Oracle Database Change Notification

Уведомление о изменении данных в СУБД, без триггеров, на подобие WebSocket

Пример потери согласованности

- **Пользователь А: подключается к БД и запрашивает данные из таблицы X**
- **Пользователь В: подключается к БД и модифицирует эти данные в таблице X**

Пользователь А видит устаревшие данные, и на основе их производит расчеты !

Варианты решения

- **Периодично опрашивать данные**
  - Потеря информации об изменениях которые произошли между опросами,
  - увеличение сетевого трафика
- **Расчеты производить на сервере в пределах одной транзакции (Пессимистичная блокировка)**
  - Высокая нагрузка на сервер
- **Расчеты производить на клиенте с использованием версионности исходных данных (Оптимистичная блокировка)**
  - Затраты на хранения (колонка версионности), дополнительная сложность

Oracle Database Change Notification

- **СУБД сама уведомит об изменениях**
  - СУБД - единственный источник “истины”
- **Как только произошло изменение СУБД сама уведомить всех заинтересованных клиентов**
  - Обновлять кэш на клиенте когда это действительно необходимо
  - Клиенты могут использовать всегда актуальный кэш
- **Сетевой трафик**
  - Трафик генерируется только в момент когда происходит изменение
  - Посылается не сами измененные данные, а только информация об изменении

Oracle Database Change Notification

Когда происходит изменение данных СУБД отсылает клиентам уведомление при операциях:

- DML (insert/update/delete)
- DDL (alter table add ..)
- Старт/Останов (startup/shutdown)

Определение объекта слежение по зависимостям (Например: через select)

Для DML-операций - нотификация только после фиксации транзакции !

Минимальные затраты на поддержку: работает на основе анализа оперативных журналов (не требуется режим archivelog)

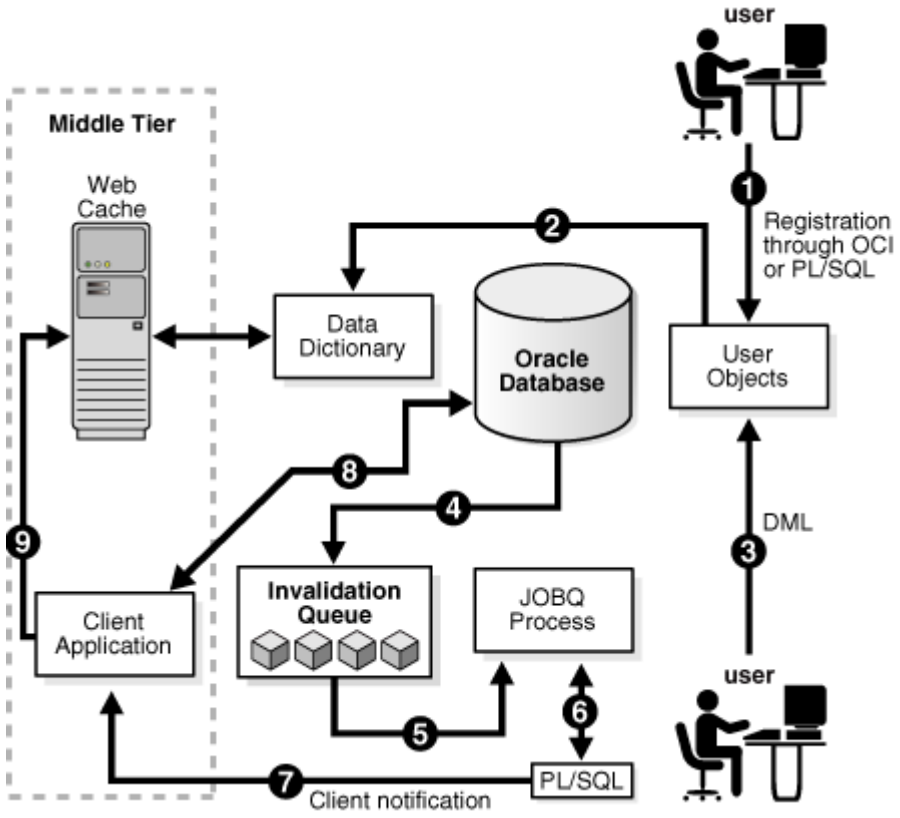
Информация передаваемая клиенту

- **Имя и схема объекта который был изменен**
- **Тип события, которое изменило данные**
  - INSERT, UPDATE, DELETE, ALTER TABLE, or DROP TABLE
  - Глобальное событие, такое как STARTUP или SHUTDOWN
    - В RAC - когда первый экземпляр стартовал или последний был остановлен
- **Для DML-операций: RowId строк, которые были модифицированы (не сами данные!)**

Типы запросов для слежения

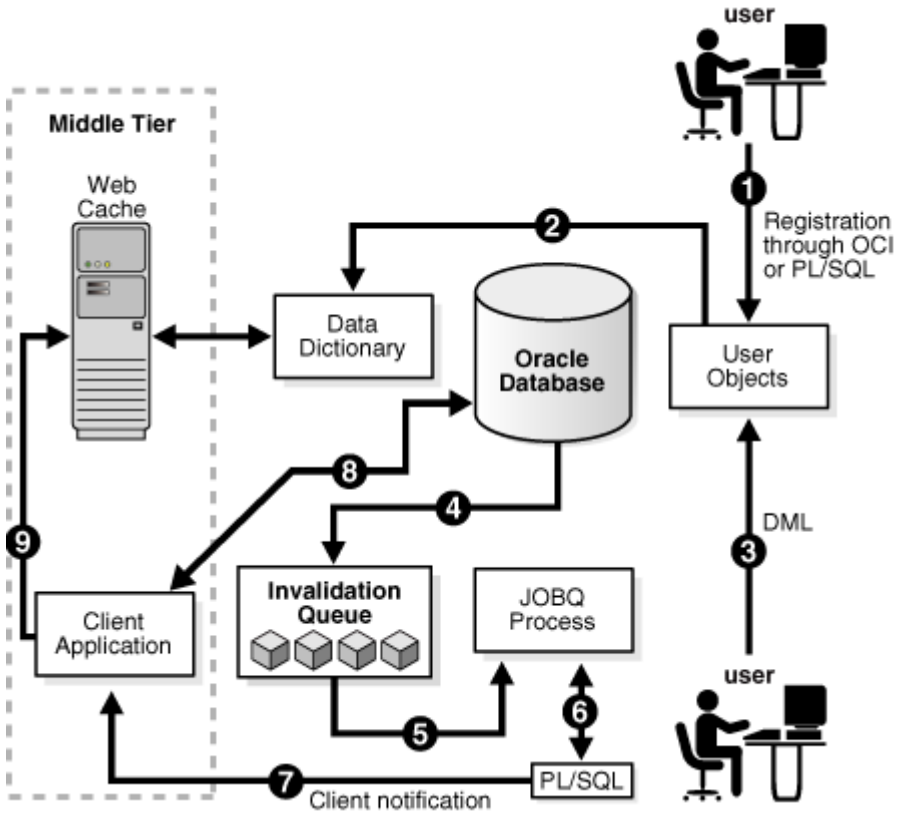
- **Все запросы:**
  - К представлениям (views)
  - Соединения (join)
  - Запросы к табличным функциям
  - REF Cursors
- **Исключая запросы использующие:**
  - Системные таблицы и представления (напр: X$, V$)
  - Запросы включающие обращение к db-link
  - Материализованные представления
- **Нотификация, если изменился любой объект от которого зависит запрос**

Общий процесс 1/9



1.В этом примере давайте предположим, что приложение кэшировало набор результатов запроса в HR.EMPLOYEES. Разработчик создает регистрацию для запроса на HR.EMPLOYEES, используя интерфейс уведомлений об изменениях PL / SQL. Кроме того, он создает хранимую процедуру PL / SQL для обработки уведомлений и предоставляет серверную процедуру PL / SQL в качестве обработчика уведомлений.

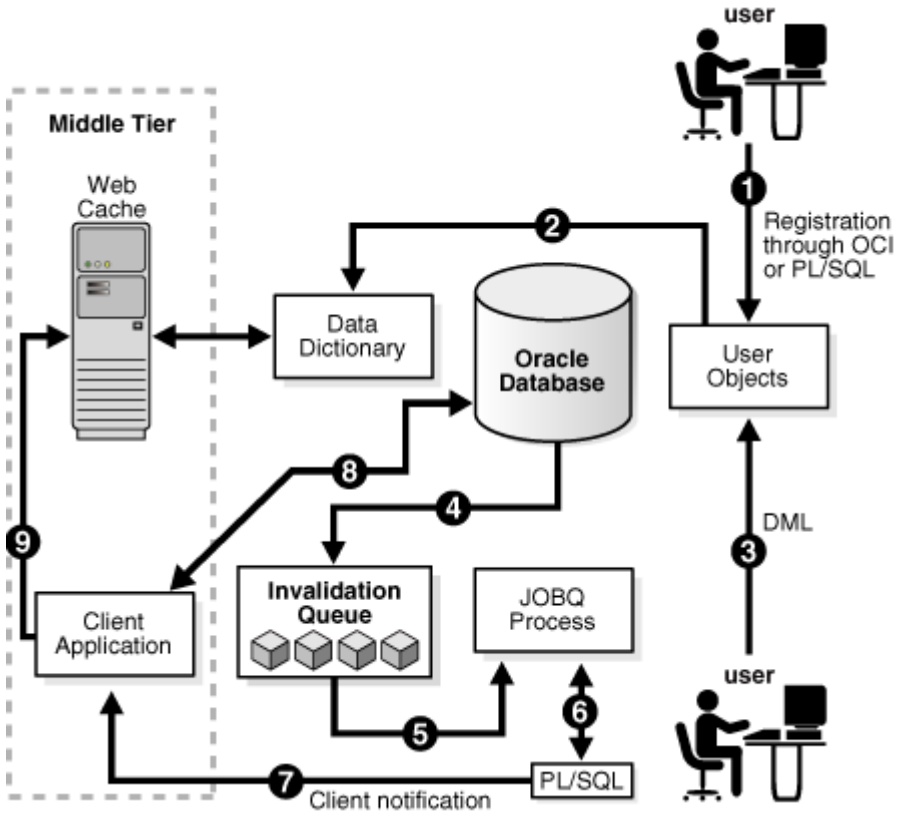
Общий процесс 2/9



1. В этом примере давайте предположим, что приложение кэшировало набор результатов запроса в HR.EMPLOYEES. Разработчик создает регистрацию для запроса на HR.EMPLOYEES, используя интерфейс уведомлений об изменениях PL / SQL. Кроме того, он создает хранимую процедуру PL / SQL для обработки уведомлений и предоставляет серверную процедуру PL / SQL в качестве обработчика уведомлений.

2.База данных заполняет регистрационную информацию в словаре данных.

Общий процесс 3/9

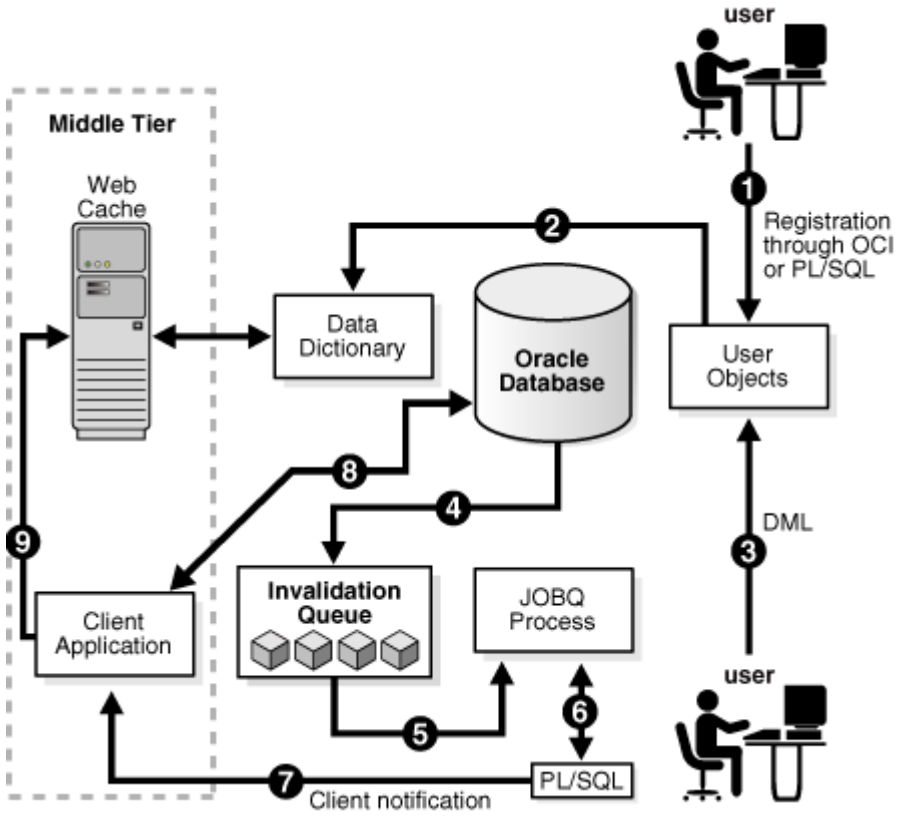


1. В этом примере давайте предположим, что приложение кэшировало набор результатов запроса в HR.EMPLOYEES. Разработчик создает регистрацию для запроса на HR.EMPLOYEES, используя интерфейс уведомлений об изменениях PL / SQL. Кроме того, он создает хранимую процедуру PL / SQL для обработки уведомлений и предоставляет серверную процедуру PL / SQL в качестве обработчика уведомлений.

2. База данных заполняет регистрационную информацию в словаре данных.

3. Пользователь изменяет один из зарегистрированных объектов с помощью операторов DML и фиксирует транзакцию. Например, пользователь обновляет строку в таблице hr.employees во внутренней базе данных. Данные для hr.employees, кэшированные на среднем уровне, теперь устарели.

Общий процесс 4/9



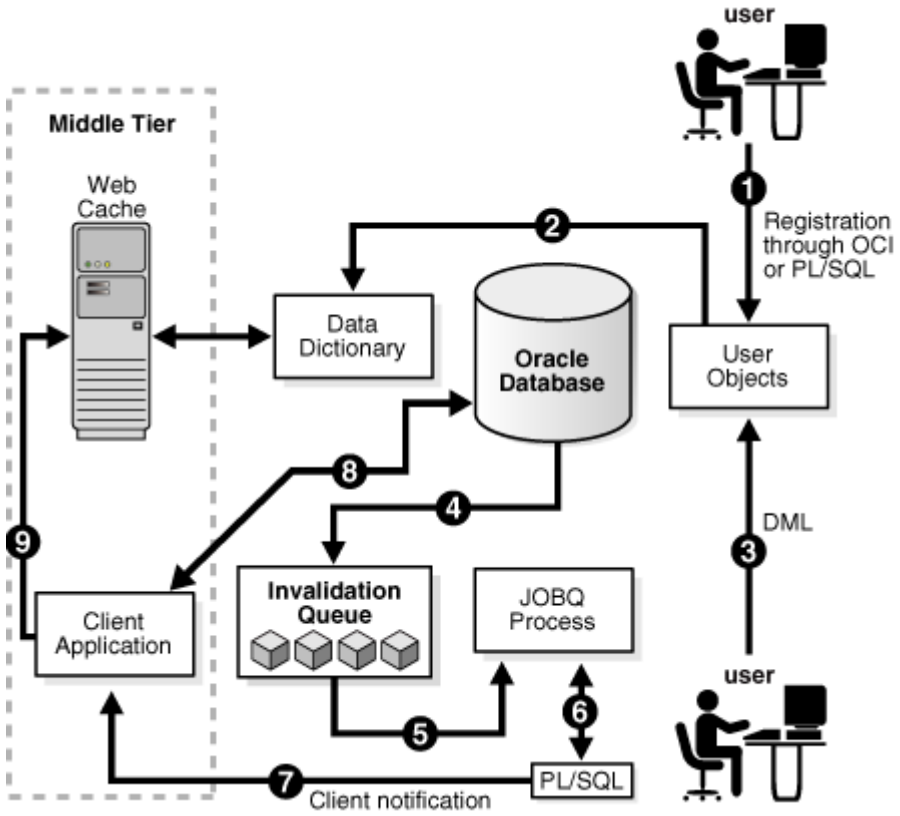
1. В этом примере давайте предположим, что приложение кэшировало набор результатов запроса в HR.EMPLOYEES. Разработчик создает регистрацию для запроса на HR.EMPLOYEES, используя интерфейс уведомлений об изменениях PL / SQL. Кроме того, он создает хранимую процедуру PL / SQL для обработки уведомлений и предоставляет серверную процедуру PL / SQL в качестве обработчика уведомлений.

2.База данных заполняет регистрационную информацию в словаре данных.

3.Пользователь изменяет один из зарегистрированных объектов с помощью операторов DML и фиксирует транзакцию. Например, пользователь обновляет строку в таблице hr.employees во внутренней базе данных. Данные для hr.employees, кэшированные на среднем уровне, теперь устарели.

4.Oracle Database добавляет сообщение, описывающее изменение во внутренней очереди.

Общий процесс 5/9



1. В этом примере давайте предположим, что приложение кэшировало набор результатов запроса в HR.EMPLOYEES. Разработчик создает регистрацию для запроса на HR.EMPLOYEES, используя интерфейс уведомлений об изменениях PL / SQL. Кроме того, он создает хранимую процедуру PL / SQL для обработки уведомлений и предоставляет серверную процедуру PL / SQL в качестве обработчика уведомлений.

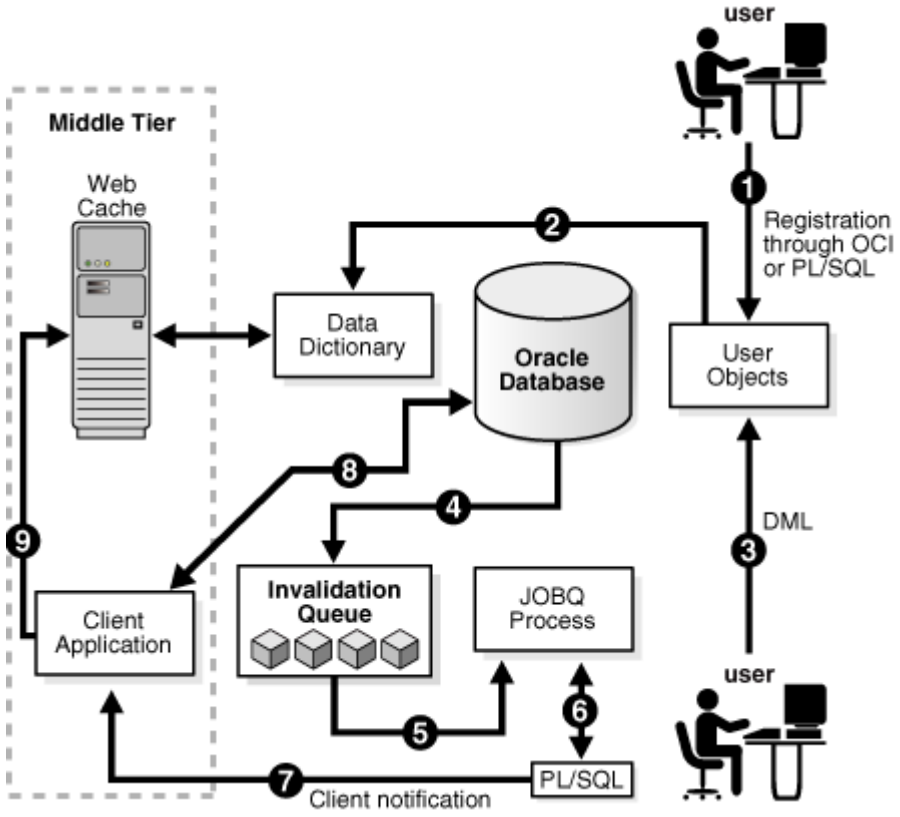
2.База данных заполняет регистрационную информацию в словаре данных.

3.Пользователь изменяет один из зарегистрированных объектов с помощью операторов DML и фиксирует транзакцию. Например, пользователь обновляет строку в таблице hr.employees во внутренней базе данных. Данные для hr.employees, кэшированные на среднем уровне, теперь устарели.

4.Oracle Database добавляет сообщение, описывающее изменение во внутренней очереди.

5.Фоновый процесс JOBQ уведомляется о новом уведомлении об изменении.

Общий процесс 6/9



1. В этом примере давайте предположим, что приложение кэшировало набор результатов запроса в HR.EMPLOYEES. Разработчик создает регистрацию для запроса на HR.EMPLOYEES, используя интерфейс уведомлений об изменениях PL / SQL. Кроме того, он создает хранимую процедуру PL / SQL для обработки уведомлений и предоставляет серверную процедуру PL / SQL в качестве обработчика уведомлений.

2. База данных заполняет регистрационную информацию в словаре данных.

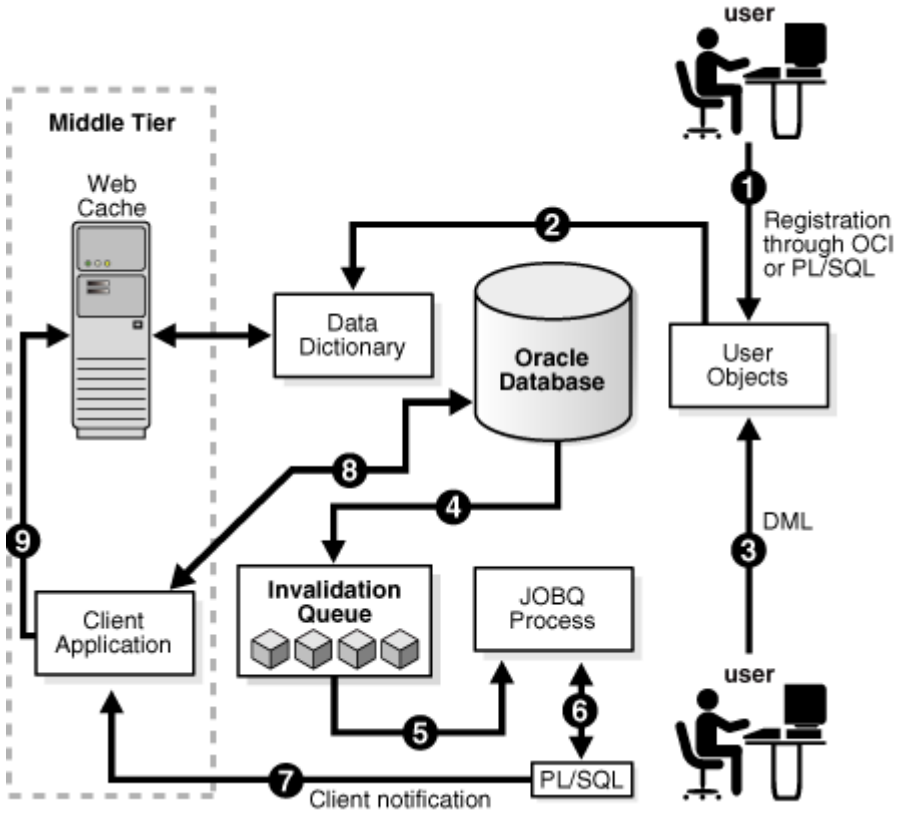
3. Пользователь изменяет один из зарегистрированных объектов с помощью операторов DML и фиксирует транзакцию. Например, пользователь обновляет строку в таблице hr.employees во внутренней базе данных. Данные для hr.employees, кэшированные на среднем уровне, теперь устарели.

4. Oracle Database добавляет сообщение, описывающее изменение во внутренней очереди.

5. Фоновый процесс JOBQ уведомляется о новом уведомлении об изменении.

6. Процесс JOBQ выполняет хранимую процедуру, указанную клиентским приложением. В этом примере JOBQ передает данные в процедуру PL / SQL на стороне сервера. Реализация процедуры обратного вызова PL / SQL определяет, как обрабатывается уведомление.

Общий процесс 7/9



1.В этом примере давайте предположим, что приложение кэшировало набор результатов запроса в HR.EMPLOYEES. Разработчик создает регистрацию для запроса на HR.EMPLOYEES, используя интерфейс уведомлений об изменениях PL / SQL. Кроме того, он создает хранимую процедуру PL / SQL для обработки уведомлений и предоставляет серверную процедуру PL / SQL в качестве обработчика уведомлений.

2База данных заполняет регистрационную информацию в словаре данных.

3Пользователь изменяет один из зарегистрированных объектов с помощью операторов DML и фиксирует транзакцию. Например, пользователь обновляет строку в таблице hr.employees во внутренней базе данных. Данные для hr.employees, кэшированные на среднем уровне, теперь устарели.

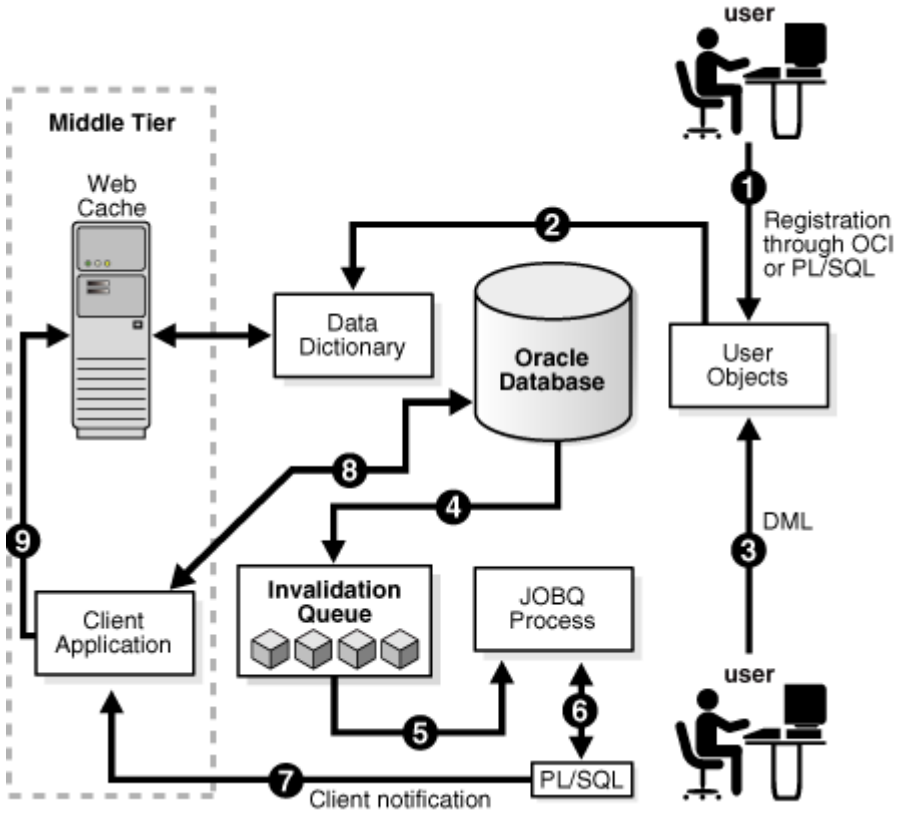
4Oracle Database добавляет сообщение, описывающее изменение во внутренней очереди.

5Фоновый процесс JOBQ уведомляется о новом уведомлении об изменении.

6Процесс JOBQ выполняет хранимую процедуру, указанную клиентским приложением. В этом примере JOBQ передает данные в процедуру PL / SQL на стороне сервера. Реализация процедуры обратного вызова PL / SQL определяет, как обрабатывается уведомление.

7Внутри серверной процедуры PL / SQL разработчик может реализовать логику для уведомления клиентского приложения среднего уровня об изменениях в зарегистрированных объектах. Например, он уведомляет приложение ROWID об измененной строке в hr.employees.

Общий процесс 8/9



1. В этом примере давайте предположим, что приложение кэшировало набор результатов запроса в HR.EMPLOYEES. Разработчик создает регистрацию для запроса на HR.EMPLOYEES, используя интерфейс уведомлений об изменениях PL / SQL. Кроме того, он создает хранимую процедуру PL / SQL для обработки уведомлений и предоставляет серверную процедуру PL / SQL в качестве обработчика уведомлений.

2.База данных заполняет регистрационную информацию в словаре данных.

3.Пользователь изменяет один из зарегистрированных объектов с помощью операторов DML и фиксирует транзакцию. Например, пользователь обновляет строку в таблице hr.employees во внутренней базе данных. Данные для hr.employees, кэшированные на среднем уровне, теперь устарели.

4.Oracle Database добавляет сообщение, описывающее изменение во внутренней очереди.

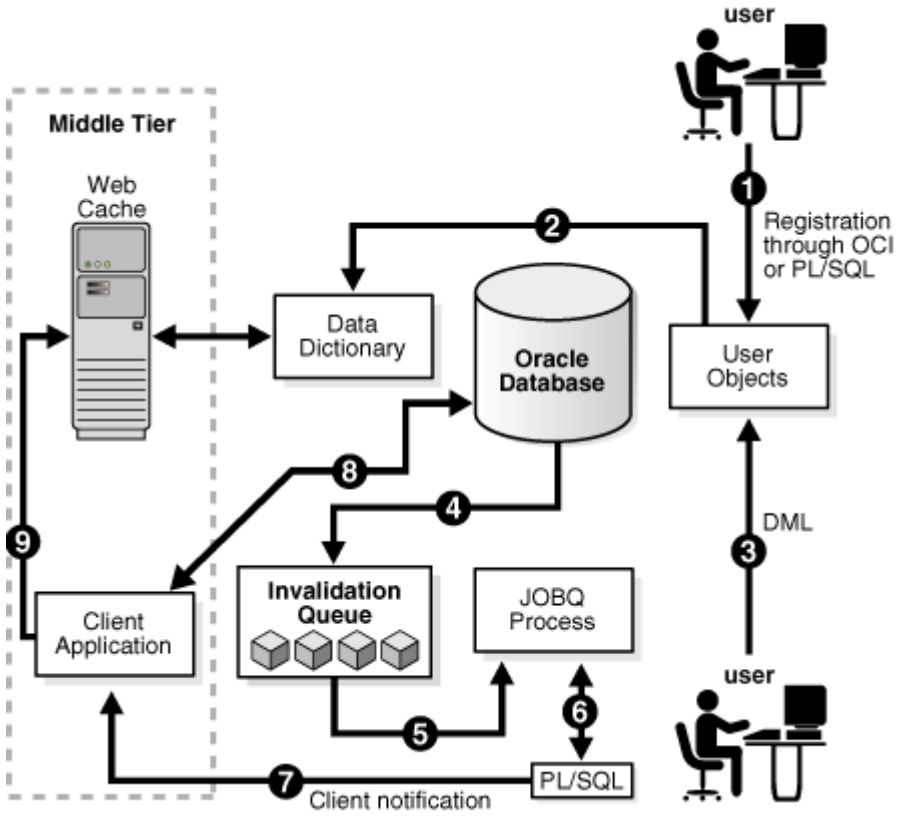
5.Фоновый процесс JOBQ уведомляется о новом уведомлении об изменении.

6.Процесс JOBQ выполняет хранимую процедуру, указанную клиентским приложением. В этом примере JOBQ передает данные в процедуру PL / SQL на стороне сервера. Реализация процедуры обратного вызова PL / SQL определяет, как обрабатывается уведомление.

7.Внутри серверной процедуры PL / SQL разработчик может реализовать логику для уведомления клиентского приложения среднего уровня об изменениях в зарегистрированных объектах. Например, он уведомляет приложение ROWID об измененной строке в hr.employees.

8.Клиентское приложение на среднем уровне запрашивает внутреннюю базу данных, чтобы получить данные в измененной строке.

Общий процесс 9/9



1. В этом примере давайте предположим, что приложение кэшировало набор результатов запроса в HR.EMPLOYEES. Разработчик создает регистрацию для запроса на HR.EMPLOYEES, используя интерфейс уведомлений об изменениях PL / SQL. Кроме того, он создает хранимую процедуру PL / SQL для обработки уведомлений и предоставляет серверную процедуру PL / SQL в качестве обработчика уведомлений.

2.База данных заполняет регистрационную информацию в словаре данных.

3.Пользователь изменяет один из зарегистрированных объектов с помощью операторов DML и фиксирует транзакцию. Например, пользователь обновляет строку в таблице hr.employees во внутренней базе данных. Данные для hr.employees, кэшированные на среднем уровне, теперь устарели.

4.Oracle Database добавляет сообщение, описывающее изменение во внутренней очереди.

5.Фоновый процесс JOBQ уведомляется о новом уведомлении об изменении.

6.Процесс JOBQ выполняет хранимую процедуру, указанную клиентским приложением. В этом примере JOBQ передает данные в процедуру PL / SQL на стороне сервера. Реализация процедуры обратного вызова PL / SQL определяет, как обрабатывается уведомление.

7.Внутри серверной процедуры PL / SQL разработчик может реализовать логику для уведомления клиентского приложения среднего уровня об изменениях в зарегистрированных объектах. Например, он уведомляет приложение ROWID об измененной строке в hr.employees.

8.Клиентское приложение на среднем уровне запрашивает внутреннюю базу данных, чтобы получить данные в измененной строке.

9.Клиентское приложение обновляет кэш новыми данными.

Процесс со стороны клиента

- **Пользователь должен иметь привилегию CHANGE NOTIFICATION**
- **Клиент “подписывается” на изменения – определяет объекты и регистрирует процедуру-обработчик (callback function)**
- **На клиенте запускается процесс слежения (change notification listener)**
- **Когда происходит изменение данных СУБД отсылает клиентам уведомление**
- **На клиенте автоматически выполняется процедура-обработчик**

Области применения

- **Применяется для контроля над данными которые больше читаются, и относительно редко меняются**
- **Для приложений имеющий свои механизмы кэширования**
- **Исследование поведения системы/приложения**