Creatio Academy

Creatio maintenance

Version 8.0

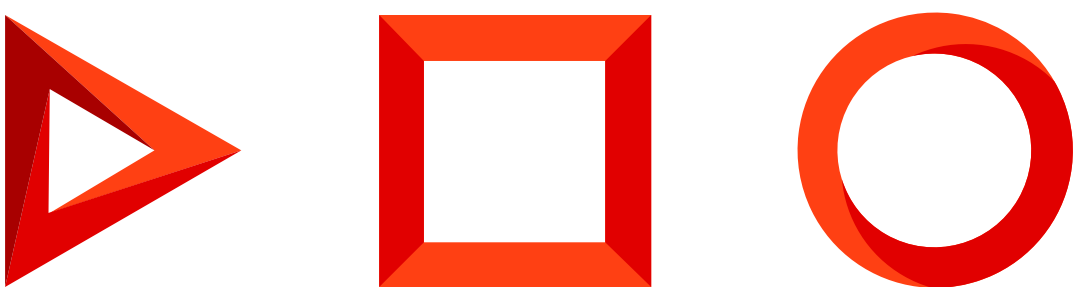

Table of Contents

Clean up the drive space

PRODUCTS: ALL CREATIO PRODUCTS

It is important to delete outdated and irrelevant records in a timely manner when you maintain a large database. As a result, the database server will use less space on its drive, which improves the database performance

Possible reasons for quick database growth

The following factors may lead to rapid database growth:

- Incorrectly set up or redundant **record permissions**. For example, the record permissions are set individually for a large number of users not arranged in groups. In this case, we recommend changing the settings and updating Creatio access permissions. Learn more in a separate guide: Access management.
- Lack of restrictions on **file upload** to Creatio. The files can be added by employees, uploaded as part of the email synchronization, or attached to self-service portal messages. We recommend limiting the size of uploaded files to 10 Mb. You can change this value in the “Attachment max size” (“MaxFileSize” code) system setting. We also recommend checking the uploaded files for relevancy and deleting the irrelevant files regularly. You can implement this feature using a business process.
- The **synchronization of all emails** from user mailboxes. We recommend selecting only the mailbox folders whose letters you must process in Creatio. For example, Important or Starred. Learn more in a separate article: Receive emails in Creatio.
- **Processes traced** for prolonged periods. Processes are usually debugged on a development environment or a test site. If you must collect the debugging information on a production site, we recommend turning the tracing off as soon as you finish analyzing the process execution problems. Learn more in a separate article: Trace process parameters.
- Incorrectly set up **business process execution logic** that causes the process to have the “Running” status for longer than needed. In this case, Creatio will keep the temporary files related to the process execution. We recommend modeling business processes so that they have definite completion conditions and do not have the “Running” status for longer than several hours. Learn more in a separate article: View process execution data.
- Incorrectly set up **data reading in business processes**. Creatio stores the values the business process retrieves as part of the [*Read data*] element execution in temporary data tables until the process finishes. If the business process does not require the values of all object columns, we recommend specifying the exact list of values to read. This will help you to reduce the amount of temporary data stored in Creatio greatly. Learn more in a separate article: [*Read data*] process element.
- Excessive **change logging**. We recommend turning on the change logging only for the sections in which you must track the data dynamics. For example, the product catalog. If you want to save the record change information, clean up the change log from the irrelevant data regularly. Learn more in a separate article: Clear the change log.
- Incorrectly set up **integration of external services** with Creatio. If the external services send a Creatio request that lacks the ForceUseSession header, they will need to re-authenticate. Learn more in developer documentation: EntityDataService.svc web service (OData 3).

- **Connecting unnecessary cultures**. This will cause Creatio to download translation files, which are then updated in each new Creatio version. We recommend connecting only those cultures that your company's employees use and removing any obsolete cultures.

Creatio database cleanup tools

There are several ways to clean up the drive space in Creatio:

- Configure the process log archival and automatic cleanup.
- Clear the change log.
- Delete section records.
- Delete data as part of a business process.

Automatic process log cleanup

Creatio can log the processes it runs. Logs help to track diagram bottlenecks and optimize them, as well as to analyze the efficiency of your employees. Creatio automatically deletes or archives the logs older than 30 days to reduce the amount of used space. The archived records remain available for the period specified in the “Archive data expiration period (days)” (“ProcessLogArchiveDataExpirationTerm” code) system setting before Creatio deletes the records automatically.

Ensure the archival is disabled to save the most space. If archival is required, you can manage when to archive and for how long to store the archived records. Learn more in a separate article: Archive the process log records.

Clear the change log

Clear the change log history to avoid storing irrelevant records in Creatio. We recommend clearing the change log regularly to ensure that the [*Change log*] section contains only the currently relevant information.

Learn more in a separate article: Clear the change log.

Delete section records

Creatio may store irrelevant section records. You can delete such records from any Creatio section individually or in bulk. If the record you want to delete is connected to other sections, Creatio will ask you to review the connections and confirm what to delete. You can delete all of the information or only the selected record and keep the connected data.

Learn more in a separate article: Delete linked records.

Delete data as part of a business process

Automate the drive cleanup with business processes. Use the [*Delete data*] process element to delete one or more records that meet specific conditions from any Creatio object. For example, create a business process that deletes all scheduled activities that were canceled. Set this process to run:

- **On a timer**, at a specific time. This is useful if you want to run the process periodically, e. g. once a month, and during the minimum load period, e. g. at night.

- **After a specific event**. This is useful if you want to run the process automatically and only when there is Creatio data to delete.
- **Manually**. This is useful if you want to run the process at any required moment.

Learn more in a separate article: [*Delete data*] process element.

Delete unused cultures

You can delete cultures that you no longer plan to use to free up additional disk space. To delete a culture, use the following script:

Microsoft SQL

```
IF OBJECT_ID('tempdb..#UsedCultures') IS NOT NULL
       DROP Table #UsedCultures
-- Get a list of used cultures
SELECT DISTINCT cult.Id
INTO #UsedCultures
FROM SysCulture cult
INNER JOIN SysAdminUnit au
       ON au.SysCultureId = cult.Id
INSERT INTO #UsedCultures
 (Id)
SELECT
 SysSettingsValue.GuidValue
FROM
 SysSettingsValue
INNER JOIN SysSettings
 ON SysSettings.Id = SysSettingsValue.SysSettingsId
WHERE
 SysSettings.Code = 'PrimaryCulture'

-- Get a list of tables from which to delete data
DECLARE TableNamesCursor CURSOR FOR
SELECT
        t3.TABLE_NAME AS ChildTableName
FROM INFORMATION_SCHEMA.REFERENTIAL_CONSTRAINTS AS t1
        INNER JOIN INFORMATION_SCHEMA.KEY_COLUMN_USAGE AS t2 ON t1.UNIQUE_CONSTRAINT_NAME = t2.C
        INNER JOIN INFORMATION_SCHEMA.KEY_COLUMN_USAGE AS t3 ON t1.CONSTRAINT_NAME = t3.CONSTRAI
WHERE
       t2.TABLE_NAME = 'SysCulture'
       and t2.COLUMN_NAME = 'Id'
       and t3.COLUMN_NAME = 'SysCultureId'
DECLARE @TableName SYSNAME
OPEN TableNamesCursor
FETCH NEXT FROM TableNamesCursor INTO @TableName
WHILE @@FETCH_STATUS = 0
BEGIN
```

```
        PRINT @TableName
        DECLARE @Sql NVARCHAR(MAX);
        SET @Sql = 'DELETE FROM ' + @TableName + '
             WHERE SysCultureId NOT IN (SELECT Id FROM #UsedCultures)';
        PRINT @Sql
        EXECUTE sp_executesql @Sql

        FETCH NEXT FROM TableNamesCursor INTO @TableName
END
CLOSE TableNamesCursor
DEALLOCATE TableNamesCursor
DELETE FROM SysCulture
WHERE Id NOT IN (SELECT Id FROM #UsedCultures)
IF OBJECT_ID('tempdb..#UsedCultures') IS NOT NULL
        DROP Table #UsedCultures
```

PostgreSQL

```
BEGIN;
-- Get a list of used cultures
CREATE TEMP TABLE "UsedCultures" ON COMMIT DROP AS
SELECT DISTINCT cult."Id"
FROM "SysCulture" cult
INNER JOIN "SysAdminUnit" au
ON au."SysCultureId" = cult."Id";
INSERT INTO "UsedCultures" ("Id")
SELECT "SysSettingsValue"."GuidValue"
FROM "SysSettingsValue"
INNER JOIN "SysSettings"
ON "SysSettings"."Id" = "SysSettingsValue"."SysSettingsId"
WHERE "SysSettings"."Code" = 'PrimaryCulture';
-- Get a list of tables from which to delete data
DO $$
DECLARE
TableNamesCursor REFCURSOR;
TableName varchar;
BEGIN
OPEN TableNamesCursor FOR
SELECT kcu.TABLE_NAME
FROM INFORMATION_SCHEMA.CONSTRAINT_COLUMN_USAGE ccu
INNER JOIN INFORMATION_SCHEMA.REFERENTIAL_CONSTRAINTS rc
    ON ccu.CONSTRAINT_CATALOG = rc.UNIQUE_CONSTRAINT_CATALOG
    AND ccu.CONSTRAINT_SCHEMA = rc.UNIQUE_CONSTRAINT_SCHEMA
    AND ccu.CONSTRAINT_NAME = rc.UNIQUE_CONSTRAINT_NAME
INNER JOIN INFORMATION_SCHEMA.KEY_COLUMN_USAGE kcu
    ON kcu.CONSTRAINT_CATALOG = rc.CONSTRAINT_CATALOG
    AND kcu.CONSTRAINT_SCHEMA = rc.CONSTRAINT_SCHEMA
```

```
    AND kcu.CONSTRAINT_NAME = rc.CONSTRAINT_NAME
WHERE ccu.COLUMN_NAME = 'Id'
  AND ccu.TABLE_SCHEMA = 'public'
  AND ccu.TABLE_NAME = 'SysCulture';

--Delete localizations
LOOP
FETCH TableNamesCursor INTO TableName;
EXIT WHEN TableName IS NULL;
RAISE NOTICE 'Deleting from table %', TableName;
EXECUTE format('DELETE FROM %I WHERE "SysCultureId" NOT IN (SELECT "Id" FROM "UsedCultures")', T
END LOOP;
CLOSE TableNamesCursor;
END $$;
DELETE FROM "SysCulture"
WHERE "Id" NOT IN (SELECT "Id" FROM "UsedCultures");
COMMIT
```

Process complex database queries faster

PRODUCTS: ALL CREATIO PRODUCTS

This functionality is available for Creatio version 7.18.4 and later.

Some Creatio database queries take a long time to process, which might affect page loading or task completion time significantly. Such queries are usually called “heavy.” They include:

- complex filters in pages and dynamic folders
- complex analytical selections in section dashboards
- complex custom queries implemented using development tools

You can accelerate the processing of heavy queries by forwarding them to a read-only database replica. This will reduce the load on the main database significantly and free up resources for the activity of users and the operation of other Creatio elements.

To set up the redirection of heavy queries, take the following steps:

1. Create a read-only database replica.
2. Configure access to the database replica in Creatio.

Step 1. Create a database replica

The procedure to create a database replica is DBMS-specific. Learn more about the process in vendor documentation:

- Create a database replica in PostgreSQL.
- Create a database replica in Microsoft SQL.
- Create a database replica in Oracle.

Step 2. Set up redirection of heavy queries

1. **Set up redirection** of heavy queries to the database replica. Perform the setup in the `Terrasoft.WebHost.dll.config` file for **Creatio .NET Core** and in the `web.config` file for Creatio **NET Framework**.

   a. Select the `UseQueryKinds` checkbox.

   ```
   <add key="UseQueryKinds" value="true" />
   ```

   b. Add the `replicaConnectionStringName="db_Replica"` value to the `db general` parameter.

   For Microsoft SQL

   ```
   <general connectionStringName="db" replicaConnectionStringName="db_Replica" securityEngine
   ```

   For PostgreSQL

   ```
   <general connectionStringName="db" replicaConnectionStringName="db_Replica" maxEntitySchema
   ```

2. **Configure access** to the database replica in Creatio. To do this, add the `db_Replica` parameter to the `ConnectionStrings.config` file:

   For Microsoft SQL

   ```
   <add name="db_Replica" connectionString="Data Source=[ Database server name ]; Initial Catalo
   ```

   For PostgreSQL

   ```
   <add name="db_Replica" connectionString="Server=[ Database server name ];Port=[ Database serv
   ```

For Oracle

```
<general connectionStringName="db" replicaConnectionStringName="db_Replica" currentSchemaName
```

For Oracle

```
<add name="db_Replica" connectionString="Data Source=(DESCRIPTION =
 (ADDRESS_LIST = (ADDRESS = (PROTOCOL = TCP)(HOST = [ Database server name ])(PORT = 1521))) (CON
```

Manage resource-intensive queries

PRODUCTS: ALL CREATIO PRODUCTS

This functionality is available for Creatio version 8.0.4 and later.

Execution of resource-intensive business operations (queries) can utilize a significant amount of database resources, which often leads to lower Creatio performance for all users. Such operations include:

- **Queries that filter strings by the “contains” mask**. For example, standard filter that has the “Name contains %Jane%” condition in the [*Contacts*] section.
- **Unfiltered, non-paginated queries** generated using OData service or server-side Creatio development tools. For example, "get all contacts" query executed when you have more than 20,000 contact records.
- **Queries sorted by a non-indexed column**.For example, [*Contacts*] section records sorted by the custom [*Account country*] column, for which no index has been created in the database.
- **Queries sorted by complex columns that contain a subquery**For example, [*Accounts*] section records sorted by the column with feedback: the sum of sales for the last year for sales related to the account.

To detect such queries, Creatio uses templates stored in the [*Query detector*] lookup.

You can apply multiple actions to resource-intensive queries to reduce the load on the database significantly and free up resources for user activity and operation of other Creatio elements. These actions are listed in the [*Query action*] lookup.

- **Limit query execution time**. Cancel the query after the time limit specified in the “Query execution timeout” (“QueryExecutionTimeout” code) system setting passes.
- **Logging**. Log only query data to the [*Query rule apply log*] lookup. This helps to track the problematic queries.
- **Limit the number of threads**. Limit the number of threads to execute the request to the value specified in the “MaxDopQueryHint thread count” (“MaxDopHintThreadsCount” code) system setting. We recommend

using a quarter to a half of total available threads, assuming one thread corresponds to one CPU core. This way, resource-intensive operations will be unable to take all available threads.

- **Cancel execution**. Cancel the query immediately.

Attention. We do not recommend changing the contents of the [*Query detector*] and [*Query action*] lookups. This can lead to violations of query handling rules.

Configure query handling rules

You can restrict the handling of heavy queries to improve database performance. To do this, define the type of the heaviest queries in your Creatio application and set rules that restrict their execution.

Note. A system administrator or user who has permissions to execute the “Can manage query rules” (“CanManageQueryRules” code) system operation can configure the rules.

The basic rule structure looks as follows:

```
If the query goes to the <name> table, the table contains more than <n> rows, and the query matc
```

To set up a rule:

1. Click → System Designer.
2. Click [*Lookups*].
3. Open the [*Query handle rule*] lookup.
4. Click [*Add*].
5. Fill out the rule parameters:
   a. Enter a unique rule name in the [*Name*] parameter. Required.
   b. Specify the schema (database table) to which the query must be executed in the [*Entity schema*] parameter. If no schema is specified, Creatio applies the rule to all schemas.
   c. Set the minimum number of table rows required for the rule to apply in the [*Min row count*] parameter. Minimum value: 1000. Required.
   d. Select the template of the query type to which to apply the rule in the [*Query Detector*] parameter. Available values are stored in the [*Query Detector*] lookup. Required.
   e. Select the action to apply to the query in the [*Query Action*] parameter. Available values are stored in the [Query Action] lookup. Required.

   Note. [*System*] checkbox determines the rule editability and priority. System rules can be configured only by cloud administrators in accordance with the cloud policies and restrictions. These rules are mandatory and cannot be modified or deleted by Creatio users.

f. Specify the rule availability in the [*Active*] — determines whether the rule is used or not.

g. Specify whether to log the Stack trace of the call of the operation that executes the query to which the rule was applied in the [*Log stack trace*] parameter. Logs are saved to the [*Query rule apply log*] lookup.

h. Specify whether to log the query execution time in the [*Log query execution time*] parameter.

6. Save the changes.
7. Repeat steps 4–6 for other needed rules.

Execute rules

Rule priorities are based on the detector, system character, schema (table), and number of table rows. When the query is executed, Creatio checks the following:

- query detectors used
- entity schema and row number
- whether [*System*] checkbox is set

If multiple rules apply to the same query detector and entity schema, Creatio applies only one system and one regular rule that match the most closely.

The examples below explain how rules affect the execution of resource-intensive queries. In the examples, the following set of rules is configured in Creatio.

Entity schema	Min number of rows	Query detector	Query action	Is System	Is Active
Like 1 rule					
Not set	20000	Text filter detector by pattern “%...”	Limit query execution time	No	Yes
Like 2 rule					
Not set	500000	Text filter detector by pattern “%...”	Cancel execution	No	Yes
Index 1 rule					
Not set	10000	Not indexed column sorting detector	Limit query execution time	No	Yes
Index 2 rule					
Not set	500000	Not indexed column sorting detector	Cancel execution	No	Yes
Activity Like 1 rule					
Activity	20000	Text filter detector by pattern “%...”	Limit query execution time	No	Yes
Activity Like rule					
Activity	1000000	Text filter detector by pattern “%...”	Cancel execution	No	Yes
Activity Like sys rule					
Activity	10000	Text filter detector by pattern “%...”	Limit the number of threads	Yes	Yes
Activity no fil no pag sys rule					
Activity	10000	Not indexed column sorting detector	Limit the number of threads	Yes	Yes

Example 1. The user searches in the [*Contacts*] section by filtering “[*Full name*] contains <Jane>” and

sorting by the non-indexed column [*Date of birth*]. The “Contacts” table has 100,000 entries.

The rule selection logic works as follows:

- The query fits two detectors: “Text filter detector by pattern "%..." and “Not indexed column sorting detector.”
- “Contact” entity schema has no specific rules.
- Fitting rules have no system rules.

As a result, the most suitable rule will be executed for each query detector.

Entity schema	Min number of rows	Query detector	Query action	Is System	Is Active
Like 1 rule					
Not set	20000	Text filter detector by pattern “%...”	Limit query execution time	No	Yes
Index 1 rule					
Not set	10000	Not indexed column sorting detector	Limit query execution time	No	Yes

Example 2. The user searches in the [*Activities*] section using the filter “[*Name*] contains <account>” and sorting by the non-indexed [*Created Date*] column. The “Activity” table has 120,000 entries.

The rule selection logic works as follows:

- The query fits two detectors: “Text filter detector by pattern "%..." and “Not indexed column sorting detector”.
- “Activity” entity schema has specific rules.
- Fitting rules include system rules.

As a result, the most suitable system rule and common rule will be executed for “Text filter detector by pattern "%..." query detector. Only the most suitable system rule will be executed for “Not indexed column sorting detector” because no common rules for this query detector were configured.

Entity schema	Min number of rows	Query detector	Query action	Is System	Is Active
Activity Like 1 rule					
Activity	20000	Text filter detector by pattern “%...”	Limit query execution time	No	Yes
Activity Like sys rule					
Activity	10000	Text filter detector by pattern “%...”	Limit the number of threads	Yes	Yes
Activity no fil no pag sys rule					
Activity	10000	Not indexed column sorting detector	Limit the number of threads	Yes	Yes

Analyze rules execution

Creatio stores the logged query execution data in the [*Query rule apply log*] lookup. For each of the configured rules, only the first operation for each of the users during the day is logged. This enables the administrator to identify Creatio bottlenecks and make appropriate changes based on the rules that are used most often.

In the log you can find the following information:

- [*Schema name*]. The root schema (database table) to which the request is made.
- [*User login*]. The user who initiated the request.
- [*Last detection date*]. Date of the last rule triggering on request.
- [*Query detector*]. Query type template for which the rule was applied.
- [*Query action*]. The action that was applied to the request.
- [*Message*]. Information that additionally reveals the reasons for triggering the rules: the number of records in the table, as well as by what criterion the trigger was triggered (for example, which heavy filter for which column).
- [*Recommendation*]. A recommendation to the system administrator how to reduce the impact of the query (for example, depending on the detector, add an index to the column, clear irrelevant/historical data in the table, strengthen the filter, etc.).
- [*Execution time, sec*]. A query execution time in database (to log this parameter, column [*Log query execution time*] of Query handle rule must be set up to “Yes”).

To view the SQL text and stack trace of the query, you can use the [*Show SQL text*] and [*Show stack trace*] buttons, which are available after clicking on the log entry. This information may be needed to analyze the request in more detail and determine the functionality that generated it. Stack trace logging is off by default. But you can turn it on in the [*Query handle rule*] lookup.

Limit the number of simultaneous DB queries

PRODUCTS: ALL CREATIO PRODUCTS

This functionality is available in Creatio version 8.0.2 and higher for MSSQL PostgreSQL DBMS.

You can improve application performance by limiting the amount of processor resources (threads) that a heavy database query can consume. This will reduce the impact of resource-intensive procedures on the work of users. Such requests are generated both in the background, for example, when executing business processes or integrations, or by users, for example, sorting registries by an aggregate column. This article describes how to configure restrictions for on-site applications. For cloud applications, these restrictions are applied automatically after upgrading to version 8.0.2.

The thread limiting mechanism works most effectively on servers with 4 or more processor cores.

The general procedure for setting restrictions consists requires steps:

1. Enable resource limit in configuration files. Read more >>>
2. Set resource limit in Creatio settings. Read more >>>

Enable resource limit for requests

Limiting the number of threads is disabled in Creatio by default. You can enable it in the ApplyMaxDopHintToUserQueries section of the configuration file:

- For **NET Framework**, these are web.config files located in the root folder of the application and in the TerrasoftWebApp folder.
- For **.NET Core**, this is the Terrasoft.WebHost.dll.config file located in the root folder of the application.

Configuration file line example

```
<configuration>
   …
     <appsettings>
       …
         <add key="ApplyMaxDopHintToUserQueries" value="true">
       …
     </appsettings>
   …
</configuration>
```

Set resource limit for requests

The system setting “Number of MaxDopQueryHint Threads” (code “MaxDopHintThreadsCount”) is intended to throttle the resources that can be used for queries to the database. To set a resource limit:

1. Open the system designer by clicking the button in the upper right corner of the application.
2. In the [*System settings*] group, click [*System settings*].
3. Open the “MaxDopQueryHint thread count” system setting (code “MaxDopHintThreadsCount”)..
4. In the [*Default value*] field, set the resource limit. We recommend using from a quarter to a half of the total number of available threads, assuming one thread corresponds to one processor core. In this way, you prevent all available threads from being occupied by resource-intensive operations.

> **Note.** The limit of half of the available number of threads allows you to perform all the necessary processes and tasks, while not slowing down the work of users. For the number of threads between 4 and 8 threads, we recommend setting the limit between 2 and 4 (i.e. half the number of available threads).

As a result, the load on the resources of the database server processors is significantly reduced, which allows you to perform resource-intensive operations without affecting the work of Creatio users.

Compile an app on a web farm

If you use a load balancer to ensure fault tolerance of your Creatio application, the processes that involve the app compilation differ. You must perform the setup on one Creatio instance, then transfer the settings to other instances.

Creatio runs the compilation automatically when you execute the following actions:

- set up record permissions
- set up the changelog
- activate an additional UI language
- install packages that contain changes created and tested on other environments into a production environment
- install a Marketplace application.

> **Note.** We recommend performing the setup outside business hours since Creatio will be unavailable.

View the general procedure to follow when Creatio compiles the app automatically below. The specifics might vary based on the load balancer you use with your Creatio application.

General setup procedure

1. **Set up a redirect** to the placeholder page that explains the reasons for the downtime in the load balancer. Learn more about setting up redirects in the vendor documentation. For example, HAProxy Enterprise.
2. **Stop all Creatio instances** except one. Learn more in the Update guide: Stopping the site.

3. **Back up** the database. Learn more in the Update guide: Creating database backup.
4. Perform the required process.
5. **Apply the changes** on the web farm level. To do this, copy the contents of the following folders to every Creatio instance:
   - WebApp/conf/ and WebApp/Terrasoft.Configuration/Pkg for Creatio **NET Framework**
   - Creatio root folder for Creatio **.NET Core**
6. **Flush** Redis.
7. **Start** the stopped Creatio instances. Learn more in the Update guide: Website starting, compilation and verification.
8. **Disable the redirect** in the load balancer.