5-дәріс.
Өнімдік және фреймдік модельдер
Пән: Интеллектуалды ақпараттық жүйелер **Алдыңғы тақырып:** Білімдерді ұсынудың логикалық модельдері
Дәрістің мақсаты
Бүгінгі дәрісте біз білімді ұсынудың екі маңызды тәсілін — **өнімдік модельді** және **фреймдік модельді** қарастырамын.
Дәріс соңында сіздер:

- өнімдік модельдің құрылымын;
- «ЕГЕР – ОНДА» ережесінің қалай жұмыс істейтінін;
- өнімдік жүйенің негізгі компоненттерін;
- жұмыс жадысының қызметін;
- ереженің іске қосылу механизмін;
- конфликт жиыны мен конфликтті шешу стратегияларын;
- тура және кері тізбектеудің өнімдік жүйедегі орнын;
- фрейм ұғымын;
- слот, слот мәні, фасет, әдепкі мән ұғымдарын;
- фреймдердегі мұрагерлікті;
- процедуралық тіркемелерді;
- өнімдік және фреймдік модельдердің айырмашылығын;
- оларды қандай жағдайда бірге қолдануға болатынын

түсіндіре алатын боласыздар.
1. Кіріспе: логикалық формуладан практикалық білімге
Өткен дәрісте біз білімдерді ұсынудың **логикалық модельдерін** қарастырдық.
Біз **факт, ереже және сұраныс** ұғымдарын талдадық. Пікірлер логикасынан бастап предикаттар логикасына өттік. Modus Ponens, Modus Tollens, унификация, Forward Chaining және Backward Chaining механизмдерінің қалай жұмыс істейтінін көрдік.
Ең бастысы, біз мына идеяны түсінуіміз қажет болған:
интеллектуалды жүйе дайын жауапты ғана сақтамауы керек, ол бар білімнің негізінде жаңа қорытынды шығара алуы керек.
Бүгін біз осы идеяны практикалық жағынан жалғастырамыз.
Бір жағдайды елестетейік.
Университетте студенттердің оқу жағдайын бақылайтын интеллектуалды жүйе бар делік.
Жүйеде студент **Мадина** туралы мынадай ақпарат бар:

- GPA = 1.8;
- академиялық қарызы бар;
- екі міндетті пәнді қайта тапсыруы қажет.

Біз жүйеден:
«Мадинаға академиялық кеңес қажет пе?»
деп сұраймыз.
Бұл сұраққа жауап беру үшін біз ұзақ логикалық формула жазуымыз мүмкін.
Бірақ сарапшының білімін қарапайым түрде де бере аламыз:
ЕГЕР студенттің GPA көрсеткіші 2.0-ден төмен БОЛСА, ОНДА академиялық кеңес ұсыну керек.
Мұндай құрылымды адам да оңай оқиды, компьютер де ереже ретінде өңдей алады.
Дәл осы жерде біз бүгінгі бірінші негізгі модельге келеміз.
Бұл — **өнімдік модель**.
Ал дәрістің екінші жартысында басқа сұрақ қоямыз:
Егер бізге студент туралы шешім шығару ғана емес, студенттің өзін — аты, курсы, мамандығы, GPA көрсеткіші, мәртебесі, пәндері сияқты барлық қасиеттерімен бірге құрылымдап сипаттау қажет болса ше?
Онда тек «ЕГЕР – ОНДА» ережелері жеткіліксіз болуы мүмкін.

Бұл жағдайда **фреймдік модель** ыңғайлы болады.
Сондықтан бүгін біз білімді екі түрлі көзқараспен қарастырамыз:
Өнімдік модель: «Қандай жағдайда не істеу керек?»
Фреймдік модель: «Объект қандай және оның қандай қасиеттері бар?»
Осы айырмашылықты дәріс соңына дейін есте ұстайық.

2. Өнімдік модель дегеніміз не?
Енді өнімдік модельге нақты анықтама берейін.
Өнімдік модель — білімді шарт пен нәтиже арасындағы байланыс түрінде, яғни «ЕГЕР шарт орындалса, ОНДА белгілі бір қорытынды жаса немесе әрекет орында» қағидасы бойынша ұсынатын модель.
Оның ең қарапайым формасы:
ЕГЕР А ОНДА В
немесе формальды түрде:
А → В
Бірақ мұнда бір нәрсеге назар аударуымыз керек.
Өткен дәрістегі логикалық импликация мен өнімдік ереже бір-біріне өте ұқсас болғанымен, олардың қолданылу мақсаты әртүрлі болуы мүмкін.
Логикалық модельде біз көбінесе:
«Осы тұжырымнан қандай логикалық салдар шығады?»
деген сұраққа жауап іздейміз.
Ал өнімдік жүйеде:
«Осы жағдай орын алғанда жүйе қандай қорытынды жасауы немесе қандай әрекет орындауы керек?»
деген практикалық сұрақ маңызды.
Мысалы:
ЕГЕР студент барлық зертханалық жұмыстарды тапсырса ЖӘНЕ рейтингі 50 балдан жоғары немесе тең болса, ОНДА студентті емтиханға жібер.
Бұл жерде жүйе нақты шешім қабылдап отыр.
Демек, өнімдік модельдің негізгі идеясы өте қарапайым:
жағдайды анықтау → сәйкес ережені табу → нәтижені шығару.
Осыдан өнімдік ереженің ішкі құрылымын қарастыруымыз керек.

3. Өнімдік ереженің құрылымы
Өнімдік ереже әдетте екі негізгі бөліктен тұрады:

Бөлік	**Мағынасы**
Антецедент	Ереженің шарты
Консеквент	Шарт орындалғандағы қорытынды немесе әрекет

Қарапайым түрде:
ЕГЕР <антецедент> ОНДА <консеквент>
Мысалы:
ЕГЕР Нұрланның кітапханаға берешегі болса, ОНДА жаңа кітап беруге уақытша тыйым сал.
Мұнда:
«Нұрланның кітапханаға берешегі бар» — антецедент.
«Жаңа кітап беруге уақытша тыйым салу» — консеквент.
Бірақ консеквент әрдайым әрекет бола бермейді.
Ол жаңа факт те болуы мүмкін.
Мысалы:
ЕГЕР студент CS101 пәнін меңгерсе ЖӘНЕ CS102 пәнін меңгерсе, ОНДА студенттің базалық программалау дайындығы бар.
Бұл жерде соңғы бөлік физикалық әрекет емес.

Жүйе білім базасына жаңа қорытынды қосып отыр:
«Базалық программалау дайындығы бар».
Сондықтан өнімдік ереженің нәтижесі екі негізгі түрде кездеседі:
1. жаңа білім алу;
2. белгілі бір әрекетті іске қосу.
Енді сұрақ туындайды:
Ереже бар. Бірақ жүйе оның шарттары орындалғанын қайдан біледі?
Ол үшін жүйеге ағымдағы жағдай туралы фактілер қажет.
Бұл фактілер **жұмыс жадысында** сақталады.

4. Өнімдік жүйенің негізгі компоненттері
Өнімдік жүйені тек ережелер жиыны деп қарастыру жеткіліксіз.
Толық өнімдік жүйеде бірнеше компонент бірге жұмыс істейді.
Негізгі төрт компонентті бөліп көрсетейік:

Компонент	**Қызметі**
Ережелер базасы	Пәндік саладағы «ЕГЕР – ОНДА» ережелерін сақтайды
Жұмыс жадысы	Ағымдағы фактілер мен аралық нәтижелерді сақтайды
Инференция механизмі	Қандай ережелер орындалатынын анықтайды
Басқару стратегиясы	Бірнеше ереже дайын болса, қайсысын бірінші орындауды таңдайды

Осы төрт компоненттің жұмысын бір тізбек ретінде қарастыруға болады:
Фактілер → сәйкес ережені іздеу → ережені іске қосу → жаңа факт немесе әрекет → қайта тексеру.
Енді осы компоненттердің ішінен алдымен жұмыс жадысын түсінейік.

5. Жұмыс жадысы
Жұмыс жадысы — жүйенің дәл қазіргі сәтте белгілі деп қабылдаған фактілерін және жұмыс барысында алынған аралық нәтижелерін сақтайтын бөлік.
Оны адамның қысқа мерзімді жадысына ұқсатуға болады.
Білім базасындағы ережелер ұзақ уақыт сақталуы мүмкін.
Ал жұмыс жадысындағы мәліметтер нақты жағдайға байланысты өзгеріп отырады.
Мысалы, университеттің жүйесінде **Әлихан** туралы бастапқы фактілер:
Student = Әлихан Course = AI201 PrerequisitePassed = yes SeatAvailable = yes
болсын.
Ереже:
ЕГЕР PrerequisitePassed = yes ЖӘНЕ SeatAvailable = yes ОНДА RegistrationAllowed = yes
Жүйе ереженің шарттарын жұмыс жадысындағы фактілермен салыстырады.
Екі шарт та орындалып тұр.
Сондықтан жаңа факт пайда болады:
RegistrationAllowed = yes
Бұл жаңа факт те жұмыс жадысына қосылады.
Демек, жұмыс жадысы өзгермейтін қойма емес.
Ол жүйе ой қорыту барысында жаңарып отырады.
Енді жүйенің ережені қалай таңдап, қалай орындайтынын қарастырайық.

6. «Сәйкестендіру – таңдау – орындау» циклі
Өнімдік жүйенің жұмысын түсінудің ең маңызды схемаларының бірі:
MATCH → SELECT → ACT
Қазақша:
Сәйкестендіру → Таңдау → Орындау

Бірінші кезең — **сәйкестендіру**.
Жүйе жұмыс жадысындағы фактілерді барлық ережелердің шарттарымен салыстырады.
Яғни:
«Қандай ережелердің шарттары дәл қазір орындалып тұр?»
деген сұраққа жауап береді.
Екінші кезең — **таңдау**.
Егер бір ғана ереже сәйкес келсе, мәселе оңай.
Бірақ бірнеше ереже бір уақытта орындалуға дайын болуы мүмкін.
Онда олардың ішінен біреуін немесе белгілі бір ретті таңдау керек.
Үшінші кезең — **орындау**.
Таңдалған ереженің нәтижесі орындалады.
Ол:

- жаңа факт қосуы;
- бұрынғы фактіні өзгертуі;
- хабарлама жіберуі;
- белгілі әрекетті іске қосуы мүмкін.

Содан кейін цикл қайта басталады.
Яғни:
Сәйкестендіру → Таңдау → Орындау → Жадыны жаңарту → қайта сәйкестендіру.
Бұл процесс қажетті қорытынды алынғанша немесе орындалатын ереже қалмағанша жалғасуы мүмкін.
Бірақ бірнеше ереже қатар дайын болғанда жүйе нақты қайсысын таңдайды?
Осы жерде **конфликт жиыны** ұғымы пайда болады.

7. Конфликт жиыны
Конфликт жиыны — шарттары ағымдағы фактілерге сәйкес келетін және дәл қазір іске қосылуға дайын ережелер жиыны.
Мысалы, жұмыс жадысында:
GPA(Асем) = 1.7
деген факт бар делік.
Ал жүйеде екі ереже бар:
R1:
ЕГЕР GPA < 2.0 ОНДА студентке академиялық кеңес ұсын.
R2:
ЕГЕР GPA < 1.8 ОНДА академиялық тәуекел туралы ескерту жаса.
Асемнің GPA көрсеткіші 1.7 болғандықтан, екі ереженің де шарты орындалады.
Сонда:
Conflict Set = {R1, R2}
болады.
Жүйе енді қай ережені бірінші орындау керектігін шешуі тиіс.
Бұл процесс **конфликтті шешу** деп аталады.

8. Конфликтті шешу стратегиялары
Конфликтті шешудің бір ғана әмбебап тәсілі жоқ.
Жүйенің мақсатына байланысты әртүрлі стратегия қолданылуы мүмкін.
1. Басымдық бойынша таңдау
Әр ережеге басымдық беріледі.
Мысалы:
R1 priority = 5 R2 priority = 10
Онда R2 бірінші орындалады.
2. Нақтылығы жоғары ережені таңдау
Шарттары нақтырақ ереже жалпы ережеден бұрын орындалуы мүмкін.

3. Жаңа фактіні қолданатын ережені таңдау
Жүйе соңғы алынған ақпаратқа сүйенетін ережеге басымдық бере алады.
4. Алдын ала берілген реттілік
Кейбір жүйелерде ережелердің орындалу реті жобалау кезінде белгіленеді.
Мұнда маңызды ой:
ережелердің өзі білімді сипаттайды, ал конфликтті шешу стратегиясы сол білімнің қалай қолданылатынын басқарады.
Енді өнімдік жүйеде қорытындының қандай бағытта жүретінін еске түсірейік.

9. Тура тізбектеу — Forward Chaining
Өткен дәрісте Forward Chaining ұғымымен таныстық.
Бүгін оны өнімдік жүйенің жұмыс механизмі ретінде қарастырамыз.
Тура тізбектеу — бастапқы фактілерден басталып, ережелерді біртіндеп қолдану арқылы жаңа қорытындыларға қарай қозғалу.
Схемасы:
ФАКТ → ЕРЕЖЕ → ЖАҢА ФАКТ → ЕРЕЖЕ → ҚОРЫТЫНДЫ
Мысалы, жүйеде **Айбек** туралы:
Attendance = 55%
деген факт пайда болды делік.
Ережелер:
R1:
ЕГЕР Attendance < 70% ОНДА AttendanceRisk = yes.
R2:
ЕГЕР AttendanceRisk = yes ОНДА SendWarning = yes.
Алдымен R1 іске қосылады.
Жұмыс жадысына:
AttendanceRisk = yes
қосылады.
Бұл жаңа факт R2 ережесінің шартын орындап тұр.
Сондықтан:
SendWarning = yes
деген нәтиже шығады.
Бұл — тура тізбектеу.
Жүйе бастапқы фактіден басталып, одан қандай салдарлар шығатынын анықтады.
Сондықтан Forward Chaining көбінесе:
«Қазіргі жағдайдан не шығады?»
деген сұраққа жауап береді.

10. Кері тізбектеу — Backward Chaining
Егер бізге барлық мүмкін қорытындылар қажет болмаса, тек бір нақты гипотезаны тексеру керек болуы мүмкін.
Онда кері тізбектеу ыңғайлы.
Кері тізбектеу — мақсаттан басталып, оны дәлелдеу үшін қандай шарттар қажет екенін кері бағытта анықтау.
Схемасы:
МАҚСАТ → ҚАЖЕТ ЕРЕЖЕ → ҚАЖЕТ ФАКТІЛЕР → ТЕКСЕРУ
Мысалы, біз:
«Динара дипломдық жобаны қорғауға жіберіле ме?»
деген сұраққа жауап іздейік.
Жүйеде:
ЕГЕР барлық міндетті пәндер аяқталса ЖӘНЕ дипломдық жоба тапсырылса ОНДА қорғауға жіберіледі

деген ереже бар.
Жүйе бірден барлық ережелерді орындамайды.
Алдымен:
«Қорғауға жіберіледі» деген қорытындыны қандай ереже береді?
деп іздейді.
Содан кейін екі шартты тексереді:
Міндетті пәндер аяқталды ма?
Дипломдық жоба тапсырылды ма?
Егер екеуі де расталса, мақсат дәлелденеді.
Осылайша:
Forward Chaining — деректен мақсатқа;
Backward Chaining — мақсаттан қажетті дерекке қарай қозғалады.
Енді өнімдік жүйелердің неге эксперттік жүйелерде кең таралғанын түсіне аламыз.

11. Өнімдік модель және эксперттік жүйе
Өнімдік модель эксперттік жүйелерге өте қолайлы.
Себебі мамандар өз тәжірибесін көбіне шарт түрінде айтады:
«Егер осындай жағдай болса, мынаны тексер».
«Егер көрсеткіш осы шектен төмен болса, мына шешімді қабылда».
Яғни сарапшының тәжірибесін ережелерге айналдыру салыстырмалы түрде табиғи.
Мысалы, техникалық қолдау маманы:
ЕГЕР компьютер желіге қосылған, БІРАҚ интернетке шығу жоқ, ОНДА шлюз параметрлерін тексер
деген тәжірибелік ережені бере алады.
Бұл жерде біз программаның негізгі алгоритмін толығымен қайта жазбай-ақ, білім базасына жаңа ереже қоса аламыз.
Сондықтан өнімдік модельдің маңызды қасиеттерінің бірі — **модульдік**.
Әр ереже білімнің жеке фрагменті ретінде сақталады.
Бірақ бұл артықшылық үлкен жүйеде жаңа мәселе туғызады.

12. Ережелер саны көбейгенде не болады?
Он ережеден тұратын жүйені басқару оңай.
Жүз ереже болса — күрделене бастайды.
Ал бірнеше мың ереже болса, мынандай мәселелер пайда болуы мүмкін:

- ережелер бірін-бірі қайталайды;
- бір-біріне қайшы келуі мүмкін;
- қандай ереже бірінші орындалатынын анықтау қиындайды;
- жүйенің неге дәл сол шешімді қабылдағанын бақылау қиындайды;
- өнімділік төмендеуі мүмкін.

Мысалы:
R1: GPA < 2.0 болса → ескерту жібер.
R2: GPA < 2.0 болса → ескерту жіберме.
Екі ереже бірдей шартта қарама-қайшы әрекет ұсынып тұр.
Бұл жерде мәселе инференция механизмінде емес.
Мәселе — **білім базасының сапасында**.
Сондықтан өнімдік жүйені жасағанда ережелерді тек жазып қою жеткіліксіз.
Оларды:
тексеру, топтау, басымдық беру, қайшылықтарды анықтау және жаңартып отыру қажет.

13. Өнімдік модельдің артықшылықтары мен шектеулері
Енді бірінші модель бойынша аралық қорытынды жасайық.

Артықшылық	**Түсіндірме**
Қарапайым құрылым	«ЕГЕР – ОНДА» формасы адамға түсінікті
Модульдік	Әр ережені жеке қосуға немесе өзгертуге болады
Түсіндірмелілік	Қандай ереже іске қосылғанын көрсетуге болады
Автоматтандыру	Шарт орындалғанда жүйе өзі әрекет ете алады
Сарапшы білімін беру	Тәжірибелік білімді ережеге айналдыру ыңғайлы

Ал шектеулері:

Шектеу	**Себебі**
Ереже санының өсуі	Үлкен базаны басқару қиындайды
Қайшылықтар	Бірнеше ереже әртүрлі нәтиже беруі мүмкін
Объект құрылымын сипаттау қиындығы	Күрделі объектіні тек ережелермен сипаттау ыңғайсыз
Ерекшеліктер	Қатаң ережеге көптеген ерекше жағдай қосылуы мүмкін

Соңғы шектеуге назар аударайық.
Өнімдік модель:
«Не істеу керек?»
деген сұраққа жақсы жауап береді.
Бірақ:
«Объектінің өзі қандай?»
деген сұраққа әрдайым ыңғайлы емес.
Осыдан дәрістің екінші негізгі бөлігіне өтеміз.

14. Неліктен фреймдік модель қажет?
Енді студент туралы толық білім сақтағымыз келеді деп елестетейік.
Студенттің:

- аты;
- туған жылы;
- білім беру бағдарламасы;
- курсы;
- тобы;
- GPA;
- академиялық мәртебесі;
- меңгерген пәндері;
- эдвайзері

бар.
Бұлардың барлығын жеке-жеке ереже түрінде жазуға болады.
Бірақ бұл өте қолайсыз.
Бізге барлық қасиетті бір объектінің айналасына жинақтайтын құрылым қажет.
Мысалы:
СТУДЕНТ
деген бір құрылым жасап, оның ішіне студентке тән барлық қасиеттерді орналастырсақ, білім әлдеқайда түсінікті болады.
Дәл осы идея **фреймдік модельдің** негізінде жатыр.

15. Фрейм дегеніміз не?
Фрейм — белгілі бір типтік объект, ұғым, жағдай немесе оқиға туралы білімді құрылымдалған түрде сипаттайтын білім бірлігі.
Қарапайым тілмен айтсам:
фрейм — объект туралы компьютерлік «карточка».
Бірақ оны кәдімгі кесте жолымен толық теңестіруге болмайды.
Фреймде тек мәндер ғана емес:

- қасиеттер;
- әдепкі мәндер;
- шектеулер;
- басқа фреймдермен байланыстар;
- мұрагерлік;
- кей жағдайда орындалатын процедуралар

сақталуы мүмкін.
Мысалы, **СТУДЕНТ** фреймін қарастырайық.

Слот	**Мән**
Аты	Аружан
Білім беру бағдарламасы	Ақпараттық жүйелер
Курс	2
GPA	3.42
Мәртебе	Белсенді
Топ	IS-24

Бүкіл кесте — Аружан туралы фреймнің қарапайым көрінісі.
Ал жеке жолдардағы **Аты**, **Курс**, **GPA**, **Мәртебе** сияқты қасиеттер **слоттар** деп аталады.
Ендеше келесі ұғымымыз — слот.

16. Слот дегеніміз не?
Слот — фрейм сипаттайтын объектінің белгілі бір қасиетін немесе басқа объектімен қатынасын көрсететін элемент.
Егер фреймді сауалнама деп қарастырсақ, слот — сол сауалнамадағы өріс.
Мысалы, **ПӘН** фреймі болсын.
Оның слоттары:

Слот	**Мән**
Код	AI201
Атауы	Жасанды интеллект
Кредит саны	5
Семестр	4
Пререквизит	CS102

Бұл жерде:
Кредит саны — слот.
5 — слоттың мәні.
Пререквизит — слот.
CS102 — оның мәні.
Демек:
фрейм → слоттардан тұрады;
слот → белгілі бір мән қабылдайды.
Бірақ слоттың тек атауы мен мәні ғана болмауы мүмкін.
Ол туралы қосымша ақпарат та берілуі мүмкін.
Оны **фасет** деп атаймыз.

17. Фасет дегеніміз не?
Фасет — слоттың өзін сипаттайтын қосымша қасиет немесе шектеу.
Яғни слот объектінің қасиетін сипаттаса, фасет сол қасиетті **қалай толтыру керек екенін** сипаттайды.
Мысалы, **GPA** слоты үшін:

Сипаттама	**Мән**
Слот	GPA
Дерек типі	нақты сан
Минимум	0.0

Максимум	4.0
Міндетті ме?	иә
Әдепкі мән	жоқ

Мұнда GPA = 3.5 — слоттың мәні.
Ал:
«мәні 0 мен 4 аралығында болуы керек»
деген шарт — слоттың **шектеуі**, яғни фасет.
Осы механизмнің арқасында фрейм тек деректі сақтап қоймайды, деректің құрылымын да бақылай алады.

18. Әдепкі мән — Default Value
Фреймдік модельдің пайдалы мүмкіндіктерінің бірі — **әдепкі мән**.
Әдепкі мән — нақты объект үшін басқа мән көрсетілмесе, автоматты түрде қолданылатын алдын ала белгіленген мән.
Мысалы, университеттегі жаңа студенттердің бастапқы мәртебесі көбіне:
«белсенді»
болсын.
Онда СТУДЕНТ фреймінде:
Мәртебе = белсенді
деген әдепкі мән бере аламыз.
Жаңа студент **Диас** жүйеге енгізілген кезде оның мәртебесі арнайы көрсетілмесе, жүйе:
Мәртебе(Диас) = белсенді
деп қабылдайды.
Ал егер Диас академиялық демалыс алса, нақты мән:
Мәртебе = академиялық демалыс
деп өзгертіледі.
Яғни әдепкі мән:
«басқа ақпарат болмаса, осы мәнді пайдалан»
деген принциппен жұмыс істейді.
Бұл көптеген ұқсас объектілерді сипаттағанда қайталанатын ақпаратты азайтады.
Осы идея бізді фреймдік модельдің тағы бір маңызды механизміне — **мұрагерлікке** алып келеді.

19. Фреймдердегі мұрагерлік
Нақты әлемдегі ұғымдар көбіне иерархия құрайды.
Мысалы:
АДАМ → СТУДЕНТ → МАГИСТРАНТ
Магистрант — студент.
Студент — адам.
Егер АДАМ фреймінде:

- аты;
- туған күні;
- байланыс нөмірі

сияқты слоттар бар болса, оларды СТУДЕНТ фреймінде қайтадан жазудың қажеті жоқ.
СТУДЕНТ бұл слоттарды АДАМ фреймінен **мұраға алады**.
Ал өзі қосымша:

- студенттік ID;
- топ;
- GPA

слоттарын қосады.
МАГИСТРАНТ болса СТУДЕНТ фреймінің қасиеттерін мұраға алып, оған:

- диссертация тақырыбы;

- ғылыми жетекші

сияқты слоттар қоса алады.
Схема:
АДАМ
↓
аты, туған күні, байланыс
СТУДЕНТ
↓
- ID, топ, GPA

МАГИСТРАНТ
↓
- диссертация, ғылыми жетекші

Бұл жерде маңызды принцип:
төменгі деңгейдегі фрейм жоғары деңгейдегі фреймнің қасиеттерін қайта жазбай пайдалана алады.
Мұрагерлік білімді қайталамауға және құрылымды жүйелі сақтауға мүмкіндік береді.

20. Мұрагерліктегі ерекшелік
Бірақ мұрагерлік әрдайым механикалық түрде жұмыс істей бермейді.
Кейде төменгі деңгейдегі фрейм ата-анасынан алған мәнді өзгертуі керек.
Мысалы:
ҚЫЗМЕТКЕР фреймінде:
Жұмыс режимі = толық күн
деген әдепкі мән болсын.
Ал:
ҚОСАЛҚЫ ҚЫЗМЕТКЕР
фреймі ҚЫЗМЕТКЕР фреймінен мұра алады, бірақ оның жұмыс режимі:
Жұмыс режимі = жарты күн
болуы мүмкін.
Бұл жағдайда төменгі деңгейдегі фрейм мұраланған әдепкі мәнді өзінің нақты мәнімен алмастырады.
Яғни мұрагерлік:
«бәрін соқыр түрде көшіру»
деген сөз емес.
Ол:
«жалпы қасиеттерді алу, ал қажет жерде нақтылау»
деген принципке негізделеді.

21. Фрейм экземпляры
Осы жерде **класс немесе типтік фрейм** мен **нақты экземплярды** ажыратып алуымыз керек.
СТУДЕНТ — жалпы құрылым.
Ал:
Санжар
— нақты студент.
Сондықтан:
СТУДЕНТ — типтік фрейм;
Санжар — сол фреймнің экземпляры.
Мысалы:
СТУДЕНТ фреймі
- Аты
- Тобы

- Курсы
- GPA
- Мәртебесі

Нақты экземпляр
Студент: Санжар

- Аты = Санжар
- Тобы = IS-24
- Курсы = 2
- GPA = 3.15
- Мәртебесі = белсенді

Бұл объектіге бағытталған программалаудағы **класс және объект** ұғымдарына ұқсайды. Бірақ оларды толық бір ұғым деп қабылдауға болмайды.

22. Фрейм мен ООП классының айырмашылығы

Сіздер программалаудан класс пен объект ұғымдарын білесіздер.
Сондықтан:
«Фрейм дегеніміз класс па?»
деген сұрақ туындауы мүмкін.
Ұқсастығы бар.
Екеуінде де:

- қасиеттер бар;
- иерархия болуы мүмкін;
- мұрагерлік қолданылады;
- нақты экземплярлар жасалады.

Бірақ мақсаттары әртүрлі.

Фрейм	**ООП классы**
Білімді ұсынуға арналған	Бағдарламаның құрылымын ұйымдастырады
Пәндік мағына маңызды	Программалық мінез-құлық маңызды
Слот, әдепкі мән, шектеу сақтайды	Өріс пен әдістерді анықтайды
Инференциямен байланысуы мүмкін	Кодтың орындалуына бағытталған

Қарапайым түрде:
класс — программаны ұйымдастыру құралы;
фрейм — білімді ұйымдастыру құралы.
Бұл шекара практикалық жүйелерде кейде жақындауы мүмкін, бірақ концептуалдық мақсаты әртүрлі.

23. Процедуралық тіркеме

Фрейм тек статикалық қасиеттер жиыны болып қалмауы мүмкін.
Кейбір слоттар өзгерген кезде белгілі бір процедура автоматты түрде орындалуы мүмкін.
Бұл механизм **процедуралық тіркеме** деп аталады.
Процедуралық тіркеме — слотпен байланысты және белгілі оқиға кезінде орындалатын процедура.
Мысалы, студент **Камила** туралы фреймде:
GPA = 1.85
болды делік.
GPA слоты өзгерген кезде жүйе:
академиялық мәртебені қайта есептеу
процедурасын іске қоса алады.
Яғни:
GPA өзгерді процедура іске қосылды академиялық мәртебе қайта есептелді.
Осылайша фрейм тек:
«объект қандай?»

деген сұраққа емес, белгілі дәрежеде:
«оның қасиеті өзгергенде не істеу керек?»
деген сұраққа да жауап бере алады.
Дегенмен күрделі әрекеттік логика қажет болғанда өнімдік ережелер ыңғайлырақ болуы мүмкін.
Осыдан екі модельді тікелей салыстырайық.

24. Өнімдік және фреймдік модельдерді салыстыру
Екі модельдің мақсаты бір — білімді компьютер өңдей алатын түрде ұсыну.
Бірақ олар білімнің әртүрлі жағына назар аударады.

Критерий	**Өнімдік модель**	**Фреймдік модель**
Негізгі бірлік	Ереже	Фрейм
Негізгі құрылым	ЕГЕР – ОНДА	Объект – слот – мән
Негізгі сұрақ	«Қандай жағдайда не болады?»	«Бұл объект қандай?»
Күшті жағы	Шешім және әрекет	Объект құрылымы
Білім түрі	Процедуралық, эвристикалық	Декларативтік, құрылымдық
Мұрагерлік	Негізгі механизм емес	Маңызды механизм
Әдепкі мән	Арнайы ережемен беріледі	Табиғи түрде қолданылады
Түсіндіру	Іске қосылған ережелер арқылы	Слоттар мен иерархия арқылы
Негізгі мәселе	Ереже саны көбейгенде басқару	Күрделі логикалық шешімдер

Бұны бір сөйлеммен есте сақтауға болады:
Өнімдік модель — шешімге бағытталған.
Фреймдік модель — объектінің құрылымына бағытталған.
Бірақ интеллектуалды жүйеде міндетті түрде тек біреуін таңдау керек пе?
Жоқ.
Көп жағдайда оларды бірге қолдану тиімді.

25. Өнімдік және фреймдік модельді бірге қолдану
Енді осы екі модельді бір жүйеге біріктірейік.
Университеттің интеллектуалды академиялық кеңесшісін жасаймыз делік.
Алдымен студент туралы құрылымдық білімді фреймммен береміз.
STUDENT фреймі

Слот	**Мән**
Name	Еркебұлан
Course	2
GPA	1.75
FailedCourses	3
Status	Active

Бұл — объектінің ағымдағы күйі.
Енді өнімдік ереже қосайық:
ЕГЕР GPA < 2.0 ЖӘНЕ FailedCourses ≥ 2 ОНДА AcademicRisk = High.
Жүйе фреймдегі мәндерді оқиды:
GPA = 1.75
FailedCourses = 3
Ереженің екі шарты да орындалып тұр.
Сондықтан:
AcademicRisk = High
деген қорытынды жасалады.
Қажет болса бұл нәтиже қайтадан студент фреймінің:
AcademicRisk
слотына жазылуы мүмкін.
Сонда процесс:

ФРЕЙМ → ФАКТІЛЕР → ЕРЕЖЕ → ҚОРЫТЫНДЫ → ФРЕЙМДІ ЖАҢАРТУ
түрінде жұмыс істейді.
Міне, бұл — гибридтік білім ұсынудың қарапайым мысалы.
Фрейм:
«Еркебұлан туралы не білеміз?»
деген сұраққа жауап береді.
Өнімдік ереже:
«Осы мәліметтерге сүйеніп қандай шешім қабылдаймыз?»
деген сұраққа жауап береді.
Осылайша екі модель бір-бірін толықтырады.

26. Қандай жағдайда қай модельді таңдау керек?
Енді практикалық тұрғыдан қарастырайық.
Егер сіз интеллектуалды жүйе жобалап отырсаңыз, бірден:
«Қай технологияны қолданамын?»
деп сұрамаңыз.
Алдымен:
«Менің білімімнің табиғаты қандай?»
деп сұраңыз.
Егер білімнің негізгі бөлігі:
«Егер осындай жағдай болса, осындай шешім қабылда»
түрінде болса, өнімдік модель ыңғайлы.
Мысалы:

- бизнес ережелері;
- мониторинг;
- техникалық диагностика;
- сараптамалық жүйе;
- қауіпсіздік ережелері;
- автоматты ескерту жүйелері.

Ал егер негізгі міндет:
күрделі объектілерді олардың қасиеттерімен және иерархиясымен сипаттау
болса, фреймдік модель ыңғайлы.
Мысалы:

- студент профилі;
- қызметкер моделі;
- медициналық карта құрылымы;
- құрылғы сипаттамасы;
- ұйым құрылымы;
- пәндік сала объектілерінің иерархиясы.

Ал егер объект туралы толық құрылым да, сол объект бойынша автоматты шешім де керек болса:
екі модельді бірге қолдануға болады.

27. Құралдар және практикалық іске асыру
Теорияны программалық құралдармен байланыстырайық.
Өнімдік ережелермен жұмыс істеуге арналған жүйелерге:
CLIPS, Drools, Jess
сияқты құралдар жатады.
CLIPS -эксперттік және ережелік жүйелерді құруда кеңінен белгілі.
Drools -Java экожүйесінде бизнес-ережелерді қолдануға мүмкіндік береді.
Фреймдік модельдер тарихында **Protégé-Frames** маңызды орын алады.

Фреймдік ортада класстар, слоттар, экземплярлар және мұрагерлік арқылы пәндік саланы құрылымдауға болады.
Қазіргі жүйелерде классикалық фрейм идеялары онтологияларда, объектілік модельдерде және білім графтарында да жалғасып келеді.
Сондықтан фреймдік модельді тек ескі технология деп қабылдауға болмайды.
Оның негізгі идеялары:
объект → қасиет → мән → иерархия → мұрагерлік
қазіргі білім инженериясында да өте маңызды.

28. Өнімдік модельді құру алгоритмі
Енді студент ретінде өнімдік модель құру қажет болса, қандай ретпен жұмыс істеу керек екенін жинақтайық.
1-қадам. Мақсатты анықтау
Жүйе қандай шешім қабылдауы керек?
2-қадам. Қажетті фактілерді анықтау
Шешім қабылдау үшін қандай ақпарат қажет?
3-қадам. Ережелерді табиғи тілде жазу
ЕГЕР ...
ОНДА ...
4-қадам. Шарттарды нақтылау
«Жоғары», «төмен», «жақсы» сияқты белгісіз ұғымдарды нақты критерийге айналдыру.
5-қадам. Ережелер арасындағы қайшылықты тексеру
Екі ереже бір жағдайда қарама-қайшы нәтиже бермей ме?
6-қадам. Басымдықты анықтау
Бірнеше ереже қатар орындалса, қайсысы маңызды?
7-қадам. Бақылау мысалдарымен тексеру
Әртүрлі фактілер енгізіп, жүйенің дұрыс қорытынды жасайтынын тексеру.
Осылайша жақсы өнімдік жүйе жай ғана көп ережеден емес, **дұрыс ұйымдастырылған және тексерілген ережелерден** тұрады.

29. Фреймдік модельді құру алгоритмі
Фреймдік модельде жұмыс реті сәл өзгеше.
1-қадам. Негізгі объектілерді анықтау
Пәндік салада қандай объектілер бар?
2-қадам. Жалпы және нақты объектілерді ажырату
Қайсысы класс, қайсысы экземпляр?
3-қадам. Слоттарды анықтау
Әр объектінің қандай қасиеттері бар?
4-қадам. Слот мәндерінің типін анықтау
Мәтін бе, сан ба, логикалық мән бе, басқа фреймге сілтеме ме?
5-қадам. Шектеулерді анықтау
Мән қандай диапазонда болуы керек?
6-қадам. Әдепкі мәндерді белгілеу
Қандай қасиеттер көбіне бірдей мән қабылдайды?
7-қадам. Иерархия құру
Қандай фрейм қандай фреймнен қасиет мұра алады?
8-қадам. Ерекшеліктерді анықтау
Мұраланған мән қай жерде өзгеруі керек?
9-қадам. Экземплярлармен тексеру
Нақты объектілерді енгізіп, модельдің жеткілікті екенін бағалау.
Мұнда маңызды қағида:
алдымен нақты бір объектіні емес, объектілердің жалпы құрылымын ойлау керек.

30. Жиі кездесетін қателер

Енді осы тақырып бойынша студенттер жиі жіберетін қателерді қарастырайық.

Бірінші қате — факт пен ережені араластыру

«Алина 3-курста оқиды» — факт.

Ал:

«Егер студент 3-курста болса, өндірістік практикаға өтініш бере алады» — ереже.

Екеуінің рөлі бірдей емес.

Екінші қате — шартты нақты анықтамау

«Егер студент жақсы оқыса...»

деген ереже компьютер үшін жеткіліксіз.

«Жақсы» деген не?

GPA ≥ 3.0 ма?

GPA ≥ 3.5 пе?

Критерий нақты болуы керек.

Үшінші қате — фреймді жай кесте деп түсіну

Фреймде тек баған мен мән емес, иерархия, мұрагерлік, әдепкі мән және шектеу сияқты білім элементтері болуы мүмкін.

Төртінші қате — әр объект үшін жеке фрейм құрылымын қайта жазу

Егер жүз студент болса, жүз рет:

аты, курс, топ, GPA

деген құрылымды жасаудың қажеті жоқ.

Бір **СТУДЕНТ** типтік фреймін жасаймыз, ал нақты студенттер оның экземплярлары болады.

Бесінші қате — бір модель барлық мәселені шешеді деп ойлау

Өнімдік модель ережелерге өте жақсы.

Фрейм объект құрылымына жақсы.

Логикалық модель формалды дәлелдеуге жақсы.

Сондықтан модель міндетке сәйкес таңдалады.

31. Дәрістің негізгі схемасы

Енді бүгінгі дәрістің екі бөлігін бір суретке біріктірейік.

Өнімдік модель

ФАКТІЛЕР ЖҰМЫС ЖАДЫСЫ ЕРЕЖЕЛЕРМЕН СӘЙКЕСТЕНДІРУ КОНФЛИКТ ЖИЫНЫ ЕРЕЖЕНІ ТАҢДАУ ЕРЕЖЕНІ ОРЫНДАУ ЖАҢА ФАКТ / ӘРЕКЕТ

Ал фреймдік модельде:

ЖАЛПЫ ҰҒЫМ ФРЕЙМ СЛОТТАРМӘНДЕР ЖӘНЕ ФАСЕТТЕР МҰРАГЕРЛІКНАҚТЫ ЭКЗЕМПЛЯР

Бірінші схема — **шешімнің қалай алынатынын** жақсы көрсетеді.

Екінші схема — **объект туралы білімнің қалай ұйымдастырылғанын** жақсы көрсетеді.

32. Қорытынды

Сонымен, бүгін біз білімді ұсынудың екі маңызды моделін тереңірек қарастырдық.

Біріншісі — **өнімдік модель**.

Оның негізгі бірлігі:

ЕГЕР – ОНДА ережесі.

Өнімдік жүйеде:

- ережелер базасы;
- жұмыс жадысы;
- инференция механизмі;
- басқару стратегиясы

бірге жұмыс істейді.

Жүйенің негізгі циклі:
сәйкестендіру → таңдау → орындау.
Бірнеше ереже қатар орындалуға дайын болғанда **конфликт жиыны** пайда болады және жүйе конфликтті шешу стратегиясын қолданады.
Forward Chaining фактілерден қорытындыға қарай, ал Backward Chaining мақсаттан қажетті фактілерге қарай қозғалады.
Екінші қарастырған тәсіліміз — **фреймдік модель**.
Фрейм белгілі бір объект, ұғым немесе жағдай туралы білімді құрылымдайды.
Фреймнің негізгі элементі — **слот**.
Слоттың мәні және оны сипаттайтын **фасеттері** болуы мүмкін.
Фреймдерде:

- әдепкі мәндер;
- иерархия;
- мұрагерлік;
- ерекшеліктер;
- процедуралық тіркемелер

қолданылуы мүмкін.
Ең басты айырмашылықты тағы бір рет қайталаймын:
Өнімдік модель: «Қандай жағдайда қандай қорытынды немесе әрекет керек?»
Фреймдік модель: «Объект қандай және оның қасиеттері қандай?»
Ал нақты интеллектуалды жүйеде бұл екеуі бір-біріне қарсы емес.
Керісінше, олар бір-бірін толықтыра алады.
Фрейм объект туралы білімді сақтайды.
Өнімдік ережелер сол білімнің негізінде шешім қабылдайды.
Сондықтан бүгінгі дәрістің негізгі тұжырымын бір сөйлеммен айтсақ:
Фрейм білімнің құрылымын көрсетсе, өнімдік ереже сол білімді қалай пайдалану керектігін көрсетеді.

33. Бақылау сұрақтары

1. Өнімдік модель дегеніміз не?
2. Өнімдік ереженің антецеденті мен консеквенті нені білдіреді?
3. Өнімдік жүйенің негізгі компоненттері қандай?
4. Жұмыс жадысының қызметі қандай?
5. «Сәйкестендіру – таңдау – орындау» циклі қалай жұмыс істейді?
6. Конфликт жиыны дегеніміз не?
7. Конфликтті шешудің қандай тәсілдері бар?
8. Forward Chaining қандай жағдайда тиімді?
9. Backward Chaining қандай жағдайда тиімді?
10. Өнімдік модельдің негізгі артықшылықтары қандай?
11. Ережелер саны өте көп болғанда қандай мәселелер пайда болады?
12. Фрейм дегеніміз не?
13. Слот дегеніміз не?
14. Слот мәні мен фасеттің айырмашылығы қандай?
15. Әдепкі мән не үшін қолданылады?
16. Фреймдердегі мұрагерлік қалай жұмыс істейді?
17. Типтік фрейм мен экземплярдың айырмашылығы қандай?
18. Фрейм мен ООП классының ұқсастығы және айырмашылығы қандай?
19. Процедуралық тіркеме дегеніміз не?
20. Өнімдік және фреймдік модельдерді бір жүйеде қалай біріктіруге болады?

34. Сабақ соңындағы шағын тапсырма

Енді бүгінгі екі модельді бір тапсырмада біріктірейік.

Университет студенттерінің ғылыми жобаға қатысуын басқаратын интеллектуалды жүйе жасау керек делік.
Студент **Меруерт** туралы мына ақпарат белгілі:

- курс — 3;
- GPA — 3.45;
- ғылыми қызығушылығы — жасанды интеллект;
- академиялық қарызы жоқ;
- ғылыми жобаға өтініш берген.

Бірінші тапсырма:
СТУДЕНТ фреймін құрыңыздар.
Кемінде бес слот анықтаңыздар.
Екінші тапсырма:
Меруертті осы фреймнің нақты экземпляры ретінде көрсетіңіздер.
Үшінші тапсырма:
Мына талапты өнімдік ережеге айналдырыңыздар:
«Егер студенттің GPA көрсеткіші 3.0-ден төмен болмаса және академиялық қарызы болмаса, онда ол ғылыми жобаға қатысуға ұсынылa алады».
Төртінші тапсырма:
Қай мәліметтер фреймде сақталатынын және қай білім өнімдік ереже түрінде берілетінін түсіндіріңіздер.
Соңғы сұрақ:
Неліктен осы есепте тек фреймді немесе тек өнімдік ережелерді қолданғаннан гөрі, екі модельді бірге қолдану ыңғайлы?
Осы сұраққа дұрыс жауап бере алсаңыздар, бүгінгі тақырыптың негізгі идеясын түсіндіңіздер.
Келесі тақырыптарда осы модельдердің интеллектуалды және эксперттік жүйелердегі нақты шешім қабылдау механизмдерімен қалай байланысатынын жалғастырамыз.