Concurrency and Recovery

In this section, basic concurrency and recovery primitives of locking, notification, and logging are addressed. The first few tables cover different kinds of locking: reader/writer, optimistic, and custom locking. In addition, deadlock detection, different granularities of locking (instance, class, pages, segments), and the point at which the locks are set are also reviewed.

The next group of tables examines another kind of concurrency primitive: triggers or change notification mechanisms. What can initiate a trigger, where the triggers execute, and the language for the triggered operations are important issues here. Alerters, a variation of triggers in which named events are generated and received, are also covered. Invocation of an alerter based on explicit invocation may be more efficient than invocation of a trigger in some applications, since an invocation predicate need not be monitored by the product.

The remaining tables in the section address other concurrency and recovery primitives: checkin/checkout, audit trails, disk recovery, and backup.

More advanced concurrency abstractions can be built on top of the primitives described in this section. The two such abstractions most commonly used are transactions and versioning. These two provide quite different ways to share data among multiple users, and they are described in the next two sections. Of course, an application-specific abstraction could be constructed directly on locking or notification mechanisms instead.

Part 3, Checklist 6, of *The Object Database Handbook* provides a discussion of concurrency and recovery.

Standard Concurrency Control

	optimistic – assumes that simultaneous access to same objects is quite rare and conflicts checked only at commit time – conflicts may result in an abort or restart – can also be achieved branching versions, see page 167	pessimistic – checks for access conflicts at the time the lock is requested with some locks queued until the conflicts are resolved – no conflicts are present at commit time	concurrency control can be specified at the system level (a description of the specification is noted: turned on/off, changed to optimistic/pessimistic)	concurrency control can be specified at the session level (a description of the specification is noted: turned on/off, changed to optimistic/pessimistic)	concurrency control can be specified at the application level (a description of the specification is noted: turned on/off, changed to optimistic/pessimistic)	concurrency control can be specified at the transaction level (a description of the specification is noted: turned on/off, changed to optimistic/pessimistic)

Advanced Concurrency Control

	many readers, one writer (MROW) – multiple readers read the pre-state of an object concurrently with one exclusive writer – also called non-blocking read consistency	read only mode in which transactions read committed data and never interfere with read-write transactions	many-readers, many writers – conflicts controlled at a sub-object level	an application can be limited in the time it can hold a granted lock	field level exclusion supported along with optimistic concurrency control

Custom Concurrency Control

	custom DBMS concurrency – possible to define custom concurrency control for the entire database manager	custom logical database concurrency – define custom concurrency control for logical databases managed by the DBMS	custom class or type concurrency – possible to define custom concurrency control on a class or type basis	custom application concurrency – possible to define custom concurrency control on an application basis

Deadlocks

	lock requests requiring polling	lock requests are queued	deadlock detection	time-outs supported	time-out time can be adjusted at the system level	time-out time can be adjusted at the application level	time-out time can be adjusted at the class or type level	support for application specific retry code upon lock conflict detection

Instance Lock Modes

	S – lock an instance object in Shared mode for read	X – lock an instance object in Exclusive mode for update

Class or Type Lock Modes

	IS – lock the class or type definition in Intention Share mode for read and lock instance objects in S mode	IX – lock a class or type definition in Intention Exclusive mode for update and instance objects in X mode	S – lock a class or type definition for read and implicitly lock instance of the class or type in S mode	X – lock a class or type definition for update and implicitly lock instances of the class or type in X mode	SIX – lock a class or type definition for read, implicitly lock instances of the class or type in S mode, explicitly lock instance to be updated in X mode

Lock Granularity

	data page	index page	group page	segment	file	object clusters or containers	isolation of single objects for page or container locking	single class or type	single object or instance	attribute or data member of a single instance	adaptive locking (starting at one granularity and changing to another)

Reference Lock Granularity

	locking an object locks objects pointed to by the references in the locked object	locking an object does not lock objects pointed to by the references in the locked object	able to lock an entire composite object – see page 45 for more information on composite objects

Lock Setting, Releasing and Promotion

	locks set automatically by the system such as when an object is dereferenced	locks released automatically by the system	locks set explicitly by the application	locks released explicitly by the application	automatic promotion of locks from read locks to write locks based on activity	automatic demotion of locks from write locks to read locks based on activity	automatic lock escalation to higher granularity such as instance to class or type if many locked instances

Change Notification or Triggers

	events trigger client-side method or function execution (client-side active notification	events trigger server-side method or function execution (server-side active notification)	active notification methods or functions can be refined on a class or type basis	events raise a flag – change found by querying (passive notification)	pre- and post-procedure invocation	allows for deferred execution of change notification or trigger	predicate can be applied for change notification or triggers	triggers or change notification can abort a transaction	supports prioritizatio n of change notification or triggers	events can result in effects inside and outside the database	triggers or change notification participate in database recovery	triggers or change notification generation have atomicity with commits

Change Notification or Trigger Types Built-in

	instance creation notification	instance update notification	instance deletion or dereferencing notification	instance query notification	notification of a checkout of objects from one type of workspace or database to another	notification of a checkin of objects from one type of workspace or database to another	make version notification

Change Notification or Trigger Execution Location

	in the server(s) (server-based)	in the client/application (client-based)

Change Notification or Trigger Implementation

	built-in to the product	provided by a third-party (product is named)

Server-Based Change Notification or Trigger Language

	C++	C#	Java	Perl	Python	Visual Basic	Smalltalk	CLOS	C	Lisp	SQL	other language

Client-Based Database Change Notification or Trigger Language

	C++	C#	Java	Perl	Python	Visual Basic	Smalltalk	CLOS	C	Lisp	SQL	other language

Alerters or Named Events

An alerter or named event notifies client applications of events that occur in the database. An application must declare its interest in an alerter or named event by listening for it.

	supports alerters or named events	alerters or named events returns parameters or result sets	client application received notification by polling	client application receives notification via callbacks causing interrupts

Checkin/Checkout of Objects

	private instances can be checked in for shared use	private instances MUST be checked in for shared use access	private classes or types can be checked in for shared use	private classes or types MUST be checked in for shared use access	shared instances can be checked out for private use	shared classes or types can be checked out for private use

Audit Trail

	supports an audit trail as part of the logging facility	audit trail events may be customized

Replication Independence

	schema replication			data replication		
		users can act as if the schema is not replicated at all			users can act as if the data is not replicated at all	
	schema represented at the physical level by many distinct copies stored at many distinct sites	the schema can be updated only on the master site	the schema can be updated at either the master site or any slave site	data represented at the physical level by many distinct copies stored at many distinct sites	data can be updated only on the master site	data can be updated at either the master site or any slave site

Replication Type

Also see Plug-Ins and Replication on page 57.

This refers to data replication only.	real-time – replication occurs as part of the same transaction	store and forward – replication occurs on a periodic basis	time-based – replication occurs on at a set time of day	event-based – replication occurs at a specific event

Replication Granularity

This refers to data replication only.	database replication – entire database, all servers are replicated	server replication – selected servers, but all data at the server is replicated	all instances replication – all instances of selected classes or types can be replicated	specific instance replication – instances of selected classes or types can be replicated	user-defined types or objects participate in replication in the same way as system-defined types or objects