# SFDC 1008

**Chart**

Claim Chart of Kathleen A. Peters, "The Design of a Change Notification Server for Clients of a Passive Object-Oriented Database Management System" (July 1992) ("Peters")

as prior art to

Asserted Claims of U.S. Patent No. 7,356,482 ("the '482 Patent")

| **'482 Patent** | PETERS THESIS |
|---|---|
| **Claim 1** | |
| A system for providing a dynamically generated application having one or more functions and one or more user interface elements; comprising: | To the extent that this preamble is construed to be limiting, Peters discloses a system for providing a dynamically generated application having one or more functions and one or more user interface element. *See, e.g.*:<br><br>Peters at Abstract:<br><br>"The major contributions of this thesis are its focus on object-oriented (as opposed to relational) data change, the design of a change notification server, and the description of a language used by clients to specify conditions of interest to monitor for change. Further, we present a model of version management, and describe how our notification server can complement an application where clients are versioning data."<br><br>Peters at § 4.2 ("Railway Network Application"):<br><br>"This application is the less-than-real-time operation of a simple railway monitoring and control system. Given a database containing a description of the railway network (e.g., *I* stations, track segments connecting stations), train schedules (e.g., routes through the network), and active train information (e.g., location, speed), various DB client processes monitor and offer control options for the movement of trains in the network."<br><br>Peters at § 4.3 ("Document Co-Authoring Application"):<br><br>"When a user is updating the text of an existing paragraph (generally, a long-term update), the DB object cannot be write-locked as this would delay concurrent reads. To allow concurrent reads and inform other users that the text is changing, the client process wanting to change a paragraph first performs a short-term update to set the specific paragraph's 'text-under-mod' variable to true. The notification server is informed by the client who |

| ‘482 Patent | PETERS THESIS |
|---|---|
| | performed the change that a committed update to ‘text-undermod’ took place. This information is passed on to interested clients (those who have issued the condition specification [I] above). When the updated text is committed some time later, interested clients are informed (again via [I] above). If the ‘text-under-mod’ variable is now false, then another client may begin a long-term text update; otherwise, the first client still has exclusive update access, but has chosen to let others see an intermediate result. , **A** non-conflicting concurrent update is allowed when a client is updating the text of a paragraph and another client wishes to change the ‘next-p’ variable of that paragraph (inserting a new paragraph, or rearranging paragraph order). The change to ‘next-p’ is a short-term update. The update- counter variable is used by the client process doing the long-term update to determine that a change has occurred to ‘next-p’ if notification does not reach it before the text update is committed.”<br><br>To the extent this reference does not teach this claim element, this reference in combination with the knowledge of one of ordinary skill in the art and the disclosures of each of the prior art references provided in Salesforce’s Petitions related to the ‘482 and ‘111 Patents renders this claim element obvious. |
| [a] a server computer; | Peters discloses a server computer. *See, e.g.*:<br><br>Peters at Abstract:<br><br>“This thesis presents a comprehensive design specification for an object-oriented database management system (OODBMS) change notification server. Objectstore is used as the example underlying OODBMS.<br><br>The major contributions of this thesis are its focus on object-oriented (as opposed to relational) data change, the design of a change notification server, and the description of a language used by clients to specify conditions of interest to monitor for change. Further, we present a model of version management, and describe how our notification server can complement an application where clients are versioning data.”<br><br>Peters at § 5.2:<br><br>“All message passing between DB client processes and the notification server is done asynchronously.” |

| ‘482 Patent | PETERS THESIS |
|---|---|
| | Peters at § 8.3 (including Figure 8-9):<br><br>“To review, clients send one or more condition specifications to the notification server to indicate what DB changes they wish to be notified of, and they send a DB update event message to the notification server for each change made to application data after the change has been committed to the OODBMS. The notification server sends an acknowledge message to clients for each condition specification it receives, and the server sends a notification message to clients for each change they have indicated an interest in.”<br><br>To the extent this reference does not teach this claim element, this reference in combination with the knowledge of one of ordinary skill in the art and the disclosures of each of the prior art references provided in Salesforce’s Petitions related to the ‘482 and ‘111 Patents renders this claim element obvious. |
| [b] one or more client computers connected to the server computer over a computer network; | Peters discloses one or more client computers connected to the server computer over a computer network. *See, e.g.*:<br><br>Peters at Abstract:<br><br>“One way to aid database clients in detecting change to shared data is to provide a notification mechanism which informs clients when the data they have an interest in has changed. Previous research has focused primarily on embedding change notification schemes in a database management system (DBMS).<br><br>. . .<br><br>The major contributions of this thesis are its focus on object-oriented (as opposed to relational) data change, the design of a change notification server, and the description of a language used by clients to specify conditions of interest to monitor for change. Further, we present a model of version management, and describe how our notification server can complement an application where clients are versioning data.”<br><br>Peters at § 4.2 (“Railway Network Application”):<br><br>“The client processes for this application are graphic user interfaces (GUIs) for train operators and station operators. There is one train operator GUI process for each active train. Each train operator GUI provides the following functions: |

| ‘482 Patent | PETERS THESIS |
|---|---|
| | - display up-to-date train information (speed limit, current speed, current location, schedule)<br>- ask for, and receive, permission to enter or leave a station<br>- update current speed and/or location of the train. There are three station operator GUI processes: a master process which monitors the entire railway network and controls train schedules, and two region processes (west and east) which monitor subsections of the network and control the tracks and trains within their field of view. A station GUI provides some or all of the following functions (depending on whether the GUI is a master or a regional process):<br>- display up-to-date network information (network diagram; train speed, location and direction of travel for each train)<br>- receive requests, and grant permission, for trains to enter or leave a station<br>- update train schedule<br>- update train speed limit<br>- update track segment status (open, closed)”<br><br>Peters at § 4.3 (“Document Co-Authoring Application”):<br><br>“When a user is updating the text of an existing paragraph (generally, a long-term update), the DB object cannot be write-locked as this would delay concurrent reads. To allow concurrent reads and inform other users that the text is changing, the client process wanting to change a paragraph first performs a short-term update to set the specific paragraph’s ‘text-under-mod’ variable to true. The notification server is informed by the client who performed the change that a committed update to ‘text-undermod’ took place. This information is passed on to interested clients (those who have issued the condition specification [I] above). When the updated text is committed some time later, interested clients are informed (again via [I] above). If the ‘text-under-mod’ variable is now false, then another client may begin a long-term text update; otherwise, the first client still has exclusive update access, but has chosen to let others see an intermediate result. , **A** non-conflicting concurrent update is allowed when a client is updating the text of a paragraph and another client wishes to change the ‘next-p’ variable of that paragraph (inserting a new paragraph, or rearranging paragraph order). The change to ‘next-p’ is a short-term update. The update- counter variable is used by the client process doing the long-term update to determine that a change has occurred to ‘next-p’ if notification does not reach it before the text update is committed.” |

DOCKET ALARM

# Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

## Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

## Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

## Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

## API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

**LAW FIRMS**
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

**FINANCIAL INSTITUTIONS**
Litigation and bankruptcy checks for companies and debtors.

**E-DISCOVERY AND LEGAL VENDORS**
Sync your system to PACER to automate legal marketing.