

ПРЕДМЕТ

< ЗДРАВСТВЕНО ИНФОРМАЦИОНИ СИСТЕМИ >

Предавање број 10

**<** **ТЕХНОЛОГИЈЕ КОЈЕ ПОДРЖАВАЈУ ЗДРАВСТВЕНЕ ИНФОРМАЦИОНЕ СИСТЕМЕ >**

|  |  |  |  |
| --- | --- | --- | --- |
| Недеља | Наставна јединица | Тематске јединице | Резултат – знања или вештине које студент треба да добије |
| 10 | Технологије које подржавају здравствене информационе системе | Системски софтвер. Управљање и приступ подацима. Моделовање релационих података. | Упознавање са технологијама за подршку здравственим информационим системима. |

Copyright © 2018 – Факултет медицинских наука Универзитета у Крагујевцу. Сва права задржана. Без претходне писмене дозволе од стране Факултета медицинских наука забрањена је репродукција, трансфер, дистрибуција или меморисање неког дела или читавих садржаја овог документа, копирањем, снимањем, електронским путем, скенирањем или на било који други начин.

Copyright © 2018 – Faculty of Medical Sciences of University of Kragujevac. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying,, recording, scanning or otherwise, without the prior written permission of Faculty of Medical Sciences.

**САДРЖАЈ**

[Технологије за подршку здравственим информационим системима 2](#_Toc5554142)

[Системски софтвер 2](#_Toc5554143)

[Оперативни системи 3](#_Toc5554144)

[Интерфејс 3](#_Toc5554145)

[Управљање и приступ подацима 4](#_Toc5554146)

[Релационе базе података 4](#_Toc5554147)

[Моделовање релационих података 5](#_Toc5554148)

Предавање бр. 10

**<ТЕХНОЛОГИЈЕ КОЈЕ ПОДРЖАВАЈУ ЗДРАВСТВЕНЕ ИНФОРМАЦИОНЕ СИСТЕМЕ>**

# Технологије за подршку здравственим информационим системима

## Системски софтвер

До ове тачке смо разговарали о апликацијама за подршку здравственим информационим системима без разматрања технологија на којима почивају. У овом делу почећемо са општом дискусијом о софтверу, а затим дефинисати програмске језике, оперативне системе и моторе интерфејса.

Постоје две основне врсте софтвера: *системски* софтвер и *апликативни* софтвер. Ове две врсте софтвера имају заједничку карактеристику - оба представљају низ рачунарских програма. Запамтите да на најосновнијем нивоу рада рачунар препознаје две ствари, електрични импулс који је укључен и електрични импулс који је искључен; ови сигнали су често представљени као 0 и 1 (или битови). Људски програмер мора да напише програмски код који преводи жеље корисника у рачунарске акције. Данас постоји много различитих програмских језика и они настављају да се развијају.

*Машински језици* су најстарији компјутерски програмски језици. Програмери машинског језика морали су буквално да преводе сваки карактер или оператер у бинарни код, приказан као група нула и јединица. Машински језици се често називају језицима прве генерације. На срећу, до педесетих година прошлог века, развијени су збирни језици, језици друге генерације, што поједностављује програмирање машинског језика. Процедурални програмски језици (трећа генерација), на пример, FORTRAN и COBOL, дошли су кратко након збирних језика, омогућавајући програмерима да напишу рачунарске програме, а да нису забринути за ручно писање машинског језика. Данас језици четврте генерације (*fourth-generation language*, 4GL), који имају многе унапред програмиране функције, омогућавају појединцима да развијају апликације без писања иједне линије програмског кода. Софтвер креира код у позадини, невидљив са становишта програмера. У одељку о управљању подацима размотрићемо структурирани језик упита (*structured query language*, SQL), што је пример 4GL.

Две друге врсте програма који се данас често користе су визуелно програмирање и објектно-оријентисано програмирање. Најчешћи тип визуелног програмирања је Microsoft Visual Basic, који дозвољава програмерима да виде коначни визуелни изглед апликације, као што су тастери, менији за скроловање и прозоре, док развијају апликацију. Објектно-оријентисани језици разликују се од традиционалних процедуралних језика јер омогућавају програмеру да ствара објекте који укључују операције (методе) повезане са подацима. На пример, објекат *master patient index* (MPI) би садржавао и MPI податке, као што су број медицинског документа сваког пацијента, презиме, име и тако даље, као и процедуре које користе ове податке, као што је додељивање броја медицинског документа, проналазак имена пацијената на основу броја медицинског документа, и тако даље. Објектно-оријентисани језици омогућавају поновно коришћење комада кода и олакшавају одржавање програма. Заједнички објектно-оријентисани програмски језици су C++ и Java.

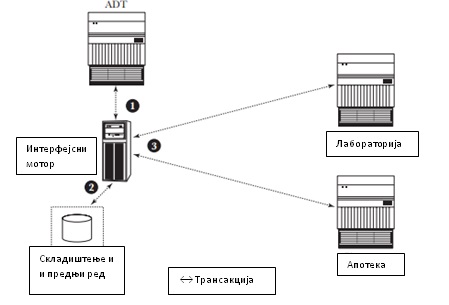
### Оперативни системи

Системски софтвер је низ програма који обављају основне рачунарске функције: на пример, управљање корисничким интерфејсом, датотекама и меморијом. Такође, системски софтвер управља периферијом рачунара, као што су штампачи, монитори и други уређаји. Системски софтвери омогућавају оснивачима да направе апликације без потребе за мануелним кодирањем основних рачунарских наредби. Најважнија компонента системских софтвера је *оперативни систем*. Оперативни систем се учитава када се компјутер укључи и он управља свим програмима које компјутер накнадно користи. Уобичајене врсте оперативних система су Windows (у неколико различитих верзија), Mac OS, Unix и Linux.

Оперативни системи могу бити *власнички* и *отвореног извора*. Власнички оперативни системи, као што су Windows и Mac OS, се купују, а прави изворни код (програм) није доступан купцима. Најпопуларнији оперативни системи су власнички. Међутим, 90-их година прошлог века, појавили су се оперативни системи отвореног извора (или невласнички) када је фински студент Линус Торвалд развио варијанту оперативног система Unix, под називом Linux. Торвалд никада није затражио права на овај оперативни систем и зато је доступан широм интернета. Linux је стекао популарност захваљујући подршци компанија комерцијалних софтвера.

### Интерфејс

Интерфејс је "софтверски програм дизајниран да поједностави креирање и управљање интерфејсом између апликационих система". Интерфејси између апликација постали су све важнији јер су системи здравствене заштите прешли на више интегрисане архитектуре. Корисници су желели да њихове различите апликације буду у могућности да комуницирају једне са другима. Хтели су да елиминишу потребу за уносом демографских података пацијента више пута у засебне системе, на пример. Заправо, корисници су почели да траже јединствени систем пријаве тако да могу приступити свим информацијама које су им потребне преко једног корисничког интерфејса.

**

*Општи приказ управљања интерфејсом*

Интерфејс је у ствари, облик средњег софтвера (*middleware*), класе софтвера који ради "између" или "у средини" апликација и оперативних система. Други примери middleware софтвера су апликације које проверавају вирусе, медицинске логичке процесоре и софтвер за шифровање података. Типични интерфејс функционише у три основна корака.

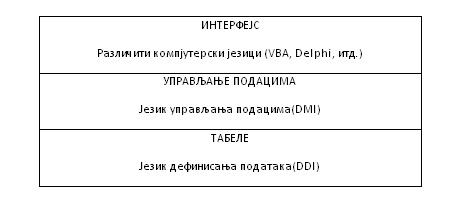
Слика изнад илуструје типичну трансакцију од једног ка другом, која укључује пријем у болницу, отпуштање и трансфер (ADT) систем. Овде, ADT систем треба да комуницира са системима лабораторије и апотеке у којима је пацијент примљен. ADT систем шаље поруку са релевантним демографским и рачунским детаљима интерфејсу. Интерфејс прима поруку, обрађује је, уколико је потребно и ставља га у ред или на листу чекања за испоруку у системе лабораторије и апотеке. Порука се накнадно прослеђује из реда у те системе. Неки интерфејси могу да раде *many-to-many* трансакције (трансакције од многих ка многима), као и *one-to-many* трансакције (трансакције од једног ка многима). Поруке су примљене од стране интерфејса из више система и затим се прослеђују у више система.

## Управљање и приступ подацима

Све апликације за здравствену заштиту које су до сада разматране захтевају податке. Електронски систем здравствених записа (ЕМR) ослања се на свеобухватне базе података, исто као и друге клиничке апликације. Подаци морају бити ускладиштени и одржавани тако да се могу преузети и користити у оквиру ових апликација. У овом делу разматрамо уобичајене типове база података и системе управљања базама података са којима су повезани. Већина наших дискусија усмеренo је на релационе базе података јер је то врста базе података која је најчешће развијена данас. Две старије врсте база података, хијерархијске и мрежне (не сме се мешати са рачунарском мрежом), можда и даље постоје у организацији здравствене заштите као компоненте застарелих апликација, али с обзиром да више немају значајно присуство на тржишту база података, о њима се овде и не дискутује. Четврти тип базе података, објектно-оријентисана база података, добила је велику пажњу у литератури током последњих година. Иако "чиста" објектно-оријентисана база података још увек није заступљена на тржишту здравствене заштите, постоје апликације са објектно оријентисаним компонентама изграђених са релацијским базама података. Ова врста хибридне базе података се назива објектно-релациона база података.

### Релационе базе података

Релационе базе података су развијене почетком 1970-их. Међутим, ове ране релационе базе података нису биле практичне због велике количине процесне снаге која је неопходна. Како су компјутери постали моћнији 1980-их и 1990-их, улога релационих база података постала је значајнија. Данас је релацијска база података доминантан тип базе података која се користи у здравству и пословању. Релацијска база података се имплементира кроз систем за управљање релацијским базама података (RDBMS). Microsoft Access је пример RDBMS за десктоп рачунарство; Оracle, Sybase и Мicrosoft SQL server су примери робуснијих RDBMS-ова који се користе се за развој већих апликација. Апликација развијена користећи RDBMS има три различите компоненте или слојa (Слика испод).

*****Слојеви у систему управљања релационом базом података*

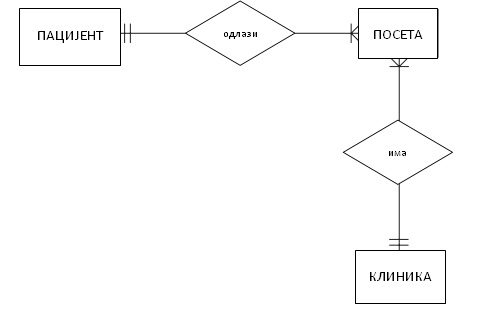
Интерфејс је развијен користећи софтвер као што су Visual Basic или Јаvа. У Microsoft Access-у овај слој је креиран са Visual Basic-oм за апликације (VBА), који је уграђен у Access пакет и користи се за креирање форми и извештаја који чине већину корисничког интерфејса. Доњи слој RDBMS-а је креиран са посебном врстом софтвера, језиком за дефинисање података (*data definition language*, DDL). DDL креира табеле структуре базе података као и однос између различитих табела. Свака табела може се посматрати као фајл, при чему је сваки ред у табели запис, а свака колона је поље или податак. Између таблица података и интерфејса, RDBMS има слој - управљање подацима (*data manipulation*, DMI). Функције овог слоја врше се управљањем података за језик (DML). DML је софтвер који омогућава кориснику да преузима, упита, ажурира и уреди податке у основним табелама.

Језик који се најчешће користи за функције DDL и DML у релацијама базе података је структурирани језик упита (SQL). SQL је пример 4GL. Корисник или програмер одређује шта се мора урадити, али не како то треба учинити. Другим речима, програмeр не мора да пројектује сложене радње које рачунар користи када извршава SQL наредбу. SQL је препознат као де факто стандард за функционисање релационе базе података. Уобичајени RDBMS производи подржавају неку врсту SQL-а, али многи од њих такође користе проширења на основни језик.

Да би се даље подржала интероперабилност између база података користећи различито управљање системима, отворен је *Оpen Database Connectivity* (ODBC) стандард за интерфејс апликацијског програма базе података (АPI). Овај стандард је уско усклађен са SQL, развила је SQL Аccess Group, а први пут је објављен 1992. ОDBC дозвољава програмима да користе SQL захтеве без потребе за познавањем власничких интерфејса база података. Коришћење база података које су у складу са ODBC-ом стандард омогућава здравственој организацији да лакше интегрише своје базе података. Организација може преместити податке из PC-базираних апликативних програма или ОD базе података у веће базе података и обрнуто, на пример.

## Моделовање релационих података

Слика испод представља пример дијаграма односа ентитета (*entity relationship diagram*, ERD), који графички приказује табеле и односе у једноставној релационој бази података. Моделовање података је битна алатка за дизајнере базе података. Иако је комплетна дискусија о ЕRD-у (и моделовању података уопште) изван оквира ових предавања, ми ћемо овде издвојити неколико кључних компонената, јер се ови модели често користе, не само као нацрт за изградњу базе података, већ и као алат за комуникацију између дизајнера и евентуалних корисника. Стога може бити неопходно за извршиоцe здравствене заштите да разумеју њихове компоненте.

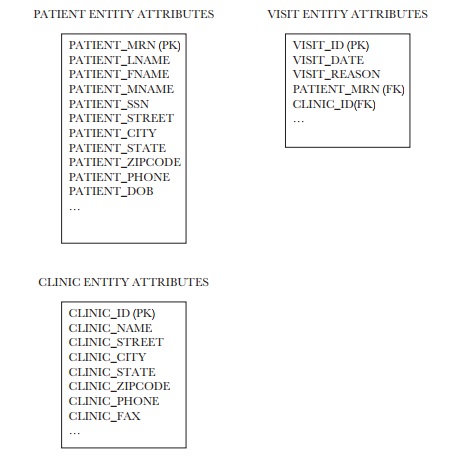


*Дијаграм односа ентитета*

***Ентитети****.* Правоугаоници у ERD дијаграму представљају ентитете. Ентитет представља особу, место или ствар чије податке организација жели да чува. Ентитети приказани у коначној верзији ERD-а биће трансформисани у табеле у релационој бази података. На слици изнад приказан је пример структуре која може бити креиран из ентитета КЛИНИКА. (Имајте на уму да су ти примери прилично поједностављени и намењени да илуструју опште концепте уместо да представе стварну праксу креирања базе података.)

***Атрибути***. Атрибути ERD-а могу бити приказани као овални облици који се шире од ентитета; међутим, чешће се могу видети наведени ентитети засебно или унутар правоугаоника ентитета (Слика изнад). Атрибути се трансформишу на поља података. Сваки ентитет у ЕRD-у мора имати јединствени идентификатор, назван свој Примарни кључ. Примарни кључ се не може дуплирати у оквиру табеле и не може садржати нулту вредност. Примарни кључ се такође користи за повезивање ентитета како би се формирале везе.

***Везе*.** Везе унутар ЕRD дијаграма могу бити приказане као дијамантски облици. Име везе је обично глагол. Постоје три могуће везе између било која два ентитета: један-ка-једном, један-ка-многима и многи-ка-многима. Везе многи-многима морају бити претворене у везе један-многима пре имплементирања релационе базе података. У нашем ЕRD примеру (Слика изнад), једна страна везе се приказује једном ознаком преко линије, а на многим странама је приказана трострука веза. Да се дешифрује однос између ентитета ПАЦИЈЕНТ и ПОСЕТА како је приказано на слици изнад, рекли бисте: ''За сваки случај ПАЦИЈЕНТ постоји много могућих случајева ПОСЕТА, а за сваки случај ПОСЕТА постоји само један могући случај ПАЦИЈЕНТ''.

*Парцијална листа атрибута за ентитете ПАЦИЈЕНТ, КЛИНИКА и ПОСЕТ*А

Припрема модела података који укључује само оне односе који треба и могу бити имплементирани у резултујућој бази података назива се нормализација. Нормализација базе података обезбеђује да се подаци чувају само на једној локацији у бази података (осим планираног вишка). Чување сваког податка на само једној локацији смањује могућност аномалија података као резултат додавања и брисања. Ово смањење редунданције података и смањени потенцијал за аномалије података је обележје релационе базе података. То је оно што је разликује од равне датотеке (*flat file*), старијег модела базе података.