Oracle Events System Demo

Copyrights

Open Source Project Repositories

The primary repository is stored at Github:

https://github.com/gaiansentience/oracle-events-system-demo/

Backup repositories are at Sourceforge and Bitbucket:

https://sourceforge.net/projects/oracle-events-system-demo/

https://bitbucket.org/gaiansentience/oracle-events-system-demo/src/master/

Project Contributors

Anthony Harper

Oracle Version Support

Minimum version Oracle 19c.

Supported versions: Oracle 19c and 21c.

Conditional Compilation: Any features coded using updated syntax that is incompatible with version 19 must implement conditional compilation to support the earlier version.

Events System Business Entities and Processes

The events system is intended for use by venue managers, event organizers, ticket resellers and customers. This may involve multiple applications and web services that are out of scope for the database project.

Venues

Venues are locations that will host events. A venue has a maximum capacity based on actual space, seating available. Events cannot exceed venue capacity.

Venue managers will schedule events at their venue, define tickets and assign tickets to resellers in blocks of any quantity.

Resellers

Resellers are companies that work with the venue to sell tickets to customers. Resellers are paid a commission for all tickets they sell. Tickets for an event are assigned to resellers in limited quantities.

Customers

Customers will purchase tickets to attend events. Customers must provide a unique email address to record their ticket purchases.

Events

Events are scheduled by the day they occur. The initial system design does not allow multiple events on the same day, but may be enhanced to allow multiple events at a venue in different time slots.

Event planned size may be at full venue capacity or less. A large venue can still plan a smaller exclusive event. Available tickets for an event are limited to the defined event size.

Event Series

Events can also be scheduled as a repeating weekly series that occurs on a certain day every week for the duration of the series. Events are identified as occurring in a series using the event series id sequence that is associated with each event created for a series.

Event Tickets can be set up and assigned for all events in a series as a group.

Customers can purchase tickets for a price category for all available events in a series using one transaction.

Event Tickets: Pricing Categories

Tickets are set up in different categories with prices and quantity available assigned by category. For example, an event may have 50 Backstage Passes, 150 VIP tickets, 200 Reserved Seating tickets and 1000 General Admission tickets.

Ticket prices in a category can be changed at any time, customers should be charged the price that is currently in effect for the ticket category. Reseller commissions are also based on the price that is in effect at the time of sale.

Ticket categories may change based on the event, sample categories might be anything: General Admission, VIP Seating, Backstage Passes, Platinum Sponsor, Early Purchase Discount, etc.

Ticket Assignments To Resellers

After ticket categories are assigned, tickets can be assigned to multiple resellers in blocks of any size. A block of tickets would all have the same ticket category. Blocks of tickets assigned to resellers can be reassigned to other resellers, or more tickets in a category can be assigned to a reseller. A reseller can be assigned blocks of tickets in multiple price categories.

For example, the event manager could assign 50 VIP tickets and 100 General Admission tickets to one reseller and 50 VIP tickets and 200 General Admission tickets to another reseller. The event manager could then assign an additional 100 General Admission tickets to the first reseller.

Any tickets not assigned to resellers can be sold directly by the venue. If all tickets have been assigned to resellers, no tickets are available directly from the venue.

Ticket Sales

A customer may purchase tickets from resellers or directly from the venue based on availability. If sold by a reseller, the sale must record the reseller.

When tickets are sold, the availability for that price category and reseller/venue should be updated immediately. All sales should validate current availability at the time of purchase. This will prevent overselling a show. If the number of available tickets is lower than the requested quantity the transaction will be canceled resulting in an error.

If tickets are purchased for an event series, the system will check availability for each event in the series. If any events do not have available tickets, those events in the series will not be included in the purchase.

All ticket sales are final. Future enhancements could support ticket refunds for canceled events.

Tickets can be purchased by ticket price category. Tickets may be purchased in multiple quantities.

All API methods for purchasing tickets require the price category and price being requested. The requested price will be validated against the price currently in effect for the price category. If the current price is lower than the requested price, the customer will be charged the current price rather than the requested price. If the requested price is lower than the current price, the transaction will be canceled, resulting in an error.

A ticket sale is defined as a quantity of tickets in a specific price category purchased by a customer from a reseller or directly from the venue at a moment in time.

Customers may make multiple purchases of tickets in multiple price categories..

Because ticket prices can be changed by the event organizers at any time up to the event, ticket sales must record the purchase price in effect at the time of sales.

Resellers are paid commissions on tickets they sell. Because ticket prices can be changed, each sales transaction made through a reseller should calculate the commission in effect at the time of sale.

Customer Tickets

The system design defines a ticket sale as a quantity of tickets in a specific price category purchased by a customer from a reseller or directly from the venue at a moment in time.

Enhancing the system to print the actual tickets will require each ticket in a purchase to have a unique serialization. Ticket serializations must be generated at the time of purchase.

Tickets should be able to be checked on admission to an event to see if the specific ticket has already been used to enter the event. If a ticket has already been used for entry, this process will return an error.

The system will support associating government issued ID information with individual tickets.

Enhancements are also planned to associate specific seating information with individual tickets.

Ticket Serialization

Ticket serialization is intended to prevent counterfeiting and aid with customer loss replacement.

The ticket serialization identifier should include an abstraction of the customer identity as well. This will aid in identifying counterfeit tickets. If tickets were just serialized numerically it would be easy to produce fakes with apparently valid serial numbers for an event.

If a customer loses their tickets and requests replacements, the serial numbers for the lost tickets should be identified as invalid and new serial numbers issued with the replacements.

Ticket serialization is also used to support validation of tickets on entry to the event. When a ticket is validated the status of the ticket is changed from ISSUED to VALIDATED. A ticket set to VALIDATED cannot be validated again and will raise an error. This prevents someone from entering the event with a duplicated ticket.

Reseller Commissions

Reseller commissions are initially set up at the reseller level. Future enhancements could support resellers negotiating a commission rate with each venue and or commissions associated with each event.

Initial system design records commissions associated with ticket sales at the time of sale. Because ticket pricing can be changed while tickets are being sold for an event, the price in effect at time of sale is recorded with the ticket purchase and is used to calculate the commission.

Enhancements should include recording when commissions are paid to a reseller. These payments should be reconcilable with the ticket sales commissions calculated.

Events System Scope

The events system is intended to demonstrate an example Oracle data structure with a PL/SQL API to interact with all client applications. Client application design and coding are out of scope for the project.

Data Structures

The design includes all tables necessary to implement the business entities and processes.

Tables are implemented using identity columns for all surrogate primary keys and unique constraints for the business key which uniquely identifies each row.

Foreign key constraints are used to enforce relations between related rows in parent and child tables.

Not null check constraints are used to require data entry in specified columns.

Database Application Programming Interface

All APIs will use stored procedures organized into packages. The initial design only uses a single schema for data and API code, planned enhancements will separate code and data schemas.

The core API uses stored procedures for all data creation and update processes. These stored procedures will contain internal validation logic to enforce business rules.

Because the core API will contain all business logic, any extended API (reporting or web services) will call the basic API internally.

The core API reporting methods return all non scalar data to calling applications using ref cursors.

The Reports API exposes these ref cursors as pipelined table functions for client applications that cannot consume ref cursors.

The web services APIs expose all API functionality to web services using JSON or XML documents for all interactions with the web service.

Creation and coding of the web services is out of scope for the project. It is envisioned that Oracle Rest Data Services should be sufficient to expose the document APIs created by the project.

Core API Functions

Create Venues. Update venue information.

Create Resellers. Set reseller commissions. Update reseller information.

Create Customers.

Create Events.

Create Event Series.

Create Ticket Categories with prices and quantities.

Create Ticket Assignments to resellers. Update ticket assignments.

Purchase a quantity of tickets from a reseller for an event and ticket pricing category.

Purchase a quantity of tickets directly from the venue for an event and ticket pricing category.

Purchase a quantity of tickets for all events in a series for a pricing category from the venue or a reseller.

Generate individual serialized tickets when a purchase is made.

[Planned] Update serialized tickets when customer requests replacement for lost tickets.

[Planned] Validate tickets on entry to an event to prevent reuse of the serialized ticket.

[Planned] Associate a government issued ID with individual tickets.

Reporting API functions

Provide reports of venue scheduled events.

Reports to support defining and updating ticket groups pricing categories and availability.

Reports to support assignment of tickets to resellers by ticket group.

Reports to display ticket availability for resellers or venues.

Reports to display ticket pricing and availability by category for an event or event series.

Reports of ticket purchases by customers for an event or event series.

Performance reports for the venue comparing reseller sales performance.

Performance reports for the venue comparing reseller sales performance for a specific event.

Commission reports for each venue displaying monthly reseller commissions by event.

Other reports to be defined as needed.

Web Services Interfaces

Web services interfaces expose the basic API and the reporting API to web service application consumers using a document based approach supporting both JSON and XML services.

Because the web services interfaces are document based, exposed methods support batch processing that would require multiple calls using the traditional approach defined in the core API.

Web services methods parse incoming requests and then call methods in the core API, keeping the business logic centralized. Action requests are then updated with the status of the action(s) taken and returned as the web service reply. The initial version still has some parameter validation logic for missing or invalid parameters, this should be moved into the core API.

Reporting methods return documents that are pre formatted using the appropriate document type views, supporting a simple select of the formatted document to be returned from the web service.

If an unrecoverable error such as an issue with parsing occurs, the web service will return a standard error document specifying the method called and the resulting error message from the database.

For batch processes that return individual status codes and messages for each item in the batch, these individual status codes are set to ERROR or SUCCESS and the status message returns the error from the API layer.

JSON API

The JSON API design includes support for JSON documents for system transactions and reports.

All JSON parsing and formatting is done using Oracle JSON SQL and PL/SQL methods.

Create or update requests that require specialized formats are documented in the package specification.

At present the test scripts contain examples of request and reply documents. Appendices to this document will be updated to document each request/reply format pair for each web service method.

XML API

The XML API design includes support for XMLdocuments for system transactions and reports.

All XML parsing and formatting is done using Oracle JSON SQL and PL/SQL methods.

No XML namespaces or schema documents have been implemented.

Create or update requests that require specialized formats are documented in the package specification.

At present the test scripts contain examples of request and reply documents. Appendices to this document will be updated to document each request/reply format pair for each web service method.