DB2/400 to MySQL

A Technical Migration Reference

Dr. Bruce Acacio
Infinite Corporation · June 2026

Abstract

This reference describes the migration of an IBM i (DB2 for i / DB2/400) database to MySQL as part of a rehosting program. It covers the structural correspondence between DB2 for i and MySQL, a complete data-type mapping, the conversion of EBCDIC-encoded data to Unicode, the preservation of referential integrity, keys, and indexes, and the treatment of logical files as views and secondary indexes. A migration workflow with end-to-end validation is presented, together with guidance on deployment targets across the three major clouds. The objective is a faithful, production-grade target schema that requires no change to the migrated application's data-access code.

Keywords: *DB2 for i, DB2/400, MySQL, schema migration, EBCDIC, Unicode, referential integrity, data types*

1. Introduction

When an IBM i estate is rehosted, the application source is recompiled largely unchanged, but the data it depends on must be carried faithfully to the target environment. Where the target database is MySQL, the migration must reproduce not only the rows but the structure, constraints, encoding, and semantics of the original DB2 for i schema, so that the recompiled RPG and COBOL programs read and write exactly as before. This document is a practitioner's reference for that task.

The guiding principle is fidelity: the migrated schema should require no change to data-access code. Every decision below - type mapping, encoding, key and index treatment - serves that goal.

2. The Source: DB2 for i Structures

DB2 for i organizes data as physical files and logical files. A physical file holds the data itself and corresponds closely to a table; a record format defines its columns. A logical file presents an alternative view or access path over one or more physical files - it may reorder or subset columns, join files, or simply provide a keyed access path - and corresponds to a view or a secondary index in relational terms. Files may contain multiple members, and data is frequently encoded in EBCDIC with packed and zoned decimal numerics.

3. The Target: MySQL Schema Model

In MySQL, each physical file becomes a table; record-format fields become columns; keyed access paths become primary keys and indexes; and logical files become either views (where they reproject or join) or secondary indexes (where they exist only to provide an access path). The default character set for the target is

utf8mb4, ensuring full Unicode coverage including supplementary characters. The storage engine is InnoDB, which provides the transactional semantics and foreign-key enforcement required to preserve referential integrity.

4. Data-Type Mapping

The following mapping reproduces DB2 for i types in MySQL with exact precision. The cardinal rule for numerics is that packed and zoned decimals map to DECIMAL with identical precision and scale - never to floating-point types, which would silently introduce rounding error in financial calculations.

DB2 for i type	MySQL type	Notes
CHAR(n)	CHAR(n) / VARCHAR(n)	Fixed CHAR for short codes; VARCHAR where trailing blanks are insignificant.
VARCHAR(n)	VARCHAR(n)	Length preserved; charset set to utf8mb4.
GRAPHIC / VARGRAPHIC	NCHAR / NVARCHAR	DBCS data mapped to Unicode columns.
DECIMAL(p,s) (packed)	DECIMAL(p,s)	Exact precision and scale preserved; no float conversion.
NUMERIC(p,s) (zoned)	DECIMAL(p,s)	Zoned decimal decoded to exact DECIMAL.
INTEGER / SMALLINT	INT / SMALLINT	Direct mapping.
BIGINT	BIGINT	Direct mapping.
DATE / TIME	DATE / TIME	Format normalized to ISO 8601.
TIMESTAMP	DATETIME(6)	Microsecond precision retained.
BLOB / CLOB	LONGBLOB / LONGTEXT	Large objects migrated with charset handling for CLOB.

Table 1. Representative DB2 for i to MySQL type mapping.

5. Character Encoding: EBCDIC to Unicode

IBM i stores character data in EBCDIC, identified by a coded character set identifier (CCSID); MySQL expects Unicode. The migration performs a CCSID-aware conversion - for example, from CCSID 037 - to UTF-8 (utf8mb4), correctly handling national-language variants and the punctuation and currency characters that naive byte-level conversion corrupts. Packed-decimal and binary fields are decoded according to their definitions rather than treated as text, and null-capable fields are preserved as nullable columns. The test of a correct conversion is that accented and multi-byte characters survive a round trip unchanged.

6. Referential Integrity, Keys, and Indexes

Primary keys defined by keyed physical files are recreated as MySQL primary keys. Referential constraints - foreign keys, with their delete and update rules - are reproduced on InnoDB so that the database itself continues to enforce the relationships the application assumes. Unique constraints and check constraints are migrated where MySQL supports them. The result is that integrity is guaranteed by the target database, not merely by application convention.

7. Logical Files as Views and Indexes

Logical files require judgment. A logical file that reprojects or joins physical files is best reproduced as a MySQL view, preserving the column set and semantics the program expects. A logical file that exists only to provide a keyed access path is reproduced as a secondary index on the underlying table, with no view required. Distinguishing the two correctly avoids both redundant objects and missing access paths.

8. Migration Workflow and Validation

A migration proceeds in stages: schema extraction from DB2 for i; automated generation of the MySQL DDL using the mapping above; an initial bulk load with encoding conversion; and, where coexistence is required, an incremental synchronization phase. Validation is non-negotiable. Row counts are compared table by table, and column checksums are computed on both sides to confirm value-level fidelity. Exceptions are published to an operational report that serves as the audit evidence of a faithful migration.

9. Deployment Targets

MySQL is deliberately the single target. It is the most widely deployed open-source database, runs natively on all three major clouds, and presents a clean, well-understood platform. Managed options include Amazon RDS and Aurora MySQL-compatible, Azure Database for MySQL Flexible Server, and Google Cloud SQL for MySQL; self-managed MySQL on the organization’s own virtual machines is equally supported. Because the migrated database is reached through standard SQL interfaces, the choice among targets is governed by the enterprise’s existing cloud agreements rather than by the migration technology.

10. Conclusion

A DB2 for i to MySQL migration succeeds when the target is indistinguishable to the application from the source: same structures, same precision, same integrity, same characters. Achieving that requires disciplined type mapping, encoding-aware conversion, faithful reproduction of keys and constraints, and rigorous validation. Done well, the data moves to a modern, portable platform while the recompiled applications continue to run exactly as they always have.

References

[1] Infinite Corporation. *Infinite DataLink: DB2/400 to MySQL Migration.* Product documentation, 2026.

[2] IBM. *DB2 for i SQL Reference.* IBM Corporation, 2025.

[3] Oracle. *MySQL 8.0 Reference Manual.* Oracle Corporation, 2025.

[4] Acacio, B. *Integrating Rehosted IBM i RPG Applications with the Mambu Cloud Banking Platform on Google Cloud.* Infinite Corporation, June 2026.

[5] IBM. *Character Data Representation Architecture (CDRA) and CCSID Reference.* IBM Corporation, 2024.