# Problēmvide

**Ikvienas informācijas sistēmas izstrādei ir jāveic problēmvides analīze un jāizveido problēmvides projektējums. Tas ietver:**

**1) prasību noteikšanu;**

**2) mērķu definēšanu;**

**3) problēmvides procesu analīzi;**

**4) vērtēšanu, kādus datus izmantosim un kādus glabāsim;**

**5) datu modeļu veidošanu;**

**6) datubāzes shēmas ģenerēšanu;**

**7) informācijas sistēmas konstruēšanu;**

**8) lietojuma daļas izveidošanu.**

**1. apakšvienība 2. apakšvienība**



D2

D01

D4

D3

D1



P3

P2

P1







D04



D02

D03

D10

D9

D8

D7

D6

D5



P8

P7

P6

P5

P4



D13

D11

D11



D12

D12



P10

P9

**3. apakšvienība 4. apakšvienība**



D16

D18

D17

D15

D14

P12

P11



**Organizācijas vadības sistēma: apakšvienības, darbinieki (P1, P2, P3,...), dokumentu (D1, D2, D3, ...) apmaiņa un dokumentu krātuves (D01, D02 ,D03,...).**

# Datu bāzes projektēšana. Trīs līmeņu datu diagramma

**Datu konceptuālais modelis (pilnīga neatkarība no DB realizācijas)**

**Lietotājs (IS pasūtītājs),**

**sistēmu analītiķis**

### Transformācija

### DB projektētājs

**Datu loģiskais modelis (izvēlēts datu bāzes vadības sistēmas tips)**

### 

### Transformācija

**DB projektētājs**

**Datu fiziskais modelis (izvēlēta konkrēta datu bāzes vadības sistēma)**

,

### DB administrators

### Realizēšana

**Datu bāze**

Datu modelēšanā izveido abstraktu reālās pasaules modeli, kas parāda, kā dotajā problēmvidē datus lietos un kā šos datus attēlos. Izdalāmi trīs datu modeļu tipi:

**1. Konceptuālie datu modeļi** (conceptual data models) – šis modelis ir neatkarīgs no tā, kādu datubāzu vadības sistēmu, tas ir, relāciju vai objektu-relāciju, izmantos informācijas sistēmā;

**2. Loģiskie datu modeļi** (logical data models) – šie modeļi ir atkarīgi no izvēlētās datubāzu vadības sistēmas, tas ir, ir būtiski, vai tā ir relāciju vai relāciju-objektu datubāzu vadības sistēma. Loģiskajā datu modelī attēlo datu realitātes, datu realitātes aprakstošos atribūtus, attieksmes starp realitātēm;

**3. Fiziskie datu modeļi** (physical data model) – šos modeļus izmanto konkrētai izvēlētai datubāzu vadības sistēmai, piemēram, Oracle datubāzu vadības sistēmai, modeļi ir detalizētāki kā loģiskie datu modeļi.

**Datu konceptuālais modelis**

1. Datu konceptuālais modelis tiek veidots **neatkarīgi no datu bāzes realizēšanas varianta** (relāciju vai objektu, vai relāciju-objektu datu bāze).

2. **Peters Čengs** (Peter Cheng) – relāciju datu bāzes konceptuālās projektēšanas pamatprincipu autors. Izveidoja **Entity Relationship (ER) diagrammu**. Viņš šādi definēja realitāti:

**"An entity is anything about which the system needs to store information."**

**3. Būtība jeb realitāte** **(entity)** ir reālās pasaules objekts, subjekts, process vai notikums, par kura realizācijām jeb eksemplāriem informācijas sistēmā nepieciešams glabāt un izgūt datus.

Pirmā literatūra par ER diagrammām

1. Chen P.P. The entity-relationship model: toward a unfied view of data, ACM Trans. On Database Systems, 1:1 (1976), p. 9 –36.

# Datu bāzes izveides un projektēšanas problēmas

Informācijas sistēmām (IS) ir divas svarīgākās komponentes: datu glabātuves (data storage) un lietojumprogrammas. Datu glabātuvē atrodas apskatāmās problēmu vides dati, no kuriem pēc lietotāja norādēm tiek izgūti un vizualizēti viņam vajadzīgie. Vairums IS realizē tikai datu glabāšanu, samērā vienkāršu datu izgūšanu un iegūto datu prezentāciju jeb izvadi (lietojumprogramma). Tiek izmantotas arī IS, kuras veic sarežģītas datu apstrādes procedūras un algoritmus: optimizācijas IS, klasifikācijas IS, prognozēšanas IS, IS ar dedukcijas iespējām.

Datu glabāšanai tiek izmantotas dažādas tehnoloģijas. Pēdējos divdesmit gados nenoliedzami populārākā datu glabāšanas tehnoloģija ir datu bāzes (pēc dažādu aptauju rezultātiem IS-ās vairāk nekā 90% gadījumos tiek lietota tieši šī tehnoloģija). Datoru programmatūru tirgū tiek piedāvātas vairāk nekā 100 dažādu iespēju līmeņu datu bāzes vadības sistēmas, kuras ļauj ērti un vienkārši realizēt nepieciešamās datu bāzes (DB).

DB izveidošana ir grūts un ļoti atbildīgs uzdevums. Galvenās problēmas ir saistītas nevis ar tehnisko realizāciju, bet gan ar DB satura definēšanu. Lai kvalitatīvi noteiktu DB saturu:

1. ir jāveic lietotāju aptaujas, kurās jānoskaidro, apskatāmajā problēmu vidē izmantojamos datus;
2. ir jāveic datu analīze un kārtošana, lai tie veidotu vienotu nepretrunīgu sistēmu;
3. ir jāizveido atbilstošas, efektīvas datoratmiņas datu glabāšanas struktūras.

Pēc iespējas vienkāršāk un skaidrāk jāmodelē apskatāmās problēmu vides datu struktūras:

1) datu vienību vērtībām jābūt nedalāmām un viena tipa (tikai skaitļi, teksta rindas un t.t.);

2) datu grupējumiem jābūt loģiski pamatotiem un viegli izprotamiem;

3) datu grupējumu saitēm jābūt skaidrām un viennozīmīgām.

Informācija par apskatāmās problēmvides datiem tiek iegūta, aptaujājot potenciālos lietotājus un analizējot eksistējošos dokumentus. Lai veiksmīgi un kvalitatīvi varētu iegūt informāciju no lietotājiem, labi jāizprot loģiskās datu struktūras, kurās viņi veido savu izpratni:

1. IS veidošanas sākumposmā tika uzskatīts, ka lietotāji IS iedomājas kā “melno kasti” un operē ar ieejas un izejas datiem;
2. vēlāk dominēja uzskats, ka lietotāji datus grupē un analizē kā tabulveida struktūras (divdimensiju tabulas);
3. attīstoties objektorientētai programmēšanai, tika izvirzīta tēze, ka nedrīkst datus apskatīt atsevišķi no to izmantojuma. Dati tika grupēti objektos atkarībā no to izmantojuma. Veidojās objektu hierarhijas, vienāda tipa objektiem piekārtojot nepieciešamo funkcionalitāti (apstrādes metodes).
4. nepieciešamība veidot sarežģītas analīzes sistēmas atklāja, ka daudzos gadījumos lietotāji savā iztēlē izmanto daudzdimensiju struktūras (datu hiperkubi). Dimensijas kalpo vaicājumu noteikumu definēšanai, lai atlasītu attiecīgās datu kopas, kuru agregētās vērtības ir nepieciešamas.

Izmantojot šos priekšstatus, tika izveidoti vairāki datu konceptuālie modeļi, kuri tika un tiek izmantoti datu bāzes projektēšanā. Populārākais no šiem modeļiem ir P. Čena realitāšu-saišu diagramma (Entity Relationship diagram) jeb ER diagramma, kas dažādās izpildes modifikācijās ir visplašāk lietotais DB datu konceptuālais modelis. Otrs populārākais modelis DB datu modelēšanai ir UML valodas klašu modelis. Šis modelis ir orientēts uz datu objektu un metožu priekšstatu.

Arī DB datu glabāšanas struktūras ir veikušas garu attīstības ceļu. Pirmās DB izmantoja datu hierarhiskās, tīklveida un CODASYL (tīkla un hierarhijas struktūru hibrīdi) struktūras. Plaša DB tehnoloģijas izmantošana iesākās līdz ar relāciju struktūru izmantošanu datu glabāšanai. Turpinājumā tika ieviestas datu objektu struktūras (objektu DB un relāciju-objektu DB) un daudzdimensiju datu glabāšanas struktūras.

Datu bāzes automatizētai projektēšanai tika izveidoti speciālas dialoga programmu paketes, kuras ļauj grafiski definēt DB konceptuālo modeli un veikt tā transformāciju uz norādītas datu bāzes vadības sistēmas datu glabāšanas struktūrām. Tehnoloģija, kad IS projektētājs savas vēlmes un priekšstatus definē grafisku diagrammu veidā un pēc tam sistēma automātiski veic DB datu glabājamo struktūru un lietojumprogrammu ģenerēšanu, tika nosaukta par CASE (Computer Aided System Engineering) tehnoloģiju. Mūsdienās tiek piedāvātas vairāk nekā 100 programmu pakešu, kas realizē šo tehnoloģiju. Parasti šīs programmu paketes sauc par CASE rīkiem. Projektētājiem ir iespējams izvēlēties savām vēlmēm vislabāk atbilstošo rīku.

Diemžēl, lai kvalitatīvi un efektīvi lietotu CASE tehnoloģiju un CASE rīkus ir daudz neatrisinātu problēmu:

1) esošo DB datu konceptuālo modeļu grafisko elementu klāsts ir nepilnīgs, dažādos CASE rīkos tas ir atšķirīgs un bieži vien nepietiekams;

2) DB konceptuālā modeļa elementi parasti tiek transformēti uz visvienkāršākām datu glabāšanas struktūrām, kas nedod vēlamo efektivitāti. Bieži vien tas rada arī nevajadzīgi sarežģītu datu glabāšanas interpretējumu.

3) neeksistē pilnvērtīgas un automātiskas DB konceptuālo modeļu elementu transformācijas uz efektīvākajām relāciju DB, relāciju-objektu DB un objektu DB datu glabāšanas struktūrām. Īpaši varētu atzīmēt, ka ļoti vājš transformāciju nodrošinājums ir relāciju-objektu DB-ēm, kuras pēdējos gados ieguvušas plašu popularitāti savu iespēju un kvalitātes dēļ. Tāpēc ļoti svarīgi ir izvērtēt konceptuālo modeļu elementu esošo transformāciju klāstu un nepieciešamos transformāciju papildinājumus.

### Pirmais datu konceptuālais modelis

**Pirmo datu modeļa grafisko attēlošanas veidu – realitāšu/saišu (Entity-Relationship (ER)) diagrammu izstrādāja Peters Čens 1976. gadā.**



# Entity-Relationship diagrammās izmantotie jēdzieni

**Apzīmējums**

**Nosaukums**

# 1. Datu grupējums jeb stingra realitāte (strong entity)

2. **Atribūts**

**3. Daudzvērtīgs atribūts**

**4. Salikts atribūts**

**Apzīmējums**

**Nosaukums**

**5. Saite**

**Apzīmējums**

**Nosaukums**

# 6. Identificējoša saite

# Entity-Relationship diagrammā izmantotie jēdzieni (turpinājums)

**7. Saite starp vairākām realitātēm (piemēram, trijām)**

**Apzīmējums**

**Nosaukums**

# 8. "Vājā" realitāte (weak entity)

**Apzīmējums**

**Nosaukums**

# 9. Pilnīga līdzdalība (total participation) - katrs realitātes X eksemplārs ir savienots ar vismaz vienu realitātes Y eksemplāru.

## Y

**X**

**Saite Z**

# 10. Daļēja līdzdalība (partial participation) - tikai daži realitātes X eksemplāri ir savienoti ar vienu realitātes Y eksemplāru.

**Saite Z**

**X**

## Y

# Entity-Relationship) diagrammā izmantotie jēdzieni (turpinājums)

# 11. Kardinalitāte:

**Viens ar daudziem ( 1 : N ). Vienam Studentam ir daudzi Pasniedzēji.**

**Saite Z**

**Students**

## Pasniedzējs

**1**

**N**

**Daudzi ar vienu ( N : 1 ). Daudziem Studentiem ir viens Pasniedzējs.**

**Saite Z**

**Students**

## Pasniedzējs

**N**

**1**

**Viens ar vienu ( 1 : 1 ). Vienam Studentam ir viens Pasniedzējs.**

**Saite Z**

**Students**

## Pasniedzējs

**1**

**N**

**Daudzi ar daudziem ( M : N ). Vienam Studentam ir daudzi Pasniedzēji un viens Pasniedzējs ir daudziem studentiem (1 : N + M : 1).**

**Saite Z**

## Pasniedzējs

**Students**

**N**

**M**

**M**

**Saite Z**

**Saite ar sevi.**

**Pasniedzējs**

**N**

## Bāzes klases (super classes) un apakšklases (sub classes)

In some cases, and entity type has numerous sub-groupings of its entities that are meaningful, and need to be explicitly represented, because of their importance.

For example, members of entity Employee can be grouped further into Secretary, Engineer, Manager and Technician.

The set listed is a subset of the entities that belong to the Employee entity, which means that every entity that belongs to one of the sub sets is also an Employee.

Each of these sub-groupings is called a **subclass**, and the Employee entity is called the **super-class**.

   ||||   -------------|    |||
  -Ssn|----EMPLOYEE    --||Name||-
  -||||-  -------|-----   |||||||
             | --|- |
            -|||d--||-
  --------------| ------------|
  -SECRETARY----| -ENGINEER----
||||||||||||||||| -||||||||||||
|||TypingSpeed|||- ||EngType|||-


An entity cannot only be a member of a subclass; it must also be a member of the super-class.

An entity can be included as a member of a number of sub classes, for example, a Secretary may also be a salaried employee, however not every member of the super class must be a member of a sub class.

# Entity-Relationship un Extended Entity-Relationship diagrammas

# Enhanced (extended) ER Diagrams (EER):

1. **contain all the basic modeling concepts of an ER diagram;**
2. **adds additional concepts:**
   * **specialization/generalization;**
   * **subclass/super class;**
   * **categories;**
   * **attribute inheritance.**
3. **extended ER diagrams use some object-oriented concepts such as inheritance;**
4. **EER is used to model concepts more accurately than the ER diagram.**

# Extended Entity-Relationship) diagrammā izmantotie jēdzieni

# 12. Mantošana (inheritance)

# Vispārinājums ("is a"). Apakšklase manto

**Bāzes klase**

**Apakšklase**

**Apakšklase**

# bāzes klases atribūtus un saites.

# Specializācija ("is a"). Bāzes klasei pievienojas

**Bāzes klase**

**Apakšklase**

**Apakšklase**

# apakšklases atribūti un saites.

**Agregācija ("part of"). Daļas un veselais**

**Bāzes klase**

**Apakšklase**

**Apakšklase**

**eksistē patstāvīgi. Viena daļa var attiekties**

**uz vairākiem veseliem.**

**Kompozīcija ("part of"). Daļas**

**Bāzes klase**

**Apakšklase**

**Apakšklase**

**neeksistē bez veselā.Absolūtā specializācija (total specialization)** – katram bāzes klases eksemplāram ir vismaz viens apakšklases eksemplārs.

**Grāmata**

**Cits tips**

**Romāns**

**Dzejoļu grāmata**

**Daļējā specializācija (partial specialization)** – ne katram bāzes klases eksemplāram ir atbilstošs pakārtototās klases eksemplārs.

**Dzejoļu grāmata**

**Grāmata**

**Romāns**

**Sadalījums (disjoint)** – katram bāzes klases eksemplāram var būt ne vairāk kā viens atbilstošais elements no apakšklasēm.

**Dzīvnieks**

**Pārklāšanās (overlapping)** – katram bāzes klases eksemplāram var atbilst vairāku apakšklašu eksemplāri.

**1**

**Suns**

**Zirgs**

**Lauva**

**Upe**

**N**

**Strauja**

**Maza**

**Liela**

**Daudzkārtīga mantošana (multiple inheritance)** – apakšklases eksemplāri ir saistīti ar vairākām bāzes klasēm.

**Bāzes klases**

**Cilvēks**

**Dzīvnieks**

**Apakšklase**

**Nāra**

**Apvienojums (union) -** apakšklase ir saistīta ar vairākām bāzes klasēm un apakšklases eksemplārs vienlaikus manto atribūtus un saites tikai no vienas bāzes klases eksemplāra.

**Pakalpojums**

**Produkts**

**Bāzes klases**

**XOR**

**Pirkums (purchase)**

**Apakšklase**

### Konceptuālo datu modeļu veidi

ER modeļī nebija ērta veida, kā attēlot dažādus reālās pasaules jēdzienus, piemēram, kolekcijas, līdz ar to laika gaitā izveidoja citus modeļu veidus, tika paplašināts, realitāšu saišu modelis, izmantoti atšķirīgi grafiskie elementi. Izplatītākie no šiem modeļu veidiem ir:

**1. Information Engineering (IE) modelis.**

**2. Ričarda Barkera modelis.**

**3. IDEF1X modelis.**

**4. Merise modelis.**

**5. Object Role modelis.**

**6. UML tipa modelis.**

**IE tipa modelim**ir vienkāršs un ērts pieraksts. Salīdzinājumā ar Peter Chen izstrādāto modeļa veidu, atšķiras saišu, realitāšu pieraksts, mantošanas struktūru attēlošana. Būtisks IE tipa modeļa trūkums ir atribūtu un unikāli identificējamo lauku attēlošana atsevišķā dokumentā - „realitāšu saraksts” (entity list). Šos elementus un citas raksturiezīmes dokumentā attēlo „burbuļidagrammu” („bubble charts”) veidā[[1]](#footnote-2).



### Ričarda Barkera konceptuālais datu modeļu veids

Viens no plašāk izmantotajiem pierakstiem ir *Ričard Barker*izstrādātais modeļa tips. Šo modeļa tipu izmanto arī Oracle rīkos. Tā īpatnība ir veids, kā pieraksta saišu nosaukumus - tam izmanto prievārdus un vārdu salikumus, kas rezultātā attēlojas teikumu veidā. Modeļa pierakstā saistošs ir atribūtu attēlošanas veids - tos attēlo pašos realitāšu taisnstūrīšos, nevis izdala atsevišķos ovālos. Apakštipi taisnstūrveidā tiek attēloti iekš supertipa taisnstūra, tādēļ pieeja var kļūt nepārskatāma, ja veido attēlojumu vairākos apakšlīmeņos. Modeļa pierakstā arī nav atļauti pārklājošies apakštipi, tomēr šo aizliegumu apiet, izveidojot apakštipu „Citi”[[2]](#footnote-3).



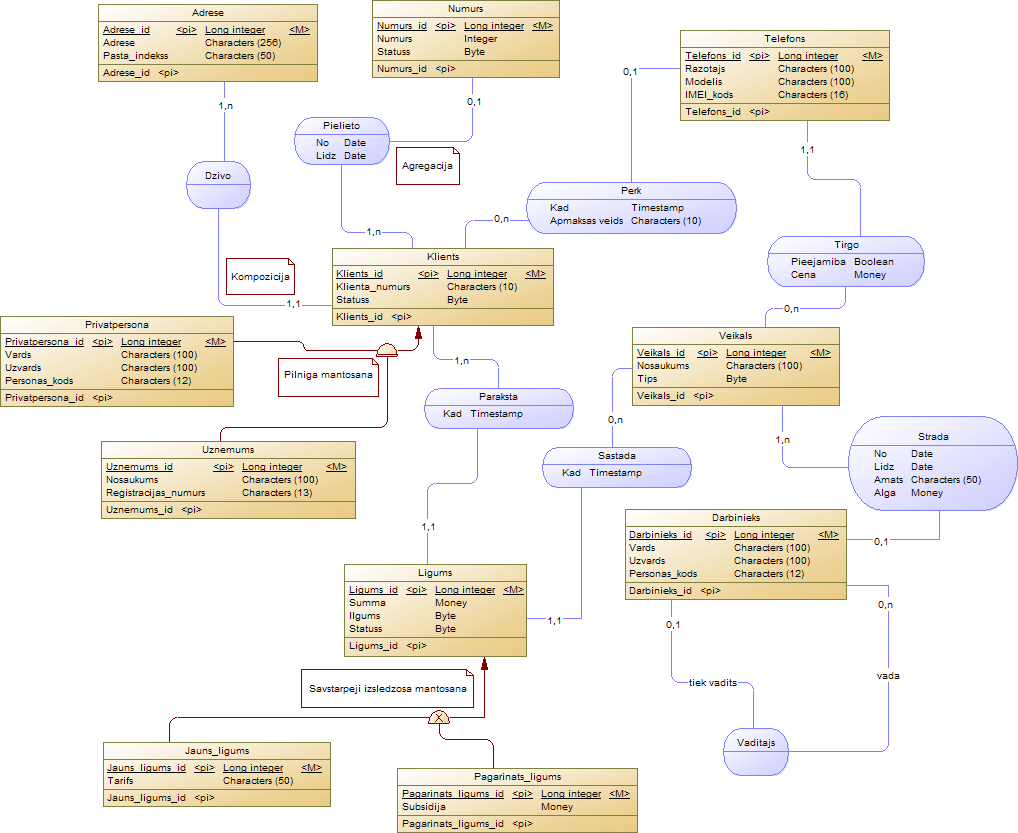
### IDEF1X konceptuālais datu modeļu veids

IDEF1X modelēšanas tipsbūtībā ir fiziskās datubāzes struktūras modelēšanas pieeja, pie kam šī tipa pieraksts ir pārāk sarežģīts un nav ieteicams to lietot datu modelēšanā.



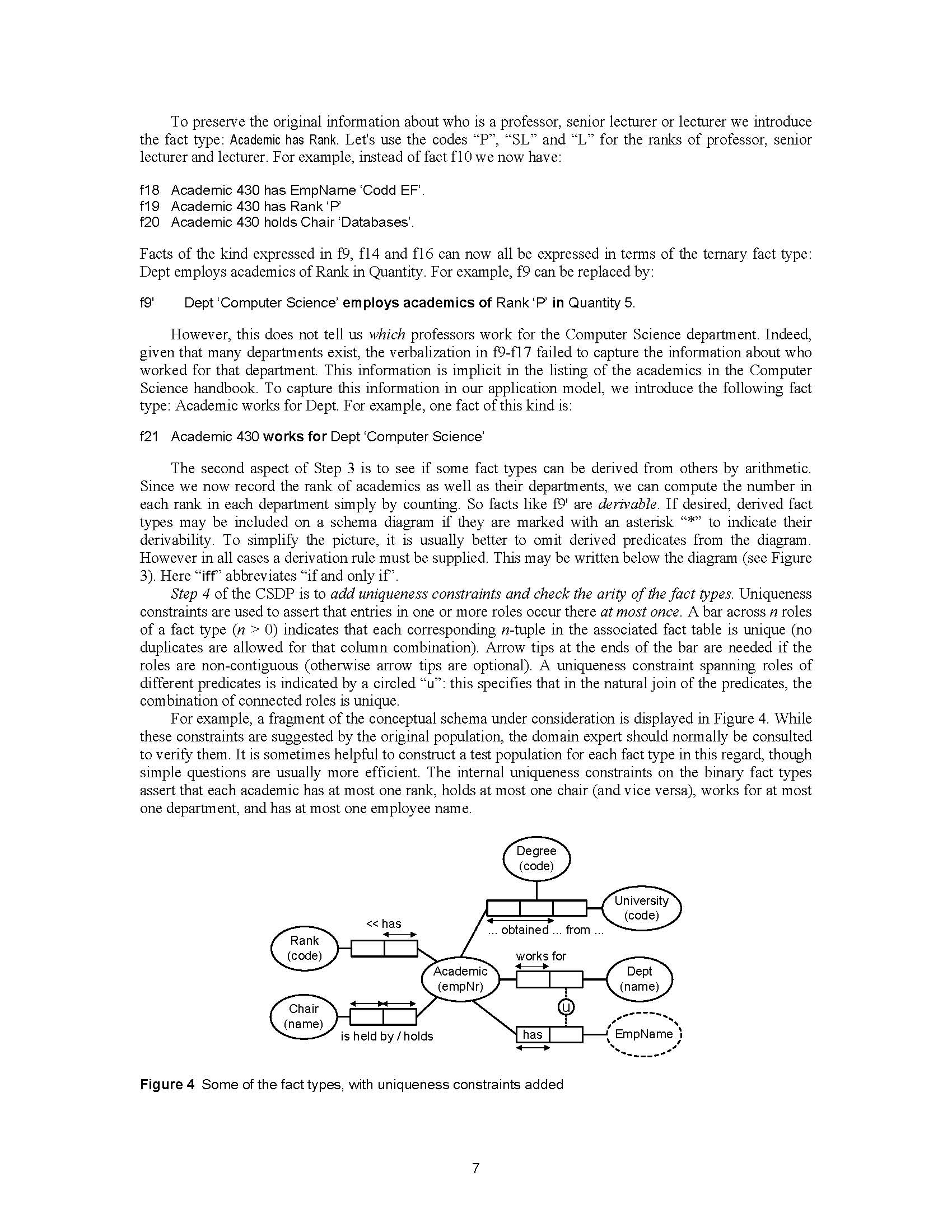
### Merise konceptuālais datu modeļu veids

Merise vēsture sākas ar 1977. gadā, Francijā, kad izstrādes grupa ar nosaukumu „Aachen” (kopa ar vadošo industrijas kompāniju pārstāvjiem) pēc valdības pasūtījuma sāka darbību projektā „Merise”. Tas ilgas 4 gadus: no1977. līdz 1981. gadam. 1979. gads kļūst par oficiālo Merise metodes dzimšanas gadu. Metodes pirmā redakcija tiek publicēta un tās būtība tiek definēta kā: «Merise — IS definēšanas metode»



### Object - Role konceptuālais datu modeļu veids

**Object-Role Modeling** (ORM, latv.: objektu lomu modelēšana), jeb **fact-oriented modeling** – ir alternatīva UML klašu diagrammai. ORM ir izvērsta valoda, lai modelētu datu bāzes modeļus konceptuālajā līmenī. ORM datu modeļi parasti satur biznesa likumus un tos ir vieglāk attīstīt nekā datu modeļus citās pieejās.

****

### Klašu diagrammas konceptuālais datu modeļu veids

Ļoti līdzīgs relāciju/saišu jeb ER diagrammas pierakstam ir UML modeļa tips – klašu diagramma, jo tā pieraksts datu struktūras attēlošanai neatšķiras no relāciju/saišu modeļa. UML modeļa tipapriekšrocības ir:

1. modeļa tipā var detalizētāk aprakstīt atribūtus;

2. var attēlot sarežģītākas situācijas, piemēram, realitāte, R1 var būt saistīta ar 0,3,6-7 vai 9 otras realitātes R2 eksemplāriem.

3. UML modeļa pierakstā tika ieviests būtisks jauns jēdziens – stereotips (stereotype). **Stereotips** (ko apzīmē ar <<>>) ļauj paplašināt UML standarta modeļa tipu un pievienot jaunus apzīmējumus, līdz ar to UML modeļa pierakstu var piemērot atbilstoši projektējumam.

UML modeļa tipa trūkumi ir:

1. lieki simboli specifisku attieksmju attēlošanai – agregācijai un kompozīcijai. Vienkāršāk pieraksts realizēts realitāšu/saišu diagrammās – saites attēlojot kā „Daļa no” vai „Vesels-daļa”. Bez tam specifisko attieksmju simboli neattēlo trigera nosacījumu, kas nosaka, ka nedrīkst dzēst vecāka datus, ja eksistē bērna-saistītie dati.

2. eksistē vairāki pieraksti (tai skaitā Maksimchuk (2001) izstrādātais, Skota Amblera izstrādātais), tomēr neviens no tiem nav pilnīgs.



### Datu modelēšana ar UML un EER diagrammām

Modeļa izstrādei nav noteikta standarta, tomēr objektu-relāciju tehnoloģiju izmantošanai, datu modelī jāvar attēlot objektorientētās pieejas jēdzienus - mantošana, tipi un apakštipi, kopas un apakškopas. Šādus jēdzienus diagrammās var attēlot, lietojot:

1. UML klašu diagrammas jēdzienu;

2. paplašināto realitāšu saišu jeb EER diagrammu, kura, atšķirībā no ER diagrammas, ļauj attēlot arī objektorientētās pieejas jēdzienus.

UML pieejā pirms datu modelēšanas ir jānorāda attēlojamā modeļa veids. Tam izmanto atbilstošo stereotipu vai vienkārši brīva UML pieraksta veidā.

| **UML modeļa stereotips** | **Modeļa veids** |
| --- | --- |
| <<Klašu modelis>>  (<<Class Model>>) | Objektorientētais vai objektu-relāciju modelis |
| <<Konceptuālais datu modelis>>  (<<Conceptual Data Model>>) | Konceptuālais datu modelis  (Conceptual data model – CDM) |
| <<Loģiskais datu modelis>>  (<<Logical Data Model>>) | Loģiskais datu modelis  (Logical data model - LDM) |
| <<Fiziskais datu modelis>>  (<<Physical Data Model>>) | Fiziskais datu modelis  (Physical data model - PDM) |
| <<File>> | Fails |
| <<Hierarchical Database>> | Hierarhiskā datu bāze |
| <<Object-Oriented Database>> | Objektorientētā datu bāze (OODB) |
| <<Object-Relational Database>> | Objektu-relāciju datu bāze |
| <<Network Database>> | Tīkla datu bāze |
| <<Relational Database>> | Relāciju datu bāze (RDB) |
| <<XML Database>> | XML datubāze |

**Datu bāzes jēdzieni**

**Datu bāzes shēma** – datu bāzes datu glabāšanas struktūras objektu un to elementu apraksts.

**Datu bāzes ārējā shēma** – datu bāzes glabāšanas struktūras objektu un to elementu skatījums no lietotāja viedokļa (katram lietotājam var būt sava ārējā shēma). Viņas apvienojot iegūstam datu bāzes shēmu.

**Datu bāzes iekšējā shēma** - datu bāzes glabāšanas struktūras objektu un to elementu skatījums no fiziskās realizācijas viedokļa.

**Datu bāzes eksemplārs (database instance)** – viena datu bāzes realizācija.

**Fiziskā datu bāze** – failu kopa ar datiem.

**Mērķi, prasības**

**Problēmvide**

**Realizēšanas iespēju analīze**

**Problēmas definēšana (galvenie mērķi, projekta robežas, resursi)**

A1

**Problēmvides modelis (datu plūsmas diagramma, sistēmas plūsmkartes,...) Informācija par datiem un izpildāmām funkcijām**

A1

**IO2**

**IO1**

A2

A1

A3

**IO3**

A1

A1

A1

**Sākotnējo DB datu tabulu iegūšana (pārveidojumu likumi)**

**Datu grupēšana (ER diagrammas izveidošana)**

**Uzlabota problēmvides modeļa izstrāde (jauna darbības algoritma izveidošana)**

IO1,IO2,IO3 -informācjas objekti, A1,A2, ... atribūti.

**DB datu tabulu kopējo īpašību pārbaude**

**DB datu tabulu kvalitātes pārbaude un dekompozīcija (dalīšana)**

**Programmas izpildāmo funkciju un to pakārtotības noteikšana**

**Funkcionālo saišu shēmas vienkāršošana (Pārveidojumu likumi)**

**Datu funkcionālo atkarību diagrammu izveidošana**

**DB struktūras definēšana (tiek izmantota datu definēšanas valoda)**

**Programmas moduļu koka izveidošana (struktūrkartes)**

**Datu bāzes sistēma**



**Programmu koda ģenerēšana**

**Dialoga interfeisa projektēšana**

**Moduļu aprakstu izveidošana (darbību diagrammas, struktūrshēmas)**

**Moduļu kvalitātes pārbaude un dekompozīcija vai agregēšana**

**Datu bāzes sistēmas projektēšanas pamatdarbību secība**

1. [Scot W Ambler „Agile Database Techniques”] [A comparison of Data Modeling Techniques, David C Hay <http://www.essentialstrategies.com/documents/comparison.pdf>]. [↑](#footnote-ref-2)
2. Tā pati atsauce, kas IE. [↑](#footnote-ref-3)