JTC

JAVA TABLE CACHE

MAKE THE MOST OF
YOUR MASTER DATA

BOI BETTER ORGANIZED INFORMATION

www.boi.at

JAVA TABLE CACHE (JTC)

MAKE THE MOST OF
YOUR MASTER DATA

Do your business applications overload your servers with master data queries?

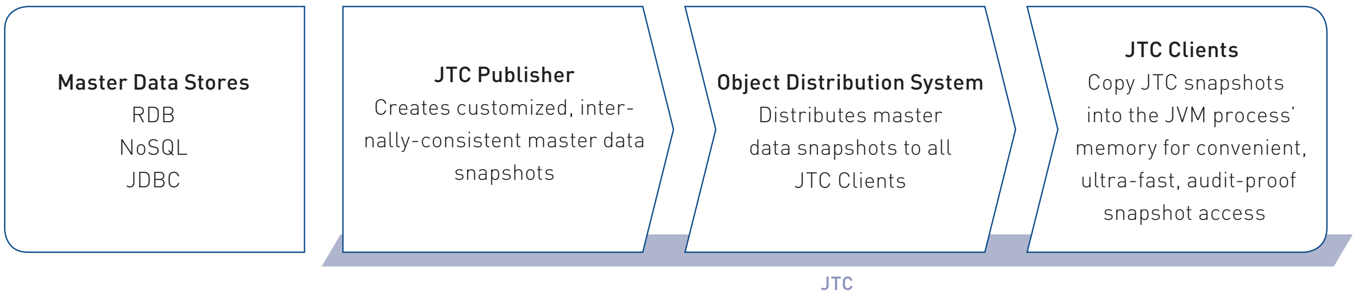
Consider using BOI's JAVA TABLE CACHE (**JTC**) to:

- Safely create customized master data snapshots from your existing master data stores
- Replicate those snapshots within your data center or across your worldwide network using your existing infrastructure
- Enjoy convenient, ultra-fast, in-memory access to your master data using plain Java in an unlimited number of JTC Clients
- Bathe in calmness: your IT architecture now supports audit-proof master data access and distribution

JTC at a Glance

JTC, a product of **BOI Software GmbH (BOI)**, delivers your master data to your Java business applications in a form that is ultra-fast to access, linearly-scalable, internally-consistent, versioned, and ultimately-traceable. **The JTC system involves three components: JTC Publisher**, an **object distribution system** of your choice, and **JTC Client**. JTC Publisher collects master data from various sources, prepares it in tabular form, and uses the object distribution system to publish the master data to the JTC Clients. JTC Client is a small pure-Java library linked into your Java business applications. It copies the master data tables from the object distribution system into the JVM process' memory, and provides a concise API to access them as in-memory indexed tables.

JTC elivers your master data to your Java business applications.



Master Data Stores
RDB
NoSQL
JDBC
JTC Publisher
Creates customized, internally-consistent master data snapshots
Object Distribution System
Distributes master data snapshots to all JTC Clients
JTC Clients
Copy JTC snapshots into the JVM process' memory for convenient, ultra-fast, audit-proof snapshot access
JTC

Figure 1: JTC concept overview

JTC is available in two editions. **JTC Server Edition** is for customers who do not already have a scalable replication system for master data distribution. JTC Server Edition includes BOI's own hierarchical object distribution system. **JTC Enterprise Edition** enables you to distribute your master data using your existing object distribution system. This may be any enterprise-class, key/value, distributed object cache supporting PUT and GET semantics.

JTC is available in two editions.

Introduction: Master Data and Business Processes

An organization's **master data** provides a single source for business data used throughout the organization's systems, applications, and processes. Because master data typically changes relatively slowly, but is widely and heavily used, it is commonly replicated throughout the organization. Master data may be as simple as a list of country code abbreviations. More complex master data might include a collection of tables parameterizing how to calculate a life insurance policy within the regulatory and business environments of every country in the world.

An organization's master data provides a single source for business data.

Master Data	Branch
Basic medical claims data	Medical insurance
Life insurance premium tables	Insurance
Interest and exchange rate tables	Credit and Banking
Shipping and postage rate tables	Logistics
Transport schedules	Logistics
Machine part replacement schedules	Manufacturing, Operations
Vehicle rental rate tables	Car rental
Retail product master data	Retailing
Human resources pay schedules	Organizational Management
Business rule parameter tables	Organizational Management
Corporate policy tables	Organizational Management
Any other type of reference data	

Table 1: Master data examples

An entire discipline has emerged concerned with the management of master data in large organizations. JTC is a highly focused product concentrated on the distribution of, and ultra-fast access to, master data.

JTC: distribution of, and ultra-fast access to, master data.

Components of a JTC System

A JTC system is composed of three components which work together to support consistent, large-scale distribution of your master data across your organization:

- JTC Publisher
- An object distribution system
- JTC Client

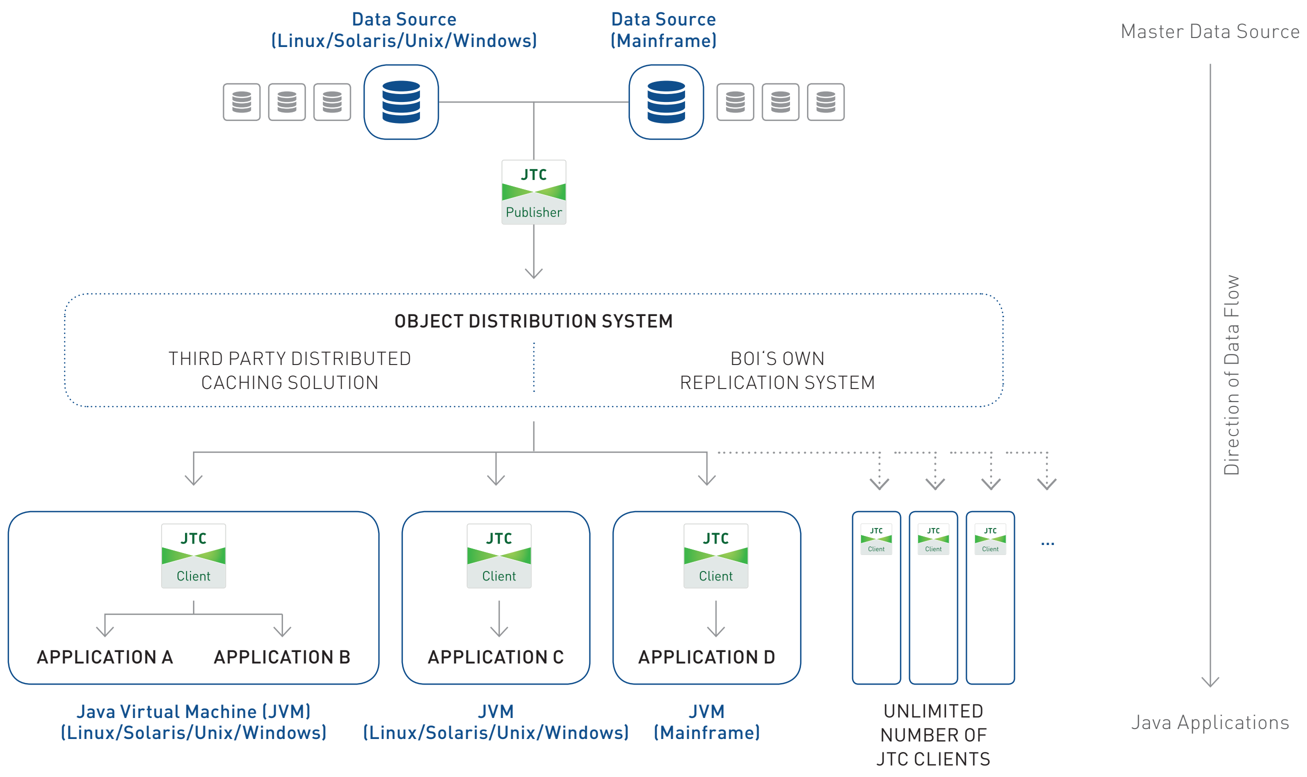


Figure 2 A JTC deployment with multiple data sources and multiple clients within multiple business applications

JTC Publisher

JTC Publisher creates JTC Snapshots based upon your master data.

JTC Publisher is a server-side component that periodically creates JTC Snapshots based upon your master data, and publishes these snapshots to the object distribution system.

A JTC Snapshot is an internally-consistent, uniquely-identified, versioned, traceable subset of your master data. Each JTC Snapshot – tagged with a unique identifier – contains a collection of tables and corresponding indexes. JTC Publisher uses your query specifications to collect and transform your master data into the snapshot's tables and indexes. Your queries (typically SQL queries) may access one or more local or remote master data sources. You specify both the structure and content of each table and index in the snapshot. You may create a snapshot periodically, programmatically, or manually.

A **JTC Table**, just like the TABLE you know from a relational database, is a matrix-like structure with columns and rows. Each column is assigned a name and a data type. Each row corresponds to a record. Since a JTC Table is simply a data structure in Java, the column data type is also a Java data type. The JTC Publisher configuration allows you to specify how to map column data types between your data source and your JTC table.

JTC Publisher acts as a central repository for all configuration related to JTC. There is no need to maintain significant configuration on each JTC Client instance: each JTC Client requires only enough configuration to locate and attach to the object distribution system.

JTC won't force you to abandon your existing master data sources. You can **continue to use whichever tools you already have for maintaining your master data**. JTC can draw master data from multiple backing data sources: relational databases; NoSQL (i.e. non-relational) databases; document databases, XML, Excel or CSV files; or any combination thereof.

Object Distribution System

In a JTC deployment, the object distribution system is the component that actively delivers scalability – the other JTC components simply stay out of the way. Information flows through the object distribution system from the JTC Publisher to the JTC Client, not vice-versa. Because of this one-way information flow, there is no need for distributed transactions or locks, and there is **no limit on the number of JTC Clients** that can receive and cache your master data snapshots.

Information flows through the object distribution system from the JTC Publisher to the JTC Client, not vice-versa.

Your object distribution system should match your scalability requirements in at least three dimensions: geographic, heterogeneous application, and homogeneous application. Geographic scalability refers to the system's ability to distribute master data over long distances, possibly even around the globe. Heterogeneous application scalability refers to the system's ability to distribute master data across many different types of applications. Homogeneous application scalability refers to the system's ability to distribute master data across many different instances of the same application.

You choose which object distribution system you use with JTC. If you do not already have a scalable replication system, use JTC Server Edition. This edition includes BOI's own hierarchical master data replication system, which BOI has optimized for a pure JTC environment.

You choose which object distribution system you use with JTC.

However, to leverage your existing distributed object cache solution, use JTC Enterprise Edition. JTC Publisher and JTC Client plug directly into Hazelcast, Infinispan and Terracotta. JTC Enterprise Edition is also compatible with other distributed object cache products. The one-way flow of information from JTC Publisher to JTC Client means JTC's only requirements for the object distribution system are:

- It must support GET and PUT operations in which keys are strings, and values are (possibly large) sequences of bytes
- PUT operations are only necessary at the publisher node; GET operations are only necessary at the client nodes
- GET operations must deliver results which are consistent with the sequence of PUT operations
- It must have a Java API

Again, you are in control: you have the freedom to select an object distribution system that meets your needs with respect to scalability, latency, and resource consumption.

JTC Client

JTC Client is a small pure-Java library that you link into your Java application to gain ultra-fast in-memory access to the JTC tables you published with JTC Publisher. JTC Client does much more than simply calling GET on the object distribution system. From the Java application's point of view, JTC Client is an in-memory database providing a degree of read consistency available only in the strictest transaction isolation level offered by standard databases.

JTC provides ultra-fast master data snapshot access from Java.

As a Java library, JTC Client helps you resolve some common business application programming difficulties. These are captured by the following questions and answers.

Are you concerned your master data accesses aren't transactionally consistent?

When running a master data query against a relational database, the entire query result must be internally consistent. One aspect of ensuring internal consistency when frequent master data updates are present, is to ensure transactional consistency of the query. That is, you must exclude the possibility of dirty reads, non-repeatable reads, and phantom reads, which may have been introduced due to other transactions running concurrently with your query. To safely achieve transactional consistency, you typically adjust the transaction isolation level to serializable.

JTC ensures transactionally consistent master data access.

Software architects often try to avoid the serializable transaction isolation level, as it can be quite taxing for the database. The reason is that the database must be capable of supporting, concurrently with the query, updates to the tables involved, possibly keeping multiple copies (or holding multiple locks) until the query has completed. Although choosing a less stringent transaction isolation level may ease the load on the database, this choice may also increase the risk of receiving transactionally-inconsistent query results.

In a JTC system, your business application queries are run against a master data snapshot instead of directly against the master data store. This architecture has two implications. First, for the business application query results to be internally consistent, the snapshot must also be internally consistent. And second, since snapshot generation is rare, the query frequency on the master data store is drastically reduced.

This frequency reduction diminishes the relevance of transaction isolation level performance. Therefore, when generating internally-consistent snapshots, you can enjoy the safety of the serializable transaction isolation level without incurring the elevated load resulting from its frequent use.

There is **no performance penalty** when a snapshot is accessed over an extended period of time from JTC Client: there are no database locks to hold or multi-version transaction records to maintain. Snapshot use in JTC Client is also thread safe: access the same or a different snapshot from different threads without hazard and without performance degradation.

Are your programmers forced to code their own searches?

JTC makes data queries easy.

If you are already bringing your master data to your applications, you may have found searching it to be frustrating and repetitive. The data is in memory, but you can't search it conveniently. The powerful query tools you used with the relational database are no longer available to the application. There are no common tools for querying tabular data inside of a Java application without moving to a full in-memory Java database. Although much faster than client-server databases, common in-memory Java databases are slow compared to JTC. They must carry the burden associated with mutable shared data (not to mention the JDBC interface).

In addition to providing fast, sequential access to your in-memory JTC Table rows, JTC Client provides convenient compile-time API's for querying by

- column or multi-column equality (i.e., compound keys)
- column or multi-column inequality (i.e., range and not-equals queries)
- filtering operators such as "<", ">", "≤", "≥", "≠", "contains", "contains-case-sensitive", "begins-with", "begins-with-case-sensitive", "ends-with", "ends-with-case-sensitive", and "regular-expression"

Do your programmers hard-code their own indexes (often in the form of hash tables)?

Your JTC client queries are automatically index-accelerated.

Performance, in addition to convenience, is often a requirement when searching through data in a Java application. Programmers commonly resort to building primitive indexes using hash tables or binary search trees. This practice leads to repetitive hard-coded indexes, something you and your programmers may find repugnant.

JTC Publisher can create and distribute indexes: you configure which JTC indexes it should create for each table. **JTC Client will make both the data tables and the corresponding index tables available for processing in-memory query requests.** The benefits for you are:

- You receive ultra-fast query results
- Your programmers don't have to create their own indexes
- You can add indexes later, as a matter of configuration; JTC will distribute them to all of your business applications
- You don't have to worry that your index and table versions don't match
- You don't have to rebuild your indexes when a new snapshot arrives

Your JTC Client queries are index-accelerated.

JTC: Advanced Master Data Distribution

At this point we've discussed each of the three components in a JTC system, and highlighted some of the features each delivers to help you distribute master data consistently throughout your organization. Now it is time to consider the system as a whole.

Time Tagged Snapshots

Imagine you need to generate a report or a quote based on yesterday's master data instead of the current master data. How are you going to accomplish that?

JTC can access multiple snapshots concurrently.

Within a business application, JTC Client allows you to choose a snapshot to use for subsequent queries. You can choose any snapshot from the sequence of snapshots available in the client's cache. If you do not explicitly select a snapshot, the default choice is the most recent snapshot. If your master data records include validity date ranges, you can select the snapshot that was valid at a particular date or time. You may also select a snapshot directly with a snapshot identifier. These snapshot selection options of the JTC Client API enable you to access older versions of your master data. You may, if you choose, still access multiple snapshots concurrently (for example, in a multi-threading environment such as a Java application server).

You control how long snapshots remain available in the cache with a simple JTC Publisher configuration parameter.

Data Spaces and Multitenancy

Do you need to run your business rules on master data records that are tailor-selected for a particular geographic region, organizational division, or process maturity level?

JTC offers multiple data spaces and multitenancy.

In JTC, you don't need to modify each business application query to insert filtering logic that selects the desired region or division. Instead, JTC allows you to create **independent data spaces within the same snapshot**. Each data space contains the same table and index structures, but with different data content.

JTC Data Spaces are often used to support multitenancy. To do so, JTC Publisher is configured to create one data space per tenant. A JTC Client must only specify which data space it wants to use; it does not need to tailor any queries, as all tables are pre-filtered during publishing.

Auditing Master Data Distribution

Are you operating in a regulatory or business environment that requires clarity and traceability with respect to which master data has been used where and when?

JTC is audit-proof.

Every JTC snapshot has a unique ID accessible via the JTC Client API. Using its own logging mechanism, every business application can log the snapshot ID corresponding to each of its query results. To complement this, you can configure JTC Publisher to archive the full content of each JTC snapshot. These features enable you to audit the exact snapshot content used by any query from any business application. Finally, by tracing every PUT operation, JTC Publisher makes its use of the object distribution system transparent.

JTC Application Performance

JTC Publisher queries your master data sources to generate master data snapshots. The object distribution system delivers these snapshots to the JTC Clients. Your business applications use JTC Client's query API to retrieve master data from the snapshots.

JTC client queries are up to 550 times faster than direct database access via JDBC.

How fast is a master data query in JTC Client compared to the same query in a client-server database system, or an in-memory Java database?

To address this question, BOI conducted experiments to compare single-table query rates in three scenarios:

1. JTC Enterprise Edition with Hazelcast via JTC Client
2. Oracle 11g Release 2 via Oracle's thin JDBC Driver
3. H2 in-memory via H2's JDBC Driver

The test environment was identical in each scenario and consisted of:

1. A virtualization host: An IBM x3650 M4 server stocked with 12 Intel Xeon E5-2630 processors (2.30 GHz)
2. A virtualized guest: SUSE Linux Enterprise Server 11 64-bit assigned four computing cores and 2GB of RAM
3. Oracle Java version 1.6

The IBM server was otherwise idle during the tests.
In each case, all test components ran within a single virtual machine.

Each test consisted of the following steps:

1. Randomly select a primary key value
2. Query a single table for all columns from the row that matches the primary key selected in step 1
3. Repeat from step 1 for a fixed number of iterations

All table fields contained variable-length character data up to a maximum of 255 characters. The individual tests varied along three dimensions:

1. table size (in both the number of rows and the number of columns)
2. the number of columns in the primary key (1 or 2)
3. the query engine (Oracle, H2 and JTC)

To allow for prepared statement compilation in Oracle and H2, and for initial table selection in JTC, the time for the first query in each test was discarded. Additionally, in each scenario, appropriate indexes were created to match the selection column(s) in each test.

The tests do not mutate data. Oracle uses pre-filled tables, H2 uses tables loaded into memory, and JTC uses snapshots. Non-volatile media characteristics, such as IOPS, disk latency, and disk throughput should not influence the test results.

The results of the tests are summarized in Table 2 below. **In these tests, JTC queries are an order of magnitude faster than H2's. JTC queries are up to 550 times faster than Oracle's queries.**

Table 2: Test results comparing query volume among Oracle, H2 and JTC test systems.

Table size		Oracle 11g Release 2 via JDBC Thin client-side driver	In-Memory H2 Database Engine via JDBC		JTC EE + Hazelcast with stable mode (without a near cache)	
		Queries/Sec	Queries/Sec	X times faster than Oracle	Queries/Sec	X times faster than Oracle
100x10	1*	10.799	434.782	40	6.250.000	579
	2*	10.504	392.156	37	3.676.470	350
1,000x25	1*	10.570	279.329	26	5.263.157	498
	2*	10.482	245.700	23	3.115.264	297
10,000x50	1*	9.765	165.016	17	3.690.036	378
	2*	9.718	159.744	16	2.267.573	233

* Columns in primary key

After studying these claims, you may be wondering, "Are these extraordinary performance gains plausible?"

Yes, these test results are plausible. BOI designed JTC's architecture for ultra-high performance. JTC brings the master data to your business application. JTC also provides a client-side query API for direct data access. Consequently, JTC Client query latencies are unaffected by:

1. the additional software layers and abstractions introduced by the JDBC API
2. runtime query compilation or prepared statement lookup
3. network traversal by query request/response packets
4. inter-process communication
5. context switching

These test results show that using JTC for master data queries has the potential to yield significant savings. The more queries you run, the more you economize on time and resources by using JTC.

BOI is aware that synthetic test scenarios never faithfully reflect how a product responds to a specific customer's IT requirements and environment. However, we are happy to work with you to explore the benefits of using JTC in your business.

Conclusion

JTC's three components assist you in maximizing the value of your organization's master data while maintaining harmony with your existing master data management tools. JTC Publisher supports you in creating versioned snapshots of your master data from virtually any single data source or combination of data sources. You choose an object distribution system to distribute your snapshots, whether it be your existing third-party enterprise object cache or BOI's own JTC Server Edition. JTC Client provides your Java business applications audit-proof, ultra-fast, in-memory, thread-safe access to your replicated snapshots. Together, the **JTC** components **provide master data access to your business applications that is scalable, convenient, safe and low-latency.**

In a JTC deployment, you retain full control over all other aspects of master data management including the processes, policies and tools you use to collect and curate master data.

We invite you to make the most of your master data with Java Table Cache.

Specifications

JTC Publisher

- Operating systems: AIX, Unix, Linux, Windows, z/OS, BS2000, VSE
- Data source compatibility: DB2, Oracle, MySQL, PostgreSQL,...
- Web-based Frontend

JTC Client

- Pure Java
- Requires Java version 1.6 or newer
- No third-party Java dependencies

Object Distribution System

JTC Server Edition

- JTC-proprietary
- Operating systems: AIX, Solaris, Unix, Linux, Windows, z/OS, BS2000, VSE

JTC Enterprise Edition

- Hazelcast
- Infinispan
- Terracotta
- Any object distribution system supporting GET and PUT operations

All names and designations may be trademarks of their respective owners.

BOI BETTER ORGANIZED INFORMATION

BOI – Your Specialist for Ultra-fast Access to Your Master Data

To learn more about JTC, please contact Frank Sinner (sales@boi.at) or read the JTC product description at www.boi.at/en/jtc.