

PIX2DB: Reconstructing Executable Relational Databases from Document Images with a 2B Vision-Language Model

Musa Peker

pekermusa@gmail.com

Özdilek <https://orcid.org/0000-0002-6495-9187>

Research Article

Keywords: table extraction; document understanding; visionlanguage models; image-to-SQL; relational schema reconstruction; synthetic data; receipts and invoices.

Posted Date: October 23rd, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-11467404/v1>

License: © ⓘ This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Additional Declarations: The authors declare no competing interests.

PIX2DB: Reconstructing Executable Relational Databases from Document Images with a 2B Vision-Language Model

Musa Peker

Ozveri R&D Center, Ozdilek Home Textile Industry and Trade Inc., Bursa, Türkiye

msapeker@gmail.com

Abstract

Tables trapped in screenshots, scans, receipts and charts are usually extracted as markup that keeps the layout but not the relational meaning: merged cells stay merged, an invoice header and its line items stay unrelated, and “\$1,250.00” stays a string. We study *image-to-database reconstruction*: given one image, produce a SQLite dump whose schema, column types, keys and rows load and can be queried as they are. Training data with exact gold databases is obtained in two ways: by rendering samples of the Spider databases at six layout levels, and by converting PubTabNet, FinTabNet and CORD annotations under the same normalization rules. A 2B model (Qwen3-VL-2B with LoRA) trained on 36,589 pairs and followed by a deterministic verify-and-repair layer reaches 0.86 table exact match and 0.99 foreign-key recall on rendered tables from unseen databases, against 0.75/0.75 for the 8B model of the same family prompted for SQL and 0.47/0.00 for an 8B HTML-to-SQLite pipeline. On held-out real images it obtains cell F1 0.81 (PubTabNet) and 0.96 (FinTabNet) and recall 0.93 on CORD receipt photos. On TabRecSet, a source not used in training, it is the best system that writes a database directly (0.52) but trails the 8B HTML pipeline (0.58), mostly in reading text from photographs. A variant trained on synthetic data alone matches the synthetic score and fails on real images, which locates the value of the real data. Code, model, data and all evaluation outputs are public.

Keywords: table extraction; document understanding; vision-language models; image-to-SQL; relational schema reconstruction; synthetic data; receipts and invoices.

1 Introduction

A spreadsheet exported as a picture, a scanned statement, the photo of a receipt: in each case the data has a relational structure that the image hides. Document understanding systems recover this structure as cell grids [21, 26, 34], as HTML or Markdown produced by OCR-free models [3, 7, 10, 23, 28], or as linearized tables for charts [14]. These outputs describe how the table *looks*. Turning them into something a program can join, sum or filter still requires decisions that the markup leaves open: which rows a merged cell belongs to, how a two-level header becomes a column name, whether “(1,209)”

Users Details			
USER ID - 10	AGE CATEGORY CODE - 18-25	USER CATEGORY CODE - Single Mother	
USER ADDRESS ID - 1	LOGIN NAME - in	DATE REGISTERED - 1980-07-04 17:28:15	

Properties		Figures	
Date on market	Date off market	Owner user id	Property type code
2002-01-06 18:44:23	1978-09-24 13:42:49	10	2
2011-12-15 01:52:28	2017-05-27 20:55:25	10	2
1976-08-01 00:52:48	1970-11-22 02:09:50	10	2


```
CREATE TABLE users (user_id INTEGER PRIMARY KEY,
age_category_code TEXT, user_category_code TEXT,
user_address_id INTEGER, login_name TEXT,
date_registered TEXT);
CREATE TABLE properties (date_on_market TEXT,
date_off_market TEXT, owner_user_id INTEGER,
property_type_code INTEGER,
FOREIGN KEY (owner_user_id) REFERENCES users(user_id));
INSERT INTO users VALUES
(10, '18-25', 'Single Mother', 1, 'in', '1980-07-04 17:28:15');
INSERT INTO properties VALUES
('2002-01-06 18:44:23', '1978-09-24 13:42:49', 10, 2), ...
```

Figure 1: A rendered document from an unseen database and the dump written by PIX2DB (abridged). The key–value header becomes its own table, the merged *Owner Id* cell is repeated in every item row, and the relation between the two blocks is stated as a foreign key.

is a negative number, and which of two tables on a page refers to the other.

We make these decisions part of the model’s output. Given an image, PIX2DB writes a SQLite dump (CREATE TABLE with types, primary and foreign keys, followed by INSERT statements) that is executed by an unmodified SQLite engine (Figure 1). The closest prior work, OptiSQL [13], generates one SQL *query* per question from a clean table image; we instead reconstruct the database once, after which every query is ordinary SQL.

The task is not solved by prompting. In a pilot on 150 rendered images, Qwen3-VL-8B [2] asked for a SQL dump produced executable output 93% of the time but a fully correct table only 61% of the time, and the same model asked for HTML that a 20-line script loads into SQLite reached 46%, failing on exactly the merged cells and forms that motivate the task. The errors were rarely misread characters. The models placed merged values in one row, invented header tables, dropped foreign keys, or wrote O’ Brien unescaped so that one bad row voided an entire INSERT.

Contributions.

- A task definition for image-to-database reconstruction

with explicit normalization conventions, and an evaluation protocol that aligns predicted and gold databases without relying on table or column names (§3).

- A data recipe that yields exact gold databases at scale: rendering samples of real relational databases at controlled layout difficulty, and converting existing table and receipt annotations into the same conventions, with every gold dump verified by execution (§4).
- PIX2DB-2b, a 2B model with a deterministic verify-and-repair layer, which outperforms $4\times$ larger models of its family on five of six evaluation sets and recovers foreign keys almost perfectly (§6).
- An analysis of what transfers: synthetic layouts teach normalization but not reading, a modest converted real set closes most of the real-image gap, and on a source held out from training the remaining gap is reading accuracy rather than structure (§6–7).

2 Related work

Table structure recognition. TSR predicts the cell grid of a cropped table, evaluated with the tree-edit-distance score TEDS [34]. Large annotated corpora come from scientific articles [4, 26, 34] and financial reports [33], and photographed tables are covered by WTW [16] and TabRecSet [30]. TableFormer [21] and the PubTables-1M detector [26] predict structure and rely on PDF text or OCR for content. A grid is a faithful description of the layout, not of the data: TEDS rewards reproducing a rowspan, whereas a database must resolve it.

OCR-free document models. Donut [7] and Pix2Struct [10] showed that an image-to-sequence model can parse documents without an OCR stage; Nougat [3], GOT [28], olmOCR [23] and Docling [1] convert pages to Markdown, LaTeX or HTML. General VLMs such as Qwen3-VL [2] read tables well enough to be our strongest baselines. None of these targets a typed relational schema or relations between tables.

Charts. DePlot [14], MatCha [15] and UniChart [18] translate plots into linearized tables, with ChartQA [17] and PlotQA [20] as standard benchmarks. We treat a chart as one more layout whose target is a typed table.

Tables, images and SQL. Text-to-SQL benchmarks [11, 12, 31] assume the database already exists. Visual table QA [19, 24, 32] answers questions over images, and recent analyses report that VLMs degrade sharply on noisy real-world tables [25]. OptiSQL [13] renders Spider 2.0 tables and generates an executable query from the image and a question; its gold database is never reconstructed. MTabVQA [24] renders Spider and BIRD tables for multi-table QA. Our use of Spider is similar in spirit, but the database itself is the output. Rendering synthetic training data with free and exact labels follows a long line of work in OCR [5].

	<i>typed schema</i>	<i>multi-table + FK</i>	<i>spans resolved</i>	<i>charts</i>	<i>photos, receipts</i>	<i>executable DB</i>
TSR models [21, 26]	×	×	×	×	○	×
Page-to-markup [1, 3, 23]	×	×	×	×	○	×
Donut-style parsing [7]	○	×	×	×	✓	×
Chart-to-table [14, 18]	×	×	–	✓	×	×
OptiSQL [13]	–	×	×	×	×	×
PIX2DB (ours)	✓	✓	✓	✓	✓	✓

Table 1: Capabilities as described in the respective papers. ○: partly, or for a subset of inputs. Donut-style parsers emit task-specific JSON fields; OptiSQL emits a query against an existing database.

Table 1 summarizes the differences.

3 Task and evaluation

Output. For an image I the system outputs a SQL text D ; the prediction is the database obtained by executing the CREATE TABLE and INSERT statements of D in an empty SQLite instance (other statements are ignored). The target follows fixed conventions: one row per record, with the value of a merged cell repeated in every row it covers; multi-level headers joined into one snake_case name; numbers stored without thousand separators or currency signs, accounting negatives as negative numbers; dates in ISO form; empty cells as NULL; derived totals rows not stored; one table per logical block, with a foreign key from line items to their header; for charts, the category and one column per series. Table and column names are chosen by the model and are not scored.

Alignment. Cell values are normalized (Unicode NFKC, dash variants unified, numbers parsed after removing separators and currency, common date formats mapped to ISO, strings case-folded); empty values are ignored. For a gold table G and a predicted table P , columns are aligned by a maximum-weight bipartite matching whose weights are the multiset overlap of normalized column values, and rows by a second matching whose weights count agreeing aligned cells [8]. The resulting number of agreeing cells $a(G, P)$ is used to match gold to predicted tables, again by maximum-weight assignment.

Metrics. *Cell precision, recall and F1* divide the total of a over matched tables by the number of non-empty predicted and gold cells. *Table exact match* requires a matched table with the same number of rows and columns and every gold cell recovered. *FK recall* is the fraction of gold foreign keys whose child and parent tables are matched to predicted tables connected by a declared foreign key. *Type accuracy* counts an aligned column as correct when its declared SQLite affinity and the stored values agree with the gold column being numeric or textual. Two cases are relaxed: values of charts

drawn without value labels match within $\tau = 0.04 \cdot \max |v|$, and receipts are scored by recall, because a correct database may legitimately contain an extra table for totals or payment.

4 Data

4.1 Rendering databases

A *scene* is a small database cut from one of the Spider [31] (CC BY-SA 4.0) SQLite files: a single table with 3–25 sampled rows and 3–9 columns, two unrelated tables, or a parent row together with 3–15 child rows that reference it through a declared foreign key. Columns with long or mostly empty values are dropped, numeric strings and integer-valued floats are coerced to numbers, and composite primary keys are not declared.

Scenes are rendered with HTML/CSS in headless Chromium at six levels (Figure 2). **L1** draws a plain grid with raw column names. **L2** adds 17 font families, colored or zebra styling, humanized headers in four case styles, thousand separators (never on identifiers or years), currency prefixes on money-like columns and long-form dates. **L3** adds merged cells on a repeated grouping column, two-level headers that group textual and numeric columns, transposed layouts for small tables and occasional derived totals rows. **L4** renders a parent row as a key–value block above the child table. **L5** photographs L3 and L4 pages with rotation up to $\pm 2.5^\circ$, uneven lighting, sensor noise, blur, contrast change, optional grayscale and JPEG quality 45–80. **L6** draws bar, horizontal bar, line and pie charts with one to three series, printing value labels for 60% of charts whose labels are exact. Synthetic receipts use monospaced narrow layouts with three item-line formats, modifiers such as “less sugar”, subtotal, tax and payment lines, and are photographed 85% of the time.

Two invariants keep the gold exact. Every displayed string maps back to the gold value without loss (formatting adds separators or symbols, never rounds or truncates). And because rendering may reorder rows and columns, for example to group merged cells, the gold scene is reordered the same way, so that the target dump reads in the order of the image. Splits are made by database: the 146 Spider training databases feed training, the 20 development databases validation, and the 40 databases that appear only in Spider’s test set our synthetic test.

4.2 Converting real annotations

PubTabNet [34] and FinTabNet [33] (both CDLA-Permissive-1.0) annotate table images with HTML. We expand spans into a grid, flatten header rows into names, repeat body rowspans, convert numbers as above and map lone dashes to NULL. In the HTML release of FinTabNet we used, tables carry no `thead`; the first row is used as header, and the second as well when the first contains group colspans. We use the original FinTabNet annotations, not the corrected FinTabNet.c [27]. Tables whose body contains colspans, typically section rows such as “Assets”, have no unambiguous relational form and are

Split	images	tables/img	rows	cols	with FK
train (synthetic)	24,000	1.38	7.5	3.7	29%
train (real)	18,794	1.00	9.4	4.6	0%
train (receipts)	3,000	1.00	7.1	3.5	0%
validation	600	1.30	7.7	3.7	22%
test (synthetic)	900	1.32	7.0	3.4	24%
test (receipts)	150	1.00	6.8	3.5	0%
PubTabNet (val)	150	1.00	9.4	4.2	0%
FinTabNet (val)	150	1.00	8.5	4.6	0%
CORD (test)	60	1.00	3.5	3.0	0%
TabRecSet (unseen)	150	1.00	6.7	3.8	0%

Table 2: Splits. Rows and columns are averages per table; “with FK” is the share of images whose gold contains a foreign key. All splits are released.

rejected, as are tables with more than 30 rows or 10 columns. Of the streamed training tables, 72% of PubTabNet and 69% of FinTabNet pass. CORD [22] (CC BY 4.0) receipts become an `items` table (name, quantity, price) built from the menu annotations.

Execution check. Every gold dump, synthetic or converted, is executed and scored against itself; any score below 1.0 is a bug in the data. The check found composite keys declared as primary keys, numbers stored as text by Spider, identifiers that start with a digit or are SQL keywords (now quoted), empty strings in numeric columns, and a header-deduplication collision in one converted table, which was removed.

4.3 Evaluation sets

Besides the synthetic test (150 images per level), we use held-out splits of the real sources: 150 PubTabNet and 150 FinTabNet validation tables and 60 CORD test receipts with at least two items, plus 150 synthetic receipts. These images are unseen, but their style is familiar from training. To measure transfer we also build **TabRecSet-150** from TabRecSet [29, 30] (CC BY 4.0), which we did not use for training; since it has been public since 2022, it may have been part of the base model’s pre-training. We take English original images (not the generated three-line and no-line variants, which repeat the same tables), one table per image, cropped by its cell polygons, with at least 60% non-empty cells. TabRecSet does not mark header rows, so the first grid row is used as header; 64% of the inspected tables pass the same conversion rules. Its annotators wrote illegible characters as #; we keep them in the gold (94 of 4,199 cells, in 26 tables). Removing these cells from scoring changes cell F1 by less than 0.01 for every system and leaves the ranking unchanged. Some images contain handwriting, and three cells contain Chinese characters; neither was filtered. Table 2 lists all splits.



Figure 2: One rendered example per synthetic level, all from the validation databases. Gold databases for these images are exact by construction; for the receipt it is a single `items` table, and the subtotal, tax and payment lines are not stored.

5 Model and repair layer

Model. PIX2DB-2b is Qwen3-VL-2B-Instruct [2] with LoRA [6] (rank 64, $\alpha = 128$, dropout 0.05) on all attention and MLP projections of the language model; the vision encoder is frozen. Images are resized to at most 1.31 MP (about 1,280 visual tokens). The loss covers only the answer, a fenced SQL block. Training mixes 14,000 synthetic images (all six levels), 3,000 synthetic receipts, 9,999 PubTabNet and 8,000 FinTabNet tables, and 795 CORD receipts seen twice: 36,589 examples, one epoch, effective batch 16, learning rate 10^{-4} with cosine decay and 3% warm-up, bf16, 2,287 steps in 5.9 hours on one A40. The adapter is merged into the base weights for release. A per-device batch of 8 ran out of memory on long batches because of the full vocabulary logits, so we used 4 with gradient accumulation of 4.

Verify and repair. Model output passes through a deterministic layer before it is returned. Statements are executed one at a time; a failing statement is retried after escaping apostrophes inside words, quoting or rewriting identifiers that are SQL keywords or free text (# of sequences¹ $\rightarrow$ of_sequences), and, for INSERTs, inserting row by row with bare composite values (9 (8), 85.0 ± 10.8) quoted as text and missing trailing cells padded with NULL. A primary key that rejects duplicate values is dropped rather than the rows. Truncated output keeps its complete tuples. Remaining problems are reported, not hidden: empty tables, values that contradict their declared type, duplicate rows and foreign keys without a parent. Statements that execute are left as they are, except that a primary key is dropped when later rows violate it.

6 Experiments

Baselines. We compare with Qwen3-VL-2B, 4B and 8B-Instruct given the same instruction as PIX2DB (Appendix A),

and with a two-stage pipeline in which Qwen3-VL-8B outputs HTML tables that keep row and column spans, and a script loads them with `pandas.read_html` (which expands spans) into SQLite. All SQL outputs, ours and the baselines', go through the same repair layer. Decoding is greedy with at most 3,072 new tokens, served by vLLM [9]. 95% confidence intervals come from 2,000 bootstrap resamples of evaluation items, and p -values from a one-sided paired bootstrap against the strongest competitor on each set.

6.1 Main results

Table 3 gives the overall picture. Fine-tuning moves the 2B model from the bottom of the table to the top on every set except TabRecSet. On the synthetic test it gains 33 points of exact match over its own base model and 11 over the 8B model; on FinTabNet and CORD the margins over the strongest competitor are 9 and 14 points. PubTabNet is the closest case: the HTML pipeline, which is well suited to flat scientific tables, is 3 points behind and the difference is not significant at the 5% level. Foreign keys separate the systems most clearly. PIX2DB declares 0.99 of the gold relations, the 8B model prompted for SQL 0.75, the smaller models a quarter, and the HTML pipeline none, because nothing in HTML expresses a relation between two tables.

6.2 Where layout matters

Figure 3 breaks the synthetic test down by level. On flat tables (L1, L2) every system is within a few points of the others, and the HTML pipeline reaches 0.85 and 0.84. With merged cells, key-value headers and photographed pages it falls to 0.18, 0.21 and 0.16, whereas PIX2DB stays at 0.89, 0.95 and 0.90. The HTML stage itself is mostly right; the failure is in the conversion, which has no notion that a rowspan is a grouping key or that a form above a table is a separate record. Charts are the exception to the pattern: apart from the untuned 2B model, all systems lie between 0.57 and 0.63 exact match, and

System	Synthetic exact	PubTabNet f1	FinTabNet f1	CORD recall	Synth. receipts recall	TabRecSet f1	FK recall
pix2db-2b (ours)	0.86 [0.83,0.88]	0.81 [0.77,0.84]	0.96 [0.95,0.98]	0.93 [0.89,0.96]	1.00 [1.00,1.00]	0.52 [0.47,0.57]	0.99
Qwen3-VL-8B → SQL	0.75 [0.73,0.78]	0.52 [0.47,0.57]	0.59 [0.54,0.64]	0.76 [0.69,0.83]	0.87 [0.84,0.90]	0.45 [0.40,0.50]	0.75
Qwen3-VL-8B → HTML → SQLite	0.47 [0.44,0.50]	0.78 [0.75,0.82]	0.87 [0.84,0.89]	0.44 [0.36,0.52]	0.38 [0.34,0.42]	0.58 [0.53,0.63]	0.00
Qwen3-VL-4B → SQL	0.67 [0.63,0.69]	0.48 [0.42,0.53]	0.48 [0.42,0.54]	0.79 [0.72,0.85]	0.77 [0.73,0.81]	0.38 [0.33,0.44]	0.25
Qwen3-VL-2B → SQL (base)	0.53 [0.50,0.56]	0.30 [0.25,0.36]	0.36 [0.31,0.41]	0.59 [0.49,0.68]	0.67 [0.62,0.72]	0.31 [0.26,0.36]	0.23

Table 3: Main results with 95% bootstrap intervals. Synthetic: table exact match on 900 images from unseen databases; PubTabNet, FinTabNet, TabRecSet: cell F1; CORD and synthetic receipts: cell recall; FK recall on the synthetic test images that contain a foreign key. Against the best competitor on each set, PIX2DB is ahead with $p < 0.001$ on synthetic, FinTabNet, CORD and synthetic receipts and with $p = 0.057$ on PubTabNet; on TabRecSet the 8B HTML pipeline is ahead by 0.06 (paired 95% interval of the difference [0.03, 0.09], $p < 0.001$; Wilcoxon signed-rank $p < 0.001$), although the marginal intervals overlap.

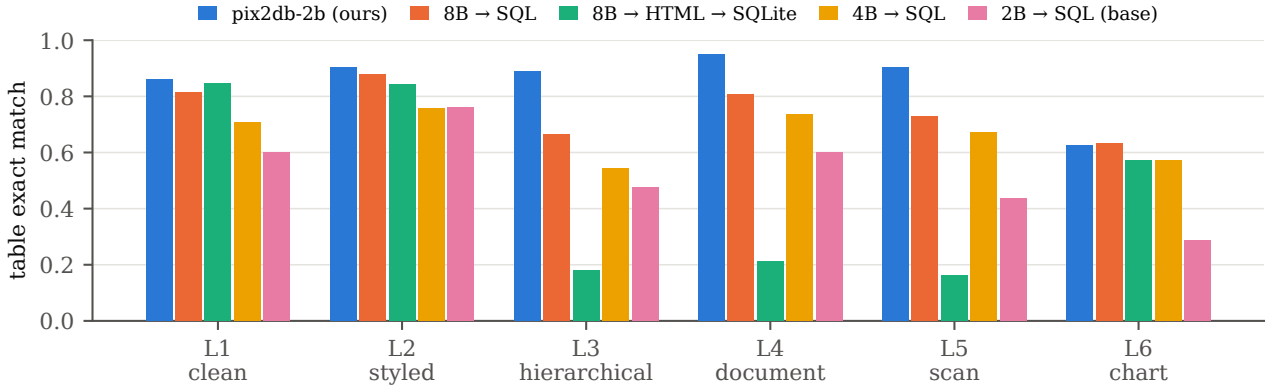


Figure 3: Table exact match per synthetic level (150 images each). The HTML pipeline is competitive on flat tables and loses most of its accuracy once merged cells, forms or scanning are involved.

PIX2DB (0.63) is not better than the 8B model. Charts without value labels are harder (0.52 exact against 0.69 with labels), but labeled charts still fail in almost a third of cases, so the difficulty lies as much in assigning marks to categories and series as in estimating values.

6.3 What the real data contributes

We trained a first version, otherwise identical, on 24k synthetic images only. Figure 4 shows that it reaches the same synthetic accuracy (0.85 vs. 0.86) and transfers poorly: cell F1 0.43 on PubTabNet, 0.45 on FinTabNet, recall 0.49 on CORD. Its failures on PubTabNet were rarely layout errors. In 82 of 150 outputs the SQL did not execute at all (28 after repair), mostly because composite values such as 9 (8) or 85.0 ± 10.8 were written as bare numbers, a format that never occurs in Spider, and in others the model looped on a CREATE TABLE it could not finish. Adding 18.8k converted real tables and receipts fixes this without costing synthetic accuracy and lifts TabRecSet from 0.44 to 0.52. The synthetic data teaches the conventions; the real data teaches what real cell contents look like.

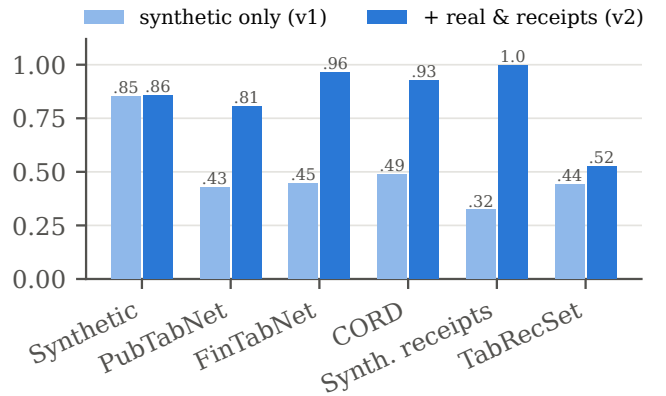


Figure 4: Same model and recipe, trained on synthetic data only (v1) or with the converted real tables and receipts added (v2). Metrics as in Table 3.

6.4 Effect of the repair layer

Figure 5 pools all six sets. The layer raises the share of outputs that execute from 0.96 to 0.99 for PIX2DB and from 0.78 to 0.87 for the untuned 2B model, but it moves pooled cell F1 by at most four points. Its value lies in the tail: on TabRecSet, where PIX2DB is out of distribution, execution rises from 0.91 to 0.99, and on PubTabNet F1 improves by 3 points (Appendix, Table 5). Repair is no substitute for training. Applied to the

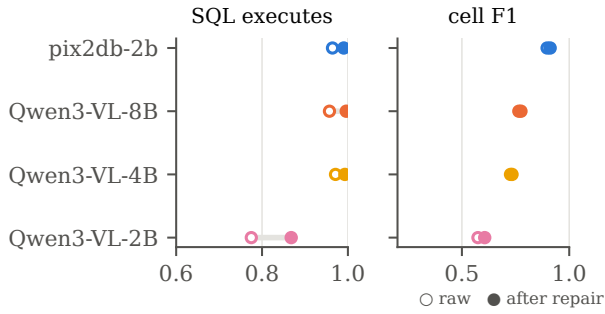


Figure 5: Share of executable outputs and cell F1 before and after the repair layer, pooled over all evaluation sets.

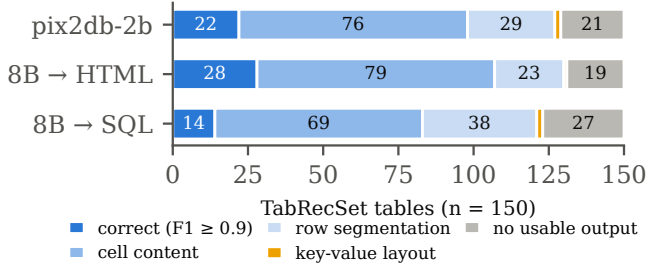


Figure 6: Outcome categories on TabRecSet-150, a source not used in our training. The thin orange segment is the key–value category (2, 1 and 2 tables from top to bottom); each row sums to 150.

base model it leaves F1 at 0.61, far below the fine-tuned model without repair (0.90).

6.5 Unseen source: TabRecSet

TabRecSet-150 is the only set on which PIX2DB loses (0.52 vs. 0.58). We sorted each prediction into one category (Figure 6): F1 of at least 0.9; usable output with the right number of rows but wrong cell contents; a wrong number of rows; a two-column key–value table; or no usable output. Key–value layouts, where our first-row-as-header gold is questionable, account for only 2 tables and do not explain the gap. The largest bucket for all systems is cell content (76 for PIX2DB, 79 for the HTML pipeline): the rows are found but text in photographed, skewed or low-resolution tables is misread. PIX2DB also segments rows wrongly more often (29 vs. 23). The 8B model reads these images better than the 2B model it is compared with, and the HTML route spends no capacity on normalization. On this source the bottleneck is reading, which points to a larger base model as the next step rather than more synthetic layouts.

6.6 Types and cost

Table 4 adds column types and output size on the synthetic test. Stored types are almost always right for every SQL system; the HTML pipeline gets 0.89 because pandas infers types from the formatted strings, so “\$1,250” stays text. PIX2DB writes the shortest outputs (195 tokens on average, against 405 for HTML), which together with its size makes it the fastest

System	types	exec	tokens	s/img
pix2db-2b (ours)	0.99	0.96	195	0.09
8B → SQL	0.98	0.95	265	0.29
8B → HTML → SQLite	0.89	–	405	0.34
4B → SQL	0.98	0.99	207	0.18
2B → SQL (base)	0.98	0.83	274	0.13

Table 4: Synthetic test. Type accuracy over aligned columns (after repair), share of raw outputs that execute, mean output tokens and amortized seconds per image with vLLM on one A40.

system: 0.09 s per image amortized over a vLLM batch on one A40. Training and every evaluation in this paper used about 12 A40 GPU-hours.

7 Discussion

Conventions are cheap to learn, reading is not. The decisions that turn a layout into a database are few and regular, and general VLMs know them but apply them inconsistently: the 8B model declares three quarters of the foreign keys and merges cells correctly about two times in three. A small model trained on a few tens of thousands of exact examples applies them almost every time. Reading text is different. The synthetic-only model read rendered text perfectly and real text poorly, and on TabRecSet a larger general model still reads better than our fine-tuned small one. We expect the recipe to carry over to a 4B or 8B base with little change, and the TabRecSet gap to shrink accordingly.

The HTML route is a strong baseline for flat tables. When a table has one header row and no merged cells, “copy the grid, let a script load it” is hard to beat, and our PubTabNet result is statistically tied with it. Its weaknesses are structural: it cannot express relations, it inherits pandas’ type inference, and it has no answer for rowspans that encode grouping. Systems that output markup should be evaluated with these cases in view, which TEDS-style metrics do not reward.

Rendering from databases. Starting from a database and producing the image inverts the usual annotation effort: labels are exact, difficulty is a parameter, and the same scene can be shown with and without merged cells or scanning. The cost is a narrow distribution of typography and cell contents, which the real-data ablation makes visible. Converting existing annotations was the cheapest way to widen it; the converter is about 150 lines of code and rejects roughly 30% of the tables it sees.

Validity of the evaluation. The metric ignores names, so a schema with unhelpful column names scores as well as a good one; naming quality is not measured. The real held-out sets share sources with training data and therefore overstate performance on unfamiliar documents, which is why we report TabRecSet separately. TabRecSet’s missing header annotation

forces a header assumption that is wrong for some tables, although the error analysis shows it affects few. All numbers come from a single training run; the intervals reflect sampling of test items, not training variance.

Limitations. Handwriting, multi-page tables and tables longer than 30 rows are outside the training distribution. Tables whose body contains section rows were excluded from both training and evaluation, so we do not know how the model handles them. Charts without value labels are read approximately. The data is English, apart from a few TabRecSet cells.

Broader impact. Receipts and invoices may contain personal data; the released data uses public datasets whose images were collected and anonymized by their authors. Extraction errors are likely to look plausible, so the warnings reported by the repair layer should be shown to users rather than discarded.

8 Conclusion

Image-to-database reconstruction asks for decisions that markup leaves open, and a 2B model trained on exact gold databases makes them reliably: merged cells become rows, forms become related tables, and the result runs in SQLite. Rendered databases teach these conventions, converted real annotations teach real content, and a deterministic repair layer turns near-misses into executable output. Reading text in unfamiliar photographs remains the open problem. Model, data, code and every evaluation output are available at <https://github.com/m-peker/pix2db> and <https://huggingface.co/musapeker/pix2db-2b>.

References

- [1] Christoph Auer, Maksym Lysak, Ahmed Nassar, Michele Dolfi, Nikolaos Livathinos, Panos Vagenas, et al. Dolving technical report, 2024.
- [2] Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, et al. Qwen3-VL technical report, 2025.
- [3] Lukas Blecher, Guillem Cucurull, Thomas Scialom, and Robert Stojnic. Nougat: Neural optical understanding for academic documents, 2023.
- [4] Zewen Chi, Heyan Huang, Heng-Da Xu, Houjin Yu, Wanxuan Yin, and Xian-Ling Mao. Complicated table structure recognition, 2019.
- [5] Ankush Gupta, Andrea Vedaldi, and Andrew Zisserman. Synthetic data for text localisation in natural images. In *Proceedings of CVPR*, 2016.
- [6] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.
- [7] Geewook Kim, Teakgyu Hong, Moonbin Yim, JeongYeon Nam, Jinyoung Park, Jinyeong Yim, Wonseok Hwang, Sangdoo Yun, Dongyoon Han, and Seunghyun Park. OCR-free document understanding transformer. In *European Conference on Computer Vision*, 2022.
- [8] Harold W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2 (1–2):83–97, 1955.
- [9] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of SOSP*, 2023.
- [10] Kenton Lee, Mandar Joshi, Iulia Turc, Hexiang Hu, Fangyu Liu, Julian Eisenschlos, Urvashi Khandelwal, Peter Shaw, Ming-Wei Chang, and Kristina Toutanova. Pix2Struct: Screenshot parsing as pretraining for visual language understanding. In *International Conference on Machine Learning*, 2023.
- [11] Fangyu Lei et al. Spider 2.0: Evaluating language models on real-world enterprise text-to-SQL workflows. In *International Conference on Learning Representations*, 2025.
- [12] Jinyang Li et al. Can LLM already serve as a database interface? A BIG bench for large-scale database grounded text-to-SQLs. In *Advances in Neural Information Processing Systems*, 2023.
- [13] Sifan Li, Hongkai Chen, Yujun Cai, Liyang Chen, Qingwen Ye, and Yiwei Wang. OptiSQL: Executable SQL generation from optical tokens, 2026.
- [14] Fangyu Liu, Julian Martin Eisenschlos, Francesco Piccinno, Syrine Krichene, Chenxi Pang, Kenton Lee, Mandar Joshi, Wenhui Chen, Nigel Collier, and Yasemin Altun. DePlot: One-shot visual language reasoning by plot-to-table translation. In *Findings of ACL*, 2023.
- [15] Fangyu Liu, Francesco Piccinno, Syrine Krichene, Chenxi Pang, Kenton Lee, Mandar Joshi, Yasemin Altun, Nigel Collier, and Julian Martin Eisenschlos. MatCha: Enhancing visual language pretraining with math reasoning and chart derendering. In *Proceedings of ACL*, 2023.
- [16] Rujiao Long, Wen Wang, Nan Xue, Feiyu Gao, Zhibo Yang, Yongpan Wang, and Gui-Song Xia. Parsing table structures in the wild. In *Proceedings of ICCV*, 2021.
- [17] Ahmed Masry, Do Xuan Long, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. ChartQA: A benchmark for question

answering about charts with visual and logical reasoning. In *Findings of ACL*, 2022.

- [18] Ahmed Masry, Parsa Kavehzadeh, Xuan Long Do, Enamul Hoque, and Shafiq Joty. UniChart: A universal vision-language pretrained model for chart comprehension and reasoning. In *Proceedings of EMNLP*, 2023.
- [19] Minesh Mathew, Dimosthenis Karatzas, and C. V. Jawahar. DocVQA: A dataset for VQA on document images. In *IEEE/CVF Winter Conference on Applications of Computer Vision*, 2021.
- [20] Nitesh Methani, Pritha Ganguly, Mitesh M. Khapra, and Pratyush Kumar. PlotQA: Reasoning over scientific plots. In *IEEE/CVF Winter Conference on Applications of Computer Vision*, 2020.
- [21] Ahmed Nassar, Nikolaos Livathinos, Maksym Lysak, and Peter Staar. TableFormer: Table structure understanding with transformers. In *Proceedings of CVPR*, 2022.
- [22] Seunghyun Park, Seung Shin, Bado Lee, Junyeop Lee, Jaeheung Surh, Minjoon Seo, and Hwalsuk Lee. CORD: A consolidated receipt dataset for post-OCR parsing. In *Document Intelligence Workshop at NeurIPS*, 2019.
- [23] Jake Poznanski, Aman Rangapur, Jon Borchardt, Jason Dunkelberger, Regan Huff, Daniel Lin, Christopher Wilhelm, Kyle Lo, and Luca Soldaini. olmOCR: Unlocking trillions of tokens in PDFs with vision language models, 2025.
- [24] Anshul Singh, Chris Biemann, and Jan Strich. MTabVQA: Evaluating multi-tabular reasoning of language models in visual space, 2025.
- [25] Anshul Singh, Rohan Chaudhary, Gagneet Singh, and Abhay Kumary. Lost in translation and noise: A deep dive into the failure modes of VLMs on real-world tables, 2025.
- [26] Brandon Smock, Rohith Pesala, and Robin Abraham. PubTables-1M: Towards comprehensive table extraction from unstructured documents. In *Proceedings of CVPR*, 2022.
- [27] Brandon Smock, Rohith Pesala, and Robin Abraham. Aligning benchmark datasets for table structure recognition. In *International Conference on Document Analysis and Recognition*, 2023.
- [28] Haoran Wei, Chenglong Liu, Jinyue Chen, Jia Wang, Lingyu Kong, Yanming Xu, Zheng Ge, Liang Zhao, Jianjian Sun, Yuang Peng, Chunrui Han, and Xiangyu Zhang. General OCR theory: Towards OCR-2.0 via a unified end-to-end model, 2024.
- [29] Fan Yang, Lei Hu, Xinwu Liu, Shuangping Huang, and Zhenghui Gu. TabRecSet: A large scale dataset for end-to-end table recognition in the wild. figshare dataset, 2023.
- [30] Fan Yang, Lei Hu, Xinwu Liu, Shuangping Huang, and Zhenghui Gu. A large-scale dataset for end-to-end table recognition in the wild. *Scientific Data*, 2023. doi: 10.1038/s41597-023-01985-8.
- [31] Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In *Proceedings of EMNLP*, 2018.
- [32] Mingyu Zheng, Xinwei Feng, Qingyi Si, Qiaoqiao She, Zheng Lin, Wenbin Jiang, and Weiping Wang. Multimodal table understanding. In *Proceedings of ACL*, 2024.
- [33] Xinyi Zheng, Douglas Burdick, Lucian Popa, Xu Zhong, and Nancy Xin Ru Wang. Global table extractor (GTE): A framework for joint table identification and cell structure recognition using visual context. In *IEEE/CVF Winter Conference on Applications of Computer Vision*, 2021.
- [34] Xu Zhong, Elaheh ShafieiBavani, and Antonio Jimeno Yepes. Image-based table recognition: Data, model, and evaluation. In *European Conference on Computer Vision*, 2020.

A Instruction

All SQL systems receive the image and this text; PIX2DB was trained with it.

Convert all data shown in this image into a SQLite database. Output only SQL, in a single ```sql code block: 1. CREATE TABLE statements: snake_case table and column names; column types INTEGER, REAL or TEXT; PRIMARY KEY and FOREIGN KEY constraints where the image implies a relationship (for example a document header and its line items). 2. INSERT statements for every record. Rules: one row per record (repeat the value of a merged cell in every row it covers); flatten multi-level headers into single column names; store numbers without thousand separators or currency symbols; write dates as YYYY-MM-DD; use NULL for empty cells; do not store total or subtotal rows (they are derived); for a chart, create one table with the category column and one column per data series; do not invent rows or columns.

The HTML pipeline instead asks for “every table and form in this image” as HTML `<table>` elements, keeping rowspan and colspan, with each key–value form written as a one-row table, and copying cell text exactly.

B Repair layer per evaluation set

Set	pix2db-2b		Qwen3-VL-8B		Qwen3-VL-2B	
	exec	F1	exec	F1	exec	F1
Synthetic	0.96→0.99	0.95→0.97	0.95→1.00	0.94→0.95	0.83→0.91	0.78→0.82
PubTabNet	0.94→0.99	0.78→0.81	0.92→0.98	0.50→0.52	0.52→0.72	0.26→0.30
FinTabNet	1.00→1.00	0.96→0.96	0.99→1.00	0.59→0.59	0.83→0.92	0.33→0.36
CORD	1.00→1.00	0.92→0.92	1.00→1.00	0.54→0.54	0.62→0.68	0.24→0.24
Synth. receipts	1.00→1.00	1.00→1.00	1.00→1.00	0.57→0.57	0.76→0.79	0.34→0.34
TabRecSet	0.91→0.99	0.51→0.52	0.94→0.99	0.42→0.45	0.71→0.87	0.29→0.31

Table 5: Share of executable outputs and cell F1 before → after repair. For receipts the F1 column shows cell F1, not the recall used in Table 3.

C Training and inference settings

Base model	Qwen3-VL-2B-Instruct (Apache-2.0)
Adapter	LoRA $r = 64$, $\alpha = 128$, dropout 0.05; q, k, v, o, gate, up, down of the language model
Vision encoder	frozen
Image budget	$\leq 1,310,720$ pixels
Optimization	AdamW, lr 10^{-4} , cosine, 3% warm-up, 1 epoch, batch $4 \times$ accumulation 4, bf16, gradient checkpointing
Data	36,589 examples (Table 2); target = gold dump in visual reading order
Hardware	1 $\times$ A40 48 GB; 5.9 h
Inference	vLLM, greedy, max 3,072 new tokens; same image budget
Seed	42 for data generation, sampling and training