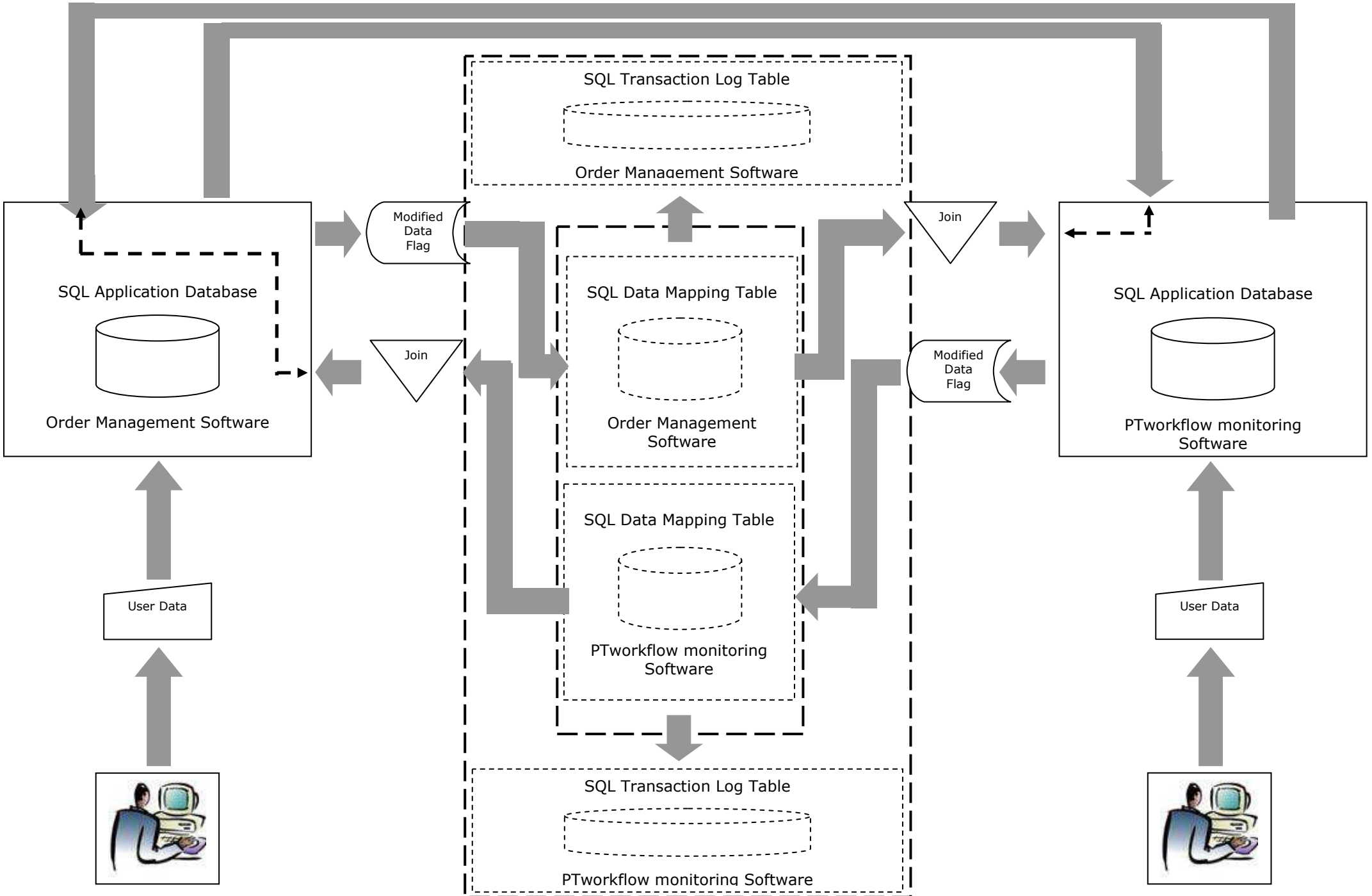
Beispiel SQL-DB-Interface PTworkflow



Funktionsprinzip:

Das Schaubild stellt eine beispielhafte SQL-basierte Datenkommunikation zwischen einer Auftragsverwaltungssoftware (z.B. Microsoft Navision, SBO (SAP Business One), etc.) und PTworkflow dar.

Der Bereich links zeigt die Applikationsdatenbank der Auftragsverwaltungssoftware, rechts befindet sich die PT Systemdatenbank.

Die mittig gestrichelt dargestellten Elemente bezeichnen die Bereiche der SQL-gesteuerten Schnittstellenfunktion.

Sowohl in der Auftragsverwaltung als auch bei PTworkflow werden Benutzereingaben wie gewohnt direkt in der zugehörigen Systemdatenbank abgelegt.
Eine zuvor festgelegte dynamische Liste von relevanten Auftragsdaten basierend auf editierbare SQL-Statements (Sharing-List) spiegelt das Mapping der „Data Mapping"-Tabellen wider, die als Schnittstellen-Pufferspeicher in der eigentlichen Datenkommunikation dienen.
Es sollte im Vorfeld definiert werden, ob beide Applikationen jeweils ihre eigene Mapping-Tabelle verwenden oder dies mit einem gemeinsamen Mapping realisiert wird.
In der Mapping-Tabelle werden beim Vorliegen von aktuellen Daten in der fremden Applikationsdatenbank entsprechende Primärschlüsselinformationen abgelegt, die auf die jeweilige Zeile der betreffenden Datentabelle(n) der Systemdatenbank oder einer Sharing-Tabelle zeigen. Zusätzlich sollte die Mapping-Tabelle ein Flag besitzen, welches die auszuführende Operation spezifiziert, sowie das zur Synchronisation der Daten erforderliche SQL-Statement.
Durch Änderung der Abfragekommandos in der Mapping-Tabelle lässt sich so auch nachträglich steuern, welche Daten mit Hilfe der Schnittstelle zwischen beiden Applikationen synchronisiert werden.

Werden in der Auftragsverwaltungssoftware projektrelevante Daten erzeugt oder bei bestehenden Aufträgen geändert, so werden die Primärschlüsselinformationen der betroffenen Datenzeilen in der Mapping-Tabelle abgelegt. Die Marker-Spalte (Flag) in der Mapping-Tabelle spezifiziert die Operation einer solchen Aktionszeile.
Vor der Datenrückgabe der durch Benutzeranfragen erzeugten SELECT's auf die PT-Systemdatenbank wird die Mapping-Tabelle bzw. die Sharing-Tabelle der benachbarten Applikation gejoint. Anhand der Markerspalte und der SQL-Statement-Spalte werden nun die gefundenen Datenupdates durch Abfragen der Sytemdatenbanktabellen synchronisiert.
Nach Abgleich der Systemdatenbanken wir das modifizierte Anfrageergebnis dem Benutzer zurückgegeben.
Eine erfolgreiche Aktualisierung kann über die Markerspalte in der Mapping-Tabelle oder in einer separaten Log-Tabelle vermerkt werden.

Durch die Kombination einer ereignis- und zeitgesteuerten Synchronisation von Daten ermöglicht das vorliegende Modell eine nahezu Echtzeitübermittlung von Aktualisierungen beim gleichzeitigen Ausnutzen des Vorteils der Trennung von System- und Schnittstellentabellen beider Applikationen. Es kann somit sehr individuell gesteuert werden, auf welche Daten die Nachbarapplikation Zugriff erhalten soll und in welcher Form Daten bereitgestellt werden müssen.
Zusätzlich bleibt die Architektur der Systemdatenbanken unverändert, was den programmtechnischen Anpassungsaufwand der Applikationen reduziert.
Fehlerpotentiale durch inkonsistente Werte in den Systemdatenbanken durch fehlerhafte Schreib- und/ oder Leseaktionen werden minimiert, da keine externe Applikation direkten Schreibzugriff auf Systemtabellen oder -daten erhält. Das ermöglicht zusätzliche Prüfroutinen der

Applikationen, ob zu modifizierende Daten den erwarteten Formaten entsprechen und die Möglichkeit, fehlerhafte Daten der Nachbarapplikation abzuweisen, bevor diese in die Systemdatenbank gelangen und dort unter Umständen zu fehlerhaften oder gar instabilem Verhalten der Anwendersoftware führen.

Performance:

Ohne Frage bindet die Integration und Abfrage einer Mapping- und/ oder Sharing-Tabelle und der nachfolgenden Synchronisation der Systemdatenbanken Ressourcen und bedeutet - je nach Umfang der zu aktualisierenden Daten - einen mehr oder weniger großen Overhead in beiden Applikationen.
Je nach der zu erwartenden Menge der Datenaktualisierungen können unterschiedliche Integrationsmodelle zur Performanceoptimierung zur Anwendung kommen.

Bei geringerem Datenaufkommen kann die Mapping-Tabelle zusätzlich als Transaktions-Log fungieren.
Dabei könnte das Anlegen und Modifizieren von Datenzeilen durch entsprechende Zeitstempel Spalten in der Mapping-Tabelle protokolliert werden. Sämtliche Transaktionen können somit direkt aus der Mapping-Tabelle nachvollzogen werden.

Basierend auf Basis Zeit und/ oder Tabellenumfang können dann von jeder Applikation automatische Routinen aufgerufen werden, die die Mapping-Tabelle bzw. die Sharing-Tabelle säubert, alte Daten entweder unwiderruflich löscht oder diese zuvor in Backuptabellen auslagert. Mit einem entsprechenden Rotationsprinzip lassen sich so sehr individuelle Lösungen erzeugen, mit dem Ziel, die aktive Mapping-Tabelle bzw. Sharing-Tabelle auf einer möglichst geringen Größe zu halten, um die Systemperformance in keiner inakzeptablen Weise zu beeinträchtigen.

Ist das zu erwartende Datenaufkommen der Mapping- und/ oder SharingTabelle größer, kann der Einsatz einer zusätzliche „Transaction Log"-Tabelle sinnvoll sein. Dabei werden sämtliche Aktionen der Mapping-Tabelle sowie die Synchronisation der Daten zwischen den Systemdatenbanken in einer separaten Logtabelle abgelegt. Die Informationen der Mapping-Tabelle bzw. Sharing-Tabelle könnte hierbei nach jeder erfolgreichen Synchronisation gelöscht werden. Damit enthält die Mapping-Tabelle bzw. Sharing-Tabelle zu jeder Zeit ausschließlich nur unverarbeitete Daten und ihre Größe reduziert sich damit auf das absolut notwendige Minimum, was zu einer erhöhten Abfrage- und Aktualisierungsgeschwindigkeit führt.

Wie schon erwähnt, können je nach Anforderung und Einsatzgebiet die separaten Log-Tabellen entfallen und/ oder die Mapping-Tabellen für jede Applikation in einer gemeinsamen Tabelle zusammengefasst werden.