دانشگاه صنعتی اصفهان

دانشکده مهندسی برق و کامپیوتر

عنوان: تکلیف اول درس پایگاه داده‌ها ۱

نام و نام خانوادگی: علیرضا ابره فروش

شماره دانشجویی: ۹۸۱۶۶۰۳

نیم سال تحصیلی: پاییز ۱۴۰۰

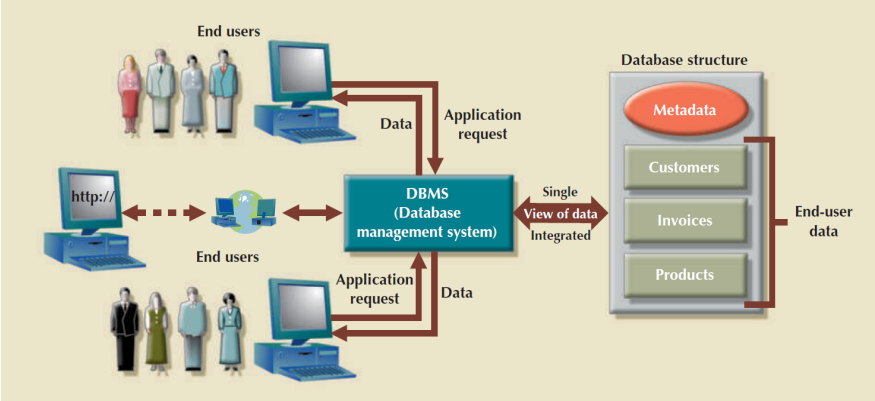
مدرّس: دکتر ناصر قدیری مدرس

دستیاران آموزشی: عارف آسمند - بهاره حاجی هاشمی - پردیس مرادبیکی - سیدمهدی موسوی

۱

۱.۱

- DBMS به عنوان واسطه بین کاربر و پایگاه داده عمل می‌کند. این ساختار پایگاه داده خود به عنوان مجموعه ای از فایل‌ها ذخیره می‌شود و تنها راه دسترسی به اطلاعات موجود در آن فایل‌ها از طریق DBMS است. شکل ۱.۱ بر این نکته تأکید می کند که DBMS به کاربر (یا برنامه کاربردی) یک نمای واحد و یکپارچه از داده‌های موجود در پایگاه داده ارائه می دهد. DBMS همه request های برنامه را دریافت و آن‌ها را به عملیات‌های پیچیده مورد نیاز برای پاسخ به این request ها ترجمه می‌کند. بسیاری از پیچیدگی‌های داخلی پایگاه داده به وسیله DBMS از برنامه های کاربردی و کاربران پنهان می‌شود. برنامه کاربردی ممکن است توسط یک برنامه‌نویس با استفاده از یک زبان برنامه‌نویسی مانند Visual Basic.NET، Java یا C# نوشته شود یا ممکن است توسط یک DBMS utility program ساخته شود. داشتن یک DBMS بین application کاربر و پایگاه داده مزایای مهمی را به ارمغان می‌آورد. اولا DBMS به داده‌ها اجازه می‌دهد که بتوانند بین چندین برنامه به اشتراک گذاشته شوند. ثانیا DBMS بسیاری از دیدگاه(view)های مختلف کاربران از داده‌ها را با هم ادغام می‌کند و در یک مخزن همه جانبه ارائه می‌دهد.



شکل ۱: *DBMS* تعاملات بین کاربر و پایگاه داده را مدیریت می‌کند.

- افزونگی داده‌ها و ناسازگاری(داده ها در چندین فرمت فایل ذخیره می شوند و در نتیجه افزونگی کمتری از اطلاعات در فایل های مختلف رخ می‌دهد)، سخت بودن دسترسی به داده‌ها(نیاز به نوشتن یک برنامه جدید برای مدیریت هر تسک جدید)، ایزولگی داده‌ها، مشکلات یکپارچگی،آپدیت اتمیک ، دسترسی همزمان توسط چند کاربر، مشکلات امنیتی

۲.۱

گام‌های طی شده توسط DBMS جهت پاسخ به درخواست کاربر(query) در شکل ۲ مشخص شده است. گام‌های پایه عبارتند از:

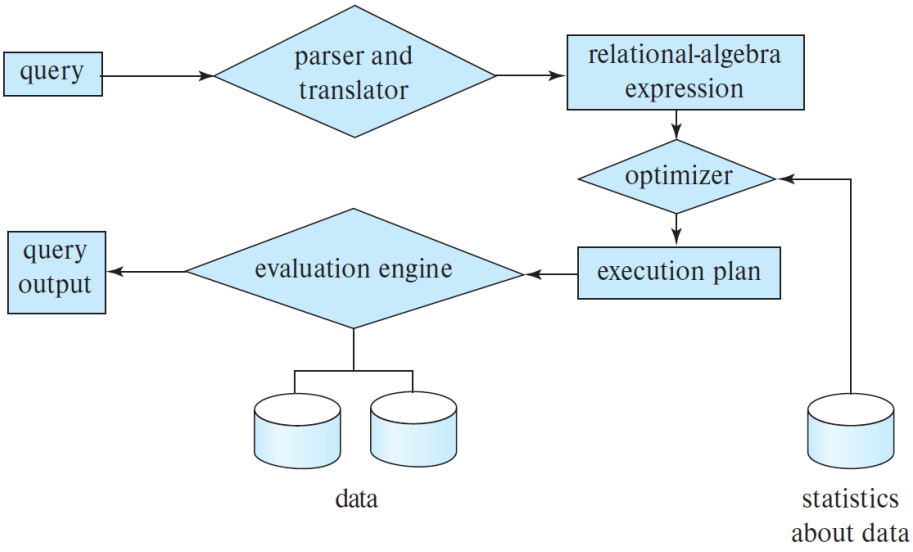
۱. پارس و ترجمه(Parsing and translation)

۲. بهینه‌سازی(Optimization)

۳. ارزیابی(Evaluation)

پیش از اینکه query processing آغاز شود، سیستم باید query را به یک فرمت قابل ترجمه تبدیل کند. یک زبان مانند SQL برای این کار مناسب می‌باشد، اما همچنان جهت نمایش داخلی query در یک سیستم مناسب نیست. یک راه مبتنی بر جبر رابطه ای وجود دارد که مناسب‌تر است.

ابتدا سیستم درخواست را به یک query تبدیل می‌کند. از اینجا به بعد پردازش بر عهده DBMS است که query را به عبارتی تحت عنوان جبر رابطه‌ای پارس و ترجمه(این ترجمه شبیه به کاری است که پارسر کامپایلر انجام می‌دهد) می‌کند و سپس بهینه(چند روش ممکن) می‌کند و در انتخاب روش به آمار و سایز جداول توجه می‌کند. حال DBMS به یک execution plan رسیده است که چگونگی اجرای دستور وارد شده توسط کاربر را نشان می‌دهد. سپس در اختیار evaluation engine قرار می‌گیرد که در این گام به سراغ داده‌ها می‌رود و آن‌ها را کنار هم می‌چیند و خروجی را برمی‌گرداند.



شکل ۲: گام‌های پردازش *query*

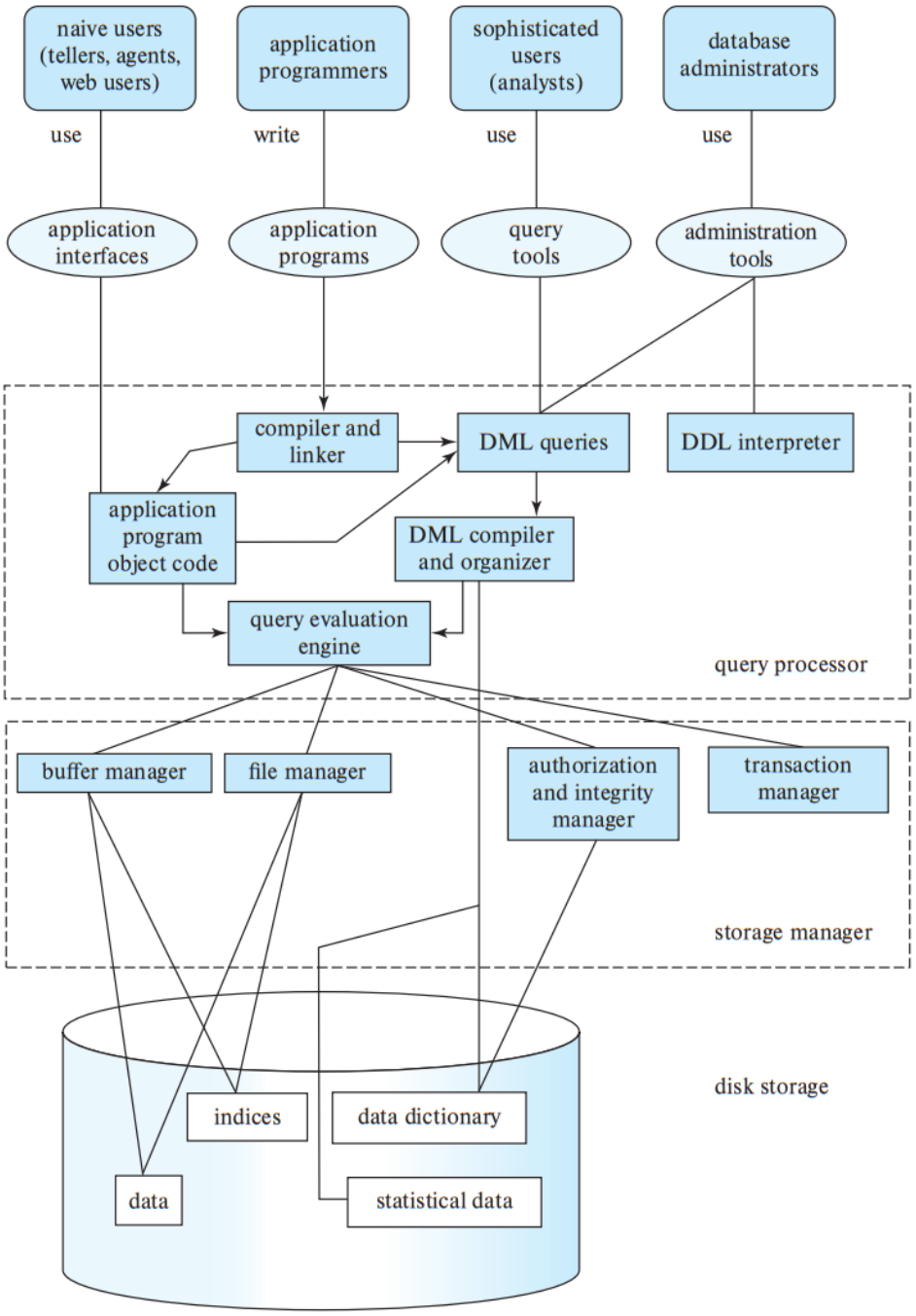
۳.۱

دو بخش اصلی DBMS عبارتند از:

۱. query processor

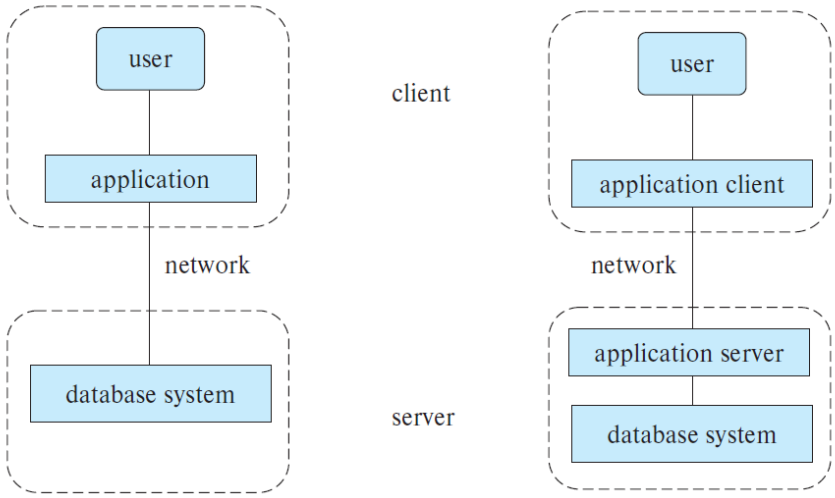
۲. storage manager

همچنین اجزای تشکیل دهنده DBMS در شکل ۳ مشخص شده است.


naive users (tellers, agents, web users)
application programmers
sophisticated users (analysts)
database administrators
use
write
use
use
application interfaces
application programs
query tools
administration tools
compiler and linker
DML queries
DDL interpreter
application program object code
DML compiler and organizer
query evaluation engine
query processor
buffer manager
file manager
authorization and integrity manager
transaction manager
storage manager
disk storage
indices
data dictionary
data
statistical data

شکل ۳

۲



شکل ۴

ها	معماری دولایه	معماری سه‌لایه
	کمتر(کندتر)	بیشتر(سریعتر)
	کمتر(کلاینت می‌تواند مستقیما با پایگاه داده تعامل داشته باشد)	بیشتر(کلاینت مجاز به تعامل مستقیم با پایگاه داده نمی‌باشد)
ی	کمتر	بیشتر(چون تعداد لایه‌ها بیشتر است در نتیجه indirectionها بیشتر است)
یری	کمتر	بیشتر(به دلیل ماژولارتی بالا امکانmaintainance آن بیشتر است)
یری	کمتر	بیشتر(امکان افزودن feature در آن ساده‌تر است)
گی	کمتر	بیشتر(به دلیل لایه‌ای بودن یک‌پارچگی بیشتر حفظ می‌شود)

جدول ۱: جدول شماره ۱

۳

۱.۳

تعدادی از جداول می‌تواند به شکل زیر تعریف شود:

application
app_id #
title
co_id @
category_id @
version
updated_on
downloads

company
co_id #
name
location

user
user_id #
name
nationality

category
category_id #
category_name
description

rating
app_id #@
user_id #@
rate

download
app_id #@
user_id #@
installation_date
uninstallation_date

۲.۳

attributeهای مشخص شده در قسمت قبل با کاراکتر # و @ به ترتیب بیانگر کلید اصلی بودن و کلید خارجی بودن در آن جدول اند.

۴

۱.۴

مقدار خاص null که در هر domain قرار دارد(به جز مواردی که خود ما محدود کرده باشیم) به این معنی است که ما اطلاعی راجع به این فیلد نداریم. توجه شود که null با صفر متفاوت است. برای مثال اگر حقوق یک کارمند صفر باشد به این معنی است که جمع دریافتی‌ها و کسری‌های او صفر می‌شود، درحالی که اگر حقوق او null باشد به این معنی است که اطلاعی از حقوق او در پایگاه داده موجود نیست. همچنین در پایگاه داده‌ی دانشگاه، null بودن نمره یک دانشجو به معنی موجود نبودن(وارد نشده بودن) آن در پایگاه داده‌ی دانشگاه است.

۲.۴

می‌توان یک relation schema به نام employment با attributeهای i_id که شناسه استاد، و dept_name که نام دانشکده است، به شکل زیر تعریف کرد:

employment
i_id
dept_name

به این شکل می‌توانیم dept_name را از instructor حذف کنیم و رابطه‌ی اشتغال را بین دانشکده و استاد برقرار کنیم(شبیه کاری که برای استاد راهنما انجام شد). از این طریق در پایگاه داده یک استاد می‌تواند در بیش از یک دانشکده مشغول به کار باشد.

۳.۴

یک راه می‌تواند اضافه کردن یک attribute تحت عنوان second_advisor به advisor باشد. از آنجایی که می‌دانیم هر دانشجو حداکثر می‌تواند دو استاد راهنما داشته باشد، در صورتی که دانشجو یک استاد راهنما داشت، تنها از first_advisor استفاده کنیم و فیلد second_advisor برابر null قرار دهیم.

۴.۴

اگر یک رکورد از student حذف شود، آنگاه رکورد مربوط به رکورد حذف شده در takes نامعتبر خواهد بود. همچنین درصورتی که یک رکورد با ID ناموجود وارد شود دچار نقض می‌شود.

۵

۱.۵

$$\Pi_{Title,\ ReturnDate}(Borrow \bowtie_{MemberID = 1356 \wedge IsReturned = false} Book)$$

۲.۵

$$\Pi_{Name}(Member \bowtie_{Member.CategoryID = \text{“}Physics\text{”}} (Book \bowtie_{Book.BookID = Borrow.BookID \wedge CategoryID = \text{“}Physics\text{”}} Borrow))$$

۳.۵

$$\Pi_{Name,\,Title}(Member \bowtie_{Member.CategoryID\,=\,Book.CategoryID}\ Book)\ -$$
$$\Pi_{Name,\,Title}(Borrow \bowtie_{Borrow.MemberID\,=\,Member.MemberID\,\wedge\,Borrow.BookID\,=\,Book.BookID}$$
$$(Member \bowtie_{Member.CategoryID\,=\,Book.CategoryID}\ Book))$$

۴.۵

$$\Pi_{Name,\,Title}(Member \bowtie_{Member.MemberID\,=\,Borrow.MemberID}$$
$$(Borrow \bowtie_{CategoryID\,=\,\text{“}Drama\text{”}\,\wedge\,IsReturned\,=\,false\,\wedge\,Today\,-\,ReturnDate\,>\,10\ Days}\ Book))$$

۵.۵

ابتدا عبارت را به شکل زیر تفکیک می‌کنیم:

$$D \leftarrow Borrow \bowtie_{Borrow.BookID\,=\,Book.BookID}\ Book$$

$$C \leftarrow D \bowtie_{Borrow.MemberID\,=\,Member.MemberID}\ Member$$

$$B \leftarrow \sigma_{Borrow.NumDays\,\times\,Book.Penalty\,\geq\,100000}(C)$$

$$A \leftarrow \Pi_{Member.Name,\,Book.Title}(B)$$

D لیست اطلاعات کتاب‌های امانت گرفته‌شده را همراه با اطلاعات امانت آن‌ها برمی‌گرداند.
C لیست اطلاعات اعضا و اطلاعات کتاب‌های امانت گرفته‌شده آن‌ها را همراه با اطلاعات امانت آن‌ها برمی‌گرداند.
B سطرهایی از C را برمی‌گرداند که جریمه دیرکرد آن‌ها بزرگتر یا مساوی ۱۰۰۰۰۰ تومان باشد.
A که همان عبارت نهایی است ستون‌های نام عضو و نام کتاب را از B برمی‌گرداند.
پس نتیجه عبارت، نام عضو و نام کتاب‌هایی که امانت گرفته‌اند و جریمه دیرکرد آن‌ها بزرگتر یا مساوی ۱۰۰۰۰۰ تومان است می‌باشد.

۶.۵

ابتدا عبارت را به شکل زیر تفکیک می‌کنیم:

$$F \leftarrow Book \bowtie_{Book.AuthorID\,=\,Author.AuthorID}\ Author$$

$$E \leftarrow F \bowtie_{Book.CategoryID\,=\,Category.CategoryID}\ Category$$

$$D \leftarrow \sigma_{Category.CategoryName\,=\,\text{“}Philosophy\text{”}\,\wedge\,Author.Name\,\neq\,\text{“}Plato\text{”}}(E)$$

$$C \leftarrow (\sigma_{IsReturned\,=\,false}(Borrow)) \bowtie_{Borrow.BookID\,=\,Book.BookID}\ Book$$

$$B \leftarrow \Pi_{Book.Title}(C)$$

$$A \leftarrow \Pi_{Book.Title}(D)$$

$$A\ -\ B$$

F اطلاعات کتاب‌ها را به همراه اطلاعات نویسنده‌ی آن‌ها برمی‌گرداند.
E اطلاعات F را به تفکیک اطلاعات موضوع آن‌ها برمی‌گرداند.
D سطرهایی از E با موضوع Philosophy که نویسنده آن‌ها Plato نیست را برمی‌گرداند.
C اطلاعات کتاب‌های امانت داده شده اما بازگردانده نشده اند را برمی‌گرداند.
B ستون نام کتاب از C را برمی‌گرداند.
A ستون نام کتاب از D را برمی‌گرداند.
A - B که همان عبارت نهایی است تفاضل B از A را برمی‌گرداند.
پس نتیجه عبارت، نام کتاب‌هایی با موضوع Philosophy که نویسنده آن‌ها Plato نیست و امانت داده شده اما بازگردانده نشده اند را برمی‌گرداند.

منابع

[1] https://www.geeksforgeeks.org/difference-between-two-tier-and-three-tier-database-architecture/

[2] https://medium.com/@gacheruevans0/2-tier-vs-3-tier-architecture-26db56fe7e9c

[3] https://www.softwaretestingclass.com/what-is-difference-between-two-tier-and-three-tier-architecture/

[4] https://www.ibm.com/nl-en/cloud/learn/three-tier-architecture

[5] https://nitrosphere.com/uncategorized/2-tier-vs-3-tier-application-architecture-could-the-winner-be-2-tier-2/