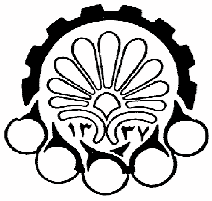
به نام آنکه جان پروانه اوست

بسمه‌تعالی



دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

دانشکده ریاضی و علوم‌کامپیوتر

پایان‌نامه کارشناسی ارشد

موضوع:

طراحی یک لایه ماناساز جنبه‌گرا با پشتیبانی از پرس‌وجوی شی‌گرا

تهیه کننده:

محمدرضا جوینده رودسری

استاد راهنما:

دکتر سید مهدی تشکری هاشمی

آبان ۱۳۷۸

دانشگاه صنعتی امیرکبیر
(پلی‌تکنیک تهران)

بسمه تعالی

فرم اطلاعات پایان‌نامه
کارشناسی- ارشد و دکترا

تاریخ:
شماره:

معاونت پژوهشی
فرم پروژه تحصیلات تکمیلی ۷

مشخصات دانشجو:

نام و نام خانوادگی: محمدرضا جوینده دانشجوی آزاد ● بورسیه ○ معادل ○

شماره دانشجوئی: دانشکده: رشته تحصیلی: گروه:

مشخصات استاد راهنما:

نام و نام خانوادگی: سید مهدی تشکری هاشمی درجه و رتبه: دانشیار

نام و نام خانوادگی: درجه و رتبه:

مشخصات استاد مشاور:

نام و نام خانوادگی: درجه و رتبه:

نام و نام خانوادگی: درجه و رتبه:

عنوان پایان‌نامه به فارسی :

طراحی یک لایه ماناساز جنبه‌گرا با پشتیبانی از پرس‌وجوی شی‌گرا

عنوان پایان‌نامه به انگلیسی:

Designing an Aspect-Oriented Persistence Layer Supporting Object-Oriented Querying

نوع پروژه: کارشناسی○ ارشد ● دکترا ○ سال تحصیلی: ۸۷-۸۸

کاربردی ● بنیادی ○ توسعه‌ای ● نظری ○

تاریخ شروع : بهمن ۸۶ تاریخ خاتمه: آبان ۸۷ تعداد واحد : ۶ سازمان تأمین کننده اعتبار :

واژه‌های کلیدی به فارسی: ماناسازی، جنبه‌گرایی، پرس‌وجوی شی‌گرا، پرس‌وجوی مجتمع در زبان، چارچوب .NET.

واژه‌های کلیدی به انگلیسی : Persistence, Aspect-Orientation, Object-Oriented Query, Language Integrated Query, .NET Framework

مشخصات ظاهری	تعداد صفحات ۹۲	تصویر● جدول● نمودار ○ نقشه ○ واژه‌نامه ●		تعداد مراجع ۳۹	تعداد صفحات ضمائم ۰
زبان متن	فارسی ●	انگلیسی ○	چکیده	فارسی ●	انگلیسی ●
یادداشت					

نظرها و پیشنهادها به منظور بهبود فعالیت‌های پژوهشی دانشگاه

استاد:

دانشجو:

امضاء استاد راهنما: **تاریخ:**

۱ : ارائه به معاونت پژوهشی به همراه یک نسخه الکترونیکی از پایان‌نامه و فرم اطلاعات پایان‌نامه بصورت PDF همراه چاپ چکیده (فارسی انگلیسی) و فرم اطلاعات پایان‌نامه
۲ : ارائه به کتابخانه دانشکده (شامل دو جلد پایان نامه به همراه نسخه الکترونیکی فرم در لوح فشرده طبق نمونه اعلام شده در صفحه خانگی کتابخانه مرکزی)
مرکزی)

چکیده

ماناسازی به مجموعه عملیاتی گفته می‌شود که عمل ذخیره و بازیابی اشیا برنامه‌های کاربردی را انجام می‌دهند. مفهوم ماناسازی بیشتر به دلیل ناهمگونی پایگاه‌های داده رابطه‌ای با مدل‌های شی‌گرا بوجود آمده است. در این پایان‌نامه ابتدا مفاهیم موجود در ماناسازی را مورد مطالعه قرار می‌دهیم. سپس جنبه‌گرایی مورد بررسی قرار می‌گیرد. جنبه‌ها موجوداتی هستند که اجرای برنامه را برای انجام عملی قطع می‌کند و فرآیندهایی به آن اضافه می‌کنند. در مورد جنبه‌گرایی ابتدا مفاهیم موجود در آن مورد بررسی قرار می‌گیرد و سپس روشی جهت پیاده‌سازی زیرساخت‌های لازم برای آن در چارچوب .NET ارائه می‌گردد. بررسی جنبه‌گرایی از آن جهت اهمیت دارد که می‌توان ماناسازی را به عنوان جنبه‌ای برای برنامه‌های کاربردی در نظر گرفت.

در این پایان‌نامه نشان داده می‌شود که چگونه بعضی از مسائل موجود در ماناسازی می‌تواند به عنوان یک جنبه مطرح شود. همچنین روشی به منظور فراهم‌کردن امکان پرس‌وجو به صورت شی‌گرا برای لایه‌های ماناساز در این پایان‌نامه معرفی می‌شود. ارائه پرس‌وجوهای شی‌گرا یکی از پرارزش‌ترین امکاناتی است که یک لایه ماناساز می‌تواند در اختیار طراحان و برنامه‌نویسان قرار دهد. چون با استفاده از پرس‌وجوهای شی‌گرا، پرس‌وجوها می‌توانند توسط خصوصیات شی‌گرایی و در همان حوزه‌ای که اشیا تعریف شده‌اند طراحی شوند و نیازی نیست که در حوزه رابطه‌ای طراحی گردند. در نتیجه امکان استفاده از پرس‌وجوهایی قدرتمندتر، ساده‌تر و با قابلیت نگهداری بیشتر بوجود می‌آید. بر اساس مولفه طراحی شده در این کار، مولفه‌ای نیز برای طراحی پرس‌وجوهای شی‌گرا بر روی گراف‌ها طراحی شده است.

کلمات کلیدی: ماناسازی، جنبه‌گرایی، پرس‌وجوی شی‌گرا، پرس‌وجوی مجتمع در زبان، چارچوب .NET.

فهرست مندرجات

پیش‌گفتار

پس از بوجود آمدن مفهوم شی‌گرایی در دهه ۱۹۶۰، عده زیادی از پژوهشگران به دلیل اینکه مدل رابطه‌ای نمی‌توانست کلیه خصوصیات این روش را پوشش دهد، به فکر ارائه روش‌های جدید به منظور ذخیره‌سازی اطلاعات افتادند.

از طرف دیگر همواره چالشی برای تهیه‌کنندگان نرم‌افزار وجود داشته و دارد و آن هزینه بالای پیاده‌سازی نرم‌افزارها است. به همین دلیل همواره این دسته به فکر کم کردن زمان پیاده‌سازی پروژه‌های نرم‌افزاری خود بوده و هستند. در راستای رسیدن به این هدف یکی از بهترین ایده‌های موجود، اتوماسیون تولید کد و همچنین استفاده از مدل‌های مبتنی بر مولفه[۱] و معماری‌های مبتنی بر سرویس[۲] است چون به خاطر امکان استفاده مجدد[۳] از آنها در پروژه‌های مختلف می‌توانند مورد استفاده قرار بگیرند. یکی از بخش‌های پروژه‌های نرم‌افزاری که می‌تواند به عنوان یکی از مهمترین کاندیداها برای اتوماسیون تولید تلقی شود، لایه مدیریت داده‌ها[۴] است. علت مهم بودن این کاندیدا از آنجا است که تقریبا در کلیه زبان‌های برنامه‌نویسی شناخته شده یک اشکال بزرگ وجود دارد. این اشکال بزرگ که تقریبا به عنوان یک اصل در دنیای نرم‌افزار پذیرفته شده است، محدود شدن طول عمر متغییرهای تعریف شده در برنامه به طول عمر

۱ Component Based
۲ Service Oriented Architechture
۳ Reusability
۴ Data Layer

برنامه است. بوضوح این محدودیت مطلوب هیچ برنامه‌نویسی نیست چون اکثر اطلاعاتی که در برنامه‌ها وجود دارند مستقل از باز و بسته‌شدن برنامه‌ها هستند بخصوص در نرم‌افزارهای بزرگ این مسئله به وفور دیده می‌شود. به عنوان نمونه هیچ شرکتی حاضر نیست که با بسته شدن برنامه‌اش اطلاعات پرسنل شرکت حذف شود و یا هیچ دانشگاهی نمی‌خواهد که با رفتن برق نمرات دانشجویان از بین برود. به همین دلیل کلیه تولیدکنندگان نرم‌افزار مجبور به استفاده از پایگاه‌های داده و نوشتن حجم زیادی برنامه، برای برقراری ارتباط با پایگاه‌های داده و استخراج و بروزرسانی اطلاعات خود هستند.

البته پژوهشگران کامپیوتر همواره سعی می‌کردند این مشکلات که ارتباط تنگاتنگی با یکدیگر دارند را حل کنند. مفهوم ماناسازی در دهه ۱۹۸۰ برای اولین بار مطرح شد و در سال ۱۹۸۷ آتکینسون[۵] در مقاله [۱] به بررسی جامعی در مورد آن پرداخت و تعدادی از مسائل کلیدی مربوط به ماناسازی را برشمرد. پس از آن دسته‌بندی که از مسائل بوجود آمد شامل سه شاخه اصلی بود. دسته اول طراحی پایگاه‌های داده شی‌گرا، دسته دوم طراحی زبان‌های با امکان ماناسازی و دسته سوم طراحی مولفه‌های ماناساز.

اولین زبانی که امکان ماناسازی را داشته در سال ۱۹۸۳ توسط آتکینسون [۲۰] معرفی شد که PSAlgol نام داشت. در ادامه زبان‌های دیگری نیز طراحی شدند که از آن جمله می‌توان به PJama[۶][۲۱]، E[۲۲]، PM3[۷][۲۴] و Glib[۲۳] اشاره کرد.

مولفه‌های ماناساز در دو شاخه نه چندان مستقل مورد بررسی قرار گرفتند که عبارتند از سرویس‌های ماناساز و لایه‌های ماناساز. در مورد سرویس‌های ماناساز در سال ۱۹۹۴ استانداردی توسط OMG[۸] برای میان‌افزار CORBA ارائه شد که جمع‌بندی از مطالعات انجام

۵ Malcolm P. Atkinson
۶ Persistent Java
۷ Persistent Modula-3
۸ Object Management Group

شده تا آن زمان را دربرداشت [۷]. همچنین به دلایل اشکالاتی که برای این استاندارد مطرح شد و تحقیقاتی که در سال‌های بعد در مورد سرویس‌های ماناساز انجام شد در سال ۲۰۰۲ استاندارد دیگری برای سرویس‌های ماناساز ارائه گشت[۸]. در مورد لایه‌های ماناساز نیز تحقیقات گسترده‌ای انجام شد که نتیجه این تحقیقات توسط اَمبر[۹] در کتاب‌های [۴]، [۵] و [٦] جمع آوری شده است.

یکی از موضوعات مهمی که در مورد لایه‌ها و سرویس‌های ماناساز مطرح شده است استفاده از پرس‌وجوهای شی‌گرا برای بازیابی اطلاعات می‌باشد. این مسئله بخصوص در زمانیکه از پایگاه‌های داده رابطه‌ای استفاده می‌شود بسیار مهم است. یکی از مهمترین استانداردهایی که در مورد این موضوع در سال ۲۰۰۰ ارائه شده است [۹]. در سال ۲۰۰٦ پرس‌وجوی مجتمع در زبان توسط میجر[۱۰] و همکارانش از شرکت مایکروسافت ارائه شد که نقطه عطفی در تحقیقات مربوط به پرس‌وجوهای شی‌گرا به شمار می‌رود [۲].

پس از معرفی مفهوم جنبه‌گرایی توسط کیزلس[۱۱] و همکارانش در سال ۱۹۹۷ در مقاله [۱۰]، ارتباط ماناسازی و جنبه‌گرای مورد توجه قرار گرفت. برای اولین بار در سال ۲۰۰۱ رشید[۱۲] موضوع ماناسازی جنبه‌های برنامه‌ها را مطرح کرد [۱۱] و دو سال بعد در مقاله دیگری مسائل موجود در ماناسازی جنبه‌گرا را مورد مطالعه قرار داد [۱۲] در ادامه این موضوع مورد توجه زیادی گرفت و مسائل مربوط به پیاده‌سازی آن نیز از این قاعده مستثنی نشد چون از یک جهت ارائه زیرساخت‌های ماناسازی جنبه‌گرا برای زبان‌های موجود و از جهت دیگر تبدیل لایه‌های ماناساز قدرتمند موجود به لایه‌های ماناساز جنبه‌گرا از اهمیت بالایی برخوردار بود.

در این پایان‌نامه به بررسی سرویس‌ها و لایه‌های ماناسازی، متدهای ماناسازی، ماناسازی

---

۹ Scott W. Amber
۱۰ Meijer
۱۱ Gregor Kiczales
۱۲ Awais Rashid

جنبه‌گرا و پرس‌وجوهای شی‌گرا پرداخته می‌شود. در فصل اول مفهوم مانا‌سازی و سپس متدهای آن مورد بررسی قرار می‌گیرند. در فصل دوم به مفهوم جنبه‌گرایی و ارائه روشی برای پیاده‌سازی آن در چارچوب NET. پرداخته می‌شود. فصل سوم بخش اصلی این پایان‌نامه را در بر می‌گیرد. در بخش اول آن در مورد مانا‌سازی جنبه‌گرا صحبت می‌شود. همچنین نشان داده می‌شود که چگونه مانا‌سازی را می‌توان به عنوان یک جنبه برای برنامه‌های NET. معرفی کرد. سپس پرس‌وجوی مجتمع در زبان مورد بررسی قرار می‌گیرد و در نهایت روشی جهت ارائه پرس‌وجوهای شی‌گرا در لایه‌های مانا‌ساز جنبه‌گرا معرفی می‌شود.

فصل ۱

ماناسازی و متدهای آن

ماناسازی به مجموعه عملیاتی گفته می‌شود که عمل ذخیره و بازیابی اشیا برنامه‌ها را انجام می‌دهند. عمل ماناسازی توسط لایه مانایی[۱] برنامه‌ها و یا سرویس‌های مانایی انجام می‌شود. در این بخش به بررسی مفاهیم پایه‌ای مربوط به ماناسازی پرداخته می‌شود. سپس خصوصیاتی که به قدرت یک لایه مانایی و یا سرویس مانایی می‌افزاید، مورد بررسی قرار می‌گیرد. در ادامه به بحث مهم شناسه‌ها پرداخته می‌شود. سپس متدهای پایه‌ای ماناسازی بررسی و مقایسه می‌شوند و درنهایت روشی برای تقویت متدهای معرفی شده بیان می‌گردد.

۱.۱ مفاهیم پایه‌ای

هر شی توسط اطلاعات درونی خود مشخص می‌شود که به این اطلاعات حالت آن شی می‌گوییم. همانطور که در شکل ۱.۱ مشاهده می‌شود، حالت شی را می‌توان به دو نوع متفاوت درنظر گرفت.

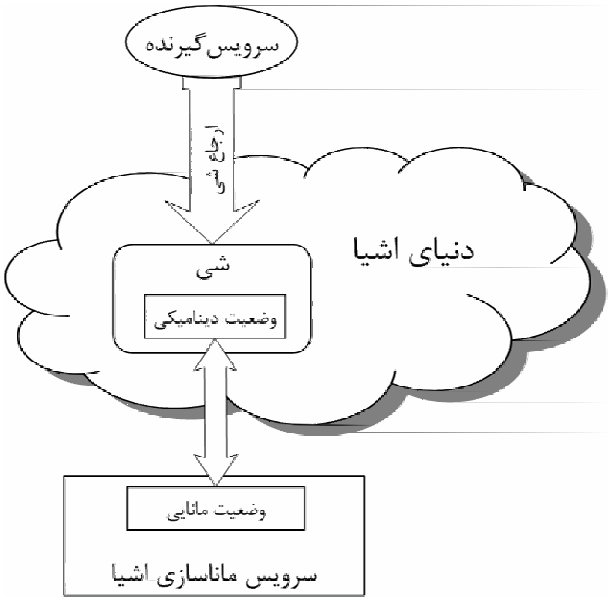
[۱] Persistence Layer

- **حالت دینامیکی**

حالت دینامیکی شی دربرگیرنده کل اطلاعات شی است و با از کار افتادن سیستم حفظ نمی‌شود. این حالت معمولا در حافظه موقت قرار دارد و ممکن است حاوی اطلاعاتی اضافی باشد.

- **حالت مانایی**

حالت مانایی شی دربرگیرنده اطلاعاتی است که بر اساس آن می‌توان حالت دینامیکی شی را بازسازی کرد. این حالت شامل اطلاعات اصلی شی است که امکان تولید کلیه اطلاعات شی از آن ممکن است.



شکل ۱.۱: حالت دینامیکی و مانایی اشیا

به عنوان نمونه اگر برای گراف دو خصیصه «ماتریس مجاورت» و «مرتبه»را در نظر بگیریم، هر دو خصیصه در وضعیت دینامیکی هر گرافی وجود دارند ولی تنها «ماتریس مجاورت» در وضعیت مانایی یک گراف قرار می‌گیرد. در شکل ۱.۱ حالت دینامیکی و مانایی یک شی در

مدل میزبان—مشتری مشخص شده است. وضعیت دینامیکی شی جزئی از خود شی محسوب می‌شود. در حالیکه وضعیت مانایی برای ساخت شی مورد استفاده قرار می‌گیرد. به همین علت وضعیت مانایی در لایه داده و یا سرویس ماناساز قرار دارد.

۲.۱ خصوصیات لایه یا سرویس مانایی

سرویس‌ها و لایه‌های مانایی رفتارهایی که برای ماناسازی حالت مانایی اشیا لازم است را بسته‌بندی[۲] می‌کنند. علاوه بر وظیفه بارگزاری و ذخیره‌کردن یک شی، یک سرویس و یا یک لایه مانایی قدرتمند باید خصوصیت‌هایی داشته باشد که در ادامه این بخش مورد بررسی قرار می‌گیرند.

۱) **مکانیزم‌های مختلف برای ماناسازی**

یک مکانیزم ماناسازی، یک تکنولوژی است که برای ذخیره، بازیابی، حذف کردن و همچنین تغییر دادن حالت مانایی اشیا مورد استفاده قرار می‌گیرد. نمونه‌هایی از مکانیزم‌های موجود عبارتند از:

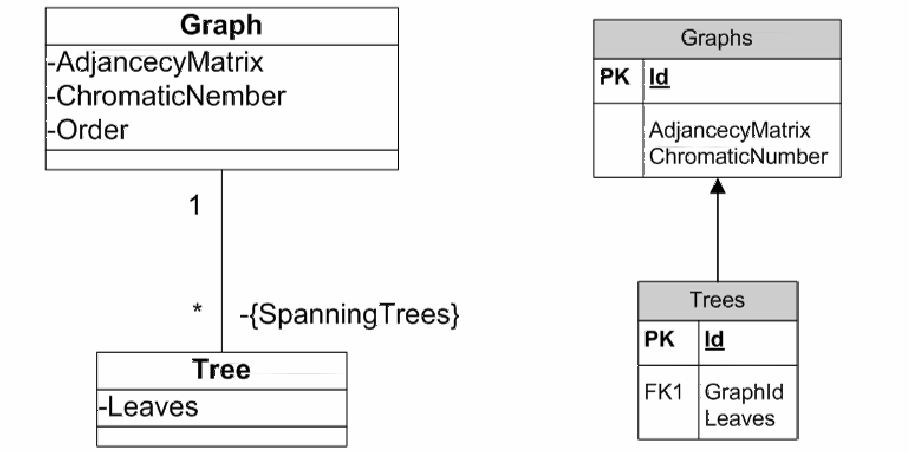
- فایل‌های مسطح[۳]
- پایگاه‌داده رابطه‌ای
- پایگاه‌داده شی‌گرا
- پایگاه‌داده توزیع‌شده

---

۲ Encapsulation
۳ Flat files

۲) متدهای مختلف ماناسازی

یک متد ماناسازی مشخص می‌کند که اشیا چگونه باید در مکانیزم‌ها ذخیره شوند. به عنوان نمونه اگر از پایگاه‌داده رابطه‌ای استفاده می‌کنیم. متدها مشخص می‌کنند که کلاس‌ها چگونه باید به جداول مرتبط شوند. در شکل ۲.۱ نمونه‌ای از یک مجموعه کلاس و مدل‌داده‌ای تولید شده برای آن آمده است. در مورد متدهای ماناسازی در بخش ۵.۱ به تفصیل بحث خواهیم کرد.



شکل ۲.۱: نمونه‌ای از یک مجموعه کلاس و مدل داده‌ای آن

۳) بسته‌بندی کامل مکانیزم‌ها ماناسازی

یک سرویس ماناساز یا لایه ماناساز ایده‌آل کلیه اعمال مکانیزم مورد استفاده را کپسوله می‌کند. به این معنی که تنها با دادن پیغام ذخیره یا حذف، سرویس یا لایه ماناساز باید کلیه اعمال مورد نیاز برای ذخیره یا حذف شدن شی مربوطه را انجام دهد و کاربر را درگیر چگونگی انجام آن نکند.

۴) **عملیات چند شی‌ای**

عملیات چند شی‌ای، به آن دسته از اعمال گفته می‌شود که با یک دستور چندین شی بارگزاری می‌شوند، تغییر می‌کنند و یا حذف می‌گردند. این اعمال در بسیاری از کاربردها بخصوص در ساخت گزارش‌ها مورد استفاده قرار می‌گیرند. به همین دلیل امکان انجام چنین اعمالی از اهمیت ویژه‌ای برخوردار است. زیرا در غیر این صورت هر دستور باید به ازای هر شی مستقلا انجام شود که به‌وضوح هزینه بسیار هنگفتی چه از لحاظ زمان اجرا و چه از لحاظ میزان اطلاعات انتقالی خواهد داشت.

۵) **اشاره‌گر**

وقتی عمل بازیابی چند شی به صورت همزمان انجام شود، باید امکان استفاده از اشاره‌گرها نیز وجود داشته باشد. چون حجم اطلاعاتی که بازیابی می‌شوند ممکن است به‌قدری زیاد باشد که بارگزاری کلیه آنها به صورت همزمان غیر ممکن باشد. یک اشاره‌گر یک ارتباط منطقی با مکانیزم ماناسازی است که توسط آن اشیا می‌توانند بارگزاری شوند. استفاده از اشاره‌گرها در بسیاری از موارد مفیدتر از بازیابی کل اطلاعات است.

٦) **اشیا پروکسی**

یک روش که در کنار اشاره‌گرها می‌تواند به افزایش سرعت دسترسی به اطلاعات کمک کند، استفاده از اشیا پروکسی است. استفاده از پروکسی‌ها از آن جهت مفید است که در بسیاری از کاربردها در یک پرس‌وجو تنها به دریافت اطلاعات کامل چند شی از میان کل مجموعه جواب نیاز داریم. در این دسته از کاربردها استفاده از اشیا پروکسی، به جای خود اشیا می‌تواند مفید باشد. به عنوان نمونه می‌توان به وب‌سایت‌های مربوط به پست‌الکترونیکی اشاره کرد که در پوشه پست‌های دریافتی به نشان دادن خلاصه‌ای از

اطلاعات پست‌ها اکتفا می‌کنند. خلاصه اطلاعات که شامل شناسه پست نیز می‌باشد، پروکسی‌هایی برای اصل پست‌ها محسوب می‌شوند. چون کاربران به صورت معمول تنها تعداد کمی از پست‌ها را مورد بررسی قرار می‌دهند، این عمل سرعت بررسی پست‌های الکترونیکی را به میزان قابل توجهی افزایش می‌دهد.

۷) **تراکنش**

مجموعه‌ای از عملیات باید بتوانند در یک تراکنش انجام شوند. یک تراکنش می‌تواند شامل ترکیبی از چند ایجاد، تغییر و حذف شی باشد. تراکنش‌ها می‌توانند به صورت تخت (تک لایه) و یا به صورت تودرتو مورد استفاده قرار گیرند. تراکنش‌ها می‌توانند دارای طول عمر کوتاه (کسری از ثانیه) و یا بلند (چند ساعت، روز و حتی ماه) باشند. همچنین تراکنش‌ها می‌توانند به صورت توزیع‌شده مورد استفاده قرار گیرند.

۸) **روش‌های مختلف برای تولید شناسه اشیا**

شناسه یک شی، فیلدی است که آن شی رابه طور یکتا مشخص می‌کند. شناسه در مدل شی‌گرایی، معادل مفهوم کلید در نظریه رابطه‌ای است. برای تولید شناسه روش‌های متفاوتی وجود دارد که در بخش ۴.۱ به تفصیل در مورد آنها صحبت خواهد شد.

۹) **امکان استفاده از کد** SQL

اگرچه استفاده از کد SQL در یک کد شی‌گرا برخلاف اصل بسته‌بندی است و باعث مخدوش شدن کد می‌شود، ولی برای بدست آوردن سرعت بیشتر در بعضی موارد نیازمند استفاده از کد SQL در کد برنامه هستیم. با توجه به این توضیحات سرویس یا لایه مانایی باید امکان دریافت کد SQL را هم داشته باشد.

۱۰) **تغییر پایگاه داده و مکانیزم**

چون امکان تغییر پایگاه داده و مکانیزم همواره وجود دارد، باید سرویس یا لایه مانا‌ساز اجازه این تغییر را به کاربران بدهد.

۳.۱ سرویس مانا‌سازی حالت‌ها

سرویس مانا‌سازی اشیا نقش کلیدی در ساختار سیستم‌های برپایه اشیا دارد. یک سرویس مانایی، همانند سایر سرویس‌ها واسط‌هایی دارد که از پیاده‌سازی‌های متفاوت که برای رسیدن به اهداف متفاوتی است پشتیبانی می‌کند. این واسط‌ها اجازه می‌دهند که مولفه‌های متفاوت با یکدیگر تعامل داشته باشند و وظایف خود را انجام دهند. اصلی‌ترین نیازمندی که در این سرویس باید پشتیبانی شود این است که به اشیا اجازه دهد تا کل یا بخشی از خود را جهت مانا‌سازی به سرویس معرفی کنند. همچنین باید یک روش وجود داشته باشد که اشیا بتوانند تصمیم بگیرند که چه بخشی از حالت دینامیکی خود را باید به سرویس معرفی کنند و چگونه می‌توانند اطلاعات خود را ذخیره و یا بازیابی کنند.

در این بخش به بررسی سرویس مانا‌سازی حالت‌ها بنا بر استاندارد OMG می‌پردازیم. OMG استانداردی برای طراحی سرویس‌مانایی بر پایه میان‌افزار CORBA ارائه کرده است [۷]. معیارهای تعیین شده در این مرجع اگرچه بر اساس میان‌افزار CORBA طراحی شده است. اما به دلیل سطح انتزاع بالایی که در طراحی آن در نظر گرفته شده، در کاربردهای دیگر نیز قابل استفاده است. هدف سرویس مانایی (سرویس مانا‌سازی حالت‌ها[۴]) ارائه یک مجموعه واسط مشترک برای مکانیزم‌های مختلفی است که همگی وظیفه ذخیره، بازیابی و مدیریت وضعیت مانایی را بر عهده دارند. این سرویس در کنار سرویس‌های دیگری می‌تواند مورد استفاده قرار

[۴] Persistent State Service

گیرد. از نمونه چنین سرویس‌هایی می‌توان به سرویس رابطه [۲۵]، سرویس تراکنش[۲۶] و سرویس مدیریت چرخه عمر [۲۷] اشاره کرد.

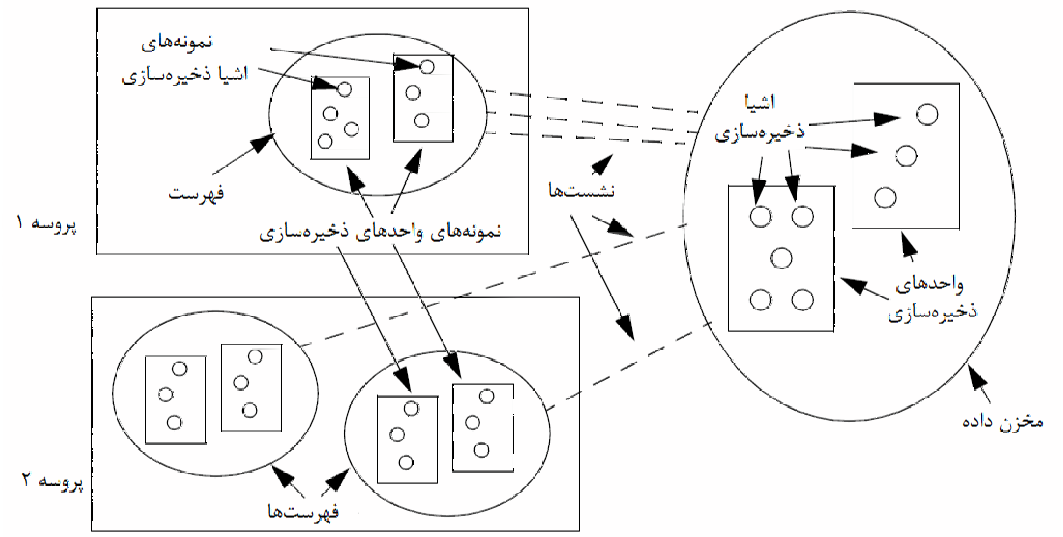
۱.۳.۱ مفاهیم پایه‌ای

سرویس ماناسازی حالت‌ها وظیفه مدیریت حالت مانایی اشیا را بر عهده دارد. یک سرویس مانایی این حالت‌ها را تحت عنوان اشیا ذخیره‌سازی[۵] و در واحدهای ذخیره‌سازی[۶] نگهداری می‌کند. واحدهای ذخیره‌سازی نیز خود در مخزن‌های داده[۷] ذخیره می‌شوند. برای اعمال تغییرات بر روی یک شی ذخیره‌سازی نیازمند داشتن یک شی در زبان برنامه‌نویسی هستیم که از آن استفاده می‌کنیم. به این اشیا نمونه‌های اشیا ذخیره‌سازی می‌گوییم. این نام از آن جهت انتخاب شده است که در زبان‌های برنامه‌نویسی شی‌گرا، به اشیا از نوع یک کلاس، نمونه‌های آن کلاس گفته می‌شود. مشابه اشیا ذخیره‌سازی برای مدیریت واحدهای ذخیره‌سازی، نیازمند داشتن نمونه‌های واحدهای ذخیره‌سازی هستیم. نمونه‌های واحدهای ذخیره‌سازی خود توسط اشیائی مدیریت می‌شوند که فهرست[۸] نامیده می‌شوند.

هر برنامه برای دسترسی به اشیا ذخیره‌سازی خود، باید یک اتصال منطقی بین پروسه خود و مخزن داده که اطلاعات آن برنامه را دارد برقرار کند. به این اتصال منطقی که می‌تواند بیش از یک مخزن داده نیز دسترسی داشته باشد یک نشست می‌گوییم. در شکل ۳.۱ مفاهیم معرفی شده در این بخش و رابطه آنها با یکدیگر مشخص شده‌اند.

---

۵ Storage Object
۶ Storage Home
۷ Datastore
۸ Catalog



شکل ۳.۱: مفاهیم پایه‌ای در سرویس ماناسازی حالت‌ها

۲.۳.۱ مخزن داده

هر مخزن داده مجموعه‌ای از واحدهای ذخیره‌سازی است. در یک مخزن داده هر واحد ذخیره‌سازی دارای یک نوع‌داده‌ای می‌باشد. همچنین در یک مخزن داده، برای هر نوع داده حداکثر یک واحد ذخیره‌سازی وجود دارد.

یک واحد ذخیره‌سازی شامل تعدادی شی ذخیره‌سازی است. هر شی ذخیره‌سازی دارای دو شناسه است که یکی درون واحد ذخیره‌سازی خود و دیگری درون فهرست مربوطه یکتا هستند. این دو شناسه به ترتیب شناسه مانایی کوچک[۹] و شناسه مانایی[۱۰] نامیده می‌شوند.

هر شی ذخیره‌سازی دارای یک نوع است که مشخص کننده اعضا و عملیات نمونه‌های آن شی ذخیره‌سازی است. یک نوع داده شی ذخیره‌سازی می‌تواند از نوع داده شی ذخیره‌سازی دیگری به ارث ببرد. یک واحد ذخیره‌سازی تنها می‌تواند اشیا ذخیره‌سازی از یک نوع خاص را

---

۹ Short-PId
۱۰ PId