Announcements

- Midterm this Thursday (October 7) in class
  - Open-book, open-notes
  - Materials up to and including last Thursday's lecture
  - Sample solution to sample midterm (handed out last week) available today
- Supplemental Problem Set #1 posted last Saturday
  - Sample solution available today

Publish/subscribe systems

- Michael Caldwell
  - Introduction to publish/subscribe
  - IBM Gryphon project
- Ayonike Akingbade
  - Oracle's expression data
- James Galdos
  - Microsoft SQL server notification service
What is Publish Subscribe?

- Information Producers (Publishers)
- Information Consumers (Subscribers)
- Events/Messages
- Many-to-many
- Publishers and Subscribers are anonymous (need not be aware of each other)

Publish/Subscribe

An Introduction to IBM's, Microsoft's and Oracle's solutions to a growing demand for interconnecting information providers and subscribers in a distributed system

10-05-2004

Group-Based Pub/Sub

- Producers publish messages with a pre-defined subject or group. e.g., cs department activity announcements
- System delivers messages to all consumers subscribing to pre-defined subject, or group members
- Easily implemented via tables which match subject categories to consumers, but...
  - Not very flexible and each subscriber must do its own filtering

Content-Based Pub/Sub

- Messages are routed on basis of content, not category. e.g., activity announcements where activity.date < today+7 and location = 'LSRC'
- A different kind of paradigm, each subscription is equivalent to a predicate which tests an event's attributes.
- Much more flexible than group-based pub/sub, but scalability and reliability become much more difficult
  - Can easily implement a group-based publish subscribe system within the framework of content-based

Content Based Pub/Sub vs. Traditional DB Design

- Content-based pub/sub presents a challenge to the existing DB paradigm:
  - Database:
    - Data is stored
    - Queries arrive and must be checked against data
  - Pub/Sub ("Inverse" of Database):
    - Queries are stored
    - Data arrives and must be checked against queries

Why Do People Need Content Based Pub/Sub Systems?

- Real Time Data Reporting:
  - stock trading, sports scores, breaking news, etc.
  - INET Book Viewer
- Message Oriented Middleware:
  - "glues" together many different applications between multiple businesses or within a single firm (mergers).
Challenges
- Matching messages to subscribers in vast distributed systems
- Message propagation.
- Guaranteeing delivery of messages despite network failure or disconnect
- Delivery of state and/or history

IBM: Gryphon
Message brokering system
- Middleware designed to service large public networks
  - Wide-area extranet or intranet that is too large or complex to be centrally administered
- Major Concerns:
  - Reliability:
    - Ensured exactly-once message delivery

IFG Approach to Message Brokering

- Information Flow Graph:
  - Nodes: information spaces
    - Sources
    - Event histories or...
    - Event interpretations
    - Schemas
  - Operations/Data Flow types:
    - Select
    - Transform
    - Collapse
    - Expand (inverse of collapse)

Types of Dataflows Supported
- Select:
  - Connects two event histories having the same schema. Associated with a predicate
- Transform:
  - Connects two event histories with different schema. Associated with a rule
- Collapse:
  - aka, interpret, applies some function to event
- Expand
  - Inverse of collapse, converts a state into an event history

Exactly-Once Delivery
- Subscribers must receive all relevant messages
- In the case of disconnect, it is not enough to simply inform subscribers that messages have been lost. The system must deliver these messages (persistent messaging).

Knowledge Graph
Exactly Once Delivery (cont’d)

- Streams: represents information for each time tick, $t$
  - Knowledge ($K(t)$) - downstream
    - $Q$: lack of knowledge
    - $D$: published at $t$
    - $D^*$: every downstream subscriber has delivered $D$
    - $S$: silence
    - $F$: final (don’t care)
    - $E$: error
  - Curiosity ($C(t)$) - upstream
    - $C$: curious
    - $A$: anti-curious/ack
    - $N$: neutral

Exactly-Once: Terminology

- A gap is a sequence of $Q$ ticks between non-$q$ ticks in a stream.
- $S\ S\ Q\ Q\ Q\ Q\ D\ D$

- Checkpoint Token: each subscriber maintains set of (pubend, timestamp) pairs which describes when they connected to each pub end
- A subscriber is in catch-up mode if it is still consuming events that it missed during disconnect
- The doubt horizon is the latest timestamp such that all ticks between latestDelivered and the timestamp are not “$Q$” (unknown)

Terminology

- Pubend: publisher hosting broker (phb)
  - only stream which stably stores information
- Subend: subscriber hosting broker (shb)

Knowledge Graph inside SHB

- SHB = Subscriber Hosting Broker
- Event Retention/Release Protocol:
  - The data of any $D$ ticks after the Doubt Horizon is not delivered until all $Q$ ticks have been resolved
Publish / Subscribe with Oracle Expressions

Outline
- Publish / Subscribe using Oracle
- Expressions
- Evaluate Operator
- Indexing
- Publishing Example

Oracle Publish / Subscribe
- Oracle provides a way for users to filter the information they would like to receive
- Allows a subscription service to send information to only users who have requested it
- Uses Expression filters to decide which users should receive the information
- Can filter information not just on type of information but the value of the information
  Example: Rather than get information on all basketball teams, get scores on teams with winning records

Expressions
- Boolean conditions
- Evaluates to true if incoming data meets a user's interests
- Must follow the SQL-WHERE clause format
- Can reference any table, column, or function in a database (called data items)

Examples of Expressions
- Model, Price, Mileage, Year and Color are all columns in a Car Table
  - EX1: Model = 'Taurus' and Price < 15000 and Mileage < 2500
  - EX2: Model = 'Mustang' and Year > 1999 and Price < 20000
  - EX3: Price < 15000 and Color = 'Red'
- Can also include functions
  - Horsepower(Model, Year) gives the horsepower for a car with the given model and year
  - EX4: Horsepower(Model, Year) > 200 and Price < 25000

How to Set Up Expressions
- Expressions are stored in a column of an expression table
- They are stored as varchar2 variables
- An expression table may include other columns such as ID of user and other demographics
- Each row will include one expression for one user
Example of Expression Table

CONSUMER Table

<table>
<thead>
<tr>
<th>CID</th>
<th>Zipcode</th>
<th>Gender</th>
<th>...</th>
<th>Interest (Expression Column)</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>27708</td>
<td>M</td>
<td></td>
<td>Model = 'Taurus' and Price &lt; 15000 and Mileage &lt; 2500</td>
</tr>
<tr>
<td>102</td>
<td>20737</td>
<td>F</td>
<td></td>
<td>Model = 'Mustang' and Year &gt; 1999 and Price &lt; 20000</td>
</tr>
<tr>
<td>105</td>
<td>27705</td>
<td>M</td>
<td></td>
<td>Price &lt; 20000 and Color = 'Red'</td>
</tr>
<tr>
<td>107</td>
<td>27708</td>
<td>M</td>
<td></td>
<td>Horsepower(Model, Year) &gt; 200 and Price &lt; 25000</td>
</tr>
</tbody>
</table>

Evaluate Operator

- Used to evaluate an expression
- Will equal 1 if the expression is true and 0 if it is false
- Takes two arguments: the column of the conditional expression and the data items for the expression

Examples using Evaluate

- The following will return the CIDs of the users whose expression evaluates to 1 (true) with the given values for Model, Price, Mileage and Year
- Ex1: Select CID From Consumer WHERE EVALUATE(consumer.interest, Model => 'Taurus', Price => 22000, Mileage => 1800, Year => 2000) = 1;
- You can also select based on other attributes in the consumer table. The following will return the CIDs of the users whose expression evaluates to 1 and who have zipcode = 27708
- Ex2: Select CID From Consumer WHERE EVALUATE(consumer.interest, Model => 'Taurus', Price => 11000, Mileage => 1800) = 1 AND zipcode = 27708;

Indexing Expressions

- Expressions can be grouped together based on the data items they reference
- Data items that are in many expressions can be evaluated faster
- Uses a Predicate Table

Predicate Table

- Each predicate (individual condition that can evaluate to true or false) in an expression is stored in a row of a predicate table
- The most commonly referenced data items in the predicates will be columns in the table.
- Each data item will also have a second column that stores the operator for the data item
- Other data items will go in the “Sparse Pred” column

Example of Predicate Table

<table>
<thead>
<tr>
<th>(GROUP 1)</th>
<th>(GROUP 2)</th>
<th>(GROUP 3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RowID</td>
<td>Op</td>
<td>Value</td>
</tr>
<tr>
<td>R1</td>
<td>=</td>
<td>'Taurus'</td>
</tr>
<tr>
<td>R2</td>
<td>=</td>
<td>'Mustang'</td>
</tr>
<tr>
<td>R3</td>
<td>&lt;</td>
<td>20000</td>
</tr>
<tr>
<td></td>
<td>&lt;</td>
<td>25000</td>
</tr>
</tbody>
</table>

RowID = Identifier of the row storing the corresponding expression in the Expression Table
Group 1 = Predicates on “Model” of a car
Group 2 = Predicates on “Price” of a car
Group 3 = Predicates with Horsepower(Model, Year)
Op = Predicate Operator
SparsePred = Other Predicates
Index Processing

- When given values for the data items, the predicates for the most commonly referenced data items are first computed.
- The Sparse Predicates are then computed for the expressions where the above predicates all evaluate to true.
- If the sparse predicates are also true for a given expression then that means the whole expression is true.

Using Expressions for Publish / Subscribe

- Expressions in Oracle only provide a way to filter what users receive what information.
- Does not deal with how to publish the information to users.
- The application is responsible for handling the content based subscriptions and writing expressions to fit the subscription criteria.

Publishing

- We can use Triggers that will check every time new data comes in if it meets a user’s interest. We can then notify such users or publish the information to them.

```
EX: CREATE TRIGGER activechk
    AFTER insert OR update ON Car
    FOR EACH ROW
    DECLARE
        cursor c1 (ditem VARCHAR2) is
            SELECT c.cid,
                FROM   Consumer c
                WHERE EVALUATE (c.Interest, ditem) = 1;
        ditem VARCHAR2(200);
    BEGIN
        ditem := Car.getVarchar
            (:new.Model, :new.Year,
            :new.Price, :new.Mileage);
        for cur in c1(ditem) loop
            notify(CID);  // notify users
        end loop;
    END
```

Summary

- Oracle uses Expressions to filter information sent to users and provide content-based subscriptions.
- Expressions can be indexed to allow for quick evaluation.
- Oracle does not have a built-in notification or publishing service.
Microsoft Notification Services

What Microsoft Did

- "Providing an easy-to-use development model that takes advantage of XML, SQL, and the Microsoft .NET common language runtime, enabling enterprises to quickly build scalable notification applications."

Microsoft Targets

Who Can Use Microsoft Notification Services?

- **Consumer Applications**: stock info, financial institution send account balances and overdraft notifications, airline flight arrivals
- **Operational Applications**: Machine is down, manager notified
- **Business Intelligence**: inventory low, send more
- **Employee Communication**: when files updated, when new reports are available

“Easy to Use”

Developers build an Application Definition File that contains central logic for sending notifications
- You need to know XML, SQL, XSLT (Extensible Stylesheet Language Transformations)
- Integrated with Visual Studio .NET – don’t need to learn new tools

Although the tools are familiar, you will see the Application Developer bears a great burden for making MSNS powerful

Create the Application Definition File (ADF)

In this file the developer defines:
- Structure of the data
- Queries used to process data
- How notifications are processed and delivered
- Operational parameters for your application

All Events and Event Providers your application deals with must be specified

All of the rules that a subscription can follow: Can a User compare new events with past notifications? Can you show the highest stock price of the past day?

MSNS is very powerful but is heavily dependent on the developer

Basic Flow: Collecting Events

1. Events happen in the world
2. The Application Developer specifies Event Providers, which give collect batches of events and give them to the application’s Event Table
3. These batches of events in the Event Table are then compared to subscriptions – what people have asked for.
Basic Flow: Processing Subscriptions

1. We have a table of events
2. The User has subscribed – asked for notifications to generate under certain conditions – the conditions are called subscription data
3. Subscription data is compared with the Events we have
4. If the event is applicable we send a notification

Basic Flow: Sending Notifications

1. Based on the events and what the user wants – we now have notifications to send out
2. The Application Developer has specified the format for each type of notification
3. Raw Notification Data is transformed into the specified format, such as XSLT
4. MSNS does NOT handle delivery itself
   Instead it sends notifications on delivery channels for other delivery services to deal with (HTML, SOAP, SMTP)

Delivery

Notification Services itself does not handle the final delivery of notifications

Notification Services uses Delivery Channels, which can be thought of as pipes to delivery services such as e-mail gateways

- SMTP protocol
- Sending notifications to operating system files, used primarily for debugging applications.
- An extensible HTTP-based protocol and a variety of HTTP and SOAP delivery services
- Custom protocols.

- Digest delivery: sends multiple notifications in one message
- Multicast delivery: Notifications formatted once and sent to many

Scalability

- Subscription logic can run in parallel
- Notification Services is multithreaded – multiprocessor computers
- Notification Services service can be deployed on one server and the actual NS database can be deployed on another
- A single instance can support multiple event providers and multiple distributors. You can distribute the event providers, generator, and distributors across two or more servers
- Partition subscriptions across multiple Notification Services instances: Applications make the subscriptions independent of each other

Types of Subscriptions

Event-triggered subscriptions are evaluated by executing a database join between new events that have arrived and a large set of subscriptions

Supported Subscriptions are:
1. Event Triggered Subscriptions
2. Subscriptions Evaluated According to a Schedule
3. Based on Event History
4. Based on Application State
5. Utilizing External Databases

Types of Subscriptions

- Event Chronicle Table II

   How to support state and history?
   Normally after events are processed they are removed.
   What about the subscriber who wants to know:
   “When a new stock price is at least 5 percent more than the price in the previous notification for the subscriber.”
   We need to store data about past events and notifications.
   Application Developer can specify all kinds of Rules that store events in Event Chronicle Tables for later:
   - Nothing fancy: just big tables with stored events for each subscription
   - You can even store information about previous notifications

VERY POWERFUL – history, state, schedules all supported
**Implementing History**

We now have a history of all the EVENTS.

But MSNS provides more:

History of Subscribers, Subscriptions, and Notifications

**ANDS/ORS**

Dealing with ANDs and ORs:

The User could specify tons of predicates

"Alert me with This AND That AND Those OR the Other Thing"

You want to handle an arbitrary number of predicates and use the power of set oriented operations (this is what SQL does)

So you break down each predicate into its own subscription row

HUGE Optimization and flexibility by having each subscription take up many rows in the subscriptions table
Summary

Microsoft has provided numerous features and the ability to create notifications on extremely complex criteria.

MSNS is very scalable but they do not provide much for notification delivery.

MSNS puts power with the developer. The Application Developer must develop all the matching rules, history rules, even provider logic, data organization, and notification formatting.

Requires application developer skilled with all aspects of process: XML, SQL, Microsoft Visual Studio .NET, XSLT and preferably delivery methods.