1. **名词解释**
2. **ERP**

**全称：**Enterprise Resource Planning 企业资源计划

**定义：**是指建立在信息技术基础上，以系统化的管理思想，为企业决策层及员工提供决策运行手段的管理平台。

**用途：**将物资资源管理（物流）、人力资源管理（人流）、财务资源管理（财流）、信息资源管理（信息流）集成一体化。从而达到最佳资源组合，取得最佳效益。通过提高物流职能部门的工作效率、减少库存、缩短生产周期以及改进服务水平来降低成本。

1. **CIMS**

**全称：**Computer Integrated Manufacturing Systems 计算机集成制造系统

**定义：**CIMS是通过计算机硬软件,并综合运用现代管理技术、制造技术、信息技术、自动化技术、系统工程技术,将企业生产全部过程中有关的人、技术、经营管理三要素及其信息与物流有机集成并优化运行的复杂的大系统。

**用途：**通过计算机网络将企业的各个生产和管理环节的数据集成管理，从而达到降低库存、提高生产效率和管理水平。

1. **SCM**

**全称：**Supply Chain Management 供应链管理

**定义：**是一个将产品、服务和信息从供应商到客户最优化传递的过程。（是一种集成的管理思想和方法，它执行供应链中从供应商到最终用户的物流的计划和控制等职能。从单一的企业角度来看，是指企业通过改善上、下游供应链关系，整合和优化供应链中的信息流、物流、资金流，以获得企业的竞争优势。）

**用途：**它把公司的制造过程、库存系统和供应商产生的数据合并在一起，从一个统一的视角展示产品建造过程的各种影响因素。

1. **EAI**

**全称：**Enterprise Application Integration 企业应用集成

**定义：**EAI是多重集成的过程，独立开发的应用程序可能使用不兼容的技术，且需要被独立管理。

**用途：**集成基于各种不同平台、用不同方案建立的异构应用，实现EAI可以归结为在应用程序之间移动信息的过程，包括使应用程序能够发送、接收信息，并对信息做出反应。

1. **SOA**

**全称：**Service-Oriented Architecture 面向服务的体系结构

**定义：**SOA是一种应用程序的组织架构，是设计原则，指导如何设计应用程序。它的基本原理是通过组件或web service 提供的分布式通信能力，把系统中的功能抽象成一个个服务。在SOA中服务通过基于消息机制的定义明确的接口和调用协议相互作用，构成应用系统。

**用途：**使得构建在各种这样的系统中的服务可以一种统一和通用的方式进行交互。

1. **IIOP**

**全称：**Internet Inter-ORB Protocol 互联网内部对象请求代理协议

**定义：**Internet上的GIOP，称为IIOP（GIOP(General Inter-ORB Protocol) 是一种通信协议，规定了两个实体：客户和服务器ORBs间的通信机制。它是一种通用协议，不能直接使用，在不同的网络上需要有不同的实现。IIOP是Internet上的GIOP）。(PPT)

IIOP是一个用于CORBA 2.0及兼容平台上的协议。（百度）

**用途：**用来在CORBA对象请求代理之间交流。还提供了JAVA RMI 和CORBA的互操作能力。

1. **IDL**

**全称：**Interface Definition language（接口定义语言）

**定义：**是CORBA规范的一部分，是跨平台开发的基础。描述性语言，用于描述接口，不定义实现，类似C中的头文件。它不是真正的编程语言，需要将它映射到相应地程序设计语言上去（映射成Client Stub Code和Server Skeleton Code）。（它提供一套通用的数据类型，并以这些数据类型来定义更为复杂的数据类型？？？）

**用途：**用于描述接口。IDL使CORBA语言无关。

1. **ORB**

**全称：**Object Request Broker 对象请求代理

**定义：**它是 CORBA 的核心组件。ORB 提供了识别和定位对象、处理连接管理、传送数据和请求通信所需的框架结构。

**用途：**在一个面向对象的分布式计算环境中为应用程序、服务器、网络设施之间分发消息提供关键通信设施。

1. **SOAP**

**全称：**Simple Object Access Protocol 简单对象访问协议

**定义：**SOAP是在松散的、分布的环境中使用XML交换结构化的和类型化的信息的一种简单协议,本身并不定义任何应用语义,只定义了一种简单的以模块化的方式包装数据的机制, 可以使用任何底层传输协议，如HTTP、FTP、SMTP等，其中最常用的是HTTP协议。

**用途：**在网络应用程序之间交换结构化和类型化数据。它是XML格式的消息交换协议，用来执行服务调用。

1. **WSDL**

**全称：**Web Services Description Language 网络服务描述性语言

**定义：**WSDL基于XML格式，定义了Web Service接口，描述了服务的操纵信息、服务提供了什么功能、服务位于何处、服务如何调用。

**用途：**用来描述Web服务和说明如何与Web服务通信。它是基于XML的组件描述，用来描述服务。

1. **MOM**

**全称：**Message Oriented Middleware 基于消息的中间件

**定义：**指利用高效可靠的消息传递机制进行平台无关的数据交流，并基于数据通信来进行分布式系统的集成。它通过消息传递来完成分布式计算环境下数据和控制的处理,采用多种机制来保证消息可靠、高效、安全。

**用途：**在不同平台之间通信，常被用来屏蔽掉各种平台及协议之间的特性，实现应用程序之间的协同，其优点在于能够在客户和服务器之间提供同步和异步的连接，并且在任何时刻都可以将消息进行传送或者存储转发。

1. **UDDI**

**全称：**Universal Description, Discovery and Integration 统一描述、发现和集成

**定义：**UDDI是一套基于Web的、分布式的、为Web服务提供的信息注册中心的实现标准规范，同时也包含一组使企业能将自身提供的Web服务注册以使别的企业能够发现的访问协议的实现标准。

**用途：**使得不同的企业能够以相同的方式描述自己提供的服务和查询对方提供的服务

可通过Web访问的注册中心，用来发布、查找服务。

1. **WSFL**

**全称：**Web Services Flow Language 网络服务流程语言

**定义：**Web Service间工作流描述，是一个描述商业过程的规范。

**用途：**将分散的、功能单一的Web服务组织成一个复杂的有机应用。

1. **ODBC**

**全称：**Open DataBase Connectivity 开放数据库互连

**定义：**微软倡导的、当前被业界广泛接受的、用于数据库访问的应用程序编程接口（API），它以X/Open和 ISO/IEC的调用级接口(CLI)规范为基础，并使用结构化查询语言（SQL）作为其数据库访问语言。

**用途：**ODBC为所有DBMS功能都定义了公共接口；ODBC定义了API和SQL语法一致层，它规定驱动程序应支持的基本功能；ODBC还提供两个函数（SQLGetInfo和SQLGetFunctions）返回关于驱动程序和DBMS能力的一般信息及驱动程序支持的函数列表。

1. **问答**
2. **应用集成大致分为几种类型，分别解决什么样的问题？**

答：

1. **表示集成：**即软件用户界面，它可以为原来基于终端的应用软件提供PC界面，或提供一个由多组件合成的应用软件；
2. **数据集成：**直接访问软件创建、维护并储存的信息，如批量文件传输，开放式数据连接ODBC，数据库访问中间件，数据转换，可以用来对多个信息源综合数据进行分析和决策，向多个应用软件提供公共信息源的只读权限，以一个信息源的信息来更新另一个数据源；
3. **功能集成：**是代码级别的软件集成，如面向消息的中间件，分布式对象技术，事务处理监控器等，它能够解决前两种方法可解决的问题，要求新软件具有其他程序的功能，在集成中暗含工作流，确保应用间的事务完整性。
4. **什么是格式良好的XML文档？**

答：格式良好的文档遵守 XML 语法，但没有 DTD 或模式。

应满足以下条件：

* 1. 元素规则：名字中不能包含空格，名字不能以数字或标点符号开头，名字不能以任何大小写的xml开头，左尖括号（<）后不可以有空格，起始和结束标签的大小写必须一致，XML文件中出现的第一个元素是根元素，根元素必须有完整的起始和结束标签，所有的子元素必须嵌套在一个根元素中，嵌套元素不可以相互重叠，子元素如果内容为空可以缩写标签。
  2. 根元素：在XML文件中有且只能够有一个根元素。XML文档必须包含在一个单一元素中。这个单一元素称为根元素，它包含文档中所有文本和所有其它元素。
  3. 必须有结束标记。在XML文档的空元素中，您可以把结束斜杠放在开始标记中。
  4. 属性必须有用引号括起的值。
  5. XML文件的第一行必须是声明该文件是XML文件以及它所使用的XML规范版本。在文件的前面不能够有其它元素或者注释。
  6. 标记之间不得交叉。
  7. 控制标记、指令和属性名称等英文要区分大小写。

|  |
| --- |
| （百度）格式良好的(well-formed)XML文档是指一个遵守XML语法规则，并遵守XML规范的文档。“格式良好”的xml文档除了要满足根元素唯一的特性之外，还包括：起始标签和结束标签应当匹配，结束标签是必不可少的；大小写应一致：XML对字母的大小写是敏感的，和是完全不同的两个标签，所以结束标签在匹配时一定要注意大小写一致；元素应当正确嵌套：子元素应当完全包括在父辈元素中。 |

1. **试解释XML中的命名空间，并描述其使用的语法？**

答：

|  |
| --- |
| （MSDN）XML命名空间将XML文档中的元素和属性名称与自定义和预定义的URI关联起来。为命名空间 URI 定义的前缀用来限定 XML 数据中的元素和属性的名称以实现此关联。 命名空间可防止元素和属性名称冲突，并允许以不同的方式处理和验证同名的元素和属性。  命名空间是使用xmlns:属性在元素上声明的，而且此属性的值就是标识该命名空间的 URI。命名空间声明的语法是xmlns:<name>=<"uri">，其中<name>是命名空间前缀的名称，<"uri">是说明命名空间 URI 的字符串。一旦声明后，前缀就可以用来限定XML文档中的元素和属性并将它们与命名空间URI关联。因为命名空间前缀在整个文档中使用，所以它的长度应较短。 |

|  |
| --- |
| （百度）当我们建立XML应用的时候，会为具体的行业应用创建特定的DTD，规定可用的元素，有时候会出现两个同名的元素在不同的地方有不同的含义，这就需要指定命名空间来避免冲突。在XML中，采用全球唯一的域名作为“Namespace”，即用URL作为Namespace。  命名空间的定义语法如下：  xmlns:[prefix]=”URI of namespace”  其中，xmlns:是必须的，它是XML的关键字；prefix是命名空间的别名。 |

1. **解释DOM和SAX解析XML文件的不同之处？**

答：

DOM是面向文档的对象式解析，定义了层次化对象模型来表示XML文档。即为XML语法中的每个概念(如元素，属性，实体，文档等)定义对应的类，而解析器在读入XML文档的时候，会建立XML语法和类之间的一一映射。

SAX是面向文档的流式解析，它是一种基于事件的解析过程，解析器顺序读取XML文档，产生一个对应的事件流，并向事件处理程序发送所捕获的各种事件，如元素开始和元素结束等。

SAX能够立即读取数据，而不是等待所有的数据被处理；不需要将整个文档一次加载到内存，效率较高；易用性的降低，流式解析编程较为复杂；单遍解析特性，不支持随机访问。

SAX解析方式无法更改数据和不支持随机访问，而DOM解析可以。DOM的层次化对象模型是一个树形结构，它将一个XML文档看作一棵节点树，每个节点代表一个XML文档中的元素。

|  |
| --- |
| （王姐姐版）   * 事件驱动的解析器SAX（Simple API for XML）  1. 顺序地处理XML数据，每次处理一个组成部分。应用程序通过回调来处理XML数据，XML解析器在解析之后并不维护元素的树结构或者任何数据 2. SAX是XML 简易应用程序编程接口(Simple API for XML),与DOM不同,它是一种事件驱动接口,它提供了处理开发者感兴趣的特定元素的方法,而不必要求在应用层次处理之前预先建立元素,因此可以不用创建没有用的结构,从本质上说,SAX是一种JAVA接口,它能给应用程序提供较大的灵活性.  * 基于树的解析器DOM（Document Object Model）  1. 将整个XML文档构造成树形结构，并提供对树中单个节点的访问。 2. 分析器将一个XML文档转换成一个对象模型的集合，即通常所说的DOM树，应用程序正是通过对DOM树的操作，来实现对XML文档数据的操作  * 不同之处：  1. DOM是基于树形结构的W3C推荐的API标准,SAX是事件驱动的有广泛支持的API标准 2. DOM适合与结构化编辑XML文档,如排序,记录移动等,SAX适合内存不足和文档结构无关的行为,如计算XML文档结点数,提取特定节点内容等 3. DOM整体装入和处理XML文档,系统资源占用大,效率低,速度慢,SAX 是一种事件驱动接口,不需要把整个XML文档加载到内存,只需要处理关心的数据,对内存的占用不会随着文档的大小而有所变化,效率高,速度快 4. SAX是一种快速而简单的接口,绝大多数的解析器可以由开发者自己编程实现.  * 选择 DOM 还是 SAX，这取决于几个因素：  1. 应用程序的目的：如果必须对数据进行更改，并且作为 XML 将它输出，则在大多数情况下，使用 DOM。与使用 XSL 转换来完成的简单结构更改不一样，如果是对数据本身进行更改，则尤其应该使用 DOM。 2. 数据的数量：对于大文件，SAX 是更好的选择。 3. 将如何使用数据：如果实际上只使用一小部分数据，则使用 SAX 将数据抽取到应用程序中，这种方法更好些。另一方面，如果知道将需要向后引用已经处理过的信息，则 SAX 可能不是正确的选择。 4. 需要速度：通常，SAX 实现比 DOM 实现快。   总结：SAX 和 DOM 不是互斥的，这一点很重要。可以使用 DOM 来创建事件的 SAX 流，可以使用 SAX 来创建 DOM 树。事实上，大多数解析器实际常常使用 SAX 来创建 DOM 树！ |

1. **描述ODBC的基本结构和工作流程**

答：

**基本结构：**

基于SQL/CLI标准的实现工具ODBC/SDK V3.0有四部分：

1）应用程序：应用程序嵌有的SQL语句在运行时被转换为若干个动态连接库中的ODBC函数

2）驱动程序管理器：负责管理和调度驱动程序

3）驱动程序：是相应于某个数据源的ODBC函数执行码，存放于动态连接库，提供给应用程序调用。一个SQL/CLI接口一般可连接若干个DBMS，故有若干个驱动程序

4）数据源：提供的数据可以是RDBMS，也可以是OODBMS或各类文件形式



**工作流程：**

1）调用驱动程序管理器，把目标数据源对相应的驱动程序调入动态连接库；

2）根据SQL语句，调用动态连接库中若干个相应的ODBC函数；

3）执行ODBC函数，把SQL语句以字符串的形式传到数据源处；

4）数据源执行所收到的SQL语句，把结果返回应用程序。

1. **数据仓库的定义，与数据库的区别，以及数据仓库用到哪些数据集成技术**

答：

**数据仓库的定义：**数据仓库是面向主题的、集成的、稳定的、随时间变化的，主要用于决策支持的数据库系统。

**ODS**（Operational Data Store）：是“面向主题的、集成的、可变的、反映当前数据值的和详细的数据的集合，用来满足企业综合的、集成的以及操作型的处理需求”。ODS像DW一样是一种面向主题，集成的数据环境，又像操作型DB一样包含着全局一致的、细节的当前的数据。

**与数据库的区别（包括ODS）：**

|  |  |  |  |
| --- | --- | --- | --- |
| **比较项目** | **数据仓库** | **ODS** | **数据库（应用系统）** |
| **建设目的** | 决策支持 | 实时监控 | 业务操作 |
| **服务对象** | 企业管理层 | 业务管理层 | 生产层 |
| **存储周期** | 长期 | 短期 | 即时 |
| **处理频率** | 非实时 | 准实时 | 实时 |
| **主要功能** | 分析功能 | 事务处理，短时分析 | 事务处理 |
| **技术实现** | OLAP | OLAP、OLTP | OLTP |
| **功能结构** | 集中 | 相对集中 | 分散 |
| **数据类型** | 明细数据,汇总数据 | 明细数据 | 明细数据 |
| **数据容量** | 非常大 | 小 | 小 |

数据仓库用到的数据集成技术：OLAP、ETL

|  |
| --- |
| （百度）1.数据库只存放在当前值，数据仓库存放历史值；  2.数据库内数据是动态变化的，只要有业务发生，数据就会被更新，而数据仓库则是静态的历史数据，只能定期添加、刷新；  3.数据库中的数据结构比较复杂，有各种结构以适合业务处理系统的需要，而数据仓库中的数据结构则相对简单；  4.数据库中数据访问频率较高，但访问量较少，而数据仓库的访问频率低但访问量却很高；  5.数据库中数据的目标是面向业务处理人员的，为业务处理人员提供信息处理的支持，而数据仓库则是面向高层管理人员的，为其提供决策支持；  6.数据库在访问数据时要求响应速度快，其响应时间一般在几秒内，而数据仓库的响应时间则可长达数几小时 |

1. **XML Schema与XML DTD的区别**

答：XML Schema 是XML文档，遵循XML语法规则，而DTD不遵守XML语法，它有其特有的语法；

数据类型：XML Schema支持数据类型，而DTD数据类型有限（与数据库数据类型不一致）

XML Schema 是一个开放的模型

XML Schema 是可扩展的，而DTD不可扩展

命名空间的集成不同：XML Schema支持命名空间，而DTD不支持，因为命名冲突

1. **什么是数据库中的元数据？它有什么作用？**

答：

元数据：是描述、解释、定位或让自己更易于检索、利用或管理一个信息资源的结构化的信息。元数经常被称作数据的数据或信息的信息。

作用：

以一种统一和稳定的方式描述和组织存储在不同介质上的信息

创建描述性元数据的一个重要原因就是要使相关信息的发现更加容易

元数据是确保未来资源将存在并持续被访问的关键

通过资源发现，元数据可以有助于电子资源的组织，使交互操作和遗产资源集成，提供数据标识和支持存档和保存变得更加容易

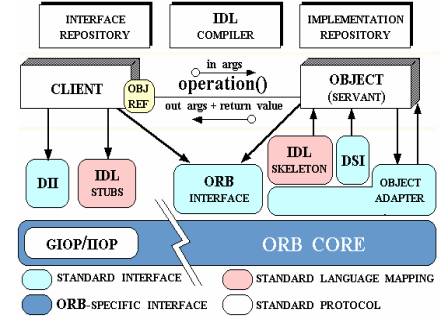
|  |
| --- |
| （王姐姐版）   * 元数据：描述数据的起源、意义和沿袭，为原始数据提供上下文环境   或元数据是对数据资源的描述，英文名称是“Metadata”，通常被解释为data about data，即关于数据的数据。元数据是信息共享和交换的基础和前提，用于描述数据集的内容、质量、表示方式、空间参考、管理方式以及数据集的其他特征。   * 分为：   静态（结构化）元数据  动态（可操作）元数据   * 关系数据库中，元数据记录物理表/属性名、域值和业务规则 * 元数据标准   1 MS：MDC、OIM  2 IBM&Oracle：OMG、CORBA  3 UML、XML   * 元数据的作用  1. 元数据主要有下列几个方面的作用：   (1)用来组织和管理空间信息，并挖掘空间信息资源，这正是数字地球的特点和优点所在。  (2)帮助数据使用者查询所需空间信息。  (3)组织和维护一个机构对数据的投资。  (4)用来建立空间信息的数据目录和数据交换中心。  (5)提供数据转换方面的信息。  SQL Server的元数据 ：  某个数据库中的表和视图的个数以及名称  某个表或者视图中列的个数以及每一列的名称、数据类型、长度、精度、描述等  某个表上定义的约束  某个表上定义的索引以及主键/外键的信息 |

1. **CORBA2.0（图、理解、应用）（这道题目的答案不准确，需要自已再找一下）**

答：

**CORBA：**Common Object Request Broker Architecture， 是一种异构平台下的语言无关的对象互操作模型。

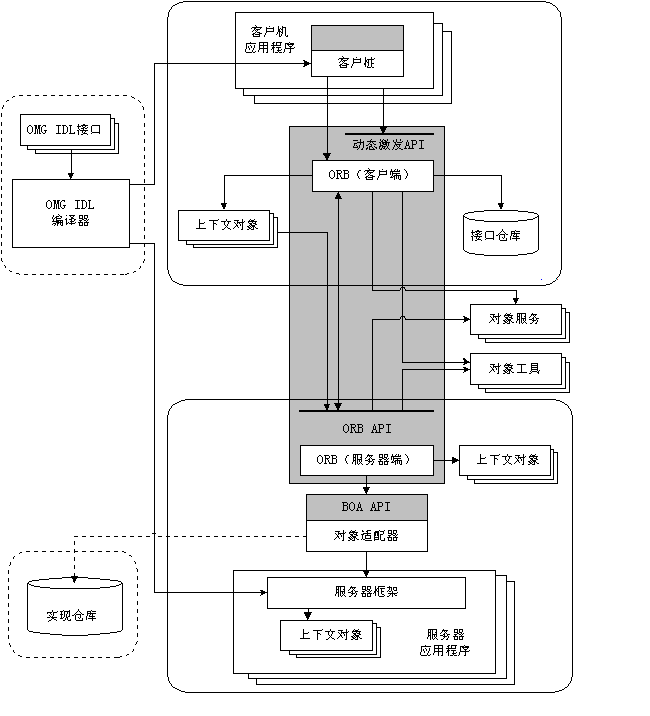
**CORBA体系结构：**



CORBA上的服务用IDL描述，IDL将被映射为某种程序设计语言如C++或Java，并且分成两分，在客户方叫IDL Stub（桩）, 在服务器方叫IDL Skeleton（骨架）。两者可以采用不同的语言。服务器方在Skeleton的基础上编写对象实现(Object Implementation)，而客户方要访问服务器对象上的方法，则要通过客户桩。而双方又要通过而ORB（Object Request Broker，对象请求代理）总线通信。

与传统的Client/Server模式（我们称为Two-tier client/server）不同，CORBA是一种multi-tier client/server architecture，更确切的说，是一种three-tier client/server模式。双重客户/服务器模式存在的问题是两者耦合太紧，它们之间采用一种私有协议通信，服务器的改变将影响到客户方。多重客户/服务器与此不同，两者之间的通信不能直接进行，而需要通过中间的一种叫代理的方式进行。在CORBA中这种代理就是ORB。通过它，客户和服务器不再关心通信问题，它们只需关心功能上的实现。从这个意义上讲，CORBA是一种中间件(Middleware)技术。

**CORBA体系结构图的理解：**

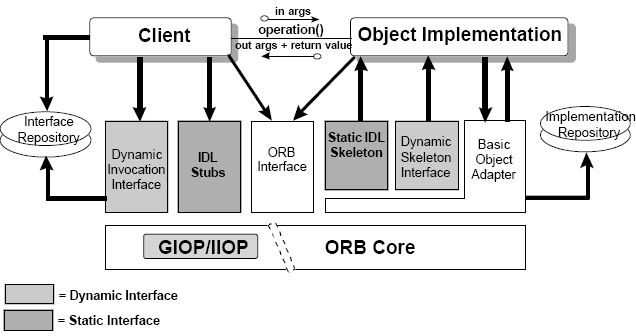


* 1. 客户机应用程序：客户机应用程序使用CORBA系统来激发定义在一些信息、数据、或应用程序对象上的操作，服务器应用程序则执行这些操作。客户机应用程序用桩类型激发API或动态激发API向服务器发送请求。
  2. 客户桩：一个客户机应用程序可以用客户桩给服务器应用程序发送请求。 客户桩通过使用OMG IDL编译器编译OMG IDL接口而产生。
  3. 动态激发API： 动态激发API是两种发送请求方式种的一种。该API是面向对象API。动态激发API在运行时从接口仓库里发现定义并用它来创建和激发向对象的请求。
  4. ORB（客户）：ORB处理请求与相应服务器和方法的选择。当一个应用程序为某个对象上的某个操作向ORB发送一个请求时，ORB检验参数是否跟接口一致并把请求转发至服务器，如果需要的话启动服务器。应用程序能够同步或异步地调用方法。 ORB是客户机应用程序的一部分。
  5. 上下文对象：上下文对象包含有关客户机、环境的信息。上下文对象被表示成一列属性及其值。

a）客户机应用程序用上下文对象来获取运行环境

* + 1. 而ORB用上下文对象中的信息来决定服务器的定位及被请求方法激活
  1. 接口仓库：存储各个接口信息的模块，例如用OMG IDL编写的接口定义、常量、类型定义等，它们被当成定义的一部分使用。接口仓库包括对指定对象有效的操作的描述和对操作有效的参数描述。接口仓库是CORBA系统的集成部分。
  2. ORB（服务器端）：ORB接收方法调度请求，激发服务器框架里的方法调度器，引导输出参数，并完成激发。
  3. 对象适配器：对象适配器处理与ORB相关的普通任务。
  4. 服务器框架：服务器框架提供调度请求到适当方法的必要代码。应用程序开发者通过编译接口定义得到服务器框架。

**CORBA的两种调用机制：**

****

1. 静态调用：在编译IDL是由编译器静态地产生IDL Stub和IDL Skeleton，Interface Repository中包含了注册了IDL定义的接口和方法和参数类型，它是一个运行时的元数据仓库。
2. 动态调用：Dynamic Invocation Interface，是CORBA提供的用来查找保存在Interface Repository中接口元数据的API，可以动态的使用这些数据产生远程方法调用；为服务器提供了运行时绑定的机制，通过这种机制，服务器可以处理那些在服务器端还不存在由IDL生成的Skeleton的CORBA对象。Dynamic Skeleton查看请求所附带的参数值，然后通过Interface Repository获得目标对象和相关的方法。
3. **试分析web service和SOA的关系**

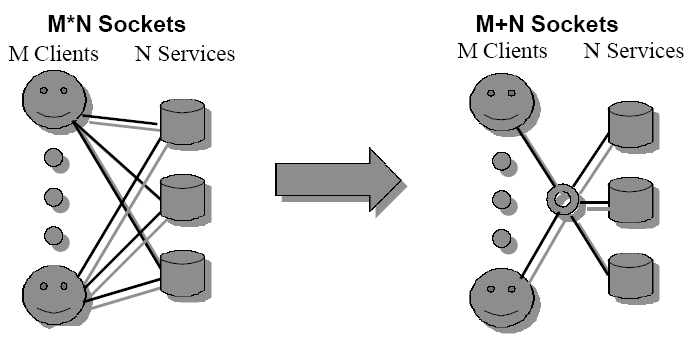
答：

|  |
| --- |
| （王姐姐版）SOA不是Web服务  1）SOA和Web服务的关系经常发生混淆：Web服务是技术规范，而SOA是设计原则。特别是Web服务中的WSDL，是一个SOA配套的接口定义标准  2）SOA是一种架构模式，而Web服务是利用一组标准实现的服务  3）Web服务是实现SOA的方式之一 |

1. **试分析消息中间件相对于分布式对象技术的优势**

答：

1. 消息中间件的优点在于能够在客户和服务器之间提供同步和异步的连接，并且在任何时刻都可以将消息进行传送或者存储转发，这也是它比远程过程调用更进一步的原因。另外消息中间件不会占用大量的网络带宽，可以跟踪事务，并且通过将事务存储到磁盘上实现网络故障时系统的恢复。当然和远程过程调用相比，消息中间件不支持程序控制的传递，不过这种功能和它的优势比起来却是无关紧要的。
2. 采用多种机制来保证消息可靠、高效、安全的传输，如：核心的异步机制保证并发的效率；多种通讯机制使用户的不同需要得到更好的解决；对树型和网状结构的支持、多路由的管理，提高了效率和适用性；采用优先级机制，区别不同的消息的优先级，有效提高网络资源的利用效率；以及断点续传、加密传输、压缩传输、滑动窗口、生命周期机制等。
3. 可扩展的体系结构：如图：简单的RPC或CORBA框架需要M\*N条连接，而消息中间件框架只需要M+N条连接：



1. 通讯程序可在不同的时间运行：消息放入适当的队列时，目标程序甚至根本不需要正在运行；即使目标程序在运行，也不意味要立即处理该消息对应用程序的结构没有约束。
2. 在复杂的应用场合中，通讯程序之间不仅可以是一对一的关系，还可以进行一对多和多对一方式，甚至是上述多种方式的组合；多种通讯方式的构造并没有增加应用程序的复杂性。
3. 使用点对点消息传送队列：由于多个接收者可使用同一队列中的消息，因此如果接收消息的顺序无关紧要，那么您可以平衡消息使用负载。要发送到队列的消息始终保留，即使没有接收者也是如此。Java客户端可使用队列浏览器对象检查队列内容。然后，它们可以根据通过该检查所获得的信息来使用消息。也就是说，虽然使用模型一般为 FIFO （first in, first out，先进先出），但如果使用者知道要使用哪些消息，即可使用消息选择器来使用未处于队列最前方的消息。管理客户端也可以使用队列浏览器来监视队列的内容。
4. 发布/ 订阅消息传送：实现了应用开发的松耦合性和企业体系结构的高度灵活，适合于企业应用的集成。消息的发送和接收根据主题进行，消息的生产者和接收者无需知道对方的存在、位置或状态， 处理过程简单，简化了应用的开发。支持动态系统重新配置，新应用的增加不会影响系统的正常运行。参与系统的组件无需知道其它组件的内部实现方式。任务的合理分布：事件订阅者和事件发布者任务分组，各负其责。另外，不同的订阅者 对事件采取的行动由各个订阅者决定。

|  |
| --- |
| （王姐姐版）  通讯程序可在不同的时间运行（传说中的异步传输）  1.1消息放入适当的队列时，目标程序甚至根本不需要正在运行  1.2即使目标程序在运行，也不意味要立即处理该消息  对应用程序的结构没有约束（可靠性更强）  2.1在复杂的应用场合中，通讯程序之间不仅可以是一对一的关系，还可以进行一对多和多对一方式，甚至是上述多种方式的组合  2.2多种通讯方式的构造并没有增加应用程序的复杂性 |

|  |
| --- |
| 1. **消息中间件中“点对点消息传送”和“发布/订阅消息传送”的异同**  * 点对点消息传送   在点对点消息传送中，消息生成方被称为发送者，而使用方则被称为接收者  它们通过被称为队列的目标来交换消息：发送者生成队列中的消息；而接收者则使用队列中的消息   * 发布/ 订阅消息传送   1 由在网络上发布消息的应用和订购特定主题消息的应用共同组成。当消息由产生者按一定的主题发布后，消息的订购者根据自己的需要，接收特定主题的消息  2 最大好处：实现了应用开发的松耦合性和企业体系结构的高度灵活，适合于企业应用的集成。   * 不同点总结：   1 “点对点消息传送”是发送者和接收者之间不存在时间上的相关性，“消息/订阅传送”是发布者与订户之间存在时间上的相关性  2 “点对点消息传送”中每条消息只能由一个接收者使用，在“消息/订阅传送”是多订户可以访问同一个主题，也就是每条消息可以由多用户接收，不同的订阅者 对事件采取的行动由各个订阅者决定。  3 “点对点消息传送”是针对消息队列进行发送消息，每个用户只能接受特定的信息，而“消息/订阅传送”是针对主题进行发送消息，而消息的获取是由用户自己觉得获取相关的消息 |