**Л8. Основі реляційного числення**

Реляційне числення грунтується на механізмі числення предикатів першого порядку. *Реляційне числення*  це система позначень для отримання необхідного відношення в термінах даних відношень. Реляційна алгебра і реляційне числення відрізняються тільки зовні, насправді вони еквівалентні. Проте мови реляційного числення – це не процедурні мови, оскільки їх засобами можна виразити все, що необхідне і необов'язково вказувати, як це отримати. Вираз в численні описує лише властивості бажаного результату, фактично не вказуючи, як його отримати. Вирази реляційної алгебри, навпаки, визначають конкретний порядок виконання операцій.

Основними поняттями числення є поняття  *змінної*  з деякою областю допустимих значень і поняття  *правильно побудованої формули* (WFF –well formulated formula), що спирається на предикати, змінні і квантори.

В залежності від області визначення змінної розрізняють  *числення кортежів*  і  *числення доменів*. У **численні кортежів** областю визначення змінних є відношення бази даних, тобто допустимим значенням кожної змінної є кортеж деякого відношення. У **численні доменів**  областю визначення змінних є домени, на яких визначені атрибути відношення бази даних, тобто допустимим значенням кожної змінної є значення деякого домена.

***Реляційне числення із змінними кортежів***

Змінна кортежу (або області значень) визначається за допомогою синтаксису мови QUEL (**QUEL** - це мова запитів до **реляційних**  баз даних, яка базується на **реляційному** численні кортежів та яка подібна до SQL https://intellect.icu/obshhaya-kharakteristika-yazyka-quel-yazyk-programmirovaniya-equel-2537) таким чином:

RANGE OF *T* IS *X1*, *X2 ., Xn*.

Тут *Т –* змінна кортежів, яка *визначається*,  *Xi* (*i* = 1, 2 ., *n*) – або ім'я відношення, або вираз числення кортежів. Нехай *Xi* є відношенням *Ri* (*i* = 1, 2 ., *n*). Відношення *R1*, *R2 ., Rn*  повинні бути сумісні за типом, тобто вони повинні мати ідентичні заголовки. Тоді змінна кортежів  *Т*  змінюється на об'єднанні цих відношень, тобто її значення в будь-який заданий час буде деяким поточним кортежем принаймні одного з цих відношень. Звичайно, якщо список ідентифікаторів виразів буде просто одним іменованим відношенням  *R*  (це звичайний випадок), то змінна кортежу просто прийматиме значення поточних кортежів одного такого відношення *R*.

При використанні змінних кортежів у формулах можна посилатися на значення атрибуту змінною (це аналогічно тому, як, наприклад, при програмуванні на мові С++  можна послатися на значення поля в змінній типу структура). Наприклад, для того, щоб послатися на значення атрибуту *A*  відношення, значення якого приймає змінна  *T*, потрібно використати конструкцію *T.A*.

Якщо компоненти кортежу можуть бути ідентифіковані за їх порядковою позицію, то можна посилатися на них за допомогою індексного посилання *T[i].*

Правильно побудована формула WFF  служить для виразу умов, що накладаються на змінні кортежів. WFF  складається з простих порівнянь скалярних значень (значень атрибутів змінних або літерально заданих констант). Складніші варіанти  WFF  будуються за допомогою логічних операцій  NOT, AND, OR, IF.THEN, і двох кванторів EXISTS  і FORALL.

Квантор EXISTS називається *квантором* *існування*,

квантор FORALL *квантором* *спільності*.

Якщо *f*  формула WFF, в якій бере участь змінна *x*, то EXISTS  *x* ( *f* ) і FORALL *x* ( *f* ) є допустимими формулами WFF. Перша формула означає: «Існує принаймні одне значення змінної *x*, що обчислення формули  *f*  для цього  *x*  дає значення істина». Друга формула означає: «Для всіх значень змінної  *x* обчислення формули  *f*  дає значення істина».

Квантор існування EXISTS визначається формально як *OR* (*АБО*)*,* щоповторюється . Тобто, якщо *R*  це відношення з кортежами *Т1*, *Т2 ., Тm*; *Т*  це змінна кортежу, яка визначається та змінюється на цьому відношенні; а *f*( *T*) – це формула, в якій використовується змінна *Т*, то формула

EXISTS *T*(*f (T) )*

є тотожньою наступній формулі WFF:

*false* OR ( *f* (*T1*)) OR … OR ( *f* (*Tm*) ).

*Приклад*. Нехай використовується відношення R:

*R* ( *D*, *F*, *W* )  
  2 3 4  
 2 3 5  
 2 4 5

Тоді  наступні вирази матимуть наведені нижче значення:

EXISTS *T* ( *T.W* > 2) : *true*EXISTS *T* ( *T.F* > 4) : *false*EXISTS *T* ( *T.D* > 2 OR *T.W* >= 5 ) : *true*

Квантор існування FORALL визначається як *AND (ТА),* щоповторюється. Іншими словами, якщо *R*, *T*  і *f (T)* такі ж, як розглядалися вищим, то формула

FORALL *T*( *f*( *T*) )

є тотожньою наступній формулі:

*true* AND ( *f*( *T1*) ) AND ... AND ( *f*( *Tm*) ).

*Приклад*. Якщо відношення *R* містить ті ж самі кортежі, що і в попередньому прикладі, то наведені нижче вирази матимуть такі значення:

FORALL *T* ( *T.D* > 2) : *false*FORALL *T* ( *T.F* > 2) : *true*FORALL *T* ( *T.D* = 2 AND *T.W* > 3) : *true*

Змінні, що входять в WFF, можуть бути *вільними*  або *зв'язаними*. Всі змінні, що входять в WFF, при побудові якої не використовувалися квантори, є **вільними**. Фактично, це означає, що якщо для будь–якого набору значень вільних змінних кортежів при обчисленні WFF набуте значення *true*, то ці значення змінних кортежів можуть входити в результуюче відношення. Нехай *f*  формула WFF, в якій змінна  *x* вільна. Якщо ім'я змінної  *x* використане відразу після квантора при побудові WFF виду EXISTS *x* ( *f* ) або FORALL *x* ( *f* ), то в цій WFF і у всіх WFF, побудованих з її участю,  *x -* зв'язана  змінна. Це означає, що таку змінну не видно за межами мінімальної WFF, що зв'язала цю змінну. При обчисленні значення такої WFF використовується не одне значення зв'язаної змінної, а вся її область визначення.

Нехай *T*  і *K*  дві змінні кортежів , визначені на відношенні *R*. Тоді, WFF EXISTS *K* (*T.A* > *K.A*) для поточного кортежу змінної *T*  приймає значення *true*  тоді і лише тоді, якщо у всьому відношенні *R*знайдеться кортеж (пов'язаний зі змінною K*)* такий, що значення його атрибуту *A* задовольняє внутрішній умові порівняння. Формула WFF FORALL *K (T.A > K.A)* для поточного кортежу змінної T приймає значення  *true* в тому і лише в тому випадку, якщо для всіх кортежів відношення  *R*(пов'язаних зі змінною K) значення атрибуту *A* задовольняють умові порівняння.

Таким чином, квантори в реляційному численні грають ту ж роль, що декларації у мові програмування. Поняття вільної змінної аналогічно поняттю глобальної змінної, описаної поза поточною процедурою. Поняття зв'язаної змінної аналогічно поняттю локальної змінної, описаної в поточній процедурі.

Отже, WFF забезпечують засоби формулювання умови вибірки з відношень БД. Щоб можна було використовувати числення для реальної роботи з БД, потрібний ще один компонент, який визначає набір і імена стовпців результуючого відношення. Цей компонент називається  *цільовим* *списком* (*target\_list*).

Цільовий список будується з цільових елементів, кожний з яких може мати наступний вигляд:

.. *T.A*[AS *X*],

де  *T*  - ім'я вільної змінної відповідної WFF, а  *A*  ім'я атрибуту відношення, на якому визначена змінна *T*, а  *X*  це ім'я атрибуту результату обчислення елементів цільового списку.

.. *T*, еквівалентно наявності підсписку *T*.*A1*, *T*.*A2*, ..., *T*.*An*, де *A1*, *A2 ..., An*  включає імена усіх атрибутів визначального відношення;

.. *N* = *T*.*A*; *N* – нове ім'я відповідного атрибуту результуючого відношення.

Останній варіант потрібний у випадках, коли в WFF використовуються декілька вільних змінних з однаковою областю визначення.

**Виразом** реляційного числення кортежів називається конструкція виду

TARGET\_LIST WHERE WFF.

Значенням виразу є відношення, тіло якого визначається WFF, а набір атрибутів та їх імена – цільовим списком.

Як вказувалося вище, для опису реляційного числення кортежів використовується синтаксис реальної мови запитів QUEL.

Якщо використовувати традиційний синтаксис мови предикатів, то опис реляційного числення із змінними кортежів виглядає таким чином:

{*t* | у (t)}

де *t*  - єдина вільна змінна, що позначає кортеж фіксованої довжини (якщо необхідно вказати арность кортежу, то використовують запис *t(i)*; *i* - арность кортежу *t*); *у* - правильно побудована формула (WFF).

На мал. 2.1 представлений огляд розглянутих елементів мови QUEL та їх еквіваленти з мови предикатів.

|  |  |  |  |
| --- | --- | --- | --- |
| **Елемент мови** | **Мова QUEL** | | **Мова предикатів** |
| Кон’юнкція | AND | ∧ | |
| Диз’юнкція | OR | ∨ | |
| Заперечення | NOT | ¬ | |
| Імплікація | IF…THEN | → | |
| Квантор існування | EXISTS | **∃** | |
| Квантор спільності | FORALL | **∀** | |

Мал. 2.1. Елементи синтаксису мови QUEL і мови предикатів

Для основних операцій реляційної алгебри можна побудувати відповідні вирази (мовою QUEL та мовою предикатів) реляційного числення на змінних кортежів.

1. Операції об'єднання (*A*∨*B*  або *A* *union* *B*) відповідають вирази:

|  |  |  |
| --- | --- | --- |
| {*t*| *A*(*t*) ∨ *B*(*t*)} |  | RANGE OF *t* IS *A*, *B* |

Необхідно отримати множину всіх кортежів *t*, причому таких кортежів, які належать відношенню *A*  або *B*.

1. Операції різниці (*A -* *B*  або *A* minus *B*) відповідають вирази:

|  |  |  |
| --- | --- | --- |
| {*t*| *A*(*t*) ∧ ( ¬*B*(*t*))} |  | RANGE OF t IS A; |
|  |  | RANGE OF *r* IS *B*; |
|  |  | *t* WHERE *t* ≠ *r* |

Розглядається  вся множина кортежів  *t*, таке, що  *t*  належить  *A*  і не належить *B*.

3. Операції  декартовий добуток (*A* ⊗*B*  або *A times B*) відповідають вирази:

{*t*(*k+m*)| ( ∃*u*)(∃*v*)(*A(u)*∧ *B(v) ∧ t[1]=u[1]*∧ … ∧ *t[k]=u[k]*∧

∧ *t*[*k+1]=v[1]*∧ … ∧ *t*[*k+m*]=*v[m])}*

RANGE OF *t* IS *A*;

RANGE OF *u* IS *A*;

RANGE OF *r* IS *B*;

RANGE OF *v* IS *B*;

(t,r) WHERE EXISTS u EXISTS v (*t[1]=u[1]* AND

AND *t[k]=u[k]* AND *r[1]=v[1]* AND … AND *r[m]=v[m]*)

Розглядається множина кортежів арности  *k+m*  (*k-*арность відношення  *A*, *m* - арность відношення *B*), таких, що існує кортеж  *u*, належний *A*, і існує кортеж  *v*, що належить *B*, причому *k*  перших компонентів кожного результуючого кортежу утворюють компоненти кортежу  *u*, а наступні  *m*  компонентів результуючого кортежу утворюють компоненти кортежу *v*.

4.  Операції проекції (  або *A*[*i1*, *i2 ., in*])  відповідають вирази:

{*t(n)*| (∃*u*)(*A(u)*∧ *t[1]= u*[*i1*] ∧… ∧ *t[n]= u*[*in*])}

RANGE OF *u* IS *A*;

*t* WHERE EXISTS *u* (*t[1]= u*[*i1*] AND **…** AND *t[n]= u*[*in*])

1. Операції вибірки (  або *A where X*  θ *Y*)  відповідають вирази:

|  |  |  |
| --- | --- | --- |
| {*t* | *A*(*t*) ∧ *X'* θ *Y'*} |  | RANGE OF *t* IS *A*; |
|  |  | *t* WHERE *X'* θ *Y'* |

де *X'* θ *Y'* - це умова *X* θ *Y*, в якій кожен операнд, що позначає компоненту *i*, замінений на *t[i].*

Як і в реляційній алгебрі, решту операцій можна отримати за допомогою наведених п'яти основних операцій.

**Приклади**

У прикладах передбачається, що діють такі змінні області значень:

RANGE OF ПА IS ПОСТАЧАЛЬНИК;

RANGE OF ТА IS ДЕТАЛЬ;

RANGE OF ПДА IS ПД;

RANGE OF ПДБ IS ПД;

RANGE OF ПДВ IS ПД.

* **Отримати номери постачальників з м.Києва зі статусом більше 20**

{*t(1)* | (**∃** *u*) (ПОСТАЧАЛЬНИК(u)∧ t[1]= u[ПНОМ] *u*[МІСТО]= 'Київ' ∧

 ∧*u*[СТАТУС]>20)}

(ПА.ПНОМ) WHERE  ПА.МІСТО='Київ' AND  ПА.СТАТУС>20

* **Отримати імена постачальників, які поставляють деталь Д2**

(ПА.ПФАМ) WHERE EXISTS ПДА ( ПДА.ПНОМ = ПА.ПНОМ

AND ПДА.ДНОМ = 'Д2' )

{*t(1)*  | (∃ *u*) (∃v) (ПД(u)∧ ПОСТАЧАЛЬНИК(v) ∧ t[1]= v[ПФАМ] ∧

∧*u*[ПНОМ]= v[ПНОМ] ∧ u[ДНОМ]= 'Д2')}

* **Отримати імена постачальників, які поставляють принаймні одну червону деталь**

(ПА.ПФАМ) WHERE EXISTS ПДА ( ПА.ПНОМ = ПДА.ПНОМ AND

EXISTS ТАК(ДА.ДНОМ = ПДА.ДНОМ AND ДА.КОЛІР = 'Червоний ))

{*t(1)* | (∃ *u*) (ПОСТАЧАЛЬНИК(u) ∧ t[1]= u[ПФАМ]  ( v)(ПД(v) ∧

∧ *u*[ПНОМ]= v[ПНОМ] ∧ (∃ w)(ДЕТАЛЬ(w) ∧

∧*w*[ДНОМ]= v[ДНОМ] ∧ w[КОЛІР]= 'Червоний')))}

* **Отримати імена постачальників, які постачають принаймні одну деталь, що поставляється постачальником П2**

(ПА.ПФАМ) WHERE EXISTS ПДА ( EXISTS ПДБ

( ПА.ПНОМ = ПДА.ПНОМ AND

ПДА.ДНОМ = ПДБ.ДНОМ AND

ПДБ.ДНОМ = 'П2' ) )

{*t(1)* | (∃ *u*) (ПОСТАЧАЛЬНИК(u) ∧ *t[1]= u*[ПФАМ] ∧ (∃ *v*) (∃ *w*) ∧

∧ (ПД(v) ∧ ПД(w) ∧ *u*[ПНОМ]= *v*[ПНОМ] ∧

∧ *v*[ДНОМ]= *w*[ДНОМ] ∧ *w*[ДНОМ]= 'П2'))}

**-  Отримати імена постачальників, які поставляють всі деталі**

(ПА.ПФАМ) WHERE FORALL ТАК ( EXISTS ПДА

( ПДА.ПНОМ = ПА.ПНОМ AND

ПДА.ДНОМ = ДА.ДНОМ ) )

{*t(1)* | (∃ *u*) (ПОСТАЧАЛЬНИК(u) ∧ *t[1]= u*[ПФАМ] ∧ (∃ *v*) (∃ *w*) ∧

∧(ДЕТАЛЬ(v) ∧ ПД(w) ∧ *w*[ПНОМ]= *u*[ПНОМ]  *w*[ДНОМ]= v[ДНОМ]))}

**-  Отримати імена постачальників, які не поставляють деталь Д2**

(ПА.ПФАМ) WHERE NOT EXISTS ПДА

( ПДА.ПНОМ = ПА.ПНОМ AND ПДА.ДНОМ = 'Д2' )

{*t(1)* | (∃ *u*) (∃ *v*) (ПД(u) ∧ ПОСТАЧАЛЬНИК(v) ∧ *t[1]=* *v*[ПФАМ] ∧

∧ (¬ (*u*[ПНОМ]= *v*[ПНОМ] ∧ *u*[ДНОМ]= 'Д2')))}

**-  Отримати номери постачальників, що поставляють принаймні всі ті деталі, які поставляє постачальник П2**

ПА.ПНОМ WHERE FORALL ПДБ (IF ПДБ.ПНОМ = 'П2' THEN

EXISTS ПДВ ( ПДВ.ПНОМ = ПА.ПНОМ AND

ПДВ.ДНОМ = ПДБ.ДНОМ))

{*t(1)* | (∃ *u*) (∃ *v*) (ПОСТАЧАЛЬНИК(u) ∧ ПД(v) ∧ *t[1]=* *u*[ПНОМ] ∧

∧ (∀ *v*) (*v*[ПНОМ]= 'П2' → (∃ *w*) (ПД(w) ∧ *w*[ПНОМ]= *u*[ПНОМ] ∧

∧ *w*[ДНОМ]= v[ДНОМ])))}

**-  Отримати номери деталей, які або важать більше 16, або поставляються постачальником П2, або і те і інше.**

ДА.ДНОМ WHERE ДА.ВЕС > 16 OR

EXISTS ПДА ( ПДА.ДНОМ = ДА.ДНОМ AND

ПДА.ПНОМ = 'П2' )

{*t(1)* | (∃ *u*) (ДЕТАЛЬ(u) ∧ *t[1]= u*[ДНОМ] *u*[ВАГА]> 16 ∨

∨ (∃ *v*) (ПД(v) ∧ *v*[ДНОМ]= *u*[ДНОМ] ∧*v*[ПНОМ]= 'П2'))}

***Реляційне числення із змінними доменів***

Реляційне числення, зорієнтоване на домени (або числення доменів), відрізняється від числення кортежів тим, що в ньому використовуються змінні доменів замість змінних кортежів, тобто змінні, що приймають свої значення в межах домена, а не відношення.

Основною формальною відмінністю числення доменів від числення кортежів є наявність додаткового набору предикатів, що дозволяють виражати так звані **умови членства**. Якщо *R* - це *n-арное* відношення з атрибутами *t1*, *t2 ..., tn*, та умова членства має вигляд

*R* *(pair, pair,…*),

де кожна пара  *pair* має вигляд  *t:v*, при цьому *v*це або константа, що задається літералом, або ім'я доменної змінної. Умова членства приймає значення  *true*  в тоді і лише тоді, якщо у відношенні *R*  існує кортеж, що містить значення вказаних атрибутів. Якщо *v–константа*, то на атрибут  *t*  задається жорстка умова, не залежна від поточних значень доменних змінних; якщо ж  *v – ім’я* доменної змінної, то умова членства може приймати різні значення при різних значеннях цієї змінної. Наприклад, обчислення виразу

ПД (ПНОМ:'П1', ДНОМ:'Д1')

дає значення  *true*, якщо і тільки якщо у відношенні ПД існує кортеж із значенням ПНОМ, рівним П1, і значенням ДНОМ, рівним Д1. Аналогічно, умова членства

ПД (ПНОМ:ПНОМА, ДНОМ:ДНОМА)

приймає значення *true*, якщо у відношенні ПД існує кортеж із значенням ПНОМ, еквівалентним поточному значенню змінної домена ПНОМА (будь-якому), і значенням ДНОМ, еквівалентним поточному значенню змінної домена ДНОМА (знову ж таки будь-якому).

У решті всіх відношень формули і вирази числення доменів виглядають схожими на формули і вирази числення кортежів. Зокрема, звичайно, розрізняються вільні і зв'язані входження доменних змінних.

Далі вважатимемо, що існують змінні доменів з іменами, що утворюються додаванням цифр 1, 2, 3 ... до відповідних імен доменів. Крім того передбачається, що в базі даних постачальників і деталей кожен атрибут має таке ж ім'я, як і відповідний йому домен, за винятком атрибутів ПФАМ і ДНАЗВ, для яких відповідний домен називається просто ІМ'Я.

Наприклад, вираз

(ПНОМ1) WHERE ПОСТАЧАЛЬНИК ( ПНОМ:ПНОМА, МІСТО: 'Вінниця' )

означає підмножину всіх номерів постачальників з міста Вінниця.

З використанням традиційного синтаксису мови предикатів реляційне числення із змінними доменів має вигляд:

{*x*1*x*2*..*.*x*k |  ψ (*x*1*x*2*..*.*x*k)},

де ψ - формула, що володіє тією властивістю, що тільки її вільні змінні доменів є різними змінними *x*1*x*2*..*.*x*k.

Реляційне числення доменів є основою більшості мов запитів, що базуються на використанні форм. Зокрема, на цьому численні базується відома мова QBE (Query-by-Example), яка була першою (і найцікавішою) мовою у сімействі мов, що базуються на табличних формах.

**Приклади**

Наведемо деякі з розглянутих вище прикладів, але в термінах числення доменів.

**- Отримати номери постачальників з Києва із статусом більше 20**

ПНОМ1 WHERE EXISTS СТАТУС1

(СТАТУС1>20 AND ПОСТАЧАЛЬНИК ( ПНОМ:ПНОМ1

СТАТУС:СТАТУС1, МІСТО:'Київ' ) )

{ ПНОМ1 | (∃СТАТУС1) (СТАТУС1>20 ∧

∧ПОСТАЧАЛЬНИК(ПНОМ1, СТАТУС1, МІСТО) ∧МІСТО='Київ')}

**-  Отримати імена постачальників, які поставляють принаймні одну червону деталь**

ИМЯ1 WHERE EXISTS ПНОМ1 EXISTS ДНОМ1

( ПОСТАЧАЛЬНИК ( ПНОМ:ПНОМ1, ПФАМ:ИМЯ1)

AND  ПД (ПНОМ: ПНОМ1, ДНОМ: ДНОМ1)

AND  ДЕТАЛЬ (ДНОМ:ДНОМ1, КОЛІР: 'Червоний'))

{ ИМЯ1 | (∃ПНОМ1) (∃ДНОМ1) (ПОСТАЧАЛЬНИК(ПНОМ1, ИМЯ1) ∧

∧ПД (ПНОМ1, ДНОМ1) ∧ ДЕТАЛЬ (ДНОМ1, КОЛІР) ∧

∧КОЛІР = 'Червоний')}

**-  Отримати імена постачальників, які поставляють принаймні одну деталь, що поставляється постачальником П2**

ИМЯ1 WHERE EXISTS ПНОМ1 EXISTS ДНОМ1

( ПОСТАЧАЛЬНИК ( ПНОМ:ПНОМ1, ПФАМ:ИМЯ1)

AND  ПД (  ПНОМ:ПНОМ1, ДНОМ:ДНОМ1 )

AND  ПД ( ПНОМ:'П2', ДНОМ:ДНОМ1 ) )

{ ИМЯ1 | (∃ПНОМ1) (∃ДНОМ1) (ПОСТАЧАЛЬНИК(ПНОМ1, ИМЯ1) ∧

∧ПД (ПНОМ1, ДНОМ1) ∧ПД (ПНОМ, ДНОМ1) ∧ПНОМ = 'П2')}

**-  Отримати імена постачальників, які поставляють всі деталі**

ИМЯ1 WHERE EXISTS ПНОМ1 ( ПОСТАЧАЛЬНИК

(ПНОМ:ПНОМ1, ПФАМ: ИМЯ1)

AND FORALL  ДНОМ1 ( IF ДЕТАЛЬ ( ДНОМ: ДНОМ1 )

THEN  ПД (ПНОМ:ПНОМ1, ДНОМ: ДНОМ1 )))

{ ИМЯ1 | (∃ПНОМ1) (ПОСТАЧАЛЬНИК (ПНОМ1, ИМЯ1) ∧

∧(∀ДНОМ1 ) ДЕТАЛЬ( ДНОМ1 ) →ПД (ПНОМ1, ДНОМ1)))}

* **Отримати імена постачальників, які не поставляють деталь Д2**

ИМЯ1 WHERE EXISTS  ПНОМ1 ( ПОСТАЧАЛЬНИК

( ПНОМ:ПНОМ1, ПФАМ:ИМЯ1 )

AND NOT  ПД (ПНОМ: ПНОМ1, ДНОМ: 'Д2' ) )

{ ИМЯ1 | (∃ПНОМ1) (ПОСТАЧАЛЬНИК (ПНОМ1, ИМЯ1) ∧

∧¬ПД (ПНОМ1, ДНОМ) ∧ДНОМ = 'Д2' ) ) }

**-  Отримати номери постачальників, що постачають принаймні всі деталі, що поставляються постачальником П2**

ПНОМ1 WHERE FORALL ДНОМ1 (IF ПД (ПНОМ: 'П2'

ДНОМ:ДНОМ1)

THEN  ПД (ПНОМ:ПНОМ1, ДНОМ:ДНОМ1))

{ ПНОМ1 | (∀ДНОМ1 ) ( ( ПД (ПНОМ, ДНОМ1) ∧

∧ПНОМ= 'П2')→ПД (ПНОМ1, ДНОМ1))}

**-  Отримати номери деталей, які або важать більше 16, або поставляються постачальником П2, або і те і інше.**

ДНОМ1 WHERE EXISTS ВЕС1

( ДЕТАЛЬ (ДНОМ:ДНОМ1, ВЕС:ВЕС1) AND  ВЕС1 > 16)

OR  ПД (ПНОМ : 'П2', ДНОМ: ДНОМ1 )

{ ДНОМ1 | (∃ВЕС1) (ДЕТАЛЬ (ДНОМ1,ВЕС1 ) ∧

∧ВЕС1>16)∨ ПД(ПНОМ,ДНОМ1) ∧ПНОМ = 'П2' )}