAN \\
        catalogId : CATID \\
\end{schema}

\begin{schema} {GoodsInSelling}
        sellId : SELLID \\
        goodsId : GOODSID \\
        sellType : SELLTYPE \\
        quantity : \nat \\
        sellPoint : \num \\
        sellMoney : \num \\
        catalogId : CATID \\
        sentFlag : BOOLEAN \\
        sentDate : DATE \\
\where
        sellPoint \geq 0 \\
        sellMoney \geq 0 \\
\end{schema}

\begin{schema} {BackOrder}
        sellId : SELLID \\
        goodsId : GOODSID \\
        sellType : SELLTYPE \\
        quantity : \nat \\
        sellPoint : \num \\
        sellMoney : \num \\
        catalogId : CATID \\
\where
        sellPoint \geq 0 \\
        sellMoney \geq 1 \\
\end{schema}

\begin{schema} {Buying}
        orderId : ORDERID \\
        goodsId : GOODSID \\
        buyQuantity : \nat \\
        pricePerUnit : \num \\
        orderDate : DATE \\
        sendDate : DATE \\
        orderOfficer : OFFID \\
        receiveOfficer : OFFID \\
        orderComment : COMMENT \\
        receiveFlag : BOOLEAN \\
\where
        pricePerUnit \geq 0 \\
\end{schema}
```

```
\begin{schema} {CustomerExt}
        customerSet : \finset Customer \\
        customerKey : \finset  CREDITNO \\
        customerAtt : CREDITNO \pfun Customer \\
\where
        \forall customer1, customer2 : Customer \\
        | customer1 \in customerSet \land customer2 \in customerSet \land \\
          customer1 \neq customer2 @  customer1.creditNo \neq customer2.creditNo \\
        \dom customerAtt \subseteq customerKey \\
\end{schema}

\begin{schema} {SupplierExt}
        supplierSet : \finset Supplier \\
        supplierKey : \finset SUPPID \\
        supplierAtt : SUPPID \pfun Supplier
\where
        \forall supplier1, supplier2 : Supplier \\
        | supplier1 \in supplierSet \land supplier2 \in supplierSet \land \\
          supplier1 \neq supplier2 @ supplier1.supplierId \neq supplier2.supplierId \\
        \dom supplierAtt \subseteq supplierKey \\
\end{schema}

\begin{schema} {CargoExt}
        cargoSet : \finset Cargo \\
        cargoKey : \finset GOODSID \\
        cargoAtt : GOODSID \pfun Cargo \\
\where
        \forall cargo1, cargo2 : Cargo \\
        | cargo1 \in cargoSet \land cargo2 \in cargoSet \land cargo1 \neq cargo2 \\
        @ cargo1.goodsId \neq cargo2.goodsId \\
        \forall cargoVar2 : Cargo \\
        @ (\exists supplierVar1 : Supplier @ cargoVar2.supplierId = supplierVar1.supplierId) \\
        \dom cargoAtt \subseteq cargoKey \\
\end{schema}

\begin{schema} {CatalogExt}
        catalogSet : \finset Catalog \\
        catalogKey : \finset (CATID \cross GOODSID) \\
        catalogAtt : (CATID \cross GOODSID) \pfun Catalog \\
\where
        \forall catalog1, catalog2 : Catalog \\
        | catalog1 \in catalogSet \land catalog2 \in catalogSet \land catalog1 \neq catalog2 \\
        @ catalog1.catalogId \neq catalog2.catalogId \lor \\
           catalog1.goodsId \neq catalog2.goodsId \\
        \forall catalogVar2 : Catalog \\
        @ (\exists cargoVar1 : Cargo @ catalogVar2.goodsId = cargoVar1.goodsId) \\
        \dom catalogAtt \subseteq catalogKey \\
\end{schema}

\begin{schema} {OfficerExt}
        officerSet : \finset Officer \\
        officerKey : \finset OFFID \\
        officerAtt : OFFID \pfun Officer \\
```

```
\where
        \forall officer1, officer2 : Officer \\
        | officer1 \in officerSet \land officer2 \in officerSet \land officer1 \neq officer2 \\
        @ officer1.officerId \neq officer2.officerId \\
        \dom officerAtt \subseteq officerKey \\
\end{schema}

\begin{schema} {SellingExt}
        sellingSet : \finset Selling \\
        sellingKey : \finset SELLID \\
        sellingAtt : SELLID \pfun Selling \\
\where
        \forall selling1,selling2 : Selling \\
        | selling1 \in sellingSet \land selling2 \in sellingSet \land selling1 \neq selling2  \\
        @ selling1.sellId \neq selling2.sellId \\
        \forall sellingVar2 : Selling \\
        @ (\exists customerVar1 : Customer @ sellingVar2.credit = customerVar1.creditNo) \\
        \forall sellingVar2 : Selling \\
        @ (\exists officerVar1 : Officer @ sellingVar2.officerId = officerVar1.officerId) \\
        \forall sellingVar2 : Selling \\
        @ (\exists catalogVar1 : Catalog @ sellingVar2.catalogId = catalogVar1.catalogId) \\
        \dom sellingAtt \subseteq sellingKey \\
\end{schema}

\begin{schema} {GoodsInSellingExt}
        goodsInSellingSet : \finset GoodsInSelling \\
        goodsInSellingKey : \finset (SELLID \cross GOODSID) \\
        goodsInSellingAtt : (SELLID \cross GOODSID) \pfun GoodsInSelling \\
\where
        \forall goodsInSelling1, goodsInSelling2 : GoodsInSelling \\
        | goodsInSelling1 \in goodsInSellingSet \land \\
          goodsInSelling2 \in goodsInSellingSet \land \\
          goodsInSelling1 \neq goodsInSelling2 \\
        @ goodsInSelling1.sellId \neq goodsInSelling2.sellId \lor \\
            goodsInSelling1.goodsId \neq goodsInSelling2.goodsId \\
        \forall goodsInSellingVar2 : GoodsInSelling \\
        @ (\exists sellingVar1 : Selling @ goodsInSellingVar2.sellId = sellingVar1.sellId) \\
        \forall goodsInSellingVar2 : GoodsInSelling \\
        @ (\exists cargoVar1 : Cargo @ goodsInSellingVar2.goodsId = cargoVar1.goodsId) \\
        \forall goodsInSellingVar2 : GoodsInSelling \\
        @ (\exists catalogVar1 : Catalog \\
            @ goodsInSellingVar2.catalogId = catalogVar1.catalogId) \\
        \dom goodsInSellingAtt \subseteq goodsInSellingKey \\
\end{schema}

\begin{schema} {BackOrderExt}
        backOrderSet : \finset BackOrder \\
        backOrderKey : \finset (SELLID \cross GOODSID) \\
        backOrderAtt : (SELLID \cross GOODSID) \pfun BackOrder \\
\where
        \forall backOrder1, backOrder2 : BackOrder \\
        | backOrder1 \in backOrderSet \land backOrder2 \in backOrderSet \land \\
```

```
        backOrder1 \neq backOrder2 \\
        @ backOrder1.goodsId \neq backOrder2.goodsId \lor \\
            backOrder1.sellId \neq backOrder2.sellId \\
        \forall backOrderVar2 : BackOrder \\
        @ (\exists sellingVar1 : Selling @ backOrderVar2.sellId = sellingVar1.sellId) \\
        \forall backOrderVar2 : BackOrder \\
        @ (\exists cargoVar1 : Cargo @ backOrderVar2.goodsId = cargoVar1.goodsId) \\
        \forall backOrderVar2 : BackOrder \\
        @ (\exists catalogVar1 : Catalog
            @ backOrderVar2.catalogId = catalogVar1.catalogId) \\
        \dom backOrderAtt \subseteq backOrderKey \\
\end{schema}

\begin{schema} {BuyingExt}
        buyingSet : \finset Buying \\
        buyingKey : \finset (ORDERID \cross GOODSID) \\
        buyingAtt : (ORDERID \cross GOODSID) \pfun Buying \\
\where
        \forall buying1, buying2 : Buying \\
        | buying1 \in buyingSet \land buying2 \in buyingSet \land buying1 \neq buying2 \\
        @ buying1.goodsId \neq buying2.goodsId \lor buying1.orderId \neq buying2.orderId \\
        \forall buyingVar2 : Buying \\
        @ (\exists officerVar1 : Officer @ buyingVar2.orderOfficer = officerVar1.officerId) \\
        \forall buyingVar2 : Buying @ \\
          (\exists officerVar1 : Officer @ buyingVar2.receiveOfficer = officerVar1.officerId) \\
        \forall buyingVar2 : Buying \\
        @ ( \exists cargoVar1 : Cargo @ buyingVar2.goodsId = cargoVar1.goodsId) \\
        \dom buyingAtt \subseteq buyingKey \\
\end{schema}

\begin{schema} {RelationshipCusSel}
        cusSel : \finset (Customer \cross Selling)
\where
        \forall cusSel1,cusSel2 : cusSel \\
        @ second~cusSel1 = second~cusSel2 \implies first~cusSel1 = first~cusSel2 \\
\end{schema}

\begin{schema} {RelationshipOffSel}
        offSel : \finset (Officer \cross Selling)
\where
        \forall offSel1,offSel2 : offSel \\
        @ second~offSel1 = second~offSel2 \implies first~offSel1 = first~offSel2 \\
\end{schema}

\begin{schema} {RelationshipOffBuy}
        offBuy : \finset (Officer \cross Buying)
\where
        \forall offBuy1,offBuy2 : offBuy \\
        @ second~offBuy1 = second~offBuy2 \implies first~offBuy1 = first~offBuy2 \\
\end{schema}

\begin{schema} {RelationshipCatSel}
        catSel : \finset (Catalog \cross Selling)
```

```
\where
        \forall catSel1,catSel2 : catSel \\
        @ second~catSel1 = second~catSel2 \implies first~catSel1 = first~catSel2 \\
\end{schema}

\begin{schema} {RelationshipCatGood}
        catGood : \finset (Catalog \cross GoodsInSelling)
\where
        \forall catGood1,catGood2 : catGood \\
        @ second~catGood1 = second~catGood2 \implies first~catGood1 = first~catGood2
\end{schema}

\begin{schema} {RelationshipCatBack}
        catBack : \finset (Catalog \cross BackOrder)
\where
        \forall catBack1,catBack2 : catBack \\
        @ second~catBack1 = second~catBack2 \implies first~catBack1 = first~catBack2
\end{schema}

\begin{schema} {RelationshipSelGood}
        selGood : \finset (GoodsInSelling \cross Selling)
\where
        \forall selGood1,selGood2 : selGood \\
        @ first~selGood1 = first~selGood2 \implies second~selGood1 = second~selGood2
\end{schema}

\begin{schema} {RelationshipSelBack}
        selBack : \finset (BackOrder \cross Selling)
\where
        \forall selBack1,selBack2 : selBack \\
        @ first~selBack1 = first~selBack2 \implies second~selBack1 = second~selBack2
\end{schema}

\begin{schema} {RelationshipCarBack}
        carBack : \finset (BackOrder \cross Cargo)
\where
        \forall carBack1,carBack2 : carBack \\
        @ first~carBack1 = first~carBack2 \implies second~carBack1 = second~carBack2
\end{schema}

\begin{schema} {RelationshipCarCat}
        carCat : \finset (Catalog \cross Cargo)
\where
        \forall carCat1,carCat2 : carCat \\
        @ first~carCat1 = first~carCat2 \implies second~carCat1 = second~carCat2
\end{schema}

\begin{schema} {RelationshipSupCar}
        supCar : \finset (Cargo \cross Supplier)
\where
        \forall supCar1,supCar2 : supCar \\
        @ first~supCar1 = first~supCar2 \implies second~supCar1 = second~supCar2
\end{schema}

\begin{schema} {RelationshipCarGood}
```

```
        carGood : \finset (Cargo \cross GoodsInSelling)
\where
        \forall carGood1,carGood2 : carGood \\
        @ second~carGood1 = second~carGood2 \implies first~carGood1 = first~carGood2
\end{schema}

\begin{schema} {RelationshipCarBuy}
        carBuy : \finset (Cargo \cross Buying)
\where
        \forall carBuy1,carBuy2 : carBuy \\
        @ second~carBuy1 = second~carBuy2 \implies first~carBuy1 = first~carBuy2
\end{schema}

\begin{zed}
        Response ::= Success | NotFound \\
\end{zed}

\begin{schema} {Success}
        r! : Response \\
\where
        r! = Success \\
\end{schema}

\begin{schema} {NotFound}
        r! : Response \\
\where
        r! = NotFound \\
\end{schema}
```

กรณีทดสอบที่ 1 มีรายละเอียดแฟ้มข้อความของแผนภาพเชิงลำดับชั้นของภาษาเอสคิวแอล ดังนี้

```
%BDDBEGIN
1:Project:Catalog:-:goodsPoint:2:NotFound
2:All:Customer:1:Customer,custPoint < 1.goodsPoint:5:NotFound
3:Project:Catalog:-:pointRef:4:NotFound
4:Any:Customer:3:Customer,custPoint >= 3.priceRef:5:NotFound
5:Intersect:2:4:-:Success:NotFound
%BDDEND
```

ข้อกำหนดเซตของแผนภาพเชิงลำดับชั้นของภาษาเอสคิวแอล มีดังนี้

```
\begin{schema} {BDD1R1}
        goodsPoint : \num \\
\end{schema}

\begin{schema} {BDD1Op1Project}
        input1? : \finset Catalog \\
        output! : \finset BDD1R1 \\
\where
        input1? \neq \emptyset \\
        \forall out : output!; in1 : input1? @ out.goodsPoint = in1.goodsPoint \\
\end{schema}

\begin{schema} {BDD1R2}
        creditNo : CREDITNO \\
```

```
        custName : NAME \\
        custAddr : ADDR \\
        creditExpire : DATE \\
        custPoint : \num \\
        custPaid : \num \\
        custTel : TEL \\
        custTax : TAXID \\
\end{schema}

\begin{schema} {BDD1Op2All}
        input1? : \finset Customer \\
        input2? : \finset BDD1R1 \\
        output! : \finset BDD1R2 \\
\where
        input1? \neq \emptyset \\
        input2? \neq \emptyset \\
        \forall out : output!; in1: input1? \\
        | ( \forall i1 : input1?; i2 : input2? | i1.custPoint < i2.goodsPoint @ true ) \\
        @ out = in1 \\
\end{schema}

\begin{schema} {BDD1R3}
        pointRef : \num \\
\end{schema}

\begin{schema} {BDD1Op3Project}
        input1? : \finset Catalog \\
        output! : \finset BDD1R3 \\
\where
        input1? \neq \emptyset \\
        \forall out : output!; in1 : input1? @ out.pointRef = in1.pointRef \\
\end{schema}

\begin{schema} {BDD1R4}
        creditNo : CREDITNO \\
        custName : NAME \\
        custAddr : ADDR \\
        creditExpire : DATE \\
        custPoint : \num \\
        custPaid : \num \\
        custTel : TEL \\
        custTax : TAXID \\
\end{schema}

\begin{schema} {BDD1Op4Any}
        input1? : \finset Customer \\
        input2? : \finset BDD1R3 \\
        output! : \finset BDD1R4 \\
\where
        input1? \neq \emptyset \\
        input2? \neq \emptyset \\
        \forall out : output!; in1: input1? \\
        | (\exists i1 : input1?; i2 : input2? | i1.custPoint < i2.pointRef @ true ) \\
        @ out = in1 \\
\end{schema}

\begin{schema} {BDD1R5}
        creditNo : CREDITNO \\
```

```
        custName : NAME \\
        custAddr : ADDR \\
        creditExpire : DATE \\
        custPoint : \num \\
        custPaid : \num \\
        custTel : TEL \\
        custTax : TAXID \\
\end{schema}

\begin{schema} {BDD1Op5Intersect}
        \Xi BDD1R2 \\
        \Xi BDD1R4 \\
        input1? : \finset BDD1R2 \\
        input2? : \finset BDD1R4 \\
        output! : \finset BDD1R5 \\
\where
        \theta BDD1R2 = \theta BDD1R4 \\
        \forall out : output!; in1: input1?; in2 : input2? @ out = \{ in1 \} \cap \{ in2 \}
\end{schema}

\begin{zed}
        BDD1 \defs \\
        BDD1Op1Project [op1input1?/input1?, op1output!/output!] \semi \\
        BDD1Op2All [op2input?/input1?, op1output!/input2?, op2output!/output!] \semi \\
        BDD1Op3Project [op3input1?/input1?, op3output!/output!] \semi \\
        BDD1Op4Any [op4input?/input1?, op3output?/input2?, op4output!/output!] \semi \\
        BDD1Op5Intersect [op2output?/input1?, op4output?/input2?, op5output!/output!] \\
\end{zed}
```

การทดสอบโดยการสร้างเงื่อนไขก่อน ของ BDD1Op1Project ด้วยคำสั่ง

try $\forall$ *input1? :* $\mathbb{F}$ *Catalog • pre BDD1Op1Project;*

เมื่อใช้คำสั่ง prove by reduce ได้ผลดังนี้

```
input1? \in \finset Catalog \\
\implies
(\exists output!: \power \lblot goodsPoint: \num \rblot
@       output! \in \finset BDD1R1 \land \lnot input1? = \{\} \land \\
        (\forall out: output!; in1: input1? @ out.goodsPoint = in1.goodsPoint))
```

การทดสอบโดยการสร้างเงื่อนไขก่อน ของ BDD1Op2All ด้วยคำสั่ง

try $\forall$ *input1? :* $\mathbb{F}$ *Customer; input2? :* $\mathbb{F}$ *BDD1R1 • pre BDD1Op2All;*

เมื่อใช้คำสั่ง prove by reduce ได้ผลดังนี้

```
input1? \in \finset Customer \land input2? \in \finset BDD1R1 \\
\implies
(\exists output!: \power \lblot creditExpire: DATE, creditNo: CREDITNO, custAddr: ADDR,
        custName: NAME, custPaid: \num, custPoint: \num, custTax: TAXID, custTel: TEL
        \rblot
@       output! \in \finset BDD1R2 \land \lnot input1? = \{\} \land \lnot input2? = \{\} \land \\
```

(\forall out: output!; in1: input1? @ out = in1))

การทดสอบโดยการสร้างเงื่อนไขก่อน ของ BDD1Op3Project ด้วยคำสั่ง

try ∀ input1? : 𝔽 Catalog • pre BDD1Op3Project;

เมื่อใช้คำสั่ง prove by reduce ได้ผลดังนี้

input1? \in \finset Catalog \\

\implies

(\exists output!: \power \lblot pointRef: \num \rblot

@ output! \in \finset BDD1R3 \land \lnot input1? = \{\} \land \\

(\forall out: output!; in1: input1? @ out.pointRef = in1.pointRef))

การทดสอบโดยการสร้างเงื่อนไขก่อน ของ BDD1Op4Any ด้วยคำสั่ง

try ∀ input1? : 𝔽 Customer; input2? : 𝔽 BDD1R3 • pre BDD1Op4Any;

เมื่อใช้คำสั่ง prove by reduce ได้ผลดังนี้

input1? \in \finset Customer \land input2? \in \finset BDD1R3 \\

\implies

(\exists output!: \power \lblot creditExpire: DATE, creditNo: CREDITNO, custAddr: ADDR,

custName: NAME, custPaid: \num, custPoint: \num, custTax: TAXID, custTel: TEL

\rblot

@ output! \in \finset BDD1R4 \land \lnot input1? = \{\} \land \lnot input2? = \{\} \land \\

(\forall out: output!; in1: input1? | \exists i1: input1?

@ \exists i2: input2? @ i1.custPoint < i2.pointRef @ out = in1))

การทดสอบโดยการสร้างเงื่อนไขก่อน ของ BDD1Op5Intersect ด้วยคำสั่ง

try ∀ BDD1R2; BDD1R4; input1? : 𝔽 BDD1R2; input2? : 𝔽 BDD1R4;

• pre BDD2Op5Intersect;

เมื่อใช้คำสั่ง prove by reduce ได้ผลดังนี้

creditNo \in CREDITNO \land custName \in NAME \\

\land custAddr \in ADDR \land creditExpire \in DATE \\

\land custPoint \in \num \land custPaid \in \num \\

\land custTel \in TEL \land custTax \in TAXID \\

\land input1? \in \finset BDD1R2 \land input2? \in \finset BDD1R4 \\

\implies

(\exists output!: \power \lblot creditExpire: DATE, creditNo: CREDITNO, custAddr: ADDR,

custName: NAME, custPaid: \num, custPoint: \num, custTax: TAXID, custTel: TEL

\rblot

@ output! \in \finset BDD1R5 \land \\

(\forall out: \finset output!; in1: input1?; in2: input2? \\

@ in1.creditNo = in2.creditNo))

การทดสอบ BDD1 ด้วยคำสั่ง

try BDD1 [op1input1? := CatalogData, op2input1? := CustomerData,

op3input1? := CatalogData, op4input1? := CustomerData];

เมื่อใช้คำสั่ง prove by reduce ได้ผลดังนี้

```
creditExpire' \in DATE \\
\land creditExpire = creditExpire' \\
\land creditNo' \in CREDITNO \\
\land creditNo = creditNo' \\
\land custAddr' \in ADDR \\
\land custAddr = custAddr' \\
\land custName' \in NAME \\
\land custName = custName' \\
\land custPaid' \in \num \\
\land custPaid = custPaid' \\
\land custPoint' \in \num \\
\land custPoint = custPoint' \\
\land custTax' \in TAXID \\
\land custTax = custTax' \\
\land custTel' \in TEL \\
\land custTel = custTel' \\
\land
 CatalogData
 \in \power \lblot catalogId: CATID, goodsId: GOODSID, goodsPoint: \num,
            goodsPrice: \num, pointRef: \num, priceRef: \num \rblot \\
\land op1output! \in \power \lblot goodsPoint: \num \rblot \\
\land CustomerData
    \in \power \lblot creditExpire: DATE, creditNo: CREDITNO, custAddr: ADDR,
                custName: NAME, custPaid: \num, custPoint: \num,
                custTax: TAXID, custTel: TEL \rblot \\
\land op2input2? \in \power \lblot goodsPoint: \num \rblot \\
\land op2output!
    \in \power \lblot creditExpire: DATE, creditNo: CREDITNO, custAddr: ADDR,
                custName: NAME, custPaid: \num, custPoint: \num,
                custTax: TAXID, custTel: TEL \rblot \\
```

```
\land op3output! \in \power \lblot pointRef: \num \rblot \\
\land op4input2? \in \power \lblot pointRef: \num \rblot \\
\land op4output!
    \in \power \lblot creditExpire: DATE, creditNo: CREDITNO, custAddr: ADDR,
                custName: NAME, custPaid: \num, custPoint: \num,
                custTax: TAXID, custTel: TEL \rblot \\
\land op5input1?
    \in \power \lblot creditExpire: DATE, creditNo: CREDITNO, custAddr: ADDR,
                custName: NAME, custPaid: \num, custPoint: \num,
                custTax: TAXID, custTel: TEL \rblot \\
\land op5input2?
    \in \power \lblot creditExpire: DATE, creditNo: CREDITNO, custAddr: ADDR,
                custName: NAME, custPaid: \num, custPoint: \num,
                custTax: TAXID, custTel: TEL \rblot \\
\land op5output!
    \in \power \lblot creditExpire: DATE, creditNo: CREDITNO, custAddr: ADDR,
                custName: NAME, custPaid: \num, custPoint: \num,
                custTax: TAXID, custTel: TEL \rblot \\
\land CatalogData \in \finset Catalog \\
\land op1output! \in \finset BDD1R1 \\
\land CustomerData \in \finset Customer \\
\land op2input2? \in \finset BDD1R1 \\
\land op2output! \in \finset BDD1R2 \\
\land op3output! \in \finset BDD1R3 \\
\land op4input2? \in \finset BDD1R3 \\
\land op4output! \in \finset BDD1R4 \\
\land op5input1? \in \finset BDD1R2 \\
\land op5input2? \in \finset BDD1R4 \\
\land op5output! \in \finset BDD1R5 \\
\land \lnot CatalogData = \{\} \\
\land \lnot CustomerData = \{\} \\
\land \lnot op2input2? = \{\} \\
\land \lnot op4input2? = \{\} \\
\land (             out \in op1output! \\
```

```
            \land in1 \in CatalogData \\
        \implies out.goodsPoint = in1.goodsPoint) \\
\land (         out\_\_0 \in op2output! \\
            \land in1\_\_0 \in CustomerData \\
        \implies out\_\_0 = in1\_\_0) \\
\land (         out\_\_1 \in op3output! \\
            \land in1\_\_1 \in CatalogData \\
        \implies out\_\_1.pointRef = in1\_\_1.pointRef) \\
\land (         out\_\_2 \in op4output! \\
            \land in1\_\_2 \in CustomerData \\
            \land i1 \in CustomerData \\
            \land i2 \in op4input2? \\
            \land i1.custPoint < i2.pointRef \\
        \implies out\_\_2 = in1\_\_2) \\
\land (         out\_\_3 \in \finset op5output! \\
            \land in1\_\_3 \in op5input1? \\
            \land in2 \in op5input2? \\
            \land in1\_\_3.creditNo \in CREDITNO \\
            \land in2.creditNo \in CREDITNO \\
            \land \lnot in1\_\_3.creditNo = in2.creditNo \\
        \implies out\_\_3 = \{\})
```

กรณีทดสอบที่ 2 มีรายละเอียดแฟ้มข้อความของแผนภาพเชิงลำดับชั้นของภาษาเอสคิวแอล ดังนี้

```
%BDDBEGIN
1:Project:Customer:-:custPoint:2:NotFound
2:Min:1:-:-:Success:NotFound
%BDDEND
```

ข้อกำหนดเซตของแผนภาพเชิงลำดับชั้นของภาษาเอสคิวแอล มีดังนี้

```
\begin{schema} {BDD2R1}
        custPoint : \num \\
\end{schema}

\begin{schema} {BDD2Op1Project}
        input1? : \finset Customer \\
        output! : \finset BDD2R1 \\
\where
        input1? \neq \emptyset \\
        \forall out : output!; in1 : input1? @ out.custPoint = in1.custPoint \\
\end{schema}
```

```
\begin{schema} {BDD2Op2Min}
        \Xi BDD2R1 \\
        output! : \num
\where
        \forall in1 : \finset BDD2R1 \\
        | in1 \neq \emptyset \land output! = min \{ custPoint \} @ true
\end{schema}

\begin{zed}
        BDD2 \defs \\
        BDD2Op1Project [op1input1?/input1?, op1output!/output!] \semi \\
        BDD2Op2Min [op2output!/output!] \\
\end{zed}
```

การทดสอบโดยการสร้างเงื่อนไขก่อน ของ BDD2Op1Project ด้วยคำสั่ง

try ∀ input1? : 𝔽 Customer • pre BDD2Op1Project;

เมื่อใช้คำสั่ง prove by reduce ได้ผลดังนี้

```
input1? \in \finset Customer \\
\implies
(\exists output!: \power \lblot custPoint: \num \rblot
@       output! \in \finset BDD2R1 \land \lnot input1? = \{\} \land \\
        (\forall out: output!; in1: input1? @ out.custPoint = in1.custPoint))
```

การทดสอบโดยการสร้างเงื่อนไขก่อน ของ BDD2Op2Min ด้วยคำสั่ง

try ∀ BDD2R1 • pre BDD2Op2Min;

เมื่อใช้คำสั่ง prove by reduce ได้ผลดังนี้

```
custPoint \in \num \\
\implies
(\exists output!: \num @ (\forall in1: \finset BDD2R1 | \lnot in1 = \{\} @ output! = custPoint))
```

การทดสอบ BDD2 ด้วยคำสั่ง

try BDD2 [op1input1? := CustomerData];

เมื่อใช้คำสั่ง prove by reduce ได้ผลดังนี้

```
custPoint' \in \num \\
\land custPoint = custPoint' \\
\land CustomerData
    \in \power \lblot creditExpire: DATE, creditNo: CREDITNO, custAddr: ADDR,
                custName: NAME, custPaid: \num, custPoint: \num,
                custTax: TAXID, custTel: TEL \rblot \\
\land op1output! \in \power \lblot custPoint: \num \rblot \\
\land op2output! \in \num \\
```

```
\land CustomerData \in \finset Customer \\
\land op1output! \in \finset BDD2R1 \\
\land \lnot CustomerData = \{\} \\
\land (          out \in op1output! \\
          \land in1 \in CustomerData \\
     \implies out.custPoint = in1.custPoint) \\
\land (          in1\_\_0 \in \finset BDD2R1 \\
          \land \lnot in1\_\_0 = \{\} \\
     \implies op2output! = custPoint)
```

กรณีทดสอบที่ 3 มีรายละเอียดแฟ้มข้อความของแผนภาพเชิงลำดับชั้นของภาษาเอสคิวแอล ดังนี้

```
%BDDBEGIN
1:Sum:BackOrder:-:quantity:Success:NotFound
%BDDEND
```

ข้อกำหนดเซตของแผนภาพเชิงลำดับชั้นของภาษาเอสคิวแอล มีดังนี้

```
\begin{zed}
        quantity == \num
\end{zed}

\begin{axdef}
        BDD3s : \num \fun quantity
\end{axdef}

\begin{axdef}
        BDD3Sum : \num \fun quantity
\where
        \forall n : \nat \\
        @  n > \# BDD3s \implies BDD3Sum n = 0 \land \\
            n \leq \# BDD3s \implies BDD3Sum n = (BDD3Sum n + 1) + BDD3s (n)
\end{axdef}
```

BDD3Sum ได้มีการประกาศเป็นข้อเท็จจริง (Axiomatic Definition) จึงไม่ต้องทำการทดสอบโดยการสร้างเงื่อนไขก่อน เนื่องจาก Z/Eves จะทำการตรวจสอบความถูกต้องของโดเมนให้โดยอัตโนมัติ ได้ผลดังนี้

```
\Local BDD1Sum \in \num \fun quantity \land n \in \nat \\
\implies
BDD1s \in \dom \# \land (       n > \# BDD1s \\
          \implies \Local BDD1Sum n = 0 \\
          \implies BDD1s \in \dom \#)
```

กรณีทดสอบที่ 4 มีรายละเอียดแฟ้มข้อความของแผนภาพเชิงลำดับชั้นของภาษาเอสคิวแอล ดังนี้

```
%BDDBEGIN
1:Update:Buying:-:-:Success:NotFound
%BDDEND
```

ข้อกำหนดเซตของแผนภาพเชิงลำดับชั้นของภาษาเอสคิวแอล มีดังนี้

```
\begin{schema} {BDD4Op1Update}
        \Delta BuyingExt \\
        buyingKey? : ORDERID \cross GOODSID \\
        newValue? : Buying \\
\where
        buyingKey? \in \dom buyingAtt \\
        buyingAtt' = buyingAtt \oplus \{ buyingKey? \mapsto newValue? \}
\end{schema}

\begin{zed}
        BDD4 \defs \\
        BDD4Op1Update [op1buyingKey?/buyingKey?, op1newValue?/newValue?] \\
\end{zed}
```

การทดสอบโดยการสร้างเงื่อนไขก่อน ของ BDD1Op1Join ด้วยคำสั่ง

try $\forall$ *BuyingExt; buyingKey? : ORDER* $\times$ *GOODSID; newValue? : Buying*

- *pre BDD4Op1Update;*

เมื่อใช้คำสั่ง prove by reduce ได้ผลดังนี้

```
buyingSet \in \finset Buying \land
buyingKey \in \finset (ORDERID \cross GOODSID) \\
\land buyingAtt \in ORDERID \cross GOODSID \pfun Buying \\
\land buyingKey? \in ORDERID \cross GOODSID \\
\land newValue? \in Buying \land
(\forall buying1: Buying; buying2: Buying
|       buying1 \in buyingSet \land buying2 \in buyingSet \land \\
        \lnot buying1 = buying2 \land \lnot (buying1.goodsId \in GOODSID \land \\
        buying2.goodsId \in GOODSID \land \lnot buying1.goodsId = buying2.goodsId) \\
@       (buying1.orderId \in ORDERID \land \\
        buying2.orderId \in ORDERID \land \\
        \lnot buying1.orderId = buying2.orderId)) \land \\
(\forall buyingVar2: Buying @
        (\exists officerVar1: @ buyingVar2.orderOfficer = officerVar1.officerId)) \land \\
(\forall buyingVar2\_\_0: Buying @
```

```
        (\exists officerVar1\_\_0: Officer @
                buyingVar2\_\_0.receiveOfficer = officerVar1\_\_0.officerId)) \land
(\forall buyingVar2\_\_1: Buying @
        (\exists cargoVar1: Cargo @ buyingVar2\_\_1.goodsId = cargoVar1.goodsId)) \\
\land \dom buyingAtt \in \power buyingKey \\
\implies
(\exists buyingKey': \power (ORDERID \cross GOODSID)
@       (\exists buyingSet': \power \lblot buyQuantity: \num, goodsId: GOODSID,
                orderComment: COMMENT, orderDate: DATE, orderId: ORDERID,
                orderOfficer: OFFID, pricePerUnit: \num, receiveFlag: BOOLEAN,
                receiveOfficer: OFFID, sendDate: DATE \rblot
        @       buyingSet' \in \finset Buying \land \\
                buyingKey' \in \finset (ORDERID \cross GOODSID) \land \\
                buyingKey? \in buyingKey' \land \\
                buyingKey? \in \dom buyingAtt \land \\
                (\forall buying1\_\_0: Buying; buying2\_\_0: Buying
                |       buying1\_\_0 \in buyingSet' \land \\
                        buying2\_\_0 \in buyingSet' \land \\
                        \lnot buying1\_\_0 = buying2\_\_0 \land \\
                        \lnot (buying1\_\_0.goodsId \in GOODSID \land \\
                        buying2\_\_0.goodsId \in GOODSID \land \\
                        \lnot buying1\_\_0.goodsId = buying2\_\_0.goodsId)
                @       (buying1\_\_0.orderId \in ORDERID \land \\
                        buying2\_\_0.orderId \in ORDERID \land \\
                        \lnot buying1\_\_0.orderId = buying2\_\_0.orderId)) \\
                        \land \dom buyingAtt \in \power buyingKey'))
```

การทดสอบ BDD4 ด้วยคำสั่ง

try BDD4 [op1buyingKey? := BuyingKeyData, op1newValue? := BuyingData];

เมื่อใช้คำสั่ง prove by reduce ได้ผลดังนี้

```
buyingSet \in \finset Buying \\
\land buyingKey \in \finset (ORDERID \cross GOODSID) \\
\land buyingAtt \in ORDERID \cross GOODSID \pfun Buying \\
\land buyingSet' \in \finset Buying \\
\land buyingKey' \in \finset (ORDERID \cross GOODSID) \\
\land BuyingKeyData \in ORDERID \cross GOODSID \\
```

```
\land BuyingData \in Buying \\
\land BuyingKeyData \in \dom buyingAtt \\
\land BuyingKeyData \in buyingKey' \\
\land buyingAtt' = buyingAtt \oplus \{(BuyingKeyData, BuyingData)\} \\
\land \dom buyingAtt \in \power buyingKey \\
\land \dom buyingAtt \in \power buyingKey' \\
\land (buying1 \in Buying \land buying2 \in Buying \land buying1 \in buyingSet \\
        \land buying2 \in buyingSet \land \lnot buying1 = buying2 \land \\
(buying1.goodsId \in GOODSID \land buying2.goodsId \in GOODSID \\
\implies
buying1.goodsId = buying2.goodsId) \\
\implies
buying1.orderId \in ORDERID \land buying2.orderId \in ORDERID \\
         \land \lnot buying1.orderId = buying2.orderId) \land \\
(buyingVar2 \in Buying \\
\implies
(\exists officerVar1: Officer @ buyingVar2.orderOfficer = officerVar1.officerId)) \land \\
(buyingVar2\_\_0 \in Buying \\
\implies
(\exists officerVar1\_\_0: Officer
        @ buyingVar2\_\_0.receiveOfficer = officerVar1\_\_0.officerId)) \land \\
(buyingVar2\_\_1 \in Buying \\
\implies
(\exists cargoVar1: Cargo @ buyingVar2\_\_1.goodsId = cargoVar1.goodsId)) \land \\
(buying1\_\_0 \in Buying \land buying2\_\_0 \in Buying \land buying1\_\_0 \in buyingSet' \\
        \land buying2\_\_0 \in buyingSet' \land \lnot buying1\_\_0 = buying2\_\_0 \\
        \land (buying1\_\_0.goodsId \in GOODSID \land \\
        buying2\_\_0.goodsId \in GOODSID \\
\implies
buying1\_\_0.goodsId = buying2\_\_0.goodsId) \\
\implies
buying1\_\_0.orderId \in ORDERID \land buying2\_\_0.orderId \in ORDERID \\
         \land \lnot buying1\_\_0.orderId = buying2\_\_0.orderId)
```

กรณีทดสอบที่ 5 มีรายละเอียดแฟ้มข้อความของแผนภาพเชิงลำดับชั้นของภาษาเอสคิวแอล ดังนี้

```
%BDDBEGIN
1:Delete:Buying:-:-:Success:NotFound
%BDDEND
```

ข้อกำหนดเซตของแผนภาพเชิงลำดับชั้นของภาษาเอสคิวแอล มีดังนี้

```
\begin{schema} {BDD5Op1Delete}
        \Delta BuyingExt \\
        buyingKey? : ORDERID \cross GOODSID \\
        delBuying? : Buying \\
\where
        buyingKey? \in \dom buyingAtt \\
        buyingSet' = buyingSet \setminus \{ delBuying? \} \\
\end{schema}

\begin{zed}
        BDD5 \defs \\
        BDD5Op1Delete [op1buyingKey?/buyingKey?, op1delBuying?/delBuying?] \\
\end{zed}
```

การทดสอบโดยการสร้างเงื่อนไขก่อน ของ BDD5Op1Delete ด้วยคำสั่ง

try $\forall$ *BuyingExt; buyingKey? : ORDER* $\times$ *GOODSID; delBuying? : Buying*

- *pre BDD5Op1Delete;*

เมื่อใช้คำสั่ง prove by reduce ได้ผลดังนี้

```
buyingSet \in \finset Buying \land
        buyingKey \in \finset (ORDERID \cross GOODSID) \land \\
        buyingAtt \in ORDERID \cross GOODSID \pfun Buying \land \\
        buyingKey? \in ORDERID \cross GOODSID \land \\
        delBuying? \in Buying \land \\
(\forall buying1: Buying; buying2: Buying
|       buying1 \in buyingSet \land \land buying2 \in buyingSet \\
        \land \lnot buying1 = buying2 \land \lnot (buying1.goodsId \in GOODSID \land \\
        buying2.goodsId \in GOODSID \land \lnot buying1.goodsId = buying2.goodsId)
@       (buying1.orderId \in ORDERID \land buying2.orderId \in ORDERID \land \\
        \lnot buying1.orderId = buying2.orderId)) \land \\
(\forall buyingVar2: Buying @
        (\exists officerVar1: Officer @
                buyingVar2.orderOfficer = officerVar1.officerId)) \land \\
(\forall buyingVar2\_\_0: Buying @
```

```
        (\exists officerVar1\_\_0: Officer @
                buyingVar2\_\_0.receiveOfficer = officerVar1\_\_0.officerId)) \land \\
(\forall buyingVar2\_\_1: Buying @
        (\exists cargoVar1: Cargo @
                buyingVar2\_\_1.goodsId = cargoVar1.goodsId)) \land \\
                \dom buyingAtt \in \power buyingKey \\
\implies
(\exists buyingAtt': \power ((ORDERID \cross GOODSID) \cross \lblot
        buyQuantity: \num, goodsId: GOODSID, orderComment: COMMENT,
        orderDate: DATE, orderId: ORDERID, orderOfficer: OFFID, pricePerUnit: \num,
        receiveFlag: BOOLEAN, receiveOfficer: OFFID, sendDate: DATE \rblot)
@       (\exists buyingKey': \power (ORDERID \cross GOODSID)
        @       buyingKey' \in \finset (ORDERID \cross GOODSID) \land \\
                buyingAtt' \in ORDERID \cross GOODSID \pfun Buying \land
                buyingKey? \in \dom buyingAtt \land \\
                (\forall buying1\_\_0: Buying; buying2\_\_0: Buying
                |       buying1\_\_0 \in buyingSet \land \\
                        buying2\_\_0 \in buyingSet \land \\
                        \lnot buying1\_\_0 = delBuying? \land \\
                        \lnot buying2\_\_0 = delBuying? \land \\
                        \lnot buying1\_\_0 = buying2\_\_0 \land \\
                        \lnot (buying1\_\_0.goodsId \in GOODSID \land \\
                        buying2\_\_0.goodsId \in GOODSID \land \\
                        \lnot buying1\_\_0.goodsId = buying2\_\_0.goodsId)
                @       (buying1\_\_0.orderId \in ORDERID \land \\
                        buying2\_\_0.orderId \in ORDERID \land \\
                        \lnot buying1\_\_0.orderId = buying2\_\_0.orderId)) \\
                        \land \dom buyingAtt' \in \power buyingKey'))
```

การทดสอบ BDD5 ด้วยคำสั่ง

try BDD5 [op1buyingKey? := BuyingKeyData, op1delBuying? := BuyingData];

เมื่อใช้คำสั่ง prove by reduce ได้ผลดังนี้

```
buyingSet \in \finset Buying \\
        \land buyingKey \in \finset (ORDERID \cross GOODSID) \\
        \land buyingAtt \in ORDERID \cross GOODSID \pfun Buying \\
        \land BuyingData \in Buying \\
```

```
        \land buyingSet' = buyingSet \setminus \{BuyingData\} \\
        \land buyingKey' \in \finset (ORDERID \cross GOODSID) \\
        \land buyingAtt' \in ORDERID \cross GOODSID \pfun Buying \\
        \land BuyingKeyData \in ORDERID \cross GOODSID \\
        \land BuyingKeyData \in \dom buyingAtt \\
        \land \dom buyingAtt \in \power buyingKey \\
        \land \dom buyingAtt' \in \power buyingKey' \\
        \land (buying1 \in Buying \land buying2 \in Buying \land buying1 \in buyingSet \\
        \land buying2 \in buyingSet \land \lnot buying1 = buying2 \\
        \land (buying1.goodsId \in GOODSID \land buying2.goodsId \in GOODSID \\
\implies
buying1.goodsId = buying2.goodsId) \\
\implies
buying1.orderId \in ORDERID \land buying2.orderId \in ORDERID \land \\
        \lnot buying1.orderId = buying2.orderId) \land (buyingVar2 \in Buying \\
\implies
(\exists officerVar1: Officer @ buyingVar2.orderOfficer = officerVar1.officerId)) \land \\
(buyingVar2\_\_0 \in Buying \\
\implies
(\exists officerVar1\_\_0: Officer
        @ buyingVar2\_\_0.receiveOfficer = officerVar1\_\_0.officerId)) \land \\
(buyingVar2\_\_1 \in Buying \\
\implies
(\exists cargoVar1: Cargo @ buyingVar2\_\_1.goodsId = cargoVar1.goodsId)) \land \\
(buying1\_\_0 \in Buying \land buying2\_\_0 \in Buying \land buying1\_\_0 \in buyingSet \\
        \land buying2\_\_0 \in buyingSet \land \lnot buying1\_\_0 = BuyingData \\
        \land \lnot buying2\_\_0 = BuyingData \land \lnot buying1\_\_0 = buying2\_\_0 \\
        \land
(buying1\_\_0.goodsId \in GOODSID \land buying2\_\_0.goodsId \in GOODSID \\
\implies
buying1\_\_0.goodsId = buying2\_\_0.goodsId) \\
\implies
buying1\_\_0.orderId \in ORDERID \land buying2\_\_0.orderId \in ORDERID \\
        \land \lnot buying1\_\_0.orderId = buying2\_\_0.orderId)
```

ภาคผนวก ค

ข้อกำหนดเซตของฐานข้อมูลจัดการชิ้นส่วน

(SPJ Database)

แฟ้มข้อความพจนานุกรมข้อมูลของฐานข้อมูลจัดการชิ้นส่วน มีรูปแบบ คือ

```
Entity Name:Attribute Name:Data Type:Key:<FK Entity-Attribute>:<CS Constraint>
```

โดยมีรายละเอียดแฟ้มข้อความพจนานุกรมข้อมูลของฐานข้อมูลจัดการชิ้นส่วน ดังนี้

```
Supplier:sNo:SNO:PK
Supplier:sName:SNAME:NK
Supplier:sStatus:\num:NK:CS \geq 0
Supplier:sCity:CITY:NK:
Part:pNo:PNO:PK
Part:pName:PNAME:NK
Part:pColor:COLOR:NK
Part:pWeight:\num:NK:CS > 0
Part:pCity:CITY:NK
SP:sNo:SNO:PK:FK Supplier-sNo
SP:pNo:PNO:PK:FK Part-pNo
SP:spQuantity:\num:NK:CS \geq 0
Project:jNo:JNO:PK
Project:jName:JNAME:NK
Project:jCity:CITY:NK
SPJ:sNo:SNO:PK:FK Supplier-sNo
SPJ:pNo:PNO:PK:FK Part-pNo
SPJ:jNo:JNO:PK:FK Project-jNo
SPJ:spjQuantity:\num:NK:CS \geq 0
```

แฟ้มข้อความพจนานุกรมความสัมพันธ์ของฐานข้อมูลจัดการชิ้นส่วนมีรูปแบบ คือ

```
Relation Name:Entity 1:Entity 2:Relation Type
```

โดยมีรายละเอียดแฟ้มข้อความพจนานุกรมความสัมพันธ์ฐานข้อมูลจัดการชิ้นส่วน ดังนี้

```
spSupplier:SP:Supplier:m-1
spPart:SP:Part:m-1
spjSupplier:SPJ:Supplier:m-1
spjPart:SPJ:Part:m-1
spjProject:SPJ:Project:m-1
```

ข้อกำหนดเซตของฐานข้อมูลจัดการชิ้นส่วน มีดังนี้

```
\begin{zed}
        [ SNO, SNAME, STATUS, CITY, \\
          PNO, PNAME, COLOR, WEIGHT, JNO, JNAME ]
\end{zed}

\begin{zed}
        BOOLEAN ::= True | False \\
\end{zed}

\begin{schema} {Supplier}
        sNo : SNO \\
        sName : SNAME \\
        sStatus : \num \\
        sCity : CITY \\
\where
        sStatus \geq 0 \\
\end{schema}

\begin{schema} {Part}
        pNo : PNO \\
        pName : PNAME \\
        pColor : COLOR \\
        pWeight : \num \\
        pCity : CITY \\
\where
        pWeight > 0 \\
\end{schema}

\begin{schema} {SP}
        sNo : SNO \\
        pNo : PNO \\
        spQuantity : \num \\
\where
        spQuantity \geq 0 \\
\end{schema}

\begin{schema} {Project}
        jNo : JNO \\
        jName : JNAME \\
        jCity : CITY \\
\end{schema}

\begin{schema} {SPJ}
        sNo : SNO \\
        pNo : PNO \\
        jNo : JNO \\
        spjQuantity : \num \\
\where
        spjQuantity \geq 0 \\
\end{schema}

\begin{schema} {SupplierExt}
        supplierSet : \finset Supplier \\
        supplierKey : \finset SNO \\
        supplierAtt : SNO \pfun Supplier \\
\where
```

```
        \forall Supplier1, Supplier2 : Supplier \\
        | Supplier1 \in supplierSet \land Supplier2 \in supplierSet \land \\
          Supplier1 \neq Supplier2 @ Supplier1.sNo \neq Supplier2.sNo \\
        \dom supplierAtt \subseteq supplierKey \\
\end{schema}

\begin{schema} {PartExt}
        partSet : \finset Part \\
        partKey : \finset PNO \\
        partAtt : PNO \pfun Part \\
\where
        \forall Part1, Part2 : Part \\
        | Part1 \in partSet \land Part2 \in partSet \land Part1 \neq Part2 \\
        @ Part1.pNo \neq Part2.pNo \\
        \dom partAtt \subseteq partKey \\
\end{schema}

\begin{schema} {SPExt}
        spSet : \finset SP \\
        spKey : \finset (SNO \cross PNO) \\
        spAtt : (SNO \cross PNO) \pfun SP \\
\where
        \forall SP1, SP2 : SP \\
        | SP1 \in spSet \land SP2 \in spSet \land SP1 \neq SP2
        @ SP1.pNo \neq SP2.pNo \lor SP1.sNo \neq SP2.sNo \\
        \forall SPVar2 : SP @ (\exists PartVar1 : Part @ SPVar2.pNo = PartVar1.pNo) \\
        \forall SPVar2 : SP
        @ (\exists SupplierVar1 : Supplier @ SPVar2.sNo = SupplierVar1.sNo) \\
        \dom spAtt \subseteq spKey \\
\end{schema}

\begin{schema} {ProjectExt}
        projectSet : \finset Project \\
        projectKey : \finset JNO \\
        projectAtt : JNO \pfun Project \\
\where
        \forall Project1, Project2 : Project \\
        | Project1 \in projectSet \land Project2 \in projectSet \land Project1 \neq Project2 \\
        @ Project1.jNo \neq Project2.jNo \\
        \dom projectAtt \subseteq projectKey \\
\end{schema}

\begin{schema} {SPJExt}
        spjSet : \finset SPJ \\
        spjKey : \finset (SNO \cross PNO \cross JNO) \\
        spjAtt : (SNO \cross PNO \cross JNO) \pfun SPJ \\
\where
        \forall SPJ1, SPJ2 : SPJ \\
        | SPJ1 \in spjSet \land SPJ2 \in spjSet \land SPJ1 \neq SPJ2 \\
        @ SPJ1.jNo \neq SPJ2.jNo \lor SPJ1.pNo \neq SPJ2.pNo \lor \\
            SPJ1.sNo \neq SPJ2.sNo \\
        \forall SPJVar2 : SPJ
        @ (\exists ProjectVar1 : Project @ SPJVar2.jNo = ProjectVar1.jNo) \\
        \forall SPJVar2 : SPJ @ (\exists PartVar1 : Part @ SPJVar2.pNo = PartVar1.pNo) \\
        \forall SPJVar2 : SPJ
        @ (\exists SupplierVar1 : Supplier @ SPJVar2.sNo = SupplierVar1.sNo) \\
        \dom spjAtt \subseteq spjKey \\
\end{schema}
```

```
\begin{schema} {RelationshipspSupplier}
        spSupplier : \finset (SP \cross Supplier)
\where
        \forall spSupplier1,spSupplier2 : spSupplier \\
        @ first~spSupplier1 = first~spSupplier2 \implies \\
            second~spSupplier1 = second~spSupplier2 \\
\end{schema}

\begin{schema} {RelationshipspPart}
        spPart : \finset (SP \cross Part)
\where
        \forall spPart1,spPart2 : spPart \\
        @ first~spPart1 = first~spPart2 \implies second~spPart1 = second~spPart2 \\
\end{schema}

\begin{schema} {RelationshipspjSupplier}
        spjSupplier : \finset (SPJ \cross Supplier)
\where
        \forall spjSupplier1,spjSupplier2 : spjSupplier \\
        @ first~spjSupplier1 = first~spjSupplier2 \implies \\
            second~spjSupplier1 = second~spjSupplier2 \\
\end{schema}

\begin{schema} {RelationshipspjPart}
        spjPart : \finset (SPJ \cross Part)
\where
        \forall spjPart1,spjPart2 : spjPart \\
        @ first~spjPart1 = first~spjPart2 \implies second~spjPart1 = second~spjPart2 \\
\end{schema}

\begin{schema} {RelationshipspjProject}
        spjProject : \finset (SPJ \cross Project)
\where
        \forall spjProject1,spjProject2 : spjProject \\
        @ first~spjProject1 = first~spjProject2 \implies \\
            second~spjProject1 = second~spjProject2 \\
\end{schema}

\begin{zed}
        Response ::= Success | NotFound \\
\end{zed}

\begin{schema} {Success}
        r! : Response \\
\where
        r! = Success \\
\end{schema}

\begin{schema} {NotFound}
        r! : Response \\
\where
        r! = NotFound \\
\end{schema}
```

กรณีทดสอบที่ 1 มีรายละเอียดแฟ้มข้อความของแผนภาพเชิงลำดับชั้นของภาษาเอสคิวแอล ดังนี้

```
%BDDBEGIN
1:Select:SP:-:pNo = "P2":2:NotFound
2:In:Supplier:1:Supplier.sNo 1.sNo:3:NotFound
3:Project:2:-:sName:Success:NotFound
%BDDEND
```

ข้อกำหนดเซตของแผนภาพเชิงลำดับชั้นของภาษาเอสคิวแอล มีดังนี้

```
\begin{schema} {BDD1R1}
        sNo : SNO \\
        pNo : PNO \\
        spQuantity : \num \\
\end{schema}

\begin{schema} {BDD1Op1Select}
        input1? : \finset SP \\
        pNoValue? : PNO \\
        output! : \finset BDD1R1 \\
\where
        input1? \neq \emptyset \\
        \forall out : output!; in1 : input1? | in1.pNo = pNoValue? @ out = in1 \\
\end{schema}

\begin{schema} {BDD1R2}
        sNo : SNO \\
        sName : SNAME \\
        sStatus : \num \\
        sCity : CITY \\
\end{schema}

\begin{schema} {BDD1Op2In}
        input1? : \finset Supplier \\
        input2? : \finset BDD1R1 \\
        output! : \finset BDD1R2 \\
\where
        input1? \neq \emptyset \\
        input2? \neq \emptyset \\
        \forall out : output!; in1: input1?; in2 : input2? \\
        | in1.sNo \in \{ in2.sNo \} @ out = in1 \\
\end{schema}

\begin{schema} {BDD1R3}
        sName : SNAME \\
\end{schema}

\begin{schema} {BDD1Op3Project}
        input1? : \finset BDD1R2 \\
        output! : \finset BDD1R3 \\
\where
        input1? \neq \emptyset \\
        \forall out : output!; in1 : input1? @ out.sName = in1.sName \\
\end{schema}

\begin{zed}
```

```
        BDD1 \defs \\
        BDD1Op1Select [op1input?/input1?, op1pNoValue?/pNoValue?,
                op1output!/output!] \semi \\
        BDD1Op2In  [op2input1?/input1?,  op2input2?/input2?,  op2output!/output!]  \semi  \\
        BDD1Op3Project [op3input1?/input1?, op3output!/output!]
\end{zed}
```

การทดสอบโดยการสร้างเงื่อนไขก่อน ของ BDD1Op1Select ด้วยคำสั่ง

try $\forall$ *input1? :* $\mathbb{F}$ *SP; pNoValue? : PNO • pre BDD1Op1Select;*

เมื่อใช้คำสั่ง prove by reduce ได้ผลดังนี้

```
input1? \in \finset SP \land pNoValue? \in PNO \\
\implies
(\exists output!: \power \lblot pNo: PNO, sNo: SNO, spQuantity: \num \rblot
@       output! \in \finset BDD1R1 \land \lnot input1? = \{\} \land \\
        (\forall out: output!; in1: input1? | in1.pNo = pNoValue? @ out = in1))
```

การทดสอบโดยการสร้างเงื่อนไขก่อน ของ BDD1Op2In ด้วยคำสั่ง

try $\forall$ *input1? :* $\mathbb{F}$ *Supplier; input2? :* $\mathbb{F}$ *BDD1R1 • pre BDD1Op2In;*

เมื่อใช้คำสั่ง prove by reduce ได้ผลดังนี้

```
input1? \in \finset Supplier \land input2? \in \finset BDD1R1 \\
\implies
(\exists output!: \power \lblot sCity: CITY, sName: SNAME, sNo: SNO, sStatus: \num \rblot
@       output! \in \finset BDD1R2 \land \lnot input1? = \{\} \land \lnot input2? = \{\} \land \\
        (\forall out: output!; in1: input1?; in2: input2? |  in1.sNo = in2.sNo @ out = in1))
```

การทดสอบโดยการสร้างเงื่อนไขก่อน ของ BDD1Op3Project ด้วยคำสั่ง

try $\forall$ *input1? :* $\mathbb{F}$ *BDD1R2 • pre BDD1Op3Project;*

เมื่อใช้คำสั่ง prove by reduce ได้ผลดังนี้

```
input1? \in \finset BDD1R2 \\
\implies
(\exists output!: \power \lblot sName: SNAME \rblot
@       output! \in \finset BDD1R3 \land \lnot input1? = \{\} \land \\
        (\forall out: output!; in1: input1? @ out.sName = in1.sName))
```

การทดสอบ BDD1 ด้วยคำสั่ง

try BDD1 [op1input1? := SPData, op1pNoValue? := P2, op2input1? := SupplierData];

เมื่อใช้คำสั่ง prove by reduce ได้ผลดังนี้

```
SPData \in \power \lblot pNo: PNO, sNo: SNO, spQuantity: \num \rblot \\
        \land op1output! \in \power \lblot pNo: PNO, sNo: SNO, spQuantity: \num \rblot \\
        \land P2 \in PNO \land \\
```

```
SupplierData \in \power \lblot sCity: CITY, sName: SNAME, sNo: SNO, sStatus: \num \\
        \rblot \land \\
op2input2? \in \power \lblot pNo: PNO, sNo: SNO, spQuantity: \num \rblot \land op2output! \\
        \in \power \lblot sCity: CITY, sName: SNAME, sNo: SNO, sStatus: \num \rblot \\
        \land
op3input1? \in \power \lblot sCity: CITY, sName: SNAME, sNo: SNO, sStatus: \num \rblot \\
\land op3output! \in \power \lblot sName: SNAME \rblot \\
\land SPData \in \finset SP \\
\land op1output! \in \finset BDD1R1 \\
\land SupplierData \in \finset Supplier \\
\land op2input2? \in \finset BDD1R1 \\
\land op2output! \in \finset BDD1R2 \\
\land op3input1? \in \finset BDD1R2 \\
\land op3output! \in \finset BDD1R3 \\
        \land \lnot SPData = \{\} \\
        \land \lnot SupplierData = \{\} \\
        \land \lnot op2input2? = \{\} \\
        \land \lnot op3input1? = \{\} \\
        \land (    out \in op1output! \\
          \land in1 \in SPData \\
          \land in1.pNo = P2 \\
\implies
out = in1) \land \\
(out\_\_0 \in op2output! \land in1\_\_0 \in SupplierData \land in2 \in op2input2? \\
          \land in1\_\_0.sNo = in2.sNo \\
\implies
out\_\_0 = in1\_\_0) \land \\
(out\_\_1 \in op3output! \land in1\_\_1 \in op3input1? \\
\implies
out\_\_1.sName = in1\_\_1.sName)
```

กรณีทดสอบที่ 2 มีรายละเอียดแฟ้มข้อความของแผนภาพเชิงลำดับชั้นของภาษาเอสคิวแอล ดังนี้

```
(Node#:Operation Name:Input Node 1:Input Node 2:Condition:Output Node:NotFound Node)
1:Join:SP:Supplier:SP.sNo = Supplier.sNo:2:NotFound
2:Select:1:-:pNo = "P2":3:NotFound
3:NotExists:Supplier:2:Supplier.sName 2.sName Supplier.sCity 2.sCity:4:NotFound
4:Project:3:-:sName sCity:Success:NotFound
```

ข้อกำหนดเซตของแผนภาพเชิงลำดับชั้นของภาษาเอสคิวแอล มีดังนี้

```
\begin{schema} {BDD2R1}
        sNo : SNO \\
        pNo : PNO \\
        spQuantity : \num \\
        sName : SNAME \\
        sStatus : \num \\
        sCity : CITY \\
\end{schema}

\begin{schema} {BDD2Op1Join}
        input1? : \finset SP \\
        input2? : \finset Supplier \\
        output! : \finset BDD2R1 \\
\where
        input1? \neq \emptyset \\
        input2? \neq \emptyset \\
        \forall out : output!; in1 : input1?; in2 : input2? | in1.sNo = in2.sNo \\
        @ out.sNo = in1.sNo  \land out.pNo = in1.pNo \land \\
            out.spQuantity = in1.spQuantity \land out.sName = in2.sName \land \\
            out.sStatus = in2.sStatus \land out.sCity = in2.sCity \\
\end{schema}

\begin{schema} {BDD2R2}
        sNo : SNO \\
        pNo : PNO \\
        spQuantity : \num \\
        sName : SNAME \\
        sStatus : \num \\
        sCity : CITY \\
\end{schema}

\begin{schema} {BDD2Op2Select}
        input1? : \finset BDD2R1 \\
        pNoValue? : PNO \\
        output! : \finset BDD2R2 \\
\where
        input1? \neq \emptyset \\
        \forall out : output!; in1 : input1? | in1.pNo = pNoValue? @ out = in1 \\
\end{schema}

\begin{schema} {BDD2R3}
        sNo : SNO \\
        sName : SNAME \\
        sStatus : \num \\
        sCity : CITY \\
\end{schema}
```

```
\begin{schema} {BDD2Op3NotExists}
        input1? : \finset Supplier \\
        input2? : \finset BDD2R2 \\
        output! : \finset BDD2R3 \\
\where
        input1? \neq \emptyset \\
        input2? \neq \emptyset \\
        \forall out : output!; in1 : input1?; in2 : input2? \\
        | in1.sName \notin \{ in2.sName \} \land in1.sCity \notin \{ in2.sCity \} \\
        @ out = in1 \\
\end{schema}

\begin{schema} {BDD2R4}
        sName : SNAME \\
        sCity : CITY \\
\end{schema}

\begin{schema} {BDD2Op4Project}
        input1? : \finset Supplier \\
        output! : \finset BDD2R4 \\
\where
        input1? \neq \emptyset \\
        \forall out : output!; in1 : input1? \\
        @ out.sName = in1.sName \land out.sCity = in1.sCity \\
\end{schema}

\begin{zed}
        BDD2 \defs \\
        BDD2Op1Join [op1input1?/input1?, op1input2?/input2?, op1output!/output!] \semi \\
        BDD2Op2Select [op1input1?/input1?, op2pNoValue?/pNoValue?,
                op2output!/output!] \semi \\
        BDD2Op3NotExists [op3input1?/input1?, op3input2?/input2?,
                op3output!/output!] \semi \\
        BDD2Op4Project [op4input1?/input1?, op4output!/output!]
\end{zed}
```

การทดสอบโดยการสร้างเงื่อนไขก่อน ของ BDD2Op1Join ด้วยคำสั่ง

try $\forall$ *input1? :* $\mathbb{F}$ *SP; input2? :* $\mathbb{F}$ *Supplier • pre BDD2Op1Join;*

เมื่อใช้คำสั่ง prove by reduce ได้ผลดังนี้

```
input1? \in \finset SP \land input2? \in \finset Supplier \\

\implies

(\exists output!: \power \lblot pNo: PNO, sCity: CITY, sName: SNAME, sNo: SNO,
        sStatus: \num, spQuantity: \num \rblot
@       output! \in \finset BDD5R1 \land \lnot input1? = \{\} \land \\
        \lnot input2? = \{\} \land \\
        (\forall out: output!; in1: input1?; in2: input2?
        |       in1.sNo = in2.sNo
        @       (out.sNo = in1.sNo \land out.pNo = in1.pNo \land \\
```

out.spQuantity = in1.spQuantity \land \\

out.sName = in2.sName \land \\

out.sStatus = in2.sStatus \land out.sCity = in2.sCity)))

การทดสอบโดยการสร้างเงื่อนไขก่อน ของ BDD2Op2Select ด้วยคำสั่ง

try $\forall$ *input1? :* $\mathbb{F}$ *BDD2R1; pNoValue? : PNO • pre BDD2Op2Select;*
เมื่อใช้คำสั่ง prove by reduce ได้ผลดังนี้

input1? \in \finset BDD5R1 \land pNoValue? \in PNO \\

\implies

(\exists output!: \power \lblot pNo: PNO, sCity: CITY, sName: SNAME, sNo: SNO,

sStatus: \num, spQuantity: \num \rblot

@ output! \in \finset BDD5R2 \land \lnot input1? = \{\} \land \\

(\forall out: output!; in1: input1? | in1.pNo = pNoValue? @ out = in1))

การทดสอบโดยการสร้างเงื่อนไขก่อน ของ BDD2Op3NotExists ด้วยคำสั่ง

try $\forall$ *input1? :* $\mathbb{F}$ *BDD2R1; pNoValue? : PNO • pre BDD2Op2Select;*
เมื่อใช้คำสั่ง prove by reduce ได้ผลดังนี้

input1? \in \finset Supplier \land input2? \in \finset BDD5R2 \\

\implies

(\exists output!: \power \lblot sCity: CITY, sName: SNAME, sNo: SNO, sStatus: \num

\rblot

@ output! \in \finset BDD5R3 \land \lnot input1? = \{\} \land \lnot input2? = \{\} \land \\

(\forall out: output!; in1: input1?; in2: input2?

| in1.sName \in SNAME \land in2.sName \in SNAME \\

\land \lnot in1.sName = in2.sName \land in1.sCity \in CITY \\

\land in2.sCity \in CITY \land \lnot in1.sCity = in2.sCity

@ out = in1))

การทดสอบโดยการสร้างเงื่อนไขก่อน ของ BDD2Op4Project ด้วยคำสั่ง

try $\forall$ *input1? :* $\mathbb{F}$ *BDD2R3 • pre BDD2Op4Project;*
เมื่อใช้คำสั่ง prove by reduce ได้ผลดังนี้

input1? \in \finset Supplier \\

\implies

(\exists output!: \power \lblot sCity: CITY, sName: SNAME \rblot

@ output! \in \finset BDD5R4 \land \lnot input1? = \{\} \land \\

(\forall out: output!; in1: input1?

@ (out.sName = in1.sName \land out.sCity = in1.sCity)))

การทดสอบ BDD2 ด้วยคำสั่ง

try BDD2 [op1input1? := SPData, op1input2? := SupplierData, op2pNoValue? := P2,
op3input1? := SupplierData];

เมื่อใช้คำสั่ง prove by reduce ได้ผลดังนี้

```
SPData \in \power \lblot pNo: PNO, sNo: SNO, spQuantity: \num \rblot \land \\
SupplierData \in \power \lblot sCity: CITY, sName: SNAME, sNo: SNO, sStatus: \num \\
        \rblot \land op1output! \in \power \lblot pNo: PNO, sCity: CITY, sName: SNAME,
        sNo: SNO, sStatus: \num, spQuantity: \num \rblot \land \\
op2input1? \in \power \lblot pNo: PNO, sCity: CITY, sName: SNAME, sNo: SNO, \\
        sStatus: \num, spQuantity: \num \rblot \land \\
op2output! \in \power \lblot pNo: PNO, sCity: CITY, sName: SNAME, sNo: SNO, \\
        sStatus: \num, spQuantity: \num \rblot \land P2 \in PNO \land \\
op3input22? \in \power \lblot pNo: PNO, sCity: CITY, sName: SNAME, sNo: SNO, \\
        sStatus: \num, spQuantity: \num \rblot \land \\
op3output! \in \power \lblot sCity: CITY, sName: SNAME, sNo: SNO, sStatus: \num \rblot \\
        \land
 op4input1? \in \power \lblot sCity: CITY, sName: SNAME, sNo: SNO, sStatus: \num \rblot \\
        \land op4output! \in \power \lblot sCity: CITY, sName: SNAME \rblot \\
\land SPData \in \finset SP \\
\land SupplierData \in \finset Supplier \\
\land op1output! \in \finset BDD2R1 \\
\land op2input1? \in \finset BDD2R1 \\
\land op2output! \in \finset BDD2R2 \\
\land op3input22? \in \finset BDD2R2 \\
\land op3output! \in \finset BDD2R3 \\
\land op4input1? \in \finset Supplier \\
\land op4output! \in \finset BDD2R4 \\
        \land \lnot SPData = \{\} \\
        \land \lnot SupplierData = \{\} \\
        \land \lnot op2input1? = \{\} \\
        \land \lnot op3input22? = \{\} \\
        \land \lnot op4input1? = \{\} \\
        \land (    out \in op1output! \\
          \land in1 \in SPData \\
          \land in2 \in SupplierData \\
          \land in1.sNo = in2.sNo \\
```

```
\implies out.sNo = in1.sNo \\
        \land out.pNo = in1.pNo \\
        \land out.spQuantity = in1.spQuantity \\
        \land out.sName = in2.sName \\
        \land out.sStatus = in2.sStatus \\
        \land out.sCity = in2.sCity) \\
        \land (   out\_\_0 \in op2output! \\
        \land in1\_\_0 \in op2input1? \\
        \land in1\_\_0.pNo = P2 \\
\implies out\_\_0 = in1\_\_0) \\
        \land (   out\_\_1 \in op3output! \\
        \land in1\_\_1 \in SupplierData \\
        \land in2\_\_0 \in op3input22? \\
        \land in1\_\_1.sName \in SNAME \\
        \land in2\_\_0.sName \in SNAME \\
        \land \lnot in1\_\_1.sName = in2\_\_0.sName \\
\implies out\_\_1 = in1\_\_1) \\
        \land (   out\_\_2 \in op4output! \\
        \land in1\_\_2 \in op4input1? \\
\implies out\_\_2.sName = in1\_\_2.sName \\
        \land out\_\_2.sCity = in1\_\_2.sCity)
```

กรณีทดสอบที่ 3 มีรายละเอียดแฟ้มข้อความของแผนภาพเชิงลำดับชั้นของภาษาเอสคิวแอล ดังนี้

```
%BDDBEGIN
1:Select:SP:-:pNo = "P2":2:NotFound
2:NotIn:Supplier:1:Supplier.sNo 2.sNo:4:NotFound
3:Project:2:-:sName:Success:NotFound
%BDDEND
```

ข้อกำหนดเซตของแผนภาพเชิงลำดับชั้นของภาษาเอสคิวแอล มีดังนี้

```
\begin{schema} {BDD3R1}
        sNo : SNO \\
        pNo : PNO \\
        spQuantity : \num \\
\end{schema}

\begin{schema} {BDD3Op1Select}
        input1? : \finset SP \\
        pNoValue? : PNO \\
        output! : \finset BDD3R1 \\
```

```
\where
        input1? \neq \emptyset \\
        \forall out : output!; in1 : input1? \\
                |       in1.pNo = pNoValue? \\
                @       out = in1 \\
\end{schema}

\begin{schema} {BDD3R2}
        sNo : SNO \\
        sName : SNAME \\
        sStatus : \num \\
        sCity : CITY \\
\end{schema}

\begin{schema} {BDD3Op2NotIn}
        input1? : \finset Supplier \\
        input2? : \finset BDD3R1 \\
        output! : \finset BDD3R2 \\
\where
        input1? \neq \emptyset \\
        input2? \neq \emptyset \\
        \forall out : output!; in1: input1?; in2 : input2? \\
                |       in1.sNo \notin \{ in2.sNo \} \\
                @       out = in1 \\
\end{schema}

\begin{schema} {BDD3R3}
        sName : SNAME \\
\end{schema}

\begin{schema} {BDD3Op3Project}
        input1? : \finset BDD3R2 \\
        output! : \finset BDD3R3 \\
\where
        input1? \neq \emptyset \\
        \forall out : output!; in1 : input1? \\
                @       out.sName = in1.sName \\
\end{schema}

\begin{zed}
        BDD3 \defs \\
        BDD3Op1Select [op1input?/input1?, op1pNoValue?/pNoValue?,
                op1output!/output!] \semi \\
        BDD3Op2NotIn [op2input1?/input1?, op2input2?/input2?,
                op2output!/output!] \semi \\
        BDD3Op3Project [op3input1?/input1?, op3output!/output!]
\end{zed}
```

การทดสอบโดยการสร้างเงื่อนไขก่อน ของ BDD3Op1Select ด้วยคำสั่ง

try $\forall$ *input1? :* $\mathbb{F}$ *SP; pNoValue? : PNO • pre BDD3Op1Select;*

เมื่อใช้คำสั่ง prove by reduce ได้ผลดังนี้

```
input1? \in \finset SP \land pNoValue? \in PNO \\

\implies

(\exists output!: \power \lblot pNo: PNO, sNo: SNO, spQuantity: \num \rblot
```

@ output! \in \finset BDD4R1 \land \lnot input1? = \{\} \land \\
(\forall out: output!; in1: input1? | in1.pNo = pNoValue? @ out = in1))

การทดสอบโดยการสร้างเงื่อนไขก่อน ของ BDD3Op2NotIn ด้วยคำสั่ง

try ∀ input1? : 𝔽 Supplier; input2? : 𝔽 BDD3R1 • pre BDD3Op2NotIn;

เมื่อใช้คำสั่ง prove by reduce ได้ผลดังนี้

input1? \in \finset Supplier \land input2? \in \finset BDD4R1 \\
\implies
(\exists output!: \power \lblot sCity: CITY, sName: SNAME, sNo: SNO, sStatus: \num \rblot
@ output! \in \finset BDD4R2 \land \lnot input1? = \{\} \land \lnot input2? = \{\} \land \\
(\forall out: output!; in1: input1?; in2: input2?
| in1.sNo \in SNO \land in2.sNo \in SNO \land \lnot in1.sNo = in2.sNo
@ out = in1))

การทดสอบโดยการสร้างเงื่อนไขก่อน ของ BDD3Op3Project ด้วยคำสั่ง

try ∀ input1? : 𝔽 BDD3R2 • pre BDD3Op3Project;

เมื่อใช้คำสั่ง prove by reduce ได้ผลดังนี้

input1? \in \finset BDD4R2 \\
\implies
(\exists output!: \power \lblot sName: SNAME \rblot
@ output! \in \finset BDD4R3 \land \lnot input1? = \{\} \land \\
(\forall out: output!; in1: input1? @ out.sName = in1.sName))

การทดสอบ BDD1 ด้วยคำสั่ง

try BDD3 [op1input? := SPData, op1pNoValue? := P2, op2input1? := SupplierData];

เมื่อใช้คำสั่ง prove by reduce ได้ผลดังนี้

SPData \in \power \lblot pNo: PNO, sNo: SNO, spQuantity: \num \rblot \land \\
op1output! \in \power \lblot pNo: PNO, sNo: SNO, spQuantity: \num \rblot \land P2 \in PNO \\
\land
SupplierData \in \power \lblot sCity: CITY, sName: SNAME, sNo: SNO, sStatus: \num \rblot \\
\land
op2input2? \in \power \lblot pNo: PNO, sNo: SNO, spQuantity: \num \rblot \land \\
op2output! \in \power \lblot sCity: CITY, sName: SNAME, sNo: SNO, sStatus: \num \rblot \\
\land
op3input1? \in \power \lblot sCity: CITY, sName: SNAME, sNo: SNO, sStatus: \num \rblot \\
\land op3output! \in \power \lblot sName: SNAME \rblot \\
\land SPData \in \finset SP \\
\land op1output! \in \finset BDD4R1 \\

```
\land SupplierData \in \finset Supplier \\
\land op2input2? \in \finset BDD4R1 \\
\land op2output! \in \finset BDD4R2 \\
\land op3input1? \in \finset BDD4R2 \\
\land op3output! \in \finset BDD4R3 \\
\land \lnot SPData = \{\} \\
\land \lnot SupplierData = \{\} \\
\land \lnot op2input2? = \{\} \\
\land \lnot op3input1? = \{\} \\
\land (          out \in op1output! \\
          \land in1 \in SPData \\
          \land in1.pNo = P2 \\
     \implies out = in1) \\
\land (          out\_\_0 \in op2output! \\
          \land in1\_\_0 \in SupplierData \\
          \land in2 \in op2input2? \\
          \land in1\_\_0.sNo \in SNO \\
          \land in2.sNo \in SNO \\
          \land \lnot in1\_\_0.sNo = in2.sNo \\
     \implies out\_\_0 = in1\_\_0) \\
\land (          out\_\_1 \in op3output! \\
          \land in1\_\_1 \in op3input1? \\
     \implies out\_\_1.sName = in1\_\_1.sName)
```

กรณีทดสอบที่ 4 มีรายละเอียดแฟ้มข้อความของแผนภาพเชิงลำดับชั้นของภาษาเอสคิวแอล ดังนี้

```
%BDDBEGIN
1: Join:SP:Supplier:SP.sNo = Supplier.sNo:2:NotFound
2:Select:1:-:pNo = "P2":3:NotFound
3:Exists:Supplier:2:Supplier.sName 2.sName:4:NotFound
4:Project:3:-:sName:Success:NotFound
%BDDEND
```

ข้อกำหนดเซตของแผนภาพเชิงลำดับชั้นของภาษาเอสคิวแอล มีดังนี้

```
\begin{schema} {BDD4R1}
          sNo : SNO \\
          pNo : PNO \\
          spQuantity : \num \\
          sName : SNAME \\
```

```
        sStatus : \num \\
        sCity : CITY \\
\end{schema}

\begin{schema} {BDD4Op1Join}
        input1? : \finset SP \\
        input2? : \finset Supplier \\
        output! : \finset BDD4R1 \\
\where
        input1? \neq \emptyset \\
        input2? \neq \emptyset \\
        \forall out : output!; in1 : input1?; in2 : input2? \\
                |       in1.sNo = in2.sNo \\
                @       out.sNo = in1.sNo  \land \\ % from input 1 (SP)
                        out.pNo = in1.pNo \land \\
                        out.spQuantity = in1.spQuantity \land \\
                        out.sName = in2.sName \land \\
                        out.sStatus = in2.sStatus \land \\
                        out.sCity = in2.sCity \\
\end{schema}

\begin{schema} {BDD4R2}
        sNo : SNO \\
        pNo : PNO \\
        spQuantity : \num \\
        sName : SNAME \\
        sStatus : \num \\
        sCity : CITY \\
\end{schema}

\begin{schema} {BDD4Op2Select}
        input1? : \finset BDD4R1 \\
        pNoValue? : PNO \\
        output! : \finset BDD4R2 \\
\where
        input1? \neq \emptyset \\
        \forall out : output!; in1 : input1? \\
                |       in1.pNo = pNoValue? \\
                @       out = in1 \\
\end{schema}

\begin{schema} {BDD4R3}
        sNo : SNO \\
        pNo : PNO \\
        spQuantity : \num \\
        sName : SNAME \\
        sStatus : \num \\
        sCity : CITY \\
\end{schema}

\begin{schema} {BDD4Op3Exists}
        input1? : \finset Supplier \\
        input2? : \finset BDD4R2 \\
        output! : \finset BDD4R3 \\
\where
        input1? \neq \emptyset \\
        input2? \neq \emptyset \\
        \forall out : output!; in1 : input1?; in2 : input2? \\
```

```
                |       in1.sName \in \{ in2.sName \} \\
                @       out = in2 \\
\end{schema}

\begin{schema} {BDD4R4}
        sName : SNAME \\
\end{schema}

\begin{schema} {BDD4Op4Project}
        input1? : \finset Supplier \\
        output! : \finset BDD4R4 \\
\where
        input1? \neq \emptyset \\
        \forall out : output!; in1 : input1? \\
                @       out.sName = in1.sName \\
\end{schema}

\begin{zed}
        BDD4 \defs \\
        BDD4Op1Join [op1input1?/input1?, op1input2?/input2?, op1output!/output!] \semi \\
        BDD4Op2Select [op2input1?/input1?, op2pNoValue?/pNoValue?,
                op2output!/output!] \semi \\
        BDD4Op3Exists [op3input1?/input1?, op3input2?/input2?,
                op3output!/output!] \semi \\
        BDD4Op4Project [op4input1?/input1?, op4output!/output!]
\end{zed}
```

การทดสอบโดยการสร้างเงื่อนไขก่อน ของ BDD4Op1Join ด้วยคำสั่ง

try $\forall$ *input1? :* $\mathbb{F}$ *SP; input2? :* $\mathbb{F}$ *Supplier • pre BDD4Op1Join;*
เมื่อใช้คำสั่ง prove by reduce ได้ผลดังนี้

```
input1? \in \finset SP \land input2? \in \finset Supplier \\

\implies

(\exists output!: \power \lblot pNo: PNO, sCity: CITY, sName: SNAME, sNo: SNO,

        sStatus: \num, spQuantity: \num \rblot

@       output! \in \finset BDD4R1 \land \lnot input1? = \{\} \land \lnot input2? = \{\} \land \\

        (\forall out: output!; in1: input1?; in2: input2? | in1.sNo = in2.sNo

        @       (out.sNo = in1.sNo \land out.pNo = in1.pNo \land \\

                out.spQuantity = in1.spQuantity \land \\

                out.sName = in2.sName \land \\

                out.sStatus = in2.sStatus \land out.sCity = in2.sCity)))
```

การทดสอบโดยการสร้างเงื่อนไขก่อน ของ BDD4Op2Select ด้วยคำสั่ง

try $\forall$ *input1? :* $\mathbb{F}$ *BDD4R1; pNoValue? : PNO • pre BDD4Op2Select;*
เมื่อใช้คำสั่ง prove by reduce ได้ผลดังนี้

```
input1? \in \finset BDD4R1 \land pNoValue? \in PNO \\

\implies
```

(\exists output!: \power \lblot pNo: PNO, sCity: CITY, sName: SNAME, sNo: SNO,

sStatus: \num, spQuantity: \num \rblot

@ output! \in \finset BDD4R2 \land \lnot input1? = \{\} \land \\

(\forall out: output!; in1: input1? | in1.pNo = pNoValue? @ out = in1))

การทดสอบโดยการสร้างเงื่อนไขก่อน ของ BDD4Op3Exists ด้วยคำสั่ง

try ∀ input1? : 𝔽 BDD4R1; pNoValue? : PNO • pre BDD4Op2Select;

เมื่อใช้คำสั่ง prove by reduce ได้ผลดังนี้

input1? \in \finset Supplier \land input2? \in \finset BDD4R2 \\

\implies

(\exists output!: \power \lblot pNo: PNO, sCity: CITY, sName: SNAME, sNo: SNO,

sStatus: \num, spQuantity: \num \rblot

@ output! \in \finset BDD3R3 \land \lnot input1? = \{\} \land \lnot input2? = \{\} \land \\

(\forall out: output!; in1: input1?; in2: input2? | in1.sName = in2.sName @ out = in2))

การทดสอบโดยการสร้างเงื่อนไขก่อน ของ BDD4Op4Project ด้วยคำสั่ง

try ∀ input1? : 𝔽 BDD4R3 • pre BDD4Op4Project;

เมื่อใช้คำสั่ง prove by reduce ได้ผลดังนี้

input1? \in \finset Supplier \\

\implies

(\exists output!: \power \lblot sName: SNAME \rblot

@ output! \in \finset BDD3R4 \land \lnot input1? = \{\} \land \\

(\forall out: output!; in1: input1? @ out.sName = in1.sName))

การทดสอบ BDD1 ด้วยคำสั่ง

try BDD4 [op1input1? := SPData, op1input2? := SupplierData, op2pNoValue? := P2,

op3input1? := Supplierdata];

เมื่อใช้คำสั่ง prove by reduce ได้ผลดังนี้

SPData \in \power \lblot pNo: PNO, sNo: SNO, spQuantity: \num \rblot \land \\

SupplierData \in \power \lblot sCity: CITY, sName: SNAME, sNo: SNO, sStatus: \num \rblot \\

\land op1output! \in \power \lblot pNo: PNO, sCity: CITY, sName: SNAME, sNo: SNO,

sStatus: \num, spQuantity: \num \rblot \land \\

op2input1? \in \power \lblot pNo: PNO, sCity: CITY, sName: SNAME, sNo: SNO,

sStatus: \num, spQuantity: \num \rblot \land \\

op2output! \in \power \lblot pNo: PNO, sCity: CITY, sName: SNAME, sNo: SNO,

sStatus: \num, spQuantity: \num \rblot \land P2 \in PNO \land \\

Supplierdata \in \power \lblot sCity: CITY, sName: SNAME, sNo: SNO, sStatus: \num \rblot \\

\land op3input2? \in \power \lblot pNo: PNO, sCity: CITY, sName: SNAME, sNo: SNO,

```
        sStatus: \num, spQuantity: \num \rblot \land \\
op3output! \in \power \lblot pNo: PNO, sCity: CITY, sName: SNAME, sNo: SNO,
        sStatus: \num, spQuantity: \num \rblot \land \\
op4input1?  \in \power \lblot sCity: CITY, sName: SNAME, sNo: SNO, sStatus: \num \rblot \\
\land op4output! \in \power \lblot sName: SNAME \rblot \\
\land SPData \in \finset SP \land SupplierData \in \finset Supplier \\
\land op1output! \in \finset BDD3R1 \land op2input1? \in \finset BDD3R1 \\
\land op2output! \in \finset BDD3R2 \land Supplierdata \in \finset Supplier \\
\land op3input2? \in \finset BDD3R2 \land op3output! \in \finset BDD3R3 \\
\land op4input1? \in \finset Supplier \land op4output! \in \finset BDD3R4 \\
\land \lnot SPData = \{\} \land \lnot SupplierData = \{\} \land \lnot op2input1? = \{\} \\
\land \lnot Supplierdata = \{\} \land \lnot op3input2? = \{\} \land \lnot op4input1? = \{\} \\
\land (out \in op1output! \land in1 \in SPData \land in2 \in SupplierData \land \\
        in1.sNo = in2.sNo \\
\implies
out.sNo = in1.sNo \land out.pNo = in1.pNo \land out.spQuantity = in1.spQuantity \\
        \land out.sName = in2.sName \land out.sStatus = in2.sStatus \\
        \land out.sCity = in2.sCity) \land
(out\_\_0 \in op2output! \land in1\_\_0 \in op2input1? \land in1\_\_0.pNo = P2 \\
\implies
out\_\_0 = in1\_\_0) \land (out\_\_1 \in op3output! \land in1\_\_1 \in Supplierdata \\
        \land in2\_\_0 \in op3input2? \land in1\_\_1.sName = in2\_\_0.sName \\
\implies
out\_\_1 = in2\_\_0) \land (out\_\_2 \in op4output! \land in1\_\_2 \in op4input1? \\
\implies
out\_\_2.sName = in1\_\_2.sName)
```

กรณีทดสอบที่ 5 มีรายละเอียดแฟ้มข้อความของแผนภาพเชิงลำดับชั้นของภาษาเอสคิวแอล ดังนี้

```
%BDDBEGIN
1:Delete:Project:-:-:Success:NotFound
%BDDEND
```

ข้อกำหนดเซตของแผนภาพเชิงลำดับชั้นของภาษาเอสคิวแอล มีดังนี้

```
\begin{schema} {BDD5Op2Delete}
        \Delta SPJExt \\
        spjKey? : SNO \cross PNO \cross JNO \\
```

```
        delSPJ? : SPJ \\
\where
        spjKey? \in \dom spjAtt \\
        spjSet' = spjSet \setminus \{ delSPJ? \} \\
\end{schema}

\begin{schema} {BDD5Op1Delete}
        \Delta ProjectExt \\
        projectKey? : JNO \\
        delProject? : Project \\
        BDD12Op2Delete \\
\where
        projectKey? \in \dom projectAtt \\
        delSPJ?.jNo = delProject?.jNo \\
        projectSet' = projectSet \setminus \{ delProject? \} \\
\end{schema}

\begin{zed}
        BDD5 \defs \\
        BDD5Op1Delete [op1projectKey?/projectKey?, op1delProject?/delProject?] \semi \\
        BDD5Op2Delete [op2spjKey?/spjKey?, op2delSPJ?/delSPJ?] \\
\end{zed}
```

การทดสอบโดยการสร้างเงื่อนไขก่อน ของ BDD5Op1Delete ด้วยคำสั่ง

try $\forall$ *SPJExt; spjKey? : SNO* $\times$ *PNO* $\times$ *JNO; delSPJ? : SPJ • pre BDD5Op2Delete;*

เมื่อใช้คำสั่ง prove by reduce ได้ผลดังนี้

```
spjSet \in \finset SPJ \land spjKey \in \finset (SNO \cross PNO \cross JNO) \\
                \land spjAtt \in SNO \cross PNO \cross JNO \pfun SPJ \\
                 \land spjKey? \in SNO \cross PNO \cross JNO \\
                \land delSPJ? \in SPJ \land \\
                (\forall SPJ1: SPJ; SPJ2: SPJ
                |       SPJ1 \in spjSet \land SPJ2 \in spjSet \land \lnot SPJ1 = SPJ2
                        \land \lnot (SPJ1.jNo \in JNO \land SPJ2.jNo \in JNO \\
                        \land \lnot SPJ1.jNo = SPJ2.jNo) \land \\
                        \lnot (SPJ1.pNo \in PNO \land SPJ2.pNo \in PNO \\
                        \land \lnot SPJ1.pNo = SPJ2.pNo)
                @       (SPJ1.sNo \in SNO \land SPJ2.sNo \in SNO \land \\
                        \lnot SPJ1.sNo = SPJ2.sNo)) \land \\
                        (\forall SPJVar2: SPJ @
                        (\exists ProjectVar1: Project @
                                SPJVar2.jNo = ProjectVar1.jNo)) \land \\
                        (\forall SPJVar2\_\_0: SPJ @
                        (\exists PartVar1: Part @ SPJVar2\_\_0.pNo = PartVar1.pNo))
```

```
                \land
        (\forall SPJVar2\_\_1: SPJ @
        (\exists SupplierVar1: Supplier @
                SPJVar2\_\_1.sNo = SupplierVar1.sNo)) \\
                \land \dom spjAtt \in \power spjKey \\
\implies
(\exists spjAtt': \power ((SNO \cross PNO \cross JNO) \cross \lblot jNo: JNO,
        pNo: PNO, sNo: SNO, spjQuantity: \num \rblot)
@       (\exists spjKey': \power (SNO \cross PNO \cross JNO)
        @       spjKey' \in \finset (SNO \cross PNO \cross JNO) \land \\
                spjAtt' \in SNO \cross PNO \cross JNO \pfun SPJ \land \\
                spjKey? \in \dom spjAtt \land \\
                (\forall SPJ1\_\_0: SPJ; SPJ2\_\_0: SPJ
                |       SPJ1\_\_0 \in spjSet \land SPJ2\_\_0 \in spjSet \land \\
                        \lnot SPJ1\_\_0 = delSPJ? \land \\
                        \lnot SPJ2\_\_0 = delSPJ? \land \\
                        \lnot SPJ1\_\_0 = SPJ2\_\_0 \land \\
                        \lnot (SPJ1\_\_0.jNo \in JNO \land \\
                        SPJ2\_\_0.jNo \in JNO \land \\
                        \lnot SPJ1\_\_0.jNo = SPJ2\_\_0.jNo) \land \\
                        \lnot (SPJ1\_\_0.pNo \in PNO \land \\
                        SPJ2\_\_0.pNo \in PNO \land \\
                        \lnot SPJ1\_\_0.pNo = SPJ2\_\_0.pNo)
                @       (SPJ1\_\_0.sNo \in SNO \land \\
                        SPJ2\_\_0.sNo \in SNO \land \\
                        \lnot SPJ1\_\_0.sNo = SPJ2\_\_0.sNo)) \land \\
                        \dom spjAtt' \in \power spjKey'))
```

การทดสอบโดยการสร้างเงื่อนไขก่อน ของ BDD5Op2Delete ด้วยคำสั่ง

try $\forall$ *ProjectExt; projectKey? : JNO; delProject? : Project • pre BDD5Op1Delete;*

เมื่อใช้คำสั่ง prove by reduce ได้ผลดังนี้

```
projectSet \in \finset Project \land projectKey \in \finset JNO \land \\
        projectAtt \in JNO \pfun Project \land projectKey? \in JNO \land \\
        delProject? \in Project \land \\
        (\forall Project1: Project; Project2: Project
        |       Project1 \in projectSet \land Project2 \in projectSet \land \\
```

```
                \lnot Project1 = Project2
        @       (Project1.jNo \in JNO \land Project2.jNo \in JNO \land \\
                \lnot Project1.jNo = Project2.jNo)) \land \\
                \dom projectAtt \in \power projectKey \\
\implies
(\exists projectAtt': \power (JNO \cross \lblot jCity: CITY, jName: JNAME, No: JNO \rblot) @
(\exists projectKey': \power JNO @
(\exists spjAtt': \power ((SNO \cross PNO \cross JNO) \cross \lblot jNo: JNO, pNo: PNO,
        sNo: SNO, spjQuantity: \num \rblot) @
(\exists spjKey': \power (SNO \cross PNO \cross JNO) @
        projectKey' \in \finset JNO \land projectAtt' \in JNO \pfun Project \land \\
        spjSet \in \finset SPJ \land spjKey \in \finset (SNO \cross PNO \cross JNO) \land \\
        spjAtt \in SNO \cross PNO \cross JNO \pfun SPJ \land \\
        spjKey' \in \finset (SNO \cross PNO \cross JNO) \land \\
        spjAtt' \in SNO \cross PNO \cross JNO \pfun SPJ \land \\
        spjKey? \in SNO \cross PNO \cross JNO \land \\
        delSPJ? \in SPJ \land spjKey? \in \dom spjAtt \land \\
        projectKey? \in \dom projectAtt \land \\
(\forall Project1\_\_0: Project; Project2\_\_0: Project
|       Project1\_\_0 \in projectSet \land Project2\_\_0 \in projectSet \land \\
        \lnot Project1\_\_0 = delProject? \land \lnot Project2\_\_0 = delProject? \land \\
        \lnot Project1\_\_0 = Project2\_\_0
@       Project1\_\_0.jNo \in JNO) \land \dom projectAtt' \in \power projectKey' \land \\
(\forall SPJ1: SPJ; SPJ2: SPJ
|       SPJ1 \in spjSet \land SPJ2 \in spjSet \land \lnot SPJ1 = SPJ2 \land \\
        \lnot (SPJ1.jNo \in JNO \land SPJ2.jNo \in JNO \land
        \lnot SPJ1.jNo = SPJ2.jNo) \land \\
        \lnot (SPJ1.pNo \in PNO \land SPJ2.pNo \in PNO \\
        \land \lnot SPJ1.pNo = SPJ2.pNo)
@       (SPJ1.sNo \in SNO \land SPJ2.sNo \in SNO \land \lnot SPJ1.sNo = SPJ2.sNo)) \land
(\forall SPJVar2: SPJ @
(\exists ProjectVar1: Project @ SPJVar2.jNo = ProjectVar1.jNo)) \land \\
(\forall SPJVar2\_\_0: SPJ @
(\exists PartVar1: Part @ SPJVar2\_\_0.pNo = PartVar1.pNo))  \land
```

```
(\forall SPJVar2\_\_1: SPJ @
(\exists SupplierVar1: Supplier @
        SPJVar2\_\_1.sNo = SupplierVar1.sNo)) \land \\
        \dom spjAtt \in \power spjKey \land \\
(\forall SPJ1\_\_0: SPJ; SPJ2\_\_0: SPJ
|       SPJ1\_\_0 \in spjSet \land SPJ2\_\_0 \in spjSet \land \\
        \lnot SPJ1\_\_0 = delSPJ? \land \lnot SPJ2\_\_0 = delSPJ? \land \\
        \lnot SPJ1\_\_0 = SPJ2\_\_0 \land \lnot (SPJ1\_\_0.jNo \in JNO \land \\
        SPJ2\_\_0.jNo \in JNO \land \lnot SPJ1\_\_0.jNo = SPJ2\_\_0.jNo) \land \\
        \lnot (SPJ1\_\_0.pNo \in PNO \land SPJ2\_\_0.pNo \in PNO \land \\
        \lnot SPJ1\_\_0.pNo = SPJ2\_\_0.pNo)
@       (SPJ1\_\_0.sNo \in SNO \land SPJ2\_\_0.sNo \in SNO \land \\
        \lnot SPJ1\_\_0.sNo = SPJ2\_\_0.sNo)) \land \\
        \dom spjAtt' \in \power spjKey' \land \\
        delSPJ?.jNo = delProject?.jNo))))
```

การทดสอบ BDD5 ด้วยคำสั่ง

try BDD5 [op1projectKey? := ProjectKeyData, op1delProject? := ProjectData,
op2spjKey? := SPJKeyData, op2delSPJ? := SPJData];

เมื่อใช้คำสั่ง prove by reduce ได้ผลดังนี้

```
delSPJ? \in \lblot jNo: JNO, pNo: PNO, sNo: SNO, spjQuantity: \num \rblot \land \\
ProjectData \in \lblot jCity: CITY, jName: JNAME, jNo: JNO \rblot \land \\
ProjectKeyData \in JNO \land \\
SPJData \in \lblot jNo: JNO, pNo: PNO, sNo: SNO, spjQuantity: \num \rblot \\
        \land SPJKeyData \in SNO \cross PNO \cross JNO \land \\
projectAtt  \in \power (JNO \cross \lblot jCity: CITY, jName: JNAME, jNo: JNO \rblot) \land \\
projectAtt' \in \power (JNO \cross \lblot jCity: CITY, jName: JNAME, jNo: JNO \rblot) \land \\
projectKey \in \power JNO \land projectKey' \in \power JNO \land \\
projectSet \in \power \lblot jCity: CITY, jName: JNAME, jNo: JNO \rblot \land \\
projectSet' \in \power \lblot jCity: CITY, jName: JNAME, jNo: JNO \rblot \land \\
spjAtt \in \power ((SNO \cross PNO \cross JNO) \cross \lblot jNo: JNO, pNo: PNO,
        sNo: SNO, spjQuantity: \num \rblot) \land \\
spjAtt' \in \power ((SNO \cross PNO \cross JNO) \cross \lblot jNo: JNO, pNo: PNO,
        sNo: SNO, spjQuantity: \num \rblot) \land \\
spjKey \in \power (SNO \cross PNO \cross JNO) \land \\
spjKey' \in \power (SNO \cross PNO \cross JNO) \land \\
```

```
spjKey? \in SNO \cross PNO \cross JNO \land \\
spjSet \in \power \lblot jNo: JNO, pNo: PNO, sNo: SNO, spjQuantity: \num \rblot \land \\
spjSet' \in \power \lblot jNo: JNO, pNo: PNO, sNo: SNO, spjQuantity: \num \rblot \land \\
(\exists spjAtt\_\_0: \power ((SNO \cross PNO \cross JNO) \cross \lblot jNo: JNO, pNo: PNO,
        sNo: SNO, spjQuantity: \num \rblot) @
(\exists spjKey\_\_0: \power (SNO \cross PNO \cross JNO) @
        (projectSet \in \finset Project \land projectKey \in \finset JNO \\
        \land projectAtt \in JNO \pfun Project \land ProjectData \in Project \\
        \land projectSet' = projectSet \setminus \{ProjectData\} \\
        \land projectKey' \in \finset JNO \land projectAtt' \in JNO \pfun Project \\
        \land spjSet \in \finset SPJ \land spjKey \in \finset (SNO \cross PNO \cross JNO) \\
        \land spjAtt \in SNO \cross PNO \cross JNO \pfun SPJ \\
        \land spjKey\_\_0 \in \finset (SNO \cross PNO \cross JNO) \\
        \land spjAtt\_\_0 \in SNO \cross PNO \cross JNO \pfun SPJ \\
        \land delSPJ? \in SPJ \land spjKey? \in \dom spjAtt \\
        \land ProjectKeyData \in \dom projectAtt \land SPJData \in SPJ \\
        \land spjSet' = spjSet \setminus (\{delSPJ?\} \cup \{SPJData\}) \\
        \land spjKey' \in \finset (SNO \cross PNO \cross JNO) \\
        \land spjAtt' \in SNO \cross PNO \cross JNO \pfun SPJ \\
        \land SPJKeyData \in \dom spjAtt\_\_0 \land \\
(\forall Project1: Project; Project2: Project
|       Project1 \in projectSet \land Project2 \in projectSet \land \lnot Project1 = Project2
@       (Project1.jNo \in JNO \land Project2.jNo \in JNO \land \lnot
        Project1.jNo = Project2.jNo)) \land \dom projectAtt \in \power projectKey \land \\
(\forall Project1\_\_0: Project; Project2\_\_0: Project
|       Project1\_\_0 \in projectSet \land Project2\_\_0 \in projectSet \\
        \land \lnot Project1\_\_0 = ProjectData \land \lnot Project2\_\_0 = ProjectData \\
        \land \lnot Project1\_\_0 = Project2\_\_0
@       Project1\_\_0.jNo \in JNO) \land \dom projectAtt' \in \power projectKey' \land \\
(\forall SPJ1: SPJ; SPJ2: SPJ
|       SPJ1 \in spjSet \land SPJ2 \in spjSet \land \lnot SPJ1 = SPJ2 \land \\
        \lnot (SPJ1.jNo \in JNO \land SPJ2.jNo \in JNO \land \land \lnot
        SPJ1.jNo = SPJ2.jNo) \land \lnot (    SPJ1.pNo \in PNO \land SPJ2.pNo \in PNO \\
        \land \lnot SPJ1.pNo = SPJ2.pNo)
```

```
@       (SPJ1.sNo \in SNO \land SPJ2.sNo \in SNO \land \lnot SPJ1.sNo = SPJ2.sNo)) \\
        \land (\forall SPJVar2: SPJ @
(\exists ProjectVar1: Project @ SPJVar2.jNo = ProjectVar1.jNo)) \land \\
(\forall SPJVar2\_\_0: SPJ @
(\exists PartVar1: Part @ SPJVar2\_\_0.pNo = PartVar1.pNo)) \land
(\forall SPJVar2\_\_1: SPJ @
(\exists SupplierVar1: Supplier @ SPJVar2\_\_1.sNo = SupplierVar1.sNo)) \land
        \dom spjAtt \in \power spjKey \land \\
(\forall SPJ1\_\_0: SPJ; SPJ2\_\_0: SPJ
|       SPJ1\_\_0 \in spjSet \land SPJ2\_\_0 \in spjSet \land \lnot SPJ1\_\_0 = delSPJ? \\
        \land \lnot SPJ2\_\_0 = delSPJ? \land \lnot SPJ1\_\_0 = SPJ2\_\_0 \\
        \land \lnot (SPJ1\_\_0.jNo \in JNO \land SPJ2\_\_0.jNo \in JNO \\
        \land \lnot SPJ1\_\_0.jNo = SPJ2\_\_0.jNo) \land \lnot (SPJ1\_\_0.pNo \in PNO \\
        \land SPJ2\_\_0.pNo \in PNO \land \lnot SPJ1\_\_0.pNo = SPJ2\_\_0.pNo)
@       (SPJ1\_\_0.sNo \in SNO \land SPJ2\_\_0.sNo \in SNO \land \\
        \lnot SPJ1\_\_0.sNo = SPJ2\_\_0.sNo)) \land \\
        \dom spjAtt\_\_0 \in \power spjKey\_\_0 \land delSPJ?.jNo = ProjectData.jNo \land \\
(\forall SPJ1\_\_1: SPJ; SPJ2\_\_1: SPJ
|       SPJ1\_\_1 \in spjSet \land SPJ2\_\_1 \in spjSet \land \lnot SPJ1\_\_1 = delSPJ? \\
        \land \lnot SPJ1\_\_1 = SPJData \land \lnot SPJ2\_\_1 = delSPJ? \\
        \land \lnot SPJ2\_\_1 = SPJData \land \lnot SPJ1\_\_1 = SPJ2\_\_1 \\
        \land \lnot (SPJ1\_\_1.jNo \in JNO \land SPJ2\_\_1.jNo \in JNO \\
        \land \lnot SPJ1\_\_1.jNo = SPJ2\_\_1.jNo) \land \lnot (     SPJ1\_\_1.pNo \in PNO \\
        \land SPJ2\_\_1.pNo \in PNO \land \lnot SPJ1\_\_1.pNo = SPJ2\_\_1.pNo)
@       (SPJ1\_\_1.sNo \in SNO \land SPJ2\_\_1.sNo \in SNO \\
        \land \lnot SPJ1\_\_1.sNo = SPJ2\_\_1.sNo)) \\
        \land \dom spjAtt' \in \power spjKey')))
```

ภาคผนวก ง

คู่มือการใช้งาน

เครื่องมือซอฟต์แวร์ R2Z2 เป็นเครื่องมือในการสร้างข้อกำหนดเซตจากแผนภาพเอนทิตีและความสัมพันธ์ และแผนภาพเชิงลำดับชั้นของภาษาเอสคิวแอล

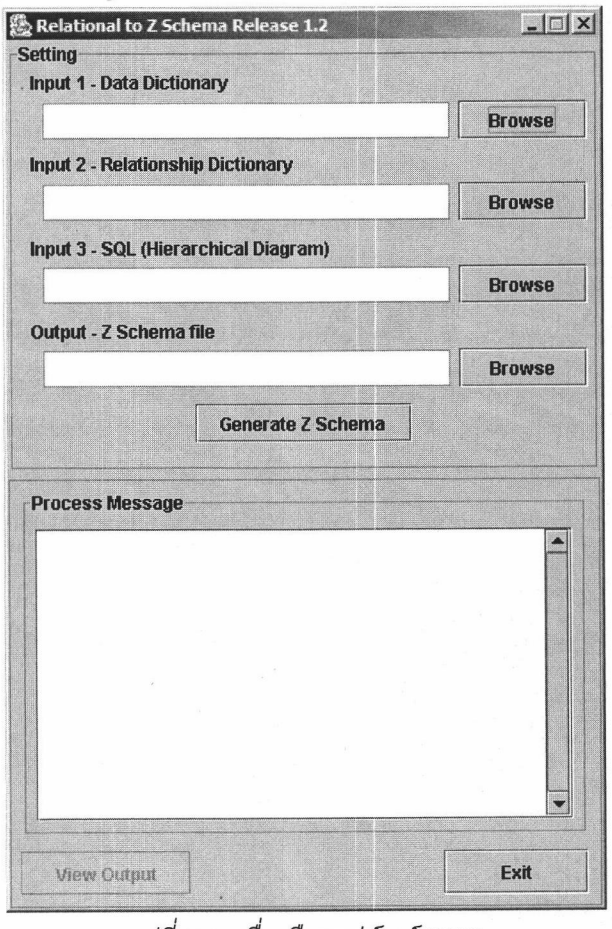
วิธีการใช้เครื่องมือซอฟต์แวร์ แบ่งได้เป็น 4 ส่วน ดังนี้

5.1 การเรียกใช้งานโปรแกรม

การเรียกใช้งานโปรแกรม สามารถเรียกใช้งานได้โดยใช้คำสั่ง java R2Z2 ดังตัวอย่าง

```
C:\class directory\java R2Z2
```

โดย class directory คือ ไดเรกทอรีที่เก็บคลาสของเครื่องมือซอฟต์แวร์ที่ผ่านการคอมไพล์แล้ว (R2Z2.class) จะปรากฏหน้าจอดังรูปที่ 5.1

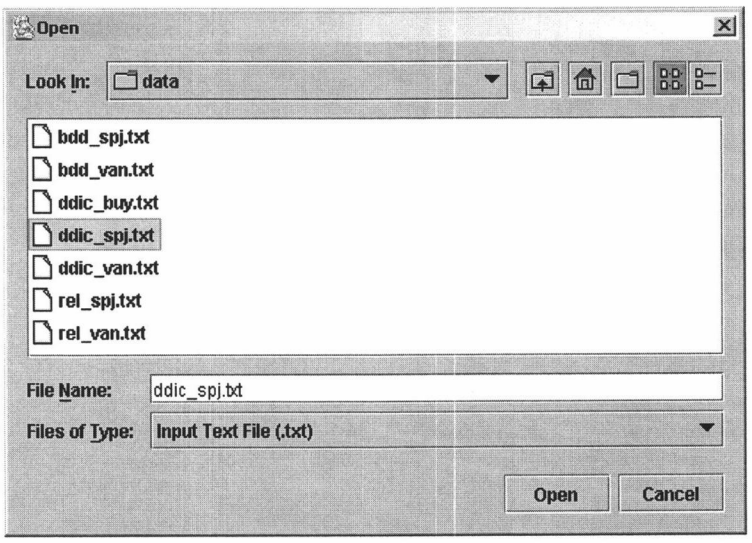


รูปที่ 5.1 เครื่องมือซอฟต์แวร์ R2Z2

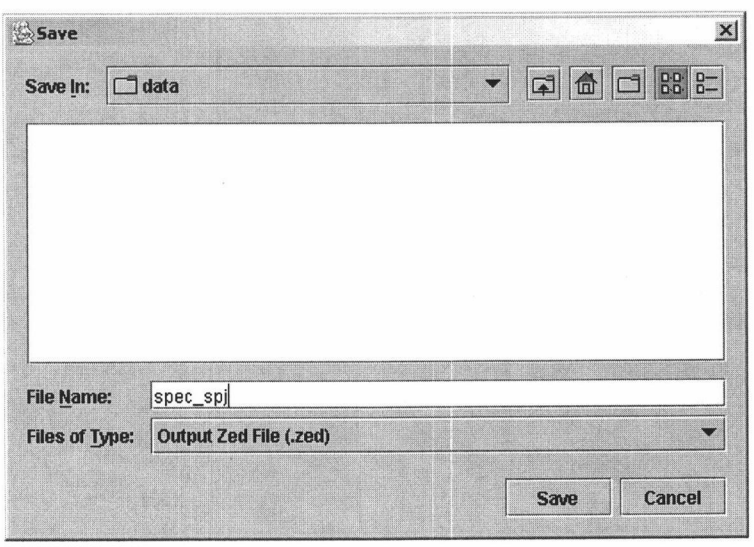
5.2 ข้อมูลเข้าและข้อมูลผลลัพธ์

จากรูปที่ 5.1 ผู้ใช้ต้องกำหนดแฟ้มข้อมูลเข้าทั้ง 3 แฟ้ม และแฟ้มผลลัพธ์ 1 แฟ้ม ดังนี้

- Input 1 - Data Dictionary คือ ชื่อแฟ้มข้อมูลเข้าที่มาจากพจนานุกรมข้อมูลของแผนภาพเอนทิตีและความสัมพันธ์ มีนามสกุลเป็น .txt เมื่อกดปุ่ม Browse จะแสดงหน้าจอรายชื่อแฟ้มข้อมูลเข้าดังรูปที่ 5.2
- Input 2 - Relationship Dictionary คือ ชื่อแฟ้มข้อมูลเข้าที่มาจากพจนานุกรมความสัมพันธ์ของแผนภาพเอนทิตีและความสัมพันธ์ มีนามสกุลเป็น .txt เมื่อกดปุ่ม Browse จะแสดงหน้าจอรายชื่อแฟ้มข้อมูลดังรูปที่ 5.2
- Input 3 - SQL (Hierarchical Diagram) คือ ชื่อแฟ้มข้อมูลเข้าที่มาจากแผนภาพเชิงลำดับชั้นของภาษาเอสคิวแอล มีนามสกุลเป็น .txt เมื่อกดปุ่ม Browse จะแสดงหน้าจอรายชื่อแฟ้มข้อมูลดังรูปที่ 5.2
- Output - Z Schema file คือ ชื่อแฟ้มข้อมูลสำหรับเก็บผลลัพธ์ข้อกำหนดเซตที่ได้จากเครื่องมือซอฟต์แวร์นี้ มีนามสกุลเป็น .zed เมื่อกดปุ่ม Browse จะแสดงหน้าจอให้ระบุชื่อแฟ้มข้อมูลผลลัพธ์ ดังรูปที่ 5.3

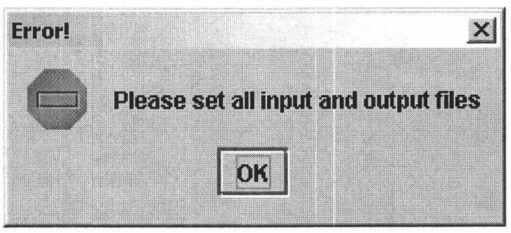


รูปที่ 5.2 รายชื่อแฟ้มข้อมูลเข้า



รูปที่ 5.3 ระบุชื่อแฟ้มข้อมูลผลลัพธ์

หากผู้ใช้ระบุชื่อแฟ้มข้อมูลเข้าหรือแฟ้มข้อมูลผลลัพธ์ไม่ครบทั้ง 4 แฟ้ม เมื่อกดปุ่ม Generate Z Schema จะแสดงข้อความเตือน ดังรูปที่ 5.4

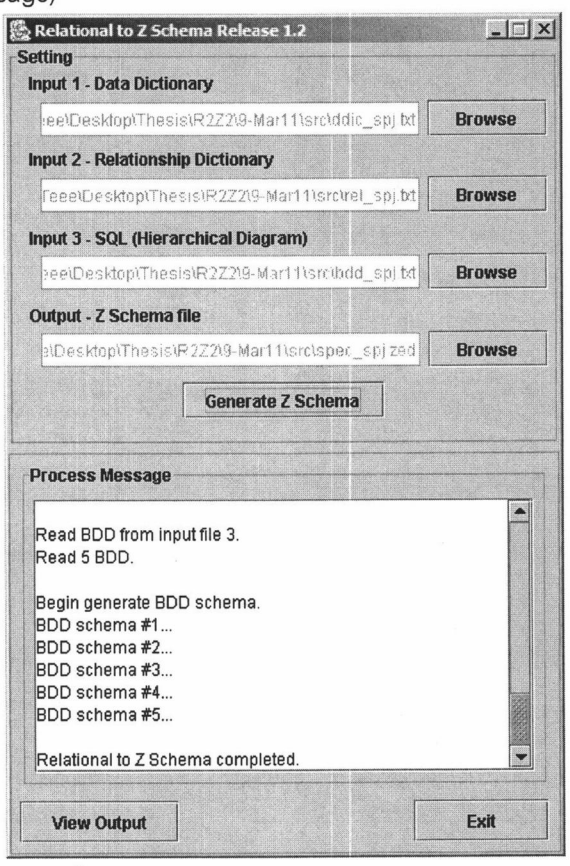


รูปที่ 5.4 ข้อความเตือนเมื่อระบุชื่อแฟ้มข้อมูลเข้าและแฟ้มข้อมูลผลลัพธ์ไม่ครบ

5.3 การสร้างข้อกำหนดเซต

เมื่อกำหนดแฟ้มข้อมูลเข้าและแฟ้มข้อมูลผลลัพธ์เรียบร้อยแล้ว ผู้ใช้สามารถทำการสร้างข้อกำหนดเซตจากข้อมูลเข้าทั้ง 3 แฟ้มที่เลือกไว้ โดยกดปุ่ม Generate Z Schema ดังรูปที่ 5.5

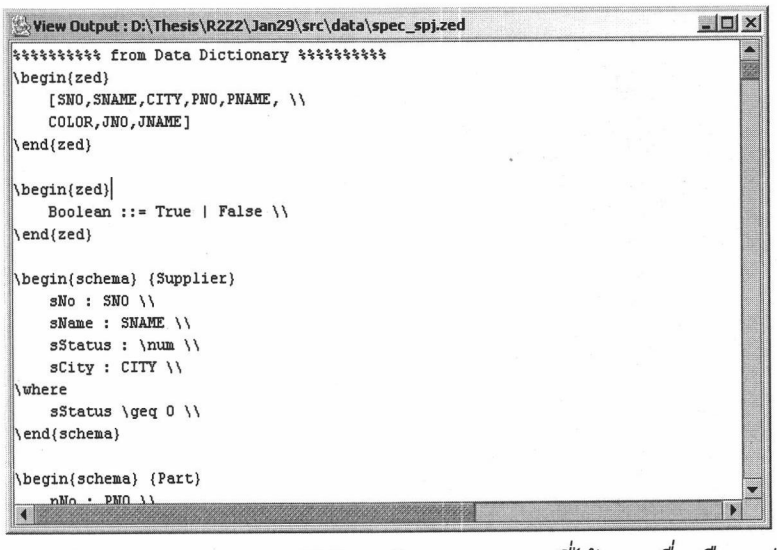
เครื่องมือซอฟต์แวร์จะทำการสร้างข้อกำหนดเซต และแสดงผลการทำงานในช่องข้อความแสดงการประมวลผล (Process Message)



รูปที่ 5.5 ผลการทำงานในการสร้างข้อกำหนดเซต

4.4.4 การดูแฟ้มข้อมูลผลลัพธ์

เมื่อเครื่องมือซอฟต์แวร์ทำการประมวลผลเพื่อสร้างข้อกำหนดเซตเรียบร้อยแล้ว ผู้ใช้สามารถดูข้อกำหนดเซตที่ได้จากแฟ้มข้อมูลผลลัพธ์ที่กำหนดไว้ โดยกดปุ่ม View Output ดังรูปที่ 5.6



รูปที่ 5.6 ข้อกำหนดเซตของคุณสมบัติเชิงพฤติกรรมของระบบที่ได้จากเครื่องมือซอฟต์แวร์

ผู้ใช้สามารถปิดหน้าต่างนี้เมื่อใช้งานเสร็จ และหากต้องการจบการทำงานของเครื่องมือซอฟต์แวร์ ให้กดปุ่ม Exit

ประวัติผู้เขียนวิทยานิพนธ์

นางสาวชนาเนตร อรรถยุกติ เกิดวันที่ 11 กรกฎาคม พ.ศ.2514 ที่จังหวัดกรุงเทพ สำเร็จการศึกษาหลักสูตรบริหารธุรกิจบัณฑิต (บธ.บ) สาขาระบบสารสนเทศ คณะบริหารธุรกิจ สถาบันเทคโนโลยีราชมงคล เมื่อปีการศึกษา 2537 จากนั้นได้เข้าทำงานที่บริษัทเงินทุนหลักทรัพย์วอลล์สตรีท จำกัด (มหาชน) และบริษัทลอจิก จำกัด ต่อมาในปีการศึกษา 2543 ได้เข้าศึกษาต่อหลักสูตรวิทยาศาสตรมหาบัณฑิต (วท.ม.) สาขาวิชาวิทยาศาสตร์คอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย