名词解释

1. ERP

全称是：Enterprise Resource Planning（企业资源计划）

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。

2. CIMS

全称是：Contemporary Integrated Manufacturing Systems（现代集成制造系统）或者 Computer Integrated Manufacture System（计算机集成制造系统）

计算机集成制造系统是 1973 年由美国学者提出的一种生产管理组织技术和方式，其基本思想是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。

3. SCM

全称是：Supply Chain Management（供应链管理）

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。

4. EAI

全称是：Enterprise Application Integration（企业应用集成）

EAI 是将基于各种不同平台、用不同方案建立的异构应用集成的一种方法和技术。

5. SOA

全称是：Service-oriented architecture（面向服务的体系结构）

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。

6. IIOP

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

提供了 JAVA RMI 和 CORBA 的互操作能力。

7. IDL

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

是 CORBA 规范的一部分，是跨平台开发的基础。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

8. ORB

ORB 的全称是：Object Request Broker（对象请求代理）。

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。

9. SOAP

全称：Simple Object Access Protocol（简单对象访问协议）。

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。

10. WSDL

全称：Web Services Description Language（web 服务描述性语言），是一个用来描述 Web 服务和说明如何与 Web 服务通信的 XML 语言。

11. MOM

全称是：message oriented middleware（基于消息的中间件），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。

12. UDDI

UDDI 的全称是：Universal Description, Discovery and Integration

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

13. ODBC

ODBC 的全称是：Open DataBase Connectivity

它是微软倡导的、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。

主要用途有: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。

问答题

1. 应用集成大致分为几种类型，分别解决什么样的问题？

- 六种类型的分法

根据技术的成熟程度，可以将集成分为 6 个层次：

平台集成；数据集成；组件集成；应用集成；流程集成；B2B 集成；

1 表示集成：软件用户界面。为原来基于终端的应用软件提供 PC 界面。提供一个由多组件合成的应用软件

（案例：1 为大型机提供 windows 界面；2 为 SAP R/3 与大型机程序提供统一的 HTML 界面；3 为多个大型机应用程序提供统一的基于 Java 的界面）

2 数据集成：直接访问软件创建、维护并存储的信息。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。

（案例：1 综合 sybase、DB2 和 SAP P/3 数据库中的数据；2 使用大型机和 Oracle 的可执行信息系统；3 允许其他应用程序在 peoplesoft 和定制的 Oracle 数据库中获取数据）

3 功能集成：代码级别的软件集成。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

（案例：1 获取用户信息，对 java 程序、大型机程序、Oracle 数据库作更新；2 把供应商的系统集成到采购系统中）

4 业务流程集成:商务逻辑。通过子流程来实现业务流程共享。支持技术和商务标准协议。支持快速实施。帮助企业获得全面业务透视能力，从而让企业可以全面掌控业务

5 B2B 集成:不同贸易协议。企业内部的多个应用系统与多个外部信息系统进行交互。M×N 交互场景（M 个内部系统与 N 个外部系统进行交互）。每个系统都有各自的数据交换格式，这些格式不一定需要被外部系统所理解。

- 三种类型的分法

根据对原有信息化应用的数据和逻辑的访问方式，应用集成层次主要分为：

1 界面层集成：主要围绕系统的用户和界面方面的信息进行集成，其特征是通过被集成对象的界面的逻辑关系完成应用集成工作。

2 数据层集成：围绕应用中的数据资源进行集成，主要提供对企业内部异构数据的互通。

3 功能层集成：基于应用系统之间的功能或业务流程的关联而是先的应用集成，实在企业业务逻辑的层面上的应用集成。

2. 什么是格式良好的 XML 文档？

格式良好的文档遵守 XML 语法，但没有 DTD 或模式。格式良好的XML文档要满足四原则：

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。XML 文档包含在一个单一元素中。文档有注释在根元素之外；那是完全合乎规则的。而不包含单一根元素的文档不管该文档可能包含什么信息，XML 解析器都会拒绝它。

3. 必须有结束标记：

不能省去任何结束标记。在 XML 文件中的标记必须正确地关闭，也就是说，在 XML 文件中，控制标记必 须有与之对应的结束标记。如果一个元素根本不包含标记，则称为空元素；HTML 换行（
）和图像（<img>）元素就是两个例子。在 XML 文档的空元素中，您可以把结束斜杠放在开始标记中。

下面的两个换行元素和两个图像元素对于 XML 解析器来说是一回事：

<img src="../img/c.gif"></img>

4. 属性必须有用引号括起的值：

XML 文档中的属性有两个规则：

1. 属性必须有值
2. 那些值必须用引号括起。

可以使用单引号，也可以使用双引号，但要始终保持一致。属性值必须要用“ ”号括起来。

除以上的四个原则外，格式良好的 XML 文件还须满足：

1. 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。
2. 标记之间不得交叉。
3. 控制标记、指令和属性名称等英文要区分大小写。

3. 试解释 XML 中的命名空间，并描述其使用的语法？

- 定义：

XML 文件可以设计一个 XML 的 DTD，产生新的标记语言。在一个 XML 文档中，可以包含由多个 DTD 描述的元素。但是这些元素之间很容易产生名字冲突。命名空间就是为解决这类问题而存在。例如：可以 books=”http://www.library.com/books:title”, 然后在文档中这样使用 <books:title>。 这样的 URI 标志就叫做名字空间。

- 命名方法：

名字空间使用两段式命名法。第一段是代表特定命名空间的“命名空间前缀”；第二段是元素或属性原来的名字； 两段之间用冒号“：”分开。 所谓名字空间前缀，就是在元素名和属性名前面增加一个标识，以唯一区分当前元素或属性来自哪一个 DTD。

- 名字空间声明的语法：

<prefix : elementName xmlns : prefix = ‘URI'/> ；声明后，prefix 就代表‘URL’。

当然，名字空间不只是根元素才可以用的，子元素同样可以使用名字空间。

名字空间声明有两种方式:直接定义方式和缺省定义方式:

- 直接定义 xmlns:<名字空间前缀> = <名字空间名>
- 缺省定义 xmlns = <名字空间名>

名字空间声明的属性名部分由两部分组成，即前缀"xmlns:"和名字空间前缀，且名字空间前缀是一个合法的 XML 名称。在缺省方式下，名字空间声明的属性名部分仅有保留属性名 xmlns，属性值部分与直接定义方式相同。

- 名称空间作用域和默认名字空间：

名字空间的作用域就是名字空间作用的范围。这个范围就是声明该名字空间的元素以及该元素中所有的子元素，除非该元素的某个子元素也声明了一个嵌套的名字空间。

XML 文档中的标记默认情况时都没有使用名字空间前缀来限制。没有使用名字空间前缀来限制的元素属于默认名字空间的控制范围。

默认名字空间的声明如下： <elementName xmlna='URI'/>

在遵循名字空间规范的 XML 文档中，标记不能包含这样的两个属性:

(1)属性名完全相同，

(2)或属性的本地部分完全相同，并且其前缀被绑定到相同的名字空间名。

具体例子见 IBM 关于 XML 的课件。

4. 解释 DOM 和 SAX 解析 XML 文件的不同之处？

- 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！

5. 描述 ODBC 的基本结构和工作流（JDBC）

- 定义：

Open DataBase Connectivity

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。

- 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。

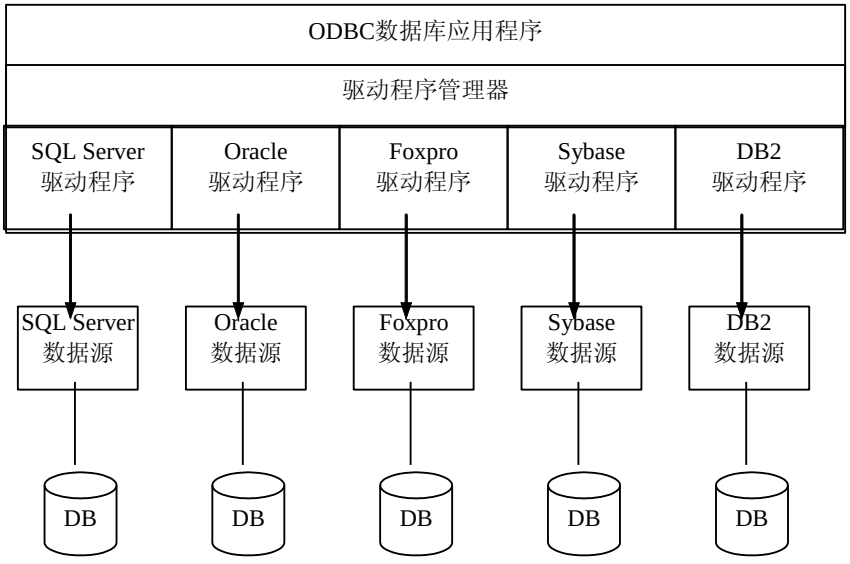
(3)驱动程序

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

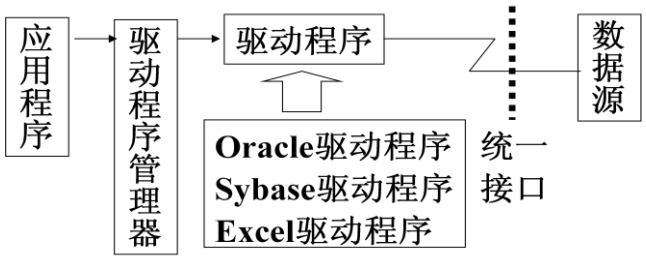
提供的数据可以是 RDBMS，也可以是 OODBMS 或各类

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。



- ODBC 数据库独立性

1 ODBC 是为最大的互用性而设计的，要求一个应用程序有用相同的源代码（不用重新编译或重新链接）访问不同的数据库管理系统(DBMS)的能力。
2 ODBC 定义了一个标准的调用层接口（CLI）。包含 X/Open 和 ISO/IEC 的 CLI 规范中的所有函数，并提供应用程序普遍需要的附加函数。
3 每个支持 ODBC 的 DBMS 需要不同的库或驱动程序，驱动程序实现 ODBC API 中的函数。当需要改变驱动程序时，应用程序不需要重新编译或者重新链接，只是动态加载新的驱动程序，并调用其中的函数即可。如果要同时访问多个 DBMS 系统，应用程序可加载多个驱动程序。
4 如何支持驱动程序取决于操作系统，例如，在 Windows 操作系统上，驱动程序是动态链接库（DLL）。

JDBC

- 定义：

基于 X / Open 的 SQL 调用级接口(CLI)
JDBC API 在其他通用 SQL 级 API(包括 ODBC)之上实现
JDBC 扩展了 Java 的能力，如使用 Java 和 JDBCAPI 就可以公布一个 Web 页，页面中带有能访问远端数据库的 Applet

- 四种 JDBC 驱动

1 基于 JDBC-ODBC 桥的 java 驱动
2 基于数据源本地驱动程序的虚拟 java 驱动
3 基于网络驱动协议的 java 驱动
4 纯 java 驱动

- JDBC VS. ODBC

5 ODBC 并不适合在 Java 中直接使用
6 完全精确地实现从 C 代码 ODBC 到 JavaAPI 写的 ODBC 的翻译也并不令人满意
7 ODBC 并不容易学习，它将简单特性和复杂特性混杂在一起，甚至对非常简单的查询都有复杂的选项；而 JDBC 刚好相反，它保持了简单事物的简单性，但又允许复杂的特性
8 JDBC 这样的 JavaAPI 对于纯 Java 方案来说是必须的

6. 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

- 定义：数据仓库是在企业管理和决策中，面向主体的、集成的、与时间相关的、不可修改的数据集合。
- 与数据库的区别

- OLTP (On-Line Transaction Processing)

Major task of traditional relational DBMS

Day-to-day operations: purchasing, inventory, banking, manufacturing, payroll, registration, accounting, etc.

- OLAP (On-Line Analytical Processing)

Major task of data warehouse system

Data analysis and decision making

Distinct features (OLTP vs. OLAP):

User and system orientation: customer vs. market

Data contents: current, detailed vs. historical, consolidated

Database design: ER + application vs. star + subject

View: current, local vs. evolutionary, integrated

Access patterns: update vs. read-only but complex queries

	OLTP	OLAP
users	clerk, IT professional	knowledge worker
function	day to day operations	decision support
DB design	application-oriented	subject-oriented
data	current, up-to-date detailed, flat relational isolated	historical, summarized, multidimensional integrated, consolidated
usage	repetitive	ad-hoc
access	read/write index/hash on prim. key	lots of scans
unit of work	short, simple transaction	complex query
# records accessed	tens	millions
#users	thousands	hundreds
DB size	100MB-GB	100GB-TB
metric	transaction throughput	query throughput, response

用到了哪些数据集成技术

1 ETL 数据清洗 抽取 集成

2 一些维的操作（上卷 下钻）

7. XML Schema 与 XML DTD 的区别

XML Schema 实际上也是 XML 的一种应用，就是将 XML DTD 重新按照 XML 语言规范来定义，这充分体现了 XML 自描述性的特点

- Schema 相比较 DTD 的优势

1 一致性。DTD 的结构和 XML 文件的结构很不相同，而 Schema 使得对 XML 的定义不必再利用一种特定的形式化的语言

2 扩展性。引入了数据类型、命名空间。

3 互换性。利用 Schema，我们能够书写 XML 文档，验证文档的合法性。通过映射机制，还可以将不同的 Schema 进行转换，以实现更高层次得数据交换。

4 规范性。同 DTD 一样，Schema 定义了 XML 文档的整体结构，如哪些元素可以出现在文档中，元素间的关系是什么，每个元素又有哪些子元素、属性，以及元素出现的顺序和次数等等。

5 易用性。XML 文档的结构已变成 Schema——一种“格式良好”的 XML 文档，用 DOM 和 SAX 去访问当然不在话下。

DTD 是已经淘汰的技术，因为 DTD 不具有规范性并且没有数据类型。

8. 什么是数据库中的元数据？他有什么作用？

- 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

9. 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。通过它可以在广域网或因特网上准确地识别、定位和访问空间信息。

(2)帮助数据使用者查询所需空间信息。比如，它可以按照不同的地理区间、指定的语言以及具体的时间段来查找空间信息资源。

(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。通过数据目录和数据交换中心等提供的空间元数据内容，用户可以共享空间信息、维护数据结果，以及对它们进行优化等。

(5)提供数据转换方面的信息。使用户在获取空间信息的同时便可以得到空间元数据信息。通过空间元数据，人们可以接受并理解空间信息，与自己的空间信息集成在一起，进行不同方面的科学分析和决策。描述空间信息的元数据标准体系内容按照部分、复合元素和数据元素来组织，它们是依次包含关系，前者包含后者，即：后者依次组成前者。具体分为 8 个基本内容部分和 4 个引用部分，由 12 个部分组成，其中标准化内容包括标识信息、数据质量信息、数据集继承信息、空间数据表示信息、空间参照系信息、实体和属性信息、发行信息以及空间元数据参考信息等内容，另外还有 4 个部分是标准化部分中必须引用的信息，它们为引用信息、时间范围信息、联系信息及地址信息。元数据标准内容体系是通过元数据网络管理系统来实现的，该系统主要由权限验证功能(服务器端验证)、输入和合法性校验功能(客户端校验)、查询功能(服务器端查询)与返回和显示功能(服务器端格式化查询结果并返回，客户端显示)等组成。利用空间元数据网络管理系统作为空间交换站的共享软件可基本上实现空间信息的网络共享。

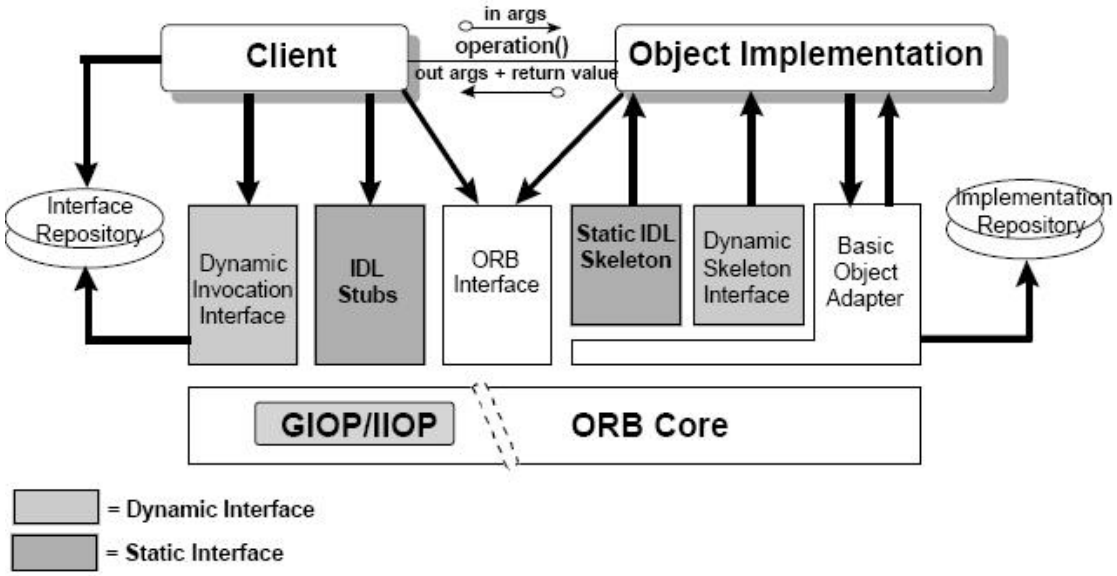
说到元数据的意义，可以从其应用目的来谈的。虽然做数据仓库言必称元数据，必称技术、业务元数据，但其到底用于何处？离开了目标去谈元数据，就发现元数据包含太多的东西，因为他是描述数据的数据嘛。

还是那客户关系系统来比喻，这个系统维护客户信息当然是有目的的，是要用这些信息进行一些自动的流程处理、去挖掘一些客户潜在的价值、做好客户服务。当然没有必要去维护客户的生命特征信息，诸如指纹、犯罪史等，这些信息跟客户关系管理的目标关系不大。元数据也是如此，你可以将所以数据的结构、大小、什么时间创建、什么时间消亡、被那些人使用等等，这些信息可以延伸得太广，如果不管目标，而试图去建一个非常完美的元数据管理体系，这是一种绝对的"自上而下"做法，必败无疑。

（以上内容来自百度）

10. CORBA2.0

CORBA2.0 体系结构



11. 是分析 web service 和 SOA 的关系

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

12. 试分析消息中间件相对于分布式对象技术的优势

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

13. 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

消息队列(MessageQueuing)：程序之间进行非直接通信的非连接模型

➢ 特点

1 多个生成方可向一个队列发送消息。生成方可共享连接或使用不同连接，但它们均可访问同一队列

2 多个接收者可使用一个队列中的消息，但每条消息只能由一个接收者使用。

3 接收者可共享连接或使用不同连接，但它们均可访问同一队列
4 发送者和接收者之间不存在时间上的相关性
5 可在运行时动态添加和删除发送者和接收者，即可根据需要扩展或收缩消息传送系统
6 消息在队列中的放置顺序与发送顺序相同，但它们的使用顺序则取决于消息失效期、消息优先级以及使用消息时是否使用选择器等因素

- 优势

1 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。
2 要发送到队列的消息始终保留，即使没有接收者也是如此。
3 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

- 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。

- 特点

1 多个生成方可向一个主题发布消息。生成方可共享连接或使用不同连接，但它们均可访问同一主题
2 多个订户可使用一个主题中的消息。订户可检索发布到一个主题中所有消息订户可共享连接或使用不同连接，但它们均可访问同一主题。
3 长期订户可能处于活动状态，也可能处于非活动状态。在它们处于非活动状态时，代理会为它们保留消息
4 可在运行时动态添加和删除发布者和订户，这样，即可根据需要扩展或收缩消息传送系统。消息发布到主题的顺序与发送顺序相同，但它们的使用顺序则取决于消息失效期、消息优先级以及使用消息时是否使用选择器等因素
5 发布者与订户之间存在时间上的相关性：主题订户只能使用在它创建订阅后发布的消息

- 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
总共有 4 个~~怎么样我都找不出第四个不同点~~OMG