**Practical Lecture 1**
**Introduction**

## Overview

- Building a distributed information system with heterogeneous clients
- The system will be implemented using Microsoft .NET technology
- Clients will be implemented using .NET and Java technology to demonstrate cross-platform interoperability in distributed systems
- The implementation is spread over 6 practical lectures

2

## Practical Session Structure

1. **Introduction**
2. Building a business component
3. Building an admin GUI
4. Introducing .NET remoting
5. Creating a web service and client website
6. Developing a Java client

3

## Pre-requisite Knowledge

- During the implementation of the system new concepts related to the development of distributed system will be explained
- The practical sessions require a good prior knowledge of the following:
  - Database principles and SQL
  - Database design using entity relationship modelling
  - Microsoft Visual Studio
  - C# programming
  - Java programming
  - System analysis and design using UML (Unified Modelling Language)

4

## Tools

- The following software tools will be used during the implementation of the system:
  - Microsoft SQL Server 2005
    - Including SQL Server Management Studio
  - Microsoft Visual Studio .NET 2005
  - NetBeans 5.0

5

## Learning Objectives

- Understand the case-study presented
- Be able to relate the design documentation (diagrams) to the case-study
- Create a database structure in SQL Server given an Entity Relationship Diagram

6

## Introduction

- In this practical session we will introduce the case-study:
  - The background
  - The requirements
  - What will be built (proposed system)
- We will explain the architecture and behaviour of the proposed system
- You will build the database required for the proposed system

7

## Case-Study

- During the course of the six practical lectures you will be incrementally implementing a system
- As each lecture builds on the work carried out in the previous lectures, it is important to do these in sequence

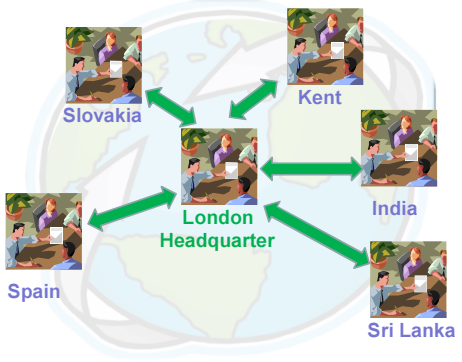
8

## Case-Study - Background

- Out Of Bounds Ltd is a London-based company which specialises in the management of software development projects
- Out Of Bounds Ltd has several teams of project managers and developers who are experts in a various fields of technology
- Some of Out Of Bounds Ltd's teams are based at their London premises, but others are at geographically distant locations (e.g. Sri Lanka, India, Spain, Slovakia, etc.)
- For certain projects, work is outsourced to other companies

9

## Out Of Bounds Ltd




10

## Case-Study – Current Situation

- Out Of Bounds Ltd is currently using Pert and Gannt charts to plan the project and set up tasks, responsibilities, timeframes, etc.
- Instructions are then given to the team leaders involved
- Communication/progress tracking is done through face-to-face meetings with local teams and through email/telephone/video conferencing with the international teams

11

## Case-Study - Requirements

- While the current provisions work, they are not very effective and it is often a difficult task for the project managers to know what the exact state of the project is
- To aid the project managers and team leaders in the project coordination task, Out Of Bounds Ltd is developing a software system which will facilitate the task
- The software system will be called – Project Tracking System (PTS)

12

## Case-Study – Requirements /2

- Project Tracking System:
  - This system will allow a manager to register projects
  - Projects are divide into tasks
  - Rules regarding the order of tasks can be set
  - Different tasks can be assigned to different teams (working for the company or outsourced to external teams)
  - The system will provide functionality for team leaders to register task progress. Project progress can be tracked by the project manager
  - The customer who commissioned the project can get summary information about the structure of the project, distribution of work and progress

13

## Users/Roles

- The system will be accessed by four different types of users:
  - Administrator/project manager
  - Team leader (internal)
  - Team leader (external)
  - Customer

14

## Users/Roles - Administrator

- The administrator is the project manager
- His/her responsibilities are:
  - setting up the project
  - dividing the project into tasks
  - assign tasks to teams
  - create new teams if necessary
  - update any changes to the project
  - track overall project progress

15

## Users/Roles – Team Leader

- A team leader is the person in charge of a particular team. A team could be:
  - Internal team – employed by Out Of Bounds Ltd
    - Local
    - International
  - External team – contracted by Out Of Bounds Ltd in order to carry out tasks
    - Could be contracted because Out Of Bounds lacks resources for a project or lacks the particular expertise
    - Could be based anywhere (locally or abroad)

16

## Users/Roles – Team Leader /2

- The responsibilities of a team leader include:
  - break tasks down into subtasks
  - estimate completion dates for new tasks
  - assign subtasks to team members
  - record progress on a particular task
  - record completed tasks
  - report expected delays, justify them and provide new estimated completion dates

17

## Users/Roles - Customer

- The customer is the individual or company who hired Out Of Bounds Ltd in order to carry out a project
- The customer is expected to liaise directly with the project manager and the use of the PTS will therefore be limited
- The customer uses the system in the following way:
  - view project structure and other project-related information
  - track overall progress of the project

18

## Requirements Overview

- One system – serving a number of geographically dispersed users
- Different types of users which require different functionality from the system
- Possibly different technologies/platforms used by different users

19

## Case-Study – Proposed System

- Need to provide different interfaces
  - We have different types of users
- Need for a **distributed** system
  - Users are in different locations using the same system
- Need for some kind of data layer
  - Information needs to be stored about projects, teams, etc.

20

## Proposed System /2

- Some knowledge / control over the platforms / applications used within Out Of Bounds Ltd
  - Allows choice of technology used for access within the company
- Little control over the platforms/applications used by externally contracted teams and customers
  - Need to provide platform independent access

21

## Proposed System /3

- All users have access to Internet
  - Enables the use of network-based technologies to achieve distribution

22

## The Solution

- Based on the above requirements we have decided to build a distributed system with a number of different user interfaces
- A database will be built using SQL Server 2005
- The main system will be built using the .NET Framework and C# programming language
- Most functionality of the system will be exposed as a web service

23

## The Solution /2

- The administrator interface can be tightly coupled to the system and will be built using .NET Remoting
- A client interface for internal teams will be built using .NET
- A client interface for external teams will be built using Java (to demonstrate the platform independence of the system)
- A website will be built for customers to access details about their project(s)

24

## Incomplete Solution

- The system we will be developing over the next six lectures is not a full-fledged commercial software solution – this would require much more time
- The solution will be built to demonstrate how the technologies work
- Many corners will be cut – validation, exception handling, performance, etc.

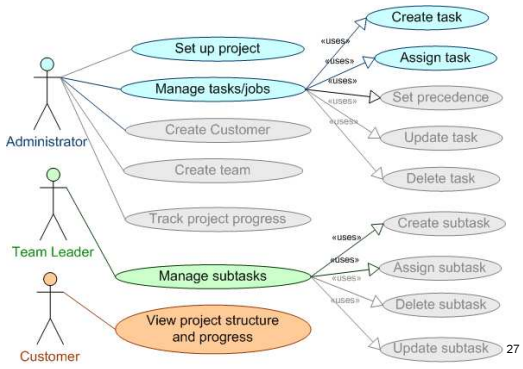
25

## Incomplete functionality

- Only a subset of the functionality identified in the system requirements will be implemented
  - In some cases this will mean that you will need to enter certain data straight on the database – this will be pointed out to you

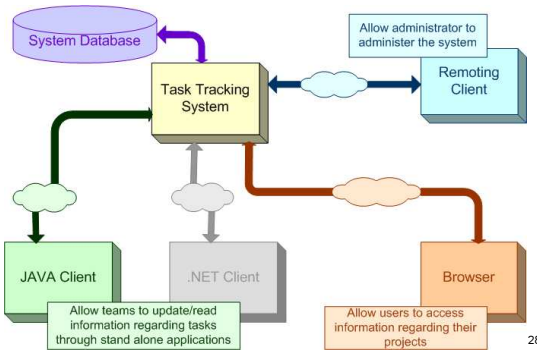
26

## Use-Case Diagram



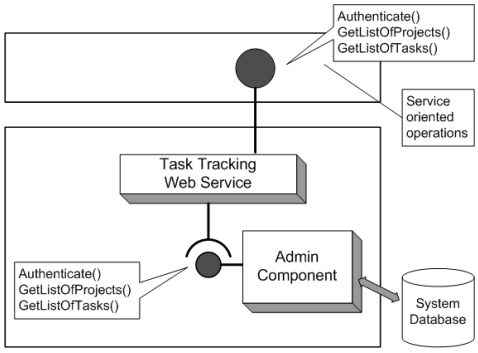

27

## The Architecture




28

## The Architecture /2




29

## The Database

- A database has been designed in order support the Project Tracking System
- These are the entities identified:
  - Project: this represents a project
  - Task: this represents a task of a particular project. A project may contain many tasks
  - Subtask: this represents a subtask of a particular task. A task may contain many subtasks

30

## The Database /2

- More entities:
  - Status: this is the status of a task or subtask
  - Predecessor: this is used to show precedence of tasks
  - Customer: this represents a client of Out of Bounds Ltd who commissioned a project
  - Team: this represents an internal or external team working for Out of Bounds Ltd

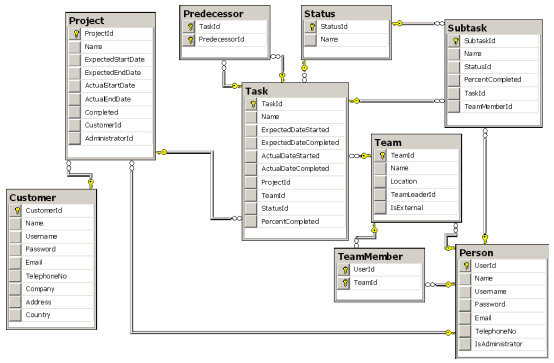
31

## The Database /3

- More entities:
  - Person: this represents administrators, team leaders and team members
  - TeamMember: this links a person to a team, showing that he/she is a team member

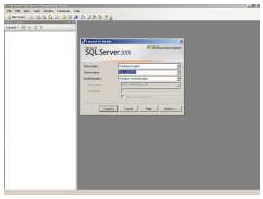
32

## The Database Design




## Create the database

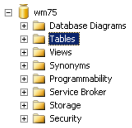
- You should now create the database and tables using Microsoft SQL Server
- In our case the database is called wm75
- You use Microsoft SQL Server Management Studio in order to administer your database



34

## Create the tables

- Once the database is created and you can access it, you should create the tables
- Expand the database and then right-click on the *Tables* option and select *New Table*
- Create the tables as shown on the following slides




35

## Project

- The primary key is ProjectId, which is of type uniqueidentifier (more on this later in the lecture)

Table - wm75.Project | Summary

| Column Name | Data Type | Allow Nulls |
|---|---|---|
| ProjectId | uniqueidentifier | ☐ |
| Name | nvarchar(50) | ☐ |
| ExpectedStartDate | datetime | ☑ |
| ExpectedEndDate | datetime | ☑ |
| ActualStartDate | datetime | ☑ |
| ActualEndDate | nchar(10) | ☑ |
| Completed | bit | ☑ |
| CustomerId | int | ☐ |
| AdministratorId | int | ☐ |
| | | ☐ |

36

## Task

- PercentCompleted reflects the percentage of completion for the task

Table - wm75.Task | Summary

| Column Name | Data Type | Allow Nulls |
|---|---|---|
| TaskId | uniqueidentifier | ☐ |
| Name | nvarchar(50) | ☐ |
| ExpectedDateStarted | datetime | ☐ |
| ExpectedDateCompleted | datetime | ☐ |
| ActualDateStarted | datetime | ☑ |
| ActualDateCompleted | datetime | ☑ |
| ProjectId | uniqueidentifier | ☐ |
| TeamId | int | ☐ |
| StatusId | int | ☐ |
| PercentCompleted | tinyint | ☑ |
| | | ☐ |

37

## Subtask

- The TeamMemberId is a foreign key from the Person table and denotes the person who has been assigned to the subtask

Table - wm75.Subtask | Summary

| Column Name | Data Type | Allow Nulls |
|---|---|---|
| SubtaskId | int | ☐ |
| Name | nvarchar(50) | ☑ |
| StatusId | int | ☐ |
| PercentCompleted | tinyint | ☑ |
| TaskId | uniqueidentifier | ☐ |
| TeamMemberId | int | ☑ |
| | | ☐ |

38

## Status

- The status entries will be the same for tasks and subtasks
- Note that both fields are set NOT to allow nulls

Table - wm75.Status | Summary

| Column Name | Data Type | Allow Nulls |
|---|---|---|
| StatusId | int | ☐ |
| Name | nvarchar(50) | ☐ |
| | | ☐ |

39

## Predecessor

- Note that we have a composite primary key. This is because the Predecessor table represents a unary relationship from the Task entity to itself

Table - wm75.Predecessor | Summary

| Column Name | Data Type | Allow Nulls |
|---|---|---|
| TaskId | uniqueidentifier | ☐ |
| PredecessorId | uniqueidentifier | ☐ |
| | | ☐ |

40

## Customer

- The password is just a string. In a real-life application, this might need to be encrypted

Table - wm75.Customer | Summary

| Column Name | Data Type | Allow Nulls |
|---|---|---|
| CustomerId | int | ☐ |
| Name | nvarchar(50) | ☑ |
| Username | nvarchar(50) | ☑ |
| Password | nvarchar(50) | ☑ |
| Email | nvarchar(50) | ☑ |
| TelephoneNo | nvarchar(50) | ☑ |
| Company | nvarchar(50) | ☑ |
| Address | nvarchar(200) | ☑ |
| Country | nvarchar(50) | ☑ |
| | | ☐ |

41

## Team

- IsExternal is a bit which denotes if a team is internal to Out of Bounds Ltd (false) or external (true)

Table - wm75.Team | Summary

| Column Name | Data Type | Allow Nulls |
|---|---|---|
| TeamId | int | ☐ |
| Name | nvarchar(50) | ☐ |
| Location | nvarchar(50) | ☐ |
| TeamLeaderId | int | ☑ |
| IsExternal | bit | ☑ |
| | | ☐ |

42

## Person

- IsAdministrator is a bit denoting whether the person is an administrator (true) or not (false)

Table - wm75.Person | Summary

| Column Name | Data Type | Allow Nulls |
|---|---|---|
| UserId | int | ☐ |
| Name | nvarchar(50) | ☐ |
| Username | nvarchar(50) | ☐ |
| Password | nvarchar(50) | ☐ |
| Email | nvarchar(50) | ☑ |
| TelephoneNo | nvarchar(50) | ☑ |
| IsAdministrator | bit | ☑ |
| | | ☐ |

43

## TeamMember

- Another composite primary key, because TeamMember is a link table which resolves the many-to-many relationship between the Team and Person entities

Table - wm75.TeamMember | Summary

| Column Name | Data Type | Allow Nulls |
|---|---|---|
| UserId | int | ☐ |
| TeamId | int | ☐ |
| | | ☐ |

44

## GUID vs. Identity

- SQLServer supports:
  - Identity
    - Automatically increments when a new record is entered
    - Ensures that a field is unique
  - GUID (Globally Unique ID)
    - The data type is called *uniqueidentifier*
    - Not automatically generated (needs to be generated and then inserted)
    - GUID should be globally unique
      - Global uniqueness is not guaranteed
      - Probability of being unique is very high

45

## GUID

- GUID is an implementation of UUID (Universally Unique Id)
- Appropriate for distributed systems as you can work with information from different sources without ids colliding
- GUID example:
  {3F2504E0-4F89-11D3-9A0C-0305E82C3301}

46

## GUID /2

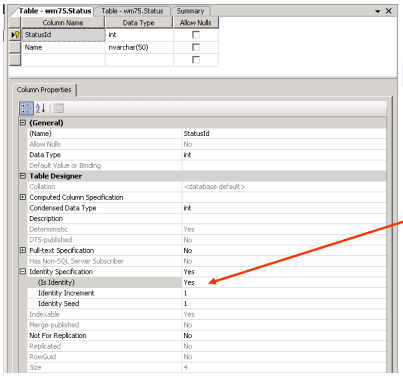
- In our system it isn't essential to use a GUID as we have only one centralised database
- We will use GUID for Projects and Tasks so potentially our system could interoperate with another system with its own projects and tasks

47

## Creating Identities

- Many of the primary keys in our tables should be identities (increment automatically)
- To turn the identity feature of a field on, you need to set the *Is Identity* to *Yes*
- The *Is Identity* property is listed under the Column Properties (see next slide)

## Creating Identities /2

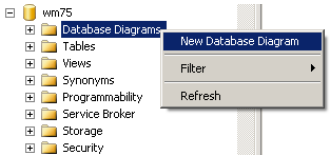


## Creating Identities /3

- You need to create identities for the primary keys of the following tables:
  - Customer (CustomerId)
  - Person (UserId)
  - Status (StatusId)
  - Subtask (SubtaskId)
  - Team (TeamId)

## Database Diagram

- Once all the tables are created, you should establish the links between them by creating a database diagram
- Right-click on *Database Diagrams* and then select *New Database Diagram*




51

## Database Diagram /2

- Add all the tables you created to the diagram and then link them by taking the primary key from one table and dropping it on the foreign key it should link to
- The tables should be linked as follows:
  - Customer (CustomerId) – Project (CustomerId)
  - Project (ProjectId) – Task (ProjectId)

52

## Database Diagram /3

- More details on linking the tables:
  - Task (TaskId) – Predecessor (TaskId)
  - Task (TaskId) – Predecessor (PredecessorId)
  - Task (TaskId) – Subtask (TaskId)
  - Status (StatusId) – Task (StatusId)
  - Status (StatusId) – Subtask (StatusId)
  - Team (TeamId) – Task (TeamId)
  - Team (TeamId) – TeamMember (TeamId)

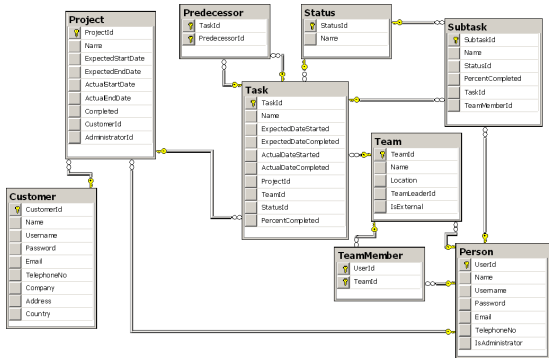
53

## Database Design /4

- More details on linking the tables:
  - Person (UserId) – Team (TeamLeaderId)
  - Person (UserId) – TeamMember (UserId)
  - Person (UserId) – Project (AdministratorId)
  - Person (UserId) – Subtask (TeamMemberId)
- When you finish, your diagram should look like the one shown on the next slide

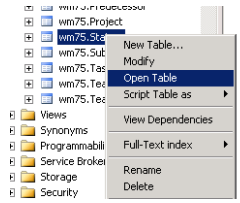
54

## The Database Design




## Populating a Table

- With the tables in place, lets put some data in
- To insert data straight on a table right-click on the table and select *Open Table*



## Populating a Table /2

- We will populate the Status table
  - StatusId is an identity and will be automatically added
  - Only enter the name for each status (as shown below)

Table - wm75.Status | Summary

| StatusId | Name |
|---|---|
| 1 | Ready to Start |
| 2 | In Progress |
| 3 | Completed |
| 4 | Waiting for predecessor |
| NULL | NULL |

## Summary

- By this point in time you should have an understanding of the case-study what it is you will build during these six practical lectures
- You should also have a database with all the required tables and established links between them

58