دانشکده مهندسی کامپیوتر

گروه مهندسی نرم افزار

شـــبکه اطلاعاتی پـــل ها

پروژه درس پایگاه داده پیشرفته

دانشجویان:

مرتضی ذاکری – محسن امیریان – امیرحسین صیادعبدی

استاد:

دکتر عین الله خنجری

اساتید حل تمرین:

امیررضا فراهانی – علی رضا محمدخانی

پاییز ۱۳۹۵

چکیده

اطلاعات مربوط به پل ها به قدری و سیع و حائز اهمیت ا ست که برای این منظور از رایانه ها و سامانه های نرم افزاری استفاده می شود. توسعه یک سامانه اطلاعاتی جامع و یکپارچه برای این منظور همواره با چالش هایی از قبیل حجم بسیار زیاد داده ها خام، استخراج اطلاعات از داده ها، مدل سازی ارتباط های بین اطلاعات و بلاخره دریافت گزارش مناسب از در سریع ترین زمان ممکن رو به رو است. به همین علت معمولا چنین سامانه هایی به صورت جداگانه برای شهرهای مختلف توسعه داده می شوند.

در این پروژه راهکاری برای طراحی و پیاده سازی یک سامانه مدیریت پل ارایه شده ا ست که چشم انداز متفاوت و آینده نگرانه تری را دنبال می کند. سامانه پیشنهادی که شبکه اطلاعاتی پل نامیده می شود، اطلاعات پل های موجود را به همراه مسیرهای اتصال آن ها ثبت و نگاهداری می کند. به این ترتیب گرافی از عناصر موجود در سامانه (پل ها، جاده و ارتباطات میان آن ها)، ایجاد می گردد که قابلیت های جدیدی همچون پیشنهاد مسیرهای بهینه برای حمل و نقل، طبقه بندی پل ها و مصور سازی ارتباط بین پل ها فراهم می گردد.

برای غلبه بر چالش های موجود عنوان شده، راه حل پیش رو، استفاده از پایگاه داده های جدید تحت عنوان NoSQLها و ایجاد یک بانک اطلاعاتی توزیع شده از داده های موجود است به نحوی که این بخش از سامانه گلوگاه سرعت و کارایی واقع نشود. علاوه بر آن ا ستفاده از مدل های گراف محور در ذخیره داده ها که چنین بانک هایی در اختیار قرار می دهند در درک بهتر روندهای مدیریتی و مدل سازی های ریاضی مورد نیاز و اعمال الگوریتم های مختلف، کمک بسزایی می کند.

واژگان کلیدی: ***سامانه مدیریت پل، شبکه اطلاعاتی پل ها، پایگاه های داده NoSQL، داده های حجیم.***

فهرست مطالب

فهرست شکل ها

فهرست جدول ها

۱ مقدمه

۱ـ۱ طرح مسئله

مسیرهای ارتباطی، جاده ها و به تبع آن پل ها، شریان های حیاتی ارتباط های فیزیکی در تمدن امروزی هستند. هرگونه سهل انگاری در روند نگهداری این شبکه به هم پیوسته، خسارات جبران ناپذیری را در پی خواهد داشت. از سوی دیگر برنامه ریزی در جهت استفاده صحیح از شبکه موجود، صرفه جویی در هزینه ها و افزایش کارایی را به دنبال خواهد داشت.

سامانه مدیریت تعمیر و نگاهداری پل ها بخش جدایی ناپذیر مدیریت کلان زیر ساخت های حمل و نقل را بخود اختصاص می دهد. در کلان شهرهای ایران نیز با توجه به توسعه شبکه حمل و نقل احساس نیاز به پیاده سازی این گونه سامانه ها به طور روز افزون احساس می شود. به دلیل اهمیت بالای این اطلاعات در مواقع بحرانی نظیر سوانح طبیعی و جنگ و نیز تفاوت ها عمده در زیرساخت های فیزیکی، طراحی و توسعه سامانه های بومی بسیار مورد تأکید است.

از آن جایی که این سامانه ها بایستی حجم وسیعی از داده ها را کنترل کنند، همراه با چالش ها موجود در زمینه موضوعات مربوط به داده های حجیم[۱] مواجه هستند. چالش دیگر مدل ذخیره سازی و مدل نمایشی داده ها است که سبب می شود استفاده از پایگاه داده های رابطه ای کلاسیک چندان مناسب نباشد. بنابراین هدف طراحی و پیاده سازی یک سامانه با ویژگی های جدید در زمینه اطلاعات مربوط به پل ها است.

[۱] Big Data

۱ـ۲ راه حل پیشنهادی

پس از برر سی های انجام شده و تحلیل و ارزیابی سامانه های م شابه، ت صمیم گرفته شد تا با اســـتفاده از فناوری های جدید در عرصـــه پایگاه داده ها، ســـامانه جدید تحت عنوان «شـــبکه اطلاعتی پل ها[۲]» ارایه گردد.

در این سامانه از پایگاه داده NoSQL به نام OrientDB استفاده شده است که یک پایگاه داده چند مدلی ا ست. مدلی که برای ذخیره سازی پل ها و ارتباط های بین آن ها به کار رفته **گراف**[۳] است.

۱ـ۳ ساختار مستند

م ستندات نهایی این پروژه به گونه ای تهیه شده که هم به صورت مرجع (جهت ارجاع به یک بخش خاص) و هم به صـــورت گزارش (جهت مطالعه ی بالا به پایین)، قابل اســـتفاده بوده و ویژگی های سامانه را به در ستی بیان کند. این مستندات در زمان های خاص خود و به صورت مجزا تهیه شده و نهایتا در یک سند ا صلی درج شده اند، که اکنون پیش روی شما ست. فصل اول شامل مقدمه، طرح مسئله و راه حل می باشد.

در فصل دوم برخی مطلعات انجام شده روی سامانه های مدیریت پل به صورت خلاصه وار بیان شـــده اســـت. در این فصـــل همچنین مقدماتی راجع به پایگاه داده های NoSQL مطرح گردیده اســـت. در پایان فصـــل هم به بیان نظریه CAP پرداخته ایم که در واقع محدودیت های موجود در سامانه های توزیعی را ترسیم می کند.

فصل سوم به معرفی و نحوه استفاده از پایگاه داده OrientDB تخصیص داده شده است. ف صل چهارم معماری فنی سامانه پی شنهادی را ت شریح می کند و نهایتا در ف صل پنجم موارد انجام شـــده مرور و ضـــمن نتیجه گیری از آن ها، پیشـــنهاداتی برای کارهای آتی عنوان گردیده است.

---

[۲] Bridge Information System

[۳] Graph Base

" Not Only SQL" حرکتی ا ست برای ت شویق تو سعه دهنده گان و تاجران ج هت باز تر کردن د ید آن ها و تو جه بیش تر آن ها به امکانات و فر صت هایی که خارج از ف ضای دیدگاه رابطه ای به داده ها، وجود دارد.

۲ مطالعات پیشین

۲ـ۱ پیشینه مدیریت پل

قدمت فرآیند تعمیر و نگهداری پل ها به صورت سنتی را می توان به قدمت ساخت و بهره برداری از پل ها دانست. ماهیت این نوع از عملیات تعمیر و نگهداری غالبا عکس العملی بوده است، یعنی بواسطه تخریب بخشی از سازه پل بهره برداران نسبت به تعمیر و یا جایگزینی اعضای پ ل اق دام م ی نمودند. پس از فاجعه و فرو ریزش پل سیلور در سال ۱۹۶۷زنگ خطر برای متصدیان بهره برداری از سیستم حمل و نقل جاده ای به صدا در آمد. پس از این فاجعه برنامه ملی بازرسی پل ها در اداره فدرال راه ایالات متحده پایه ریزی شد.

تخریب پل رودخانه میانوس در سال ۱۹۸۳و اختلالات گسترده ترافیکی ناشی از آن در شمال شرق آمریکا بر گستردگی موضوع افزود و بواسطه آن نیاز به برنامه ریزی و ایجاد سیستم جامع مدیریتی برای نگهداری و تعمیر پل ها بیش از پیش احساس شد. در همین راستا در سال ۱۹۸۵برنامه تحقیقات بزرگراهی پروژه ای در راستای راه اندازی یک سیستم موثر مدیریت پل ها تعریف و پیاده سازی نمود. در سال ۱۹۹۱قانون جامع بهره وری حمل و نقل چند سطحی کلیه ایالت ها را ملزم نمود تا نسبت به توسعه، پیاده سازی و اجرای سی ستم مدیریت تعمیر و نگهداری پل ها تا اکتبر ۱۹۹۸اقدام نمایند. بوا سطه قانون یاد شده برخی تلاش های ایالتی نیز در برخی ایالت ها مانند پنسیلوانیا، کارولینای شمالی و ایندیانا منجر به توسعه سیستم های مدیریت تعمیر و نگهداری پل ها بصورت محلی و منطقه ای گردید.

مدیریت پل همگی اقدامات و فعالیت ها در طول عمر پل را از ساخت تا تعویض برای تامین ایمنی و کارایی، شامل می شود. گزارش OECD در خصوص مدیریت پل ها، سیستم مدیریت پل را به عنوان «ابزاری برای کمک به متولیان بزرگ راه ها و مدیریت پل در انتخاب

رشد بهینه شبکه پـــل هـــا که با سیاست های متولیان و اهداف دراز مدت و محدودیت های بودجه سازگار با شد»، تعریف کرده ا ست. مدیریت پل در خ صوص اولویت بنـــدی احتیاجات نگهداری، برنامه ریزی روش کار اجرای نگهداری و بهینه سـازی هزینه طول عمر پل می باشـد. به طور خاص تر مدیریت پل در ارتبـاط بـا کارهای زیر است:

- ارزیابی شرایط پل
- ارزیابی ظرفیت باربری
- پیش بینی زوال
- تعیین راهکارهای موجود و اثربخشی آنها، طول عمر و هزینه
- ضوابط اولویت بندی احتیاجات نگهداری
- تصمیم گیری برای تخصیص منابع
- توسعه سیستم مدیریت اطلاعات

برآورد مناسـب ترین اسـتراتژی نگهداری برای تعداد بسـیار زیادی پل مسـئله بزرگ و پیچیده ای است؛ زیرا، پارامترهای متعددی برای تعیین اقتصـادی تـرین راه حل وجود دارد. در نتیجه تنها راه عملی برای مدیریت پل اثر بخش، ا ستفاده از سامانه ها و نرم افزارهای رایانه ای است. تعدادی از این سامانه ها عبارتند از:

- AASHTOWare™ (اداره راه ایالات متحده)
- dTIMSCT (کانادا)
- Exor Highway (بریتانیا)
- HONSEN (استرالیا)

۲ـ۲ پایگاه های داده NoSQL

امروزه به دلیل فراگیر شدن شبکه های گسترده رایانه ای به ویژه اینترنت و به وجود آمدن نرم افزارهای تحت وب با کاربران بسیار زیاد، دیگر پایگاه داده های RDBMS[۴] یا سنتی پا سخ گوی نیاز این نرم افزارها، تو سعه دهندگان و ا ستفاده کنندگان آن نمی با شند. از دلایل آن می توان به نگه داری داده ها با حجم بسیار زیاد، سرعت بالا در خواندن و نوشتن نام برد. به همین دلایل

[۴] Relational Database Managements System

پایگاه داده های نسل بعدی یعنی NoSQL ها در اواخر دهه اول هزاره سوم به وجود آمدند و بسیار سریع در حال پیشرفت هستند.

از ویژگی های NoSQL ها می توان به نحوه ذخیره سازی و نگه داری داده ها به صورت توزیع شده، نبودن رابطه به صورت جداولی، عموما متن باز بودن، بدون Schema بودن و قابلیت گسترش پذیری در سرور های مختلف با محل های جغرافیایی متفاوت اشاره کرد. در این پایگاه داده ها دیگر محدودیت های ساختار های خاص، نرمال سازی و غیر نرمال سازی و بسیاری موارد دیگر را نمی بینیم.

۲ـ۲ـ۱ مقایسه NoSQL ها و RDBMS ها

برای درک بهتر مفهوم پایگاه داده های NoSQL و نحوه عملکر آن ها شاید بهتر با شد تا ویژگی های آن ها را درکنار ویژگی های پایگاه داده های رابطه قرار داد و مقایسه کرد. جدول ۲-۱ مقایسه ای کوچک میان پایگاه داده های NoSQL یا غیر رابطه ای و RDBMS یا رابطه ای را نشان می دهد.

جدول ۲-۱. مقایسه پایگاه داده های NoSQL با پایگاه داده های رابطه ای

	RDBMS	NoSQL
انواع	یک نوع با امکانات مختلف	*Key-Value, Column Store, Document Store, Graph*
تاریخچه توسعه	در دهه ۱۹۷۰ برای مقابله با اولین موج از برنامه های ذخیره داده، توسعه داده شدند	در دهه ابتدایی هزاره سوم برای مقابله با محدودیت های پایگاه های داده SQL خصوصاً نگرانی حول مقیاس، تکرار و ذخیره داده بدون ساختار، توسعه یافتند
نمونه	*Microsoft SQL Server, MySQL, Oracle*	*Aerospike, Sophia, BigTable, MongoDB, Cassandra, HBase, Neo4j,* ***OrientDB***
	جدول (رابطه) شامل ستون ها و ردیف ها	بر اساس نوع پایگاه داده متفاوت است. برای مثال، انبارهای key-value مشابه پایگاه های داده SQL عمل می کنند، اما تنها دو ستون دارد (key و Value) به همراه اطلاعات پیچیده تری که

مدل نگهداری داده		برخی مواقع درون ستون value ذخیره می شود. پایگاه های داده مبتنی بر سند، تمامی داده های مرتبط با یکدیگر را در یک "سند" در JSON، XML و یا سایر فرمت ها که مقدار ها را به صورت سلسله مراتبی در خود جای می دهند، ذخیره می کنند.
شِما (اسکیما)	با حرکت از رکورد یک به رکورد دو، ساختار داده‌ای یکسان می باشد.	با حرکت از رکورد یک به رکورد دو، ممکن است با دو ساختار داده‌ای متفاوت مواجه شوید (بدون شِما).
مقیاس پذیری	عمودی؛ به این معنی که یک سِرور تنها به منظور غلبه بر تقاضای افزایش یافته، باید به میزان زیادی تقویت شود. امکان دارد که پایگاه های داده SQL در میان سِرور های متعددی توزیع شود، اما عموماً نیاز به مهندسی مازاد قابل توجهی است.	افقی؛ به این معنی که برای افزایش ظرفیت، یک راهبر پایگاه داده به راحتی می تواند سِرور و یا مواردی همچون فضای ابری را اضافه نماید. پایگاه داده، در صورت لزوم به طور خودکار داده را در میان سرور توزیع می کند.
مدل توسعه	ترکیبی از منبع باز (مانند MySQL و Postgres) و منبع بسته (مانند پایگاه داده Oracle)	اکثراً منبع باز
پشتیبانی تراکنش ها	اکثراً به صورت کامل پشتیبانی می کنند	در شرایط خاص و در سطوح خاص (سطح سند در مقابل سطح پایگاه داده)
زبان مدل	زبان برنامه نویسی استاندارد شده SQL. در برخی انواع مختلف مانند PL/SQL و T-SQL.	سمت برنامه نویسی - زبان های مختلف مانند : xml - Json - Java Script , ...

پایایی	امکان پیکربندی برای پایایی زیاد و قوی	وابسته به محصول، برخی پایایی زیاد را فراهم می آورند (برای مثال MongoDB) در حالیکه برخی پایایی مشروط و نسبی فراهم می آورند.

۲ـ۲ـ۲ طبقه بندی ساختاری NoSQL ها

در این بخش اشاره کوتاهی بر انواع پایگاه داده های NoSQL کرده و به صورت مختصر پیرامون هرکدام توضیح خواهیم داد. پایگاه داده های NoSQL بر اساس **نوع ذخیره سازی و ارتباط داده ها** به چهار دسته کلی تقسیم می شوند:

۱) Key-Value Store (کلید - مقدار)،
۲) Column Family Store (خانواده ستونی)،
۳) Document Store (سندگرا)،
۴) Graph Based (مبتنی بر گراف).

در ذیل شرح مختصری از هر کدام از این دسته ها، موارد کاربرد آن ها، مزایا و معایب هریک و بلاخره چندین نمونه توزیع موجود در بازار، آمده است.

- **کلید – مقدار:** ساده‌ترین حالت از دسته‌بندی‌های NoSQL دسته، کلید-مقدار می‌باشد و معمولا در سیستم هایی مورد استفاده قرار می گیرد که داده‌ها از یکدیگر متمایز هستند و اصولا در دسترس بودن داده‌ها نسبت به مواردی نظیر پایائی اهمیت بیشتری دارد. در این معماری فقط یک کلید داریم (که مانند کلید اصلی در پایگاه داده های رابطه ای عمل می کنند) و یک مقدار داریم که مقدار معادل آن کلید را باز می گرداند. از مزایای این دسته می توان به سرعت بالا در درج کردن و خواندن اطلاعات، پیاده سازی و قابلیت توسعه پذیری آسان اشاره کرد.
  - چند نمونه محصول: LevelDB، Redis و Aerospike.
- **خانواده ستونی:** پایگاه های داده ستونی با توسعه کلید-مقدارها به وجود آمده اند. این پایگاه های داده در واقع به جای یک جفت کلید-مقدار، می توانند برای هر رکورد چندین جفت کلید-مقدار داشته باشند. در این نوع نیازی به ساختار نداریم و هر رکورد

می تواند چندین ستون با تعداد صفات متفاوت داشته باشند. از مزایای این دسته می تواند ذخیره سازی میزان وسیع و متفاوتی از رکوردها با مقادیر بسیار باشد.

- چند نمونه محصول: Amazon SimpleDB، Cassandra و HBase.

- **سند گرا:** این دسته از پایگاه داده ها نیز مانند دسته اول یعنی کلید-مقدار و دسته دوم ستونی می باشند ولی با این تفاوت که در این سیستم دسته بندی داده های مرتبط با یکدیگر در قالب یک فایل سند می باشند. از متن ساده گرفته تا یک ایمیل یا عکس و ... یک سند می باشند. اما با وجود قدرت بسیار بالایی که این نوع پایگاه های داده دارند، خواندن و نوشتن در آن ها بسیار وقت گیر است. از مزایای این دسته می توان به ذخیره سازی مقدار زیادی داده های بی ربط نام برد.
  - چند نمونه محصول: MongoDB، Elastic Search، CouchDB و RavenDB.
- **مبتنی بر گراف:** این دسته از پایگاه های داده به داده ها، از دید کاملا متفاوتی نسبت به دسته های قبلی نگاه می کند. داده ها را مانند یک گراف به هم مرتبط می کند و ساختار یک درخت یا گراف را به داده ها می دهد. در این پایگاه داده، رکوردها هنگام درج شدن، توسط یک یا چند صفت به هم مرتبط می شوند؛ نتیجه این که انجام عملیات ریاضی و الگوریتمیک بسیار ساده تر از دسته های دیگر است. کاربرد این دسته زمانی است که ارتباطات معین و مشخصی میان رکوردها وجود دارد؛ مثل *شبکه های اجتماعی*. از مزایای این دسته می توان به مناسب بودن برای تحقیقات علمی و فنی اشاره کرد.
  - چند نمونه محصول: Neo4J، InfoGrid، AllegroGraph وSparksee.

برخی از پایگاه های داده NoSQL همزمان از چندین مدل پشتیبانی می کنند. به عنوان مثال هم دارای امکاناتی برای ذخیره و بازیابی اسناد و هم دارای امکاناتی برای کار با گراف ها هستند. چنین پایگاه داده هایی را Multi-Model هم می نامند و برای کاربردهای ترکیبی ایده آل به نظر می رسند. از جمله ابزارهای مطرح در این زمینه پایگاه داده OrientDB است، که موضوع اصلی فصل سوم این نوشتار می باشد.

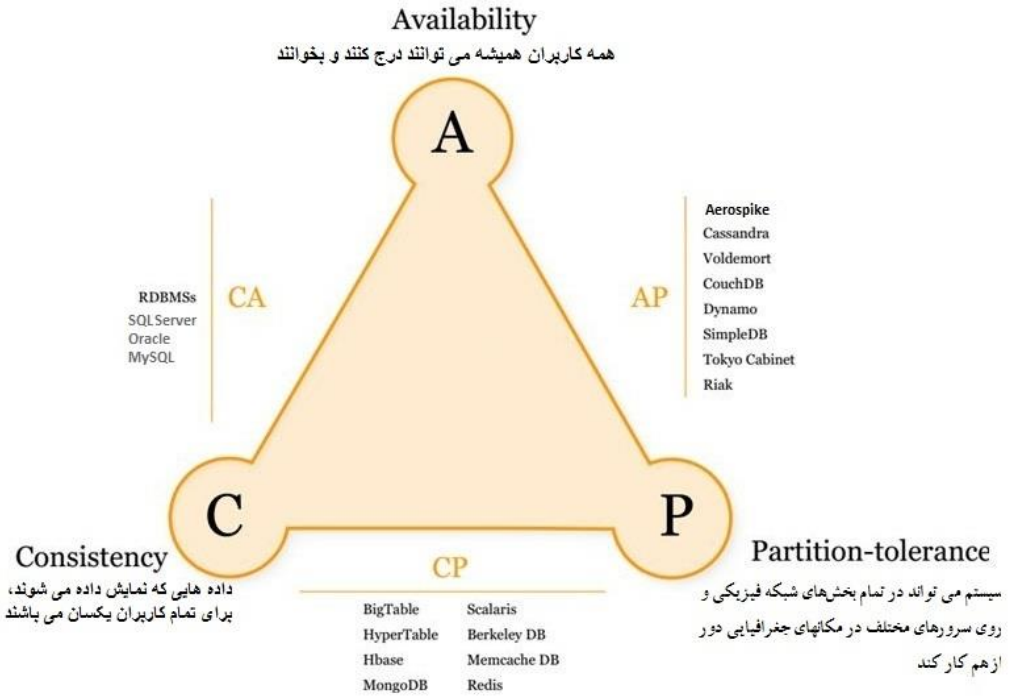
۲ـ۳ نظریه CAP

با توجه به گستردگی و تنوع زیاد موجود در محصولات NoSQL بر خلاف محصولات رابطه ای انتخاب یک محصول متناسب برای استفاده در پروژه های مرتبط، در نگاه اول ساده به نظر نمی رسد. نظریه CAP که در این جا تشریح می شود، راهنمای بسیار خوبی برای انتخاب پایگاده داده مناسب با نیازهای نرم افزار موردی می باشد. نظریه **CAP** از ۳ راس Availability، Consistency و Partition-tolerance تشکیل شده است.

Availability یا **دسترسی پذیری**، به این معنا می باشد که کاربران همیشه بتوانند عملیات درج کردن و خواندن را داشته باشند. یعنی اگر Server اصلی و یا یک Server دیگر دچار مشکل شد ارتباط کاربران قطع نشود و همچنان دسترسی داشته باشند. یک بیان دیگر دسترسی پذیری اذعان می دارد که همه درخواست‌ها در مورد *موفقیت آمیز* یا *ناموفق بودن* عملیات جواب مناسبی را دریافت کنند.

Consistency یا **سازگاری** یعنی داده هایی که تمام کاربران مشاهده می کنند از هر جا برای تمام آن ها داده های یکسان نمایش داده شود و هیچ کاربری نسبت به دیگری داده های کمتر، بیش تر یا اشتباهی یا متفاوتی را نبیند که یعنی پایگاه داده تضمین می کند که یک فقره از اطلاعات همیشه و همه جا یکسان ذخیره سازی شده باشند.

Partition-telorance به این معنا می باشد که سرور های مختلف در مکان های مختلف جغرافیایی به طوری توزیع شوند که سیستم به طور یکپارچه با همه گره ها در ارتباط می باشد و کار می کند. به عبارت دیگر سامانه به کار خودش ادامه دهد حتی در صورت پیش‌آمدن شکست شبکه‌ای در مقیاس گسترده. شکل ۲-۱ یک شمای مفهومی از ارتباط این سه عامل را در قالب رئوس یک مثلث نشان می دهد.



شکل ۲-۱. شمای مفهومی رئوس مثلث CAP

طبق این نظریه پایگاه داده ها در واقعیت فقط می توانند بر اســـاس دو رأس شـــکل بگیرند. البته پایگاه داده هایی نیز ه ستند که قابلیت های راس سوم را هم، به ا ستثنا برخی از قابلیت ها، دارند ولی در اکثریت فقط دو رأس می توانند تجمیع شـــوند. به طور کلی این نظریه بیان می کند در **سامانه های توزیع شده این سه عامل نمی توانند همزمان برقرار باشند**.

تفسیر جمله آخر پاراگراف فوق این است که مدل داده ای مورد نیاز ما یا باید CA باشد که در این حالت ا ستفاده از پایگاه داده های رابطه ای منا سب می با شد. یا باید AP با شد که در این صورت پایگاه داده های غیر رابطه ای یا NoSQL پی شنهاد داده می شوند که معمولا از نوع کلید-مقدار و یا ستونی ه ستند و یا CP با شد که بیش تر پایگاه داده هایی که از نوع NoSQL و در دسته های ستونی، سندگرا و یا مبتنی بر گراف هستند پیشنهاد می شوند که البته در شکل ۲-۱ چند نمونه از این پایگاه داده ها بر اساس اضلاع مثلث پیشنهاد شده اند.

۳ توزیع غیر رابطه ای OrientDB

۳ـ۱ انگیزه

در سال های اخیر، شاهد انفجار تعداد زیادی از پایگاه داده های NoSQL و محصولات مربوط به آن ها بوده ایم. واژه NoSQL به معنای یک کمپین برای مقابله با زبان SQL نیست! در واقع OrientDB از نحو SQL هم پشتیبانی می کند. تعریفی زیر را می توان تعریفی مناسب برای NoSQL در نظر گرفت:

> *" Not Only SQL" حرکتی ا ست برای ت شویق تو سعه دهنده گان و تاجران جهت باز تر کردن دید آن ها و توجه بیش تر آن ها به امکانات و فرصت هایی که خارج از فضای دیدگاه رابطه ای به داده ها، وجود دارد.*

جایگزین های سیستم های مدیریت پایگاه داده های رابطه ای[۵] مدت هاست که وجود دارند. اما مواردی که در اولویت بوده اند، زمینه هایی مثل مخابرات، پزشکی، CAD[۶] و... هستند. گرایش به جایگزین های NoSQL مانند OrientDB به طور چشم گیری در حال افزایش است. جای تعجب نیست که تعداد زیادی از شرکت های بزرگ اینترنتی مانند Google، Amazon، Facebook، Foursquare و Twitter از راه حل های NoSQL در محیط های تولید خود استفاده می کنند.

چه انگیزه ای باعث می شود شرکت ها راحتی دنیای پایگاه داده های رابطه ای ترک کنند؟ این م سئله اساساً مرتبط با برطرف کردن هرچه بهتر نیازهای داده ای امروزی، می باشد. تعدادی از زمینه های کلیدی این نیازها عبارتند از:

- عملکرد
- مقیاس پذیری (اغلب بسیار بزرگ)
- رد پای کوچکتر
- بهره وری توسعه دهندگان و راحتی کار آنها
- انعطاف پذیری طرح

[۵] RDBMS

[۶] Computer Aided Design

در سال های نه چندان دور، تو سعه دهندگان سی ستم هایی طراحی می کردند که می توان ست هزاران کاربر را به صورت همزمان مدیریت کند. اما امروزه اتصال و سرویس دهی به میلیون ها کاربر در یک لحظه، دیگر چندان دور از انتظار نیست.

راه حل های مبنتی بر NoSQL، به طور کلی یک مسیر قدرتمند، مقیاس پذیر و انعطاف پذیر را برای نیاز های داده ای فراهم می کنند.

۲_۳ Multi-Model

موتور OrientDB از مدل های Graph، Document، Key-Value و Objects پشتیبانی می کند. بنابراین OrientDB می تواند به عنوان جایگزین مناسبی برای محصولی باشد که در یکی از زمینه های بالا کار می کند. در واقع می توان گفت این محصول به طور کامل Multi-Model است. ضمناً این توانایی ها تنها واسط هایی برای موتور پایگاه داده نیست، بلکه خود موتور برای پشتیبانی از هر چهار مدل ذکر شده طراحی شده است. این مسئله، تفاوت اصلی OrinetDB را با سایر محصولات Multi-Model نشان می دهد. زیرا آن ها تنها یک لایه ی اضافی با یک API افزوده اند که در واقع سایر مدل ها تقلید (شبیه سازی) می کند.

۳_۳ نسخه های برنامه

محصول OrientDB در دو نسخه موجود است:

نسخه عمومی[۷]: این نسخه به عنوان یک پروژه ی متن باز[۸] تحت مجوز آپاچی ۲[۹] منتشر شد. بر اساس این مجوز، استفاده رایگان از پروژه های متن باز و تجاری بلامانع می باشد.

نسخه Enterprise: یک نرم افزار تجاری است و بر روی نسخه ی عمومی نوشته شده است. این نسخه که توسط تیم سازنده ی موتور OrientDB توسعه داده شده است، علاوه بر ویژگی های نسخه عمومی، ویژگی های زیر را نیز دارد:

- پشتیبان گیری و بازیابی بصورت مداوم
- پشتیبان گیری زمان بندی شده و کامل

[۷] Community Edition
[۸] Open Source
[۹] Apache 2

- Querry Profiler
- مانیتورینگ زنده با هشدار های قابل تنظیم
- و ...

۳_۴ نصب OrientDB

هر دو نسخه ی OrientDB قابلیت اجرا بر روی تمام سیستم عامل هایی که ماشین مجازی جاوا را پشتیبانی می کنند، دارند. به عنوان مثال:

- Linux و تمام مشتقات آن
- Mac OS
- Window از نسخه ی 95/NT به بعد
- Solaris
- HP-UX
- IBM AIX

همچنین لازم به ذکر است برای اجرای OrientDB، نصب نسخه ی ۱,۷ به بعد جاوا نیز مورد نیاز است. برای دریافت این نسخه ها می توان به آدرس زیر مراجعه کرد.

http://orientdb.com/download/

برای نصب OrientDB می توان از یکی از روش های Binary Installation و یا Source Code Installation استفاده کرد. در روش اول یک پکیج از پیش ترجمه[۱۰] شده فراهم شده است. کافی است بر اساس سیستم عامل خود، پکیج مربوط را دانلود کرده و به کامپیوتر خود انتقال دهیم. این پکیج شامل تمامی فایل هائیست که برای اجرای OrientDB مورد نیاز هستند.

اما در روش دوم که Source Code Installation است، می توانیم با دریافت متن کد نسخه ی عمومی، که قابل دریافت از سایت GitHub است، آن را ترجمه کرده و استفاده کنیم. لازمه ی این فرآیند، نصب ابزارهای Git و Apache Maven از قبل بر روی کامپیوتر است.

[۱۰] Compile

۳ـ۵ راه اندازی Server

پس از اینکه OrientDB روی سیستم نصب شد، نیاز به راه اندازی Server پایگاه داده است. برای این کار می بایست به محل نصب برنامه رفته، وارد پوشه bin شده و server.bat را اجرا کنیم که بعد از اجرای آن با تصویر زیر مواجه می شویم:

برای راه اندازی Server در سیستم های بر پایه Unix، می بایست فایل server.sh را در پوشه bin اجرا کنیم.

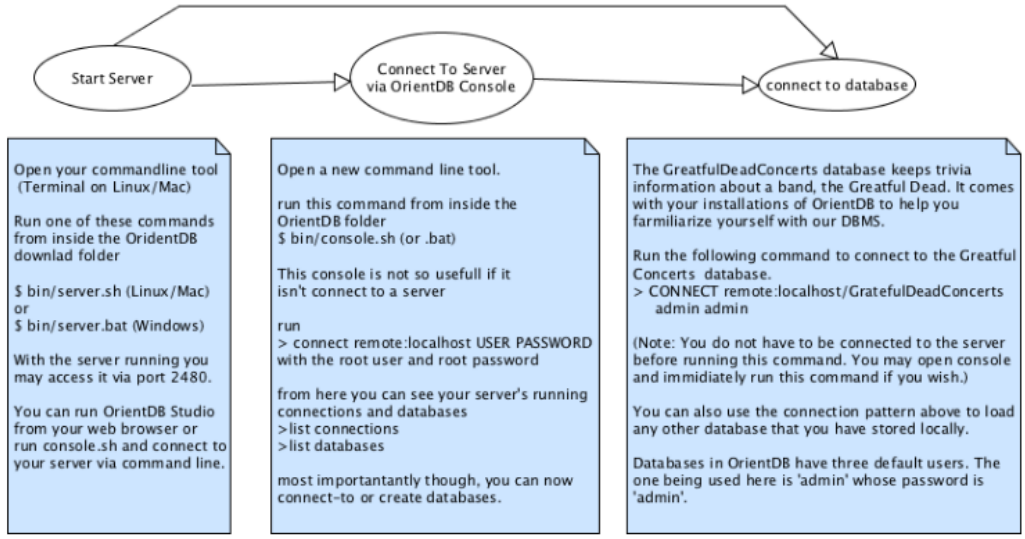
ضمناً جهت متوقف کردن Server می بایست از فایل shutdown.bat در ویندوز و یا shutdown.sh در سیستم های بر پایه Unix استفاده کنیم.

۳ـ۶ اتصال به Server

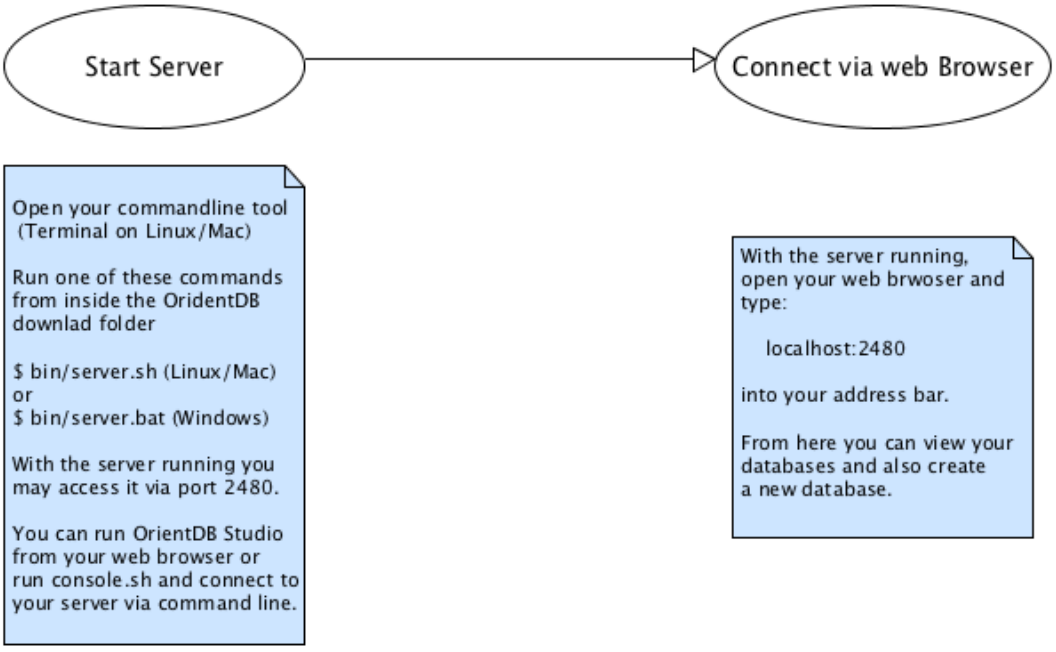
پس از راه اندازی Server، به دو روش می توان به آن متصل شد. روش اول با استفاده از console و روش دوم با مرورگر وب می باشد.

به طور پیشفرض OrientDB برای اتصال به محیط خارجی، بر روی دو پورت مختلف شنود انجام می دهد. یکی پورت ۲۴۲۴ برای اتصال از طریق console و دیگری پورت ۲۴۸۰ برای اتصال از طریق مرورگر وب.

در شکل های زیر راه اندازی Server و اتصال به آن به روش های گفته شده را می توان به صورت خلاصه مشاهده کرد:

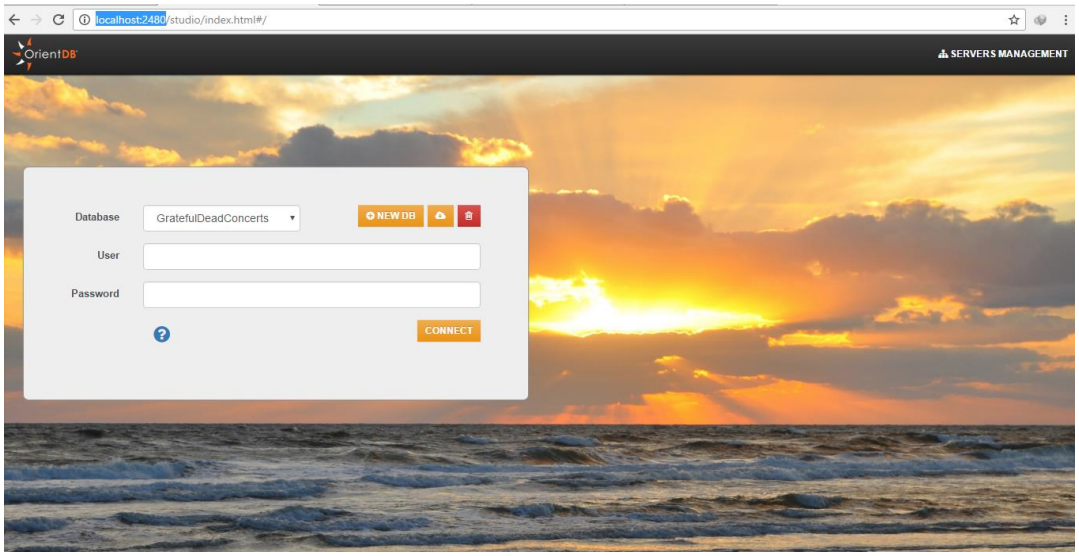


شکل ۳-۱ اتصال به OrientDB از طریق خط فرمان



شکل ۳-۲ اتصال با استفاده از OrientDB Studio

ما برای اتصال و کار با OrientDB از OreintDB Studio استفاده می کنیم. برای این کار همان طور که در شکل توضیح داده شده است، با مرور گر خود وارد آدرس http://localhost:2480 می شویم.



شکل ۳-۳واسط تحت وب OrientDB

۳ـ۷ محیط OrientDB

در صفحه ای که مشاهده می کنید امکان ایجاد یک پایگاه داده جدید، حذف و یا اتصال به یک پایگاه داده موجود و یا وارد کردن[۱۱] یک پایگاه داده، وجود دارد.

۳ـ۸ رابط های برنامه کاربردی[۱۲] و درایور

۳ـ۸ـ۱ توابع[۱۳]

توابع در OrientDB مانند سایر زبان ها، به صورت واحدی اجرایی از کد ها است که تعدادی پارامتر به عنوان ورودی دریافت می کند و نتیجه ای را باز می گرداند. توابع در OrientDB همانند Stored Procedures ها در RDBMS ها هستند. ویژگی های این توابع عبارتند از:

[۱۱] Import
[۱۲] API
[۱۳] Functions

ماندگار هستند.

- می توان به زبان های SQL و Javascript آن ها را نوشت.
- قابلیت اجرا در SQL، Java، REST و Studio را دارند.
- می توانند یکدیگر را فراخوانی کنند.
- از قابلیت بازگشتی پشتیبانی می کنند.
- به صورت خودکار تطبیق پارامتر ها را با مکان و نام انجام می دهند.

۳ـ۸ـ۲ ساخت توابع

علاوه بر توابعی که در OrientDB بصورت پیش فرض وجود دارند، می توانیم توابع جدیدی ایجاد کنیم. شیوه ی ایجاد تابع در Studio به این صورت است که:

- وارد پانل Functions می شویم.
- یک نام برای تابع خود انتخاب می کنیم. مثلاً sum
- پارامترهای مورد نظر را اضافه می کنیم. به عنوان مثال
- در کادر مربوطه کد اجرایی تابع را می نویسیم. به عنوان مثال:

```
return parseInt(a) + parseInt(b);
```

- تابع را با گزینه Save، ذخیره می کنیم.

ضمناً از قسمت پایین صفحه می توان خروجی تابع را مشاهده کرد.

۳ـ۸ـ۳ فراخوانی توابع در SQL

تمامی توابع موجود در پایگاه داده به طور خودکار در موتور SQL نیز ثبت می شوند. شیوه ی استفاده از تابع sum - که در بخش قبل تعریف شد – به شکل زیر است:

```
SELECT SUM(3,4)
```

و یا:

```
SELECT SUM(salary,bonus) AS total FROM Employee
```

در این مثال تابع sum بر روی تمامی رکورد های کلاس Employee اجرا می شود و مقادیر فیلد های salary و bonus را با یکدیگر جمع می کند.

۳ـ۹ استفاده از توابع با API جاوا

۴ شبکه اطلاعاتی پل ها

۴ـ۱ ماژول های سامانه

شبکه اطلاعاتی پل ها از پنج پیمانه نرم افزاری یا ماژول ا صلی ت شکیل شده ا ست. هر ماژول کارکرد تعریف شده و به خصوصی دارد که طی فاز مدل سازی نیازمندی ها و با توجه به امکان سنجی های صورت گرفته وجود آن در سامانه تشخیص داده شده است. این ماژول ها عبارتند از:

۱- ماژول مدیریت سوال
۲- ماژول مدیریت پل
۳- ماژول گزارش گیر
۴- ماژول مدیریت کاربر
۵- ماژول پشتیبان گیر (برون ریز - درون ریز)

این ماژول وظیفه نگهداری و مدیریت پرسش های موجود در سامانه را به عهده دارد. ماژول مدیریت سوال تنها برای توسعه دهندگان است و نمود کاربری ندارد.

ماژول مدیریت پل همان گونه که از عنوان آن بر می آید، مهم ترین قطعه سامانه را پیاده سازی می کند که اعمال مختلف درج، ویرایش، ت شکیل پر س شنامه و ... را برای یک پل فراهم می آورد.

ماژول گزارش گیر مسئولیت تولید کلیه گزارش های چاپی و غیر چاپی یک پل موجود در سامانه را برعهده دارد. این ماژول امکاناتی در ارتباط با گزارش ها، از قبیل برون ریزی های مورد نیاز فراهم می کند که از آن می توان به عنوان نوعی جست و جو گر پل نیز استفاده نمود.

ماژول مدیریت کاربر، برخلاف دو ماژول قبلی، یک وظیفه سیستمی اصلی بر عهده ندارد و تنها به مدیر یت کاربران نرم افزار و اعمال آن ها در نرم افزار می پردازد. مهم ترین وظیفه این ماژول در سامانه ایجاد سطوح د ستر سی متفاوت برای کاربران ا ست به نحوی که هیچ کاربری نتواند به قسمت هایی که اجازه ندارد، د سترسی پیدا کند و بالعکس هر کاربری به

آن چه که باید بتواند به راحتی دستیابی داشته باشد. این ماژول امکانات اضافه تری علاوه بر آن چه گفته شـــده نیز فراهم می کند از جمله ثبت نقاط ورود و خروج هر کاربر، که در موارد لزوم می تواند مفید واقع شود.

در نهایت ماژول پشتیبان گیر وظیفه گرفتن پشتیبان از اطلاعات پل های ثبت شده را بر عهده خواهد داشت. پشتیبانی گیری به صورت استخراج یک یا مجموعه ای از پل ها در قالب فایل هایی با پســوند ***brg*** صــورت می گیرد. این روش یک فایل قابل حمل و کم حجم به ازای اطلاعات هر پل ایجاد می کند که بعدا نرم افزار می تواند آن بخواند.

۴ـ۲ مهندسی نرم افزار سامانه

این بخش برای بیان متدلوژی تو سعه مورد ا ستفاده، فازهای طی شده برای تولید و نگاه داشت محصول و به طور کلی اصول مهندسی نرم افزار سامانه نگارش شده است.

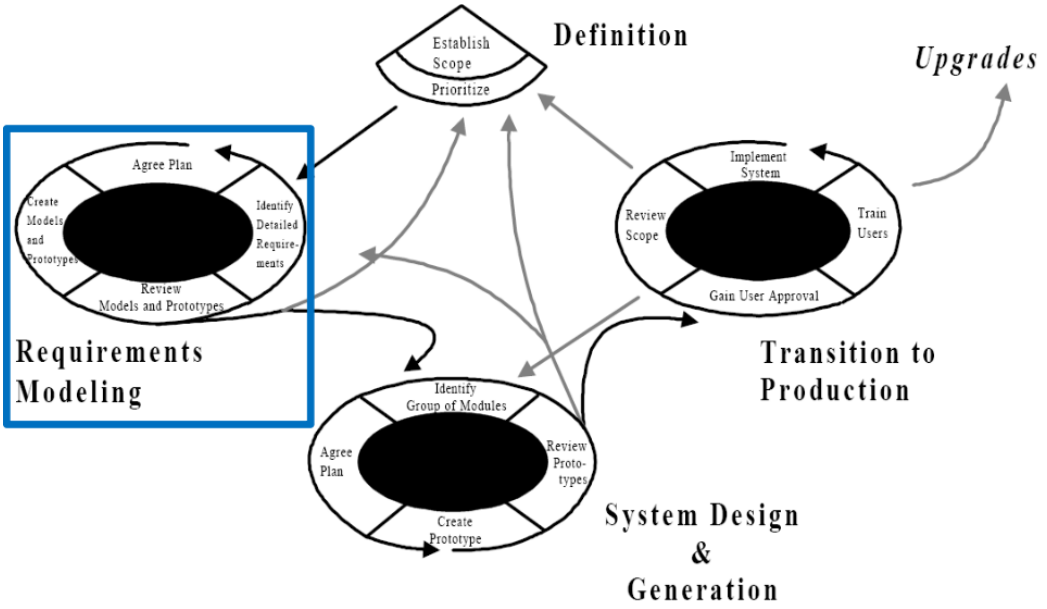
۴ـ۲ـ۱ متدلوژی توسعه

با توجه به پیچیدگی یک ســامانه نرم افزاری و نیاز های قابل گســترش یک نرم افزار، تدوین یک معماری کلی و مســتند ســازی آن نقش انکار ناپذیری برای مهیا کردن این گونه ســامانه ها بر عهده دارد. یک معماری خوب برای ایجاد یک دید کلی نه تنها باعث افزایش انعطاف پذیری و قدرت ا ستفاده مجدد و بزرگ تر سازی سامانه می شود، بلکه پیاده سازی آن را هم آسان تر ساخته و امکان مدیریت بخش های مختلفی که وابستگی آن ها کم است، فراهم می سازد؛ در واقع معماری خوبی که سامانه را به ماژول های تا حد امکان مستقل از هم تقسیم می نماید به ما این اجازه را می دهد تا هر بخش را به صــورت مســتقل و موازی اجرا کرده و همچنین قدرت استفاده از ابزار های مهندسی نرم افزار را برای ما مهیا می کند.

این سامانه با بهره گیری از متدلوژی توسعه ی نرم افزار CDM[۱۴]، مدل توسعه سریع[۱۵]، از شــرکت اوراکل[۱۶]، و به صــورت کاملا شــی گرا[۱۷] و با رعایت الگوهای موجود در این زمینه طراحی و تحلیل، طراحی و پیاده ســازی گردیده اســت. این متدلوژی شــامل چهار فاز تعریف،

[۱۴] Custom Development Method
[۱۵] Fast-Track
[۱۶] Oracle®
[۱۷] Object-Oriented

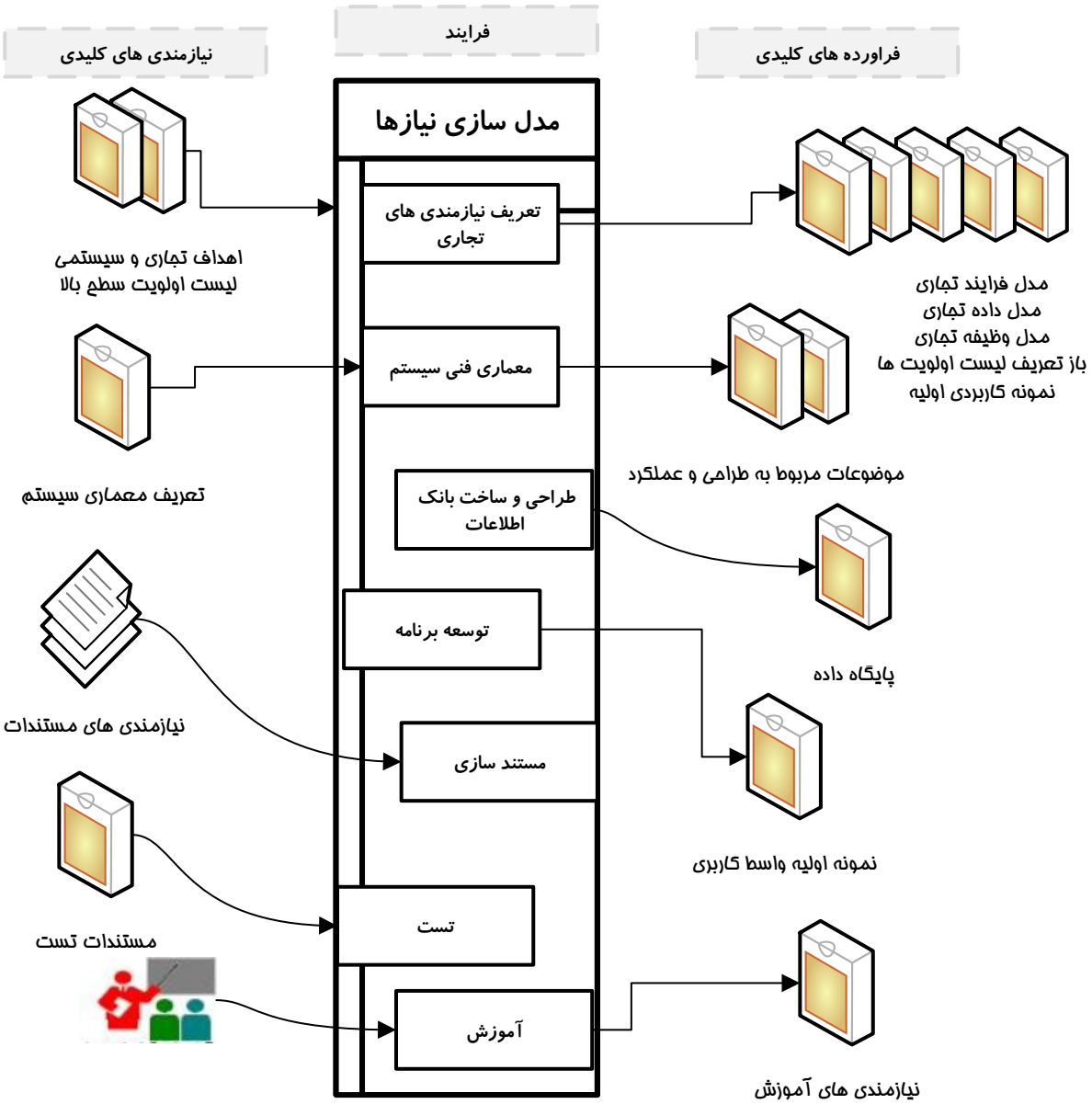
مدل سازی نیازمندی ها، طراحی و تولید و انتقال و عملیاتی سازی نهایی می با شد. فاز تعریف در فصل اول بیان شد. در این جا به بررسی سامانه در قالب سه فاز باقی مانده می پردازیم. فازهای نامبرده در شکل ۴-۱به تصویر کشیده شده اند.



شکل ۴-۱. فازهای متدلوژی *CDM FAST TRACK* و جایگاه فاز تحلیل و مدل سازی نیازمندی ها

۴ـ۲ـ۲ مدل سازی نیازمندی ها

فاز تحلیل نیازمندی ها به طور مستقیم حاصل از نتایج صحبت کارفرما بوده و توسعه دهندگان از مجموعه مصاحبه های انجام شده تحلیل های لازم برای طراحی سامانه را استخراج و باز مهند سی کرده اند. از جمله فراورده های کلیدی این فاز می توان به ساختار و پیکر بندی نهایی مدل داده، شمای اولیه واسط کاربری برنامه، مدل منطق تجاری و نیز نیازمندی های آموزش کار با برنامه اشاره کرد. فراورده های نامبرده به عنوان ورودی، مورد مصرف فاز بعدی (طراحی و تولید) قرار خواهند گرفت. موارد ذکر شده به همراه جزئیات داخلی فاز در شکل شکل ۴-۲ به تصویر کشیده شده اند.



شکل ۴-۲. نیازمندی های و فراورده های کلیدی فاز تحلیل و مدل سازی نیازها

۵ نتیجه گیری و کارهای آتی

۶ منابع و ماخذ

❊❊❊