

**دانشکده مهندسی کامپیوتر**

گروه مهندسی نرم افزار

شـبکه اطلاعاتی پـل 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 1395

**چکیده**

اطلاعات مربوط به پل ها به قدری وسیع و حائز اهمیت است 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 است 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 استفاده 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**

[1 مقدمه 1](#_Toc467445262)

[1ـ1 طرح مسئله 1](#_Toc467445263)

[1ـ2 راه حل پیشنهادی 2](#_Toc467445264)

[1ـ3 ساختار مستند 2](#_Toc467445265)

[2 مطالعات پیشین 3](#_Toc467445266)

[2ـ1 پیشینه مدیریت پل 3](#_Toc467445267)

[2ـ2 پایگاه های داده NoSQL 4](#_Toc467445268)

[2ـ2ـ1 مقایسه NoSQL ها و RDBMS ها 5](#_Toc467445269)

[2ـ2ـ2 طبقه بندی ساختاری NoSQL ها 7](#_Toc467445270)

[2ـ3 نظریه CAP 9](#_Toc467445271)

[3 توزیع غیر رابطه ای OrientDB 11](#_Toc467445272)

[3ـ1 انگیزه 11](#_Toc467445273)

[3ـ2 Multi-Model 12](#_Toc467445274)

[3ـ3 نسخه های برنامه 12](#_Toc467445275)

[3ـ4 نصب OrientDB 13](#_Toc467445276)

[3ـ5 راه اندازی Server 14](#_Toc467445277)

[3ـ6 اتصال به Server 14](#_Toc467445278)

[3ـ7 محیط OrientDB 16](#_Toc467445279)

[3ـ8 رابط های برنامه کاربردی و درایور 16](#_Toc467445280)

[3ـ8ـ1 توابع 16](#_Toc467445281)

[3ـ8ـ2 ساخت توابع 17](#_Toc467445282)

[3ـ8ـ3 فراخوانی توابع در SQL 17](#_Toc467445283)

[3ـ9 استفاده از توابع با API جاوا 18](#_Toc467445284)

[4 شبکه اطلاعاتی پل ها 19](#_Toc467445285)

[4ـ1 ماژول های سامانه 19](#_Toc467445286)

[4ـ2 مهندسی نرم افزار سامانه 20](#_Toc467445287)

[4ـ2ـ1 متدلوژی توسعه 20](#_Toc467445288)

[4ـ2ـ2 مدل سازی نیازمندی ها 21](#_Toc467445289)

[5 نتیجه گیری و کارهای آتی 23](#_Toc467445290)

[6 منابع و ماخذ 24](#_Toc467445291)

**فهرست شکل ها**

[شکل ‏2‑1. شمای مفهومی رئوس مثلث CAP 10](#_Toc467446059)

[شکل ‏3‑1 اتصال به OrientDB از طریق خط فرمان 15](#_Toc467446060)

[شکل ‏3‑2 اتصال با استفاده از OrientDB Studio 15](#_Toc467446061)

[شکل ‏3‑3واسط تحت وب OrientDB 16](#_Toc467446062)

[شکل ‏4‑1. فازهای متدلوژی CDM Fast Track و جایگاه فاز تحلیل و مدل سازی نیازمندی ها 21](#_Toc467446063)

[شکل ‏4‑2. نیازمندی های و فراورده های کلیدی فاز تحلیل و مدل سازی نیازها 22](#_Toc467446064)

**فهرست جدول ها**

[جدول ‏2‑1. مقایسه پایگاه داده های NoSQL با پایگاه داده های رابطه ای 5](#_Toc467446201)

# مقدمه

## 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 مديريت تعمير و نگاهداري پل ها بخش جدايي ناپذير مديريت كلان زير ساخت هاي حمل و نقل را بخود اختصاص مـي دهـد. در كلان شهرهاي ايران نيز با توجه به توسعه شبكه حمل و نقل احساس نياز به پياده سازي اين گونه سامانه ها به طور روز افزون احساس مي 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[[1]](#footnote-1) 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.

## راه حل پیشنهادی

پس از بررسی های انجام شده و تحلیل و ارزیابی سامانه های مشابه، تصمیم گرفته شد تا با استفاده از فناوری های جدید در عرصه پایگاه داده ها، سامانه جدید تحت عنوان «شبکه اطلاعتی پل ها[[2]](#footnote-2)» ارایه گردد.

در این سامانه از پایگاه داده NoSQL به نامOrientDB استفاده شده است که یک پایگاه داده چند مدلی است. مدلی که برای ذخیره سازی پل ها و ارتباط های بین آن ها به کار رفته **گراف[[3]](#footnote-3)** است.

## ساختار مستند

مستندات نهایی این پروژه به گونه ای تهیه شده که هم به صورت مرجع (جهت ارجاع به یک بخش خاص) و هم به صورت گزارش (جهت مطالعه ی بالا به پایین)، قابل استفاده بوده و ویژگی های سامانه را به درستی بیان کند. این مستندات در زمان های خاص خود و به صورت مجزا تهیه شده و نهایتا در یک سند اصلی درج شده اند، که اکنون پیش روی شماست. 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 شده است. در این فصل همچنین مقدماتی راجع به پایگاه داده های NoSQL مطرح گردیده است. در پایان فصل 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. فصل چهارم معماری فنی سامانه پیشنهادی را تشریح می کند و نهایتا در فصل پنجم موارد انجام شده مرور و ضمن نتیجه گیری از آن ها، پیشنهاداتی برای کارهای آتی عنوان گردیده است.

**"Not Only SQL"حرکتی است برای تشویق توسعه 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.**

# مطالعات پیشین

## پیشینه مدیریت پل

قدمت فرآيند تعمير و نگهداري پل ها به صورت سنتي را مي توان به قدمت ساخت و بهره برداري از پل ها دانست. ماهيت اين نوع از عمليات تعميـر و نگهداري غالبا عكس العملي بوده است، يعني بواسطه تخريب بخشي از سازه پل بهره برداران نسبت به تعمير و يا جايگزيني اعضاي پـل اقـدام مـي نمودند. پس از فاجعه و فرو ريزش پل سيلور در سال 1967زنگ خطر براي متصديان بهره برداري از سيستم حمل و نقل جاده اي بـه صـدا در آمـد. پس از اين فاجعه برنامه ملي بازرسي پل ها در اداره فـدرال راه ايـالات متحـده پايـه ريـزي شـد.

تخريـب پـل رودخانـه ميـانوس در سـال 1983و اختلالات گسترده ترافيكي ناشي از آن در شمال شرق آمريكا بر گستردگي موضوع افزود و بواسطه آن نياز بـه برنامـه ريـزي و ايجـاد سيسـتم جـامع مديريتي براي نگهداري و تعمير پل ها بيش از پيش احساس شد. در همين راستا در سال 1985برنامه تحقيقات بزرگراهي پـروژه اي در راسـتاي راه اندازي يك سيستم موثر مديريت پل ها تعريف و پياده سازي نمود. در سال 1991قانون جامع بهره وري حمل و نقل چند سطحي كليه ايالـت هـا را ملزم نمود تا نسبت به توسعه، پياده سازي و اجراي سيستم مديريت تعمير و نگهداري پل ها تا اكتبر 1998اقدام نمايند. بواسطه قانون ياد شده برخي تلاش هاي ايالتي نيز در برخي ايالت ها مانند پنسيلوانيا، كاروليناي شمالي و اينديانا منجر به توسـعه سيسـتم هـاي مديريت تعمير و نگهداري پل ها بصورت محلي و منطقه اي گرديد.

مديريت پل همگی اقدامات و فعالیت ها در طول عمر پل را از ساخت تا تعويض براي تامين ايمني و كارايي، شامل مي شود. گـزارش OECD در خصوص مديريت پل ها، سيستم مديريت پل را به عنوان «ابزاري براي كمك به متوليان بزرگ راه ها و مديريت پل در انتخاب رشد بهينه شبكه پـل هـا كه با سياست هاي متوليان و اهداف دراز مدت و محدوديت هاي بودجه سازگار باشد»، تعريف كرده است. مديريت پل در خصوص اولويت بنـدي احتياجات نگهداري، برنامه ريزي روش كار اجراي نگهداري و بهينه سازي هزينه طول عمر پل مي باشد. به طور خاص تر مديريت پل در ارتبـاط بـا كارهاي زير است:

* ارزيابي شرايط پل
* ارزيابي ظرفيت باربري
* پيش بيني زوال
* تعيين راهكارهاي موجود و اثربخشي آنها، طول عمر و هزينه
* ضوابط اولويت بندي احتياجات نگهداري
* تصميم گيري براي تخصيص منابع
* توسعه سيستم مديريت اطلاعات

برآورد مناسب ترين استراتژي نگهداري براي تعداد بسیار زيادي پل مسئله بزرگ و پيچيده اي است؛ زيرا، پارامترهاي متعددي براي تعيين اقتصـادي تـرين راه حل وجود دارد. در نتيجه تنها راه عملي براي مديريت پل اثر بخش، استفاده از سامانه ها و نرم افزارهاي رایانه ای است. تعدادی از این سامانه ها عبارتند از:

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

## 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 [[4]](#footnote-4)RDBMS یا سنتی پاسخ گوی نیاز این نرم افزارها، توسعه دهندگان و استفاده کنندگان آن نمی باشند. 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 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.

### مقایسه NoSQL ها و RDBMS ها

برای درک بهتر مفهوم پایگاه داده های NoSQL و نحوه عملکر آن ها شاید بهتر باشد تا ویژگی های آن ها را درکنار ویژگی های پایگاه داده های رابطه قرار داد و مقایسه کرد. جدول ‏2‑1 مقایسه ای کوچک میان پایگاه داده های NoSQL یا غیر رابطه ای و RDBMS یا رابطه ای را نشان می دهد.

جدول ‏2‑1. 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 1970 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*** |
| مدل نگهداری داده | 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 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. |

### 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:

1. Key-Value Store (کلید - مقدار)،
2. Column Family Store (خانواده ستونی)،
3. Document Store (سندگرا)،
4. 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 و.Sparksee

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.

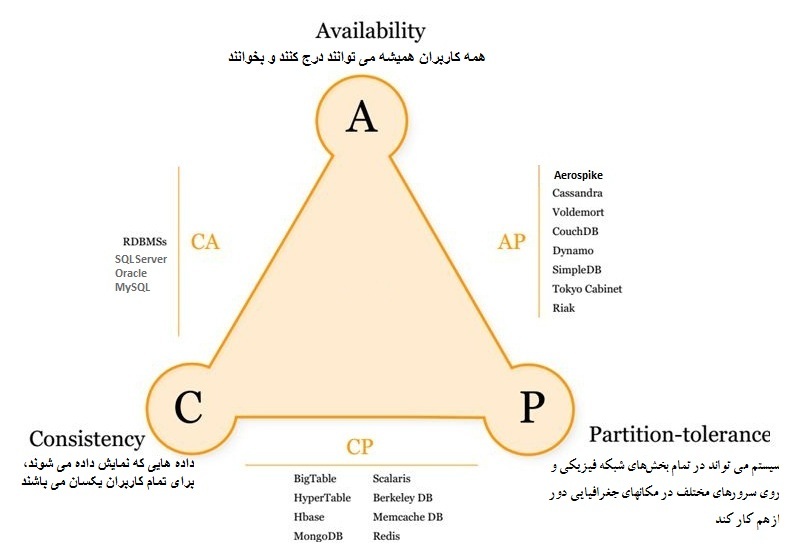
## نظریه CAP

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 3 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 ‏2‑1 یک شمای مفهومی از ارتباط این سه عامل را در قالب رئوس یک مثلث نشان می دهد.



شکل ‏2‑1. شمای مفهومی رئوس مثلث CAP

طبق این نظریه پایگاه داده ها در واقعیت فقط می توانند بر اساس دو رأس شکل بگیرند. البته پایگاه داده هایی نیز هستند که قابلیت های راس سوم را هم، به استثنا برخی از قابلیت ها، دارند ولی در اکثریت فقط دو رأس می توانند تجمیع شوند. 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 استفاده از پایگاه داده های رابطه ای مناسب می باشد. یا باید AP باشد که در این صورت پایگاه داده های غیر رابطه ای یا NoSQL پیشنهاد داده می شوند که معمولا از نوع کلید-مقدار و یا ستونی هستند و یا CP باشد 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 ‏2‑1 چند نمونه از این پایگاه داده ها بر اساس اضلاع مثلث پیشنهاد شده اند.

# توزیع غیر رابطه ای OrientDB

## 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 است برای تشویق توسعه 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[[5]](#footnote-5) مدت هاست که وجود دارند. اما مواردی که در اولویت بوده اند، زمینه هایی مثل مخابرات، پزشکی، CAD [[6]](#footnote-6) 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 مسئله 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

در سال های نه چندان دور، توسعه دهندگان سیستم هایی طراحی می کردند که می توانست 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.

## 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.

## نسخه های برنامه

محصول OrientDB در دو نسخه موجود است:

**نسخه عمومی**[[7]](#footnote-7)**:** این نسخه به عنوان یک پروژه ی متن باز[[8]](#footnote-8) تحت مجوز آپاچی 2[[9]](#footnote-9) 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
* Querry Profiler
* مانیتورینگ زنده با هشدار های قابل تنظیم
* و ...

## 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 1.7 به بعد جاوا نیز مورد نیاز است. برای دریافت این نسخه ها می توان به آدرس زیر مراجه 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[[10]](#footnote-10) شده فراهم شده است. کافی است بر اساس سیستم عامل خود، پکیج مربوط را دانلود کرده و به کامپیوتر خود انتفال دهیم. این پکیج شامل تمامی فایل ها ئیست 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.

## 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.

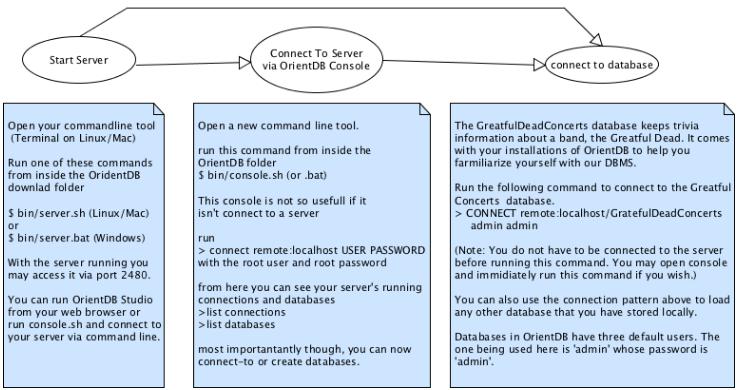
ضمناً جهت متوقف کردن Server می بایست از فایل shutdown.bat در ویندوز و یا shutdown.sh در سیستم های بر پایه Unix استفاده کنیم.

## اتصال به Server

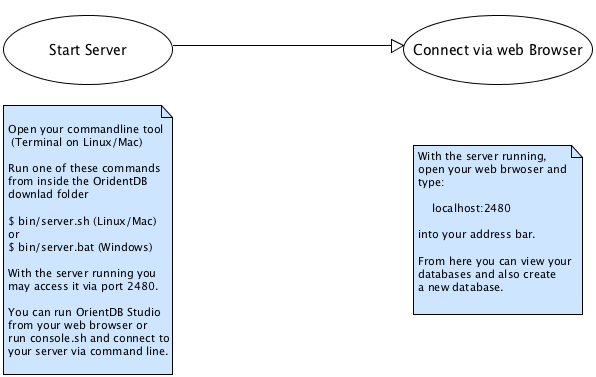
پس از راه اندازی Server، به دو روش می توان به آن متصل شد. روش اول با استفاده از console و روش دوم با مرورگر وب می باشد.

به طور پیشفرض OrientDB برای اتصال به محیط خارجی، بر روی دو پورت مختلف شنود انجام می دهد. یکی پورت 2424 برای اتصال از طریق console و دیگری پورت 2480 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:

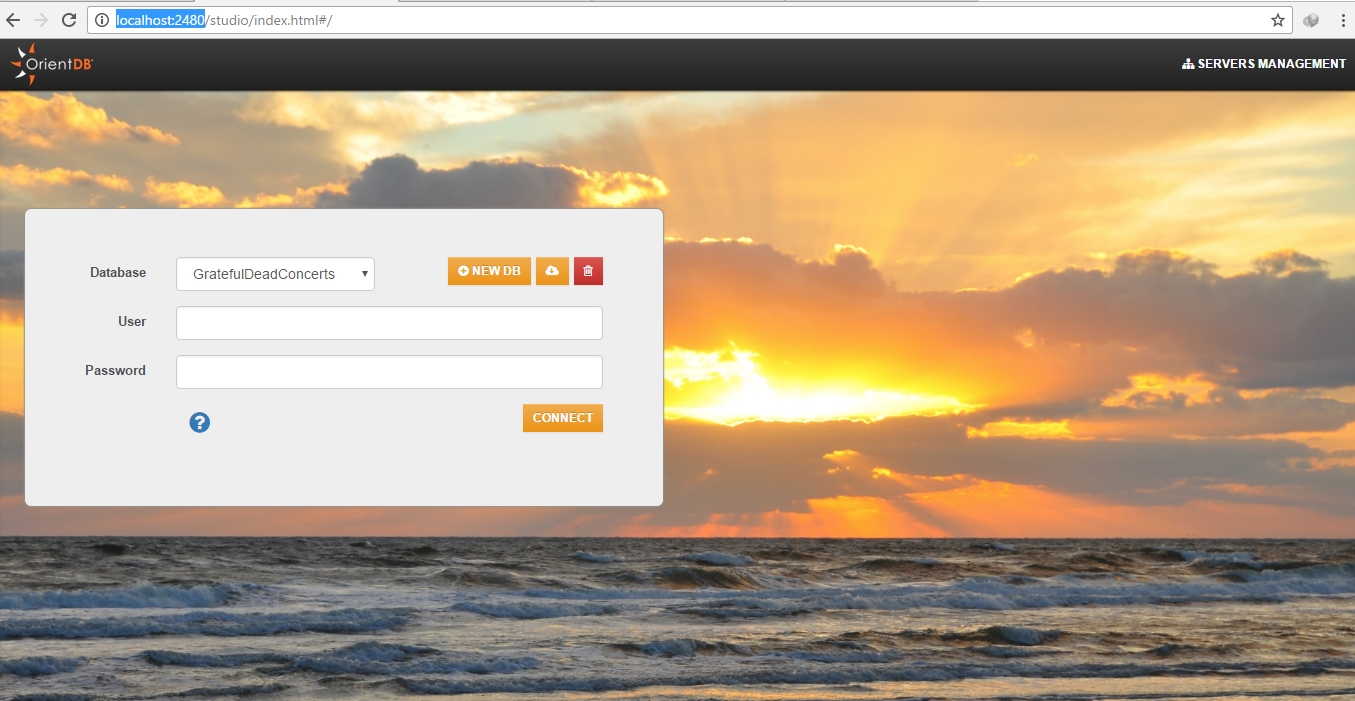


شکل ‏3‑1 اتصال به OrientDB از طریق خط فرمان



شکل ‏3‑2 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 مرورگر خود وارد آدرس <http://localhost:2480> می شویم.



شکل ‏3‑3واسط تحت وب OrientDB

## 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[[11]](#footnote-11) یک پایگاه داده، وجود دارد.

## رابط های برنامه کاربردی[[12]](#footnote-12) و درایور

### توابع[[13]](#footnote-13)

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:

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.

### 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 Functionsمی 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.

### 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.

## استفاده از توابع با API جاوا

# شبکه اطلاعاتی پل ها

## ماژول های سامانه

شبکه اطلاعاتی پل ها از پنج پیمانه نرم افزاری یا ماژول اصلی تشکیل شده است. 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:

1. ماژول مدیریت سوال
2. ماژول مدیریت پل
3. ماژول گزارش گیر
4. ماژول مدیریت کاربر
5. 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، تشکیل پرسشنامه 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 مدیریت کاربران نرم افزار و اعمال آن ها در نرم افزار می پردازد. مهم ترین وظیفه این ماژول در سامانه ایجاد سطوح دسترسی متفاوت برای کاربران است به نحوی که هیچ کاربری نتواند به قسمت هایی که اجازه ندارد، دسترسی 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 شده 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 پسوند ***.brg*** صورت 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.

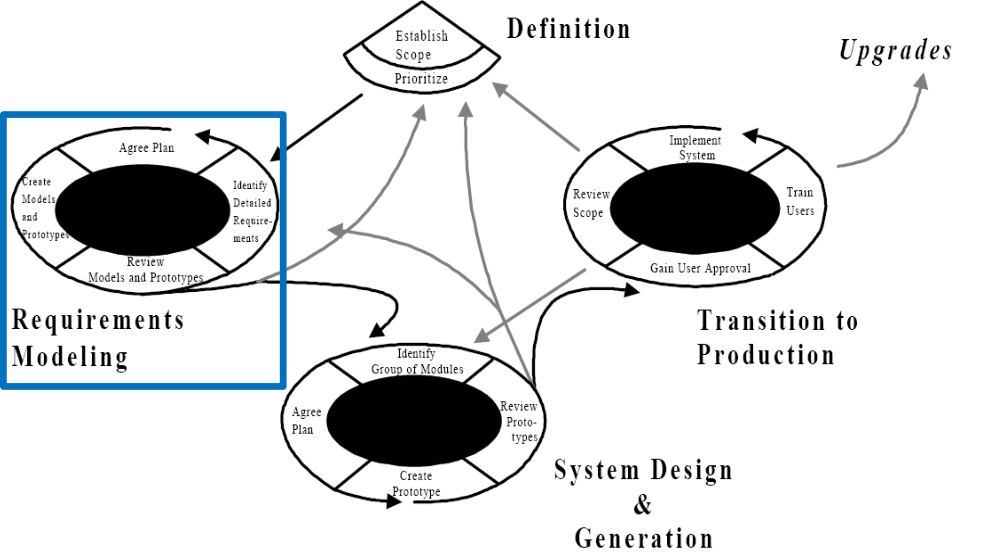
## مهندسی نرم افزار سامانه

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

### متدلوژی توسعه

با توجه به پیچیدگی یک سامانه نرم افزاری و نیاز های قابل گسترش یک نرم افزار، تدوین یک معماری کلی و مستند سازی آن نقش انکار ناپذیری برای مهیا کردن این گونه سامانه ها بر عهده دارد. یک معماری خوب برای ایجاد یک دید کلی نه تنها باعث افزایش انعطاف پذیری و قدرت استفاده 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 صورت مستقل 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 [[14]](#footnote-14)CDM، مدل توسعه سریع[[15]](#footnote-15)، از شرکت اوراکل[[16]](#footnote-16)، و به صورت کاملا شی گرا[[17]](#footnote-17) و با رعایت الگوهای موجود در این زمینه طراحی و تحلیل، طراحی و پیاده سازی گردیده است. این متدلوژی شامل چهار فاز تعریف، مدل سازی نیازمندی ها، طراحی و تولید و انتقال و عملیاتی سازی نهایی می باشد. 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 ‏4‑1به تصویر کشیده شده اند.



شکل ‏4‑1. فازهای متدلوژی CDM Fast Track و جایگاه فاز تحلیل و مدل سازی نیازمندی ها

### 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 مهندسی 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 ‏4‑2 به تصویر کشیده شده اند.



شکل ‏4‑2. نیازمندی های و فراورده های کلیدی فاز تحلیل و مدل سازی نیازها

# نتیجه گیری و کارهای آتی

# منابع و ماخذ

\*\*\*

1. Big Data [↑](#footnote-ref-1)
2. Bridge Information System [↑](#footnote-ref-2)
3. Graph Base [↑](#footnote-ref-3)
4. Relational Database Managements System [↑](#footnote-ref-4)
5. RDBMS [↑](#footnote-ref-5)
6. Computer Aided Design [↑](#footnote-ref-6)
7. Community Edition [↑](#footnote-ref-7)
8. Open Source [↑](#footnote-ref-8)
9. Apache 2 [↑](#footnote-ref-9)
10. Compile [↑](#footnote-ref-10)
11. Import [↑](#footnote-ref-11)
12. API [↑](#footnote-ref-12)
13. Functions [↑](#footnote-ref-13)
14. Custom Development Method [↑](#footnote-ref-14)
15. Fast-Track [↑](#footnote-ref-15)
16. Oracle® [↑](#footnote-ref-16)
17. Object-Oriented [↑](#footnote-ref-17)