Episode Book

EPISODE #6370

Why Modern Audio Datasets Ship as Parquet Files

PUBLISHED	30 October 2026 at 14:48 UTC
DURATION	31:00
LISTEN	myweirdprompts.com/episode/audio-datasets-in-parquet-13343/
HOSTS	Corn, Herman, Monty, Neville
SCRIPT MODEL	openai/gpt-6.1-sol
VOICES	chatterbox-regular (chatterbox-local)
PIPELINE	V5.25
SOURCES	20 cited by the research (listed at the end)
BOOK GENERATED	31 October 2026 at 05:58 UTC

SYNOPSIS

Parquet was designed for untidy tabular records, but dataset repositories increasingly use it to bundle audio recordings alongside transcripts and timestamps. The file structure splits data into row groups, column chunks, and pages, allowing readers across different programming languages to selectively decompress columns without streaming full datasets. From footer-based metadata and dictionary fallback mechanisms to custom compression codecs, storing raw audio inside columnar files introduces distinct architectural trade-offs between packaging convenience and model training workflows.

TRANSCRIPT



Corn

Daniel wants to know why Parquet has become the thing datasets arrive in, even when the dataset is mostly sound. Recordings, transcripts, speaker information, all bundled together and handed to somebody whose software you have never met. He wants to get inside that file, then work out whether we should actually ship an audio corpus that way.



Herman

The distribution part explains the appeal immediately. You can open the same bytes with Python through pyarrow, with Rust through arrow-rs, with Java through parquet-java, or directly in DuckDB and Polars. The author doesn't have to persuade the recipient to adopt the author's entire computing environment.



Corn

A dataset without a compulsory new personality.



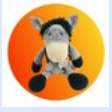
Herman

That matters more than another impressive compression chart. A format becomes useful infrastructure when somebody can publish it and stop being involved in every subsequent read.



Corn

Although audio seems like an awkward tenant. I understand separating a table into columns when I only want durations. Less obvious why I should put the recording itself beside them. You know what you're like, Herman. You'll admire the footer and accidentally put the whole microphone cupboard inside it.



Herman

Only the equipment with useful metadata. The loose cables remain an unsolved classification problem.



Herman

Parquet came out of the Hadoop ecosystem, with Google's Dremel work on nested records as a major influence. Nested data was part of the design from the beginning. Lists inside records, optional values, structures inside structures. That becomes important when your transcript contains words, timestamps, and speaker assignments rather than one uninterrupted string.



Corn

The reputation is rectangular office furniture. The actual design expected untidy records.



Herman

Its own motivation statement says the storage substrate should be useful to all frameworks without extensive, difficult dependencies. It explicitly refuses to play favorites. That's a remarkably durable position in an industry where every tool wants to become your landlord.



Corn

Hugging Face gives us the practical evidence. Its Dataset Viewer automatically converts hosted datasets to Parquet, up to five gigabytes. Even if you didn't choose Parquet when publishing, somebody downstream may choose it for you.



Herman

Their audio guidance goes further and recommends embedding audio bytes directly in Parquet for suitable datasets. You can carry multiple recordings, transcripts, and other information together. Binary data is a native fit, without inventing a textual representation for every byte.



Corn

I wouldn't mistake the viewer's convenience for a universal endorsement, though. A format can be excellent for previewing a dataset and still be awkward for training on long recordings.



Herman

Would you personally prefer a directory of recordings and a separate manifest?



Corn

For full podcast episodes, probably. I can inspect the original recordings without a library understanding the packaging. For short speech clips with word timings, I'd accept the bundle. The useful distinction is how people will consume it, not whether the repository looks tidy.



Herman

That distinction will decide the row groups, too. Parquet lets you choose the physical organization, but it doesn't know whether a row means a taxi journey or an hour of audio. A reasonable default for one can be ridiculous for the other.



Corn

The file does not know how heavy the row is.



Herman

Picture an ordinary table cut horizontally into batches. Those batches are row groups. Within each row group, each column has its own column chunk. Each column chunk contains pages. File, row group, column chunk, page. For nested schemas, those chunks correspond to the physical leaf columns.



Corn

Meaning the rows remain logically together, but their values aren't written as complete records one after another. All the durations for that group live together. All the speaker identifiers live together. The recordings live in their own chunk.



Herman

Yes. A column chunk is contiguous in the file. A page is the working unit for encoding and compression. Those boundaries give different operations somewhere sensible to happen. You can schedule row groups independently, read selected chunks, and decompress pages without treating the entire dataset as one compressed stream.



Corn

My initial guess would have been one compressed stream per column. Pages save you from inflating every recording just to reach one near the end.



Herman

They can. Although... finding an individual row still depends on what indexes are present and what the reader uses. Columnar storage doesn't promise perfect point lookup. It promises useful boundaries, and readers can exploit them.



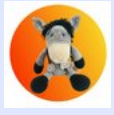
Herman

The physical file starts with a four-byte magic marker, PAR1. Then come the column chunks, arranged by row group. After the data comes the file metadata, then a four-byte little-endian length telling you how big that metadata is, then PAR1 again.



Corn

The contents page is at the back.



Herman

Written there deliberately. The writer can produce the data in a single pass, learn where everything ended up, and write the map afterward. A reader opens the end first, reads the metadata length, finds the metadata, and discovers which chunks it wants.



Corn

Remote storage makes that more than an aesthetic choice. With range requests, I can fetch the tail and then the duration column instead of downloading the recordings. Provided the server and reader cooperate.



Herman

Provided they cooperate, yes. The footer gives offsets and sizes. It doesn't force an HTTP server to honor range requests or prevent a client from downloading everything anyway.



Corn

A selective reader attached to an indiscriminate download button.



Herman

Statistics make the selectivity stronger. Column chunks can carry minimum and maximum values. Optional column indexes provide page-level statistics too. If you're asking for recordings shorter than a threshold, a reader can reject a row group whose minimum duration already exceeds it.



Corn

Unless every group contains everything from a cough to an opera.



Herman

Then the range is broad and the statistics can't rule much out. Organization matters. Sorting or clustering around common filters can make skipping effective. Merely writing statistics doesn't guarantee that a query avoids much data.



Corn

I'd be wary of sorting only by speaker for a training release. That might help someone extract a speaker, but a person reading sequentially gets long runs of the same voice. The storage optimization starts making choices about the learning pipeline.



Herman

That's a real trade-off. Shuffle during consumption, or publish a layout suited to the expected access pattern. What would you prioritize if strangers were downloading it?



Corn

Stable recording identifiers and documented ordering first. Then moderate shards that can be processed independently. I wouldn't silently optimize around my own training loop and call it neutral.



Herman

The physical type system takes a similarly restrained approach. Booleans, integers, floating-point values, and byte arrays form the core. There are ordinary and fixed-length byte arrays. Logical annotations explain what those physical values mean.



Corn

Such as a string being bytes with a string annotation, rather than a wholly separate physical species.



Herman

Right. There are thirty-two-bit and sixty-four-bit integer types, plus the older ninety-six-bit type you'll encounter around timestamps. No dedicated sixteen-bit physical integer is necessary because a wider integer with efficient encoding can store small values economically.



Corn

That is almost suspiciously restrained for a file format.



Herman

The complications move into the logical meanings and compatibility rules. Small physical vocabulary, richer annotations. An older reader may understand the underlying bytes without understanding every newer interpretation.



Corn

Which makes preserving meaning the harder part of compatibility. Opening successfully is a low bar if the timestamp changes its religion.



Herman

Encoding is the next layer, and it's separate from compression. First choose how values are represented. Then pass the resulting page data through a compression codec. You can change one without changing the other.



Corn

Repeated speaker identifiers are the obvious example. Writing the same long identifier repeatedly is wasteful before any general compressor gets involved.



Herman

Dictionary encoding creates a dictionary for a column chunk and writes its dictionary page first. Subsequent values become integer dictionary indexes. Those indexes use run-length encoding and bit packing. Repeated values can become runs; small index ranges need very few bits.



Corn

Forty speakers don't need forty elaborate names repeated in every row.



Herman

Precisely the useful case. But every recording is usually different. Build a dictionary of unique audio blobs and you've mostly constructed a second place to keep the same recordings. When a dictionary becomes too large, by size or number of distinct values, the writer falls back to plain encoding.



Corn

The dictionary has become the dataset with an index attached.



Herman

That fallback is central to the audio story. There are other encodings too. Delta binary packing works with integer differences. Delta length byte arrays encode lengths efficiently. Delta byte arrays can exploit shared prefixes. Byte stream split reorganizes bytes from fixed-width values so a compressor can find patterns more easily.



Corn

You're reaching the stage where you start looking fondly at integer differences.



Herman

I was already there. There's also ALP for floating-point values, marked as preview in the recent format release. That qualification matters. A feature existing in the specification doesn't mean every library can read it.



Corn

Your installation habits suggest you consider that a reader's personal challenge.



Herman

My installation habits are why I now appreciate test files.



Herman

The bit-packing details are lovely, though. The hybrid packs from the least significant bit upward. That allows little-endian processors to load machine words and unpack values with shifts and masks. A choice that looks fiddly in a document removes work from an operation repeated across enormous columns.



Corn

I knew the register was coming.



Herman

You did. Now compression comes after that representation. Snappy, GZIP, Brotli, ZSTD, LZO, and LZ4 Raw are among the supported choices. Uncompressed is also valid. Different columns can use different codecs.



Corn

I'd spend compression effort on transcripts before assuming an already compressed recording has much left to give.



Herman

Sensible. There's an older LZ4 variant with troublesome framing inherited from Hadoop. It's deprecated in favor of LZ4 Raw. Apart from that legacy exception, Parquet feeds page data to the compression library without adding another Parquet-specific framing layer.



Corn

Even compression has a legacy luggage compartment.



Herman

Interoperability reaches into small details. Writers are discouraged from producing multi-member GZIP pages by default because historical readers haven't all handled them. Both sides saying GZIP doesn't guarantee they agree on every permitted arrangement.



Corn

This is the bit people erase when they say a format is supported. They mean the files they've happened to receive.



Herman

We go live now to Monty, apparently at a carnival honoring today's topic.

M

Monty

I am standing here in the middle of the Parquet Carnival, and behind me the samba band is going absolutely wild because, my friends, the float representing the columnar layout just rolled past and it is magnificent, every row group a separate dancer, every page compressed and encoded and ready for the parade.



Corn

Monty, can anybody there read the footer?

M

Monty

Now the float for the audio bytes is approaching, and the crowd is throwing transcripts and metadata into the air like confetti, and I cannot hear myself think over these drums, but I can tell you the whole avenue smells of fresh parquet and pure joy.

M

Monty

This is Monty, reporting live from the Parquet Carnival in the middle of the samba band, and I am signing off before the tuba section reaches me.



Herman

The practical answer to reader compatibility is testing across implementations. Parquet-testing maintains a corpus specifically so implementations can check whether they read one another's files. For creating files in Python, pyarrow is the obvious starting point.



Corn

Pandas can use it underneath, but I'd distinguish the convenient table interface from the actual storage implementation. Otherwise you think you're choosing a format when you're inheriting an engine's defaults.



Herman

Pyarrow defaults include dictionary encoding, Snappy compression, and statistics. Its default row group target is roughly a million rows, limited by the table size. That is a perfectly consequential choice to inherit without noticing.



Corn

A million audio recordings in one row group sounds like a hostage situation.



Herman

You'd change it. Another subtlety is that pyarrow's default logical format version is two point six, while its default data page version is one point zero. Those are separate settings, not a contradictory description of the same switch.



Corn

Put that distinction on the packaging before somebody changes both because newer sounds healthier.



Herman

Rust users have the parquet crate in arrow-rs, with parquet derive for mapping plain structures. Java has parquet-java, the reference implementation. DuckDB and Polars are excellent for inspection and querying without building a dedicated application.



Corn

I'd open an unfamiliar file in DuckDB first and inspect the schema and a small selection of metadata. Not decode all the sound to prove it arrived.



Herman

Hugging Face Datasets adds the domain-specific layer. It can understand that a particular structured column represents audio. Parquet itself doesn't define an audio feature, a decoder, or a promise that playback works.



Corn

Here's the schema I'd actually want to hand Daniel. One row per clip, a stable identifier, audio, transcript, duration, speaker information, and explicit units for any timings. Not a transcript whose timestamps depend on the original editor's memory.



Herman

Hugging Face's convention makes the audio column a structure with binary bytes and a string path. The README's feature declaration identifies it as audio. That is a platform convention layered over Parquet, not an official audio type in the Parquet specification.



Corn

The path can preserve a useful filename even when the bytes are inline. It shouldn't secretly be a location on the publisher's laptop.



Herman

Exactly that distinction. A standalone recipient can extract the bytes using a generic reader. To get automatic audio decoding, their software needs to understand the convention and have the decoder dependencies available.



Corn

One file, several understandings of the file.



Herman

A concrete example is arda-argmax's simchoir-parquet dataset. Its audio is inline mono WAV at sixteen kilohertz. Alongside it are lists of word tokens, word start and end times, speaker assignments, recording identifiers, and recording properties.



Corn

Nested columns earn their keep there. You don't need a separate word table with a recording identifier repeated beside every token, then a join just to reconstruct one utterance.



Herman

Dremel-style record shredding preserves the structure through leaf values and information about nesting and repetition. On read, the lists are reconstructed. You're storing nested records column-wise, not simply flattening every nested value into unrelated rows.



Corn

I'd still check that the token list and timing lists line up. The format can faithfully preserve my mistake.



Herman

It absolutely can. Simchoir also carries a manifest JSON column containing the full lhotse Cut description. That's a useful escape hatch for reconstructing a richer application object while keeping frequently queried properties in proper columns.



Corn

JSON for the information you may need later, columns for the information you keep asking about. I like that better than pretending every consumer wants the entire original object.



Herman

Its initial subset has a thousand mixtures and forty speakers, spread over five shards. Provenance files sit alongside the shards. The release is a package with supporting context, not a magical solitary file that explains itself.



Corn

Shards are worth emphasizing. Bundling doesn't have to mean one enormous download. You can have consistent independent pieces and still avoid shipping a million loose clips.



Herman

Another example, speechio_test, repackages a Mandarin speech recognition benchmark into Parquet with embedded audio. It covers twenty-seven test sets and roughly sixty-six hours. The attraction is a consistent distribution interface across a corpus.



Corn

Now the compression claim needs narrowing. A WAV container often holds uncompressed PCM. We shouldn't describe WAV itself as already compressed and then act surprised if ZSTD reduces it.



Herman

Correct. I was about to say audio gains little... no, compressed audio gains little from another general compressor. PCM may shrink substantially. The gain depends on the payload. FLAC or another already compressed stream usually leaves much less redundancy for ZSTD.



Corn

None of that makes Parquet a better audio codec. It stores the bytes you chose. Dictionary encoding helps recurring metadata, not thousands of unique recordings.



Herman

Speaker identifiers and language labels are especially favorable. Durations and timestamps may benefit from appropriate encodings. Transcript tokens have repetition too, although the actual gain depends on how they're represented. There isn't one compression story for the whole file.



Corn

I can have beautifully compressed speaker labels attached to three gigabytes of stubborn sound.



Herman

You can, and still have an excellent dataset. What do you gain from keeping the audio inline, rather than merely linking it?



Corn

The row carries its own payload. No broken relative paths, no missing object on another host, no extra small-file download for each clip. I can copy the shards and keep the relationship intact. The price is that updating a transcript or replacing a recording generally means rewriting the relevant artifact.



Herman

That's the price of an immutable distribution snapshot. External audio with a metadata sidecar lets you change the table without republishing all the recordings. Inline audio makes a snapshot easier to move and verify together.



Corn

For something downloaded by strangers, I'd favor that snapshot if the clips are short. For our full episodes, I'd keep the original media separate. I don't want every correction to a title dragging the sound through another release.



Herman

Hugging Face's guidance draws a similar boundary. It recommends Parquet for audio files smaller than about one megabyte and small row groups, around a hundred rows. For larger recordings, it recommends WebDataset or sharing the originals, optionally with metadata.



Corn

A hundred rows is dramatically different from pyarrow's million-ish default. That's not fine tuning. That's remembering what you put in the cells.



Herman

Small groups bound the amount of nearby material a reader may have to deal with. But let's be precise... row group size is not the same as compression block size. Pages are compressed individually. Page indexes and reader behavior affect how directly a particular row can be reached.



Corn

Still, if I make the groups gigantic, I've made planning, buffering, and skipping less convenient. If I make them tiny, I grow the footer and add overhead.



Herman

Right. The hundred-row recommendation is practical guidance, not a universal optimum. Long recordings can dominate a group even when its row count looks modest. You have to think in bytes and access patterns as well.



Corn

WebDataset's TAR shards make more sense when training wants to stream successive examples and their associated files. The metadata-query advantage matters less if you're going to read everything anyway.



Herman

Parquet becomes particularly attractive when people select examples by language, duration, or speaker, then decode the selected sound. WebDataset is strong for sequential consumption. Loose media plus a Parquet metadata sidecar is another perfectly coherent design.



Corn

A sidecar is allowed to be the right answer. Bundling everything shouldn't become a matter of personal honor.



Herman

For the downloader using Hugging Face Datasets, the normal path is to select the repository, configuration, and split, optionally stream it, and iterate examples. Its audio feature hands decoding to torchcodec, which wraps FFmpeg.



Corn

Worth distinguishing compressed bytes, decoded samples, and sample rate. A downstream tool may want a floating-point waveform. Another may simply want to save the original recording without touching it.



Herman

Both are possible. Older examples often show direct access to an audio array. Current decoder-backed interfaces can expose an audio decoder object instead, so follow the installed library's interface when requesting samples. The result can be a floating-point waveform, but Parquet isn't doing that conversion.



Corn

The decoder is another dependency that survived the single-file promise.



Herman

The promise was transport, Corn. Playback still needs playback software.



Corn

I concede the promise. I retain the eyebrow.



Herman

You can also remove an unwanted manifest column from your dataset view or project only the columns you need. With suitable readers, leaving the audio column out means you can analyze metadata without loading the audio payload. The lhotse manifest can be used to reconstruct a CutSet if that's your workflow.



Corn

Before publishing, I'd test those two experiences separately. Can someone inspect metadata with a generic tool? Can someone decode a clip through the advertised audio stack? A successful upload demonstrates neither.



Herman

Compatibility is where newer can actively hurt. The author of Carpet changed the default to Parquet version two, then rolled it back after discovering that the older Pandas environment at work couldn't read the result.



Corn

Modernized straight out of opening.



Herman

Pyarrow's documentation explicitly warns that logical versions two point four and two point six may not be readable everywhere. Version one point zero maximizes compatibility, provided its available types meet your needs. Data page versions remain a separate consideration.



Corn

I'd choose the oldest feature set that preserves the actual schema. Not the oldest at any cost. Losing timestamp meaning to win an opening contest seems a poor exchange.



Herman

Nor does version two guarantee smaller files. In the comparison described by the Carpet author, the NYC taxi dataset with ZSTD went from four hundred thirty megabytes to four hundred forty-four. The newer choice was larger.



Corn

Fourteen megabytes of progress.



Herman

Another dataset improved only slightly once ZSTD was involved. Uncompressed comparisons looked much more dramatic. That tells you to test the complete configuration, not infer the outcome from the version label.



Corn

Your reader matrix matters more than your victory over an uncompressed baseline.



Herman

The format's success makes evolution difficult. Supporting a new feature in one writer doesn't help if the recipients' query engines reject it. There's been a long-running unresolved discussion about what the version two core actually comprises.



Corn

Then the release needs a real dataset card. Schema, units, conventions, dependencies, licensing, provenance, and a small example someone can open. Those aren't decoration around the artifact. They're how the next person knows what the artifact means.



Herman

The Hub viewer isn't a substitute. The open-index Hacker News archive packages more than forty-seven million items as Parquet, yet its viewer has displayed a CreateCommitError. A failed preview doesn't establish that the underlying files are bad.



Corn

Nor does a working preview establish that my local reader can open them. I'd want the files and their documentation to remain useful when the website's thumbnail has given up.



Herman

Hilbert left a card for our next contributor. Specialist in archival transfer, practical data handling, assorted departmental assistance, and available most afternoons. Neville Muddledorf.

N

Neville

That's me. The afternoons are the internship. I wrote that bit. Uncle Hilbert said I should sound established, but I have only established which cupboard has the spare headphones. He's technically my great uncle twice removed, then sideways. I was helping

N

Neville (continued)

with his little archive of recordings of doors closing.



Corn

An appropriately decisive sound collection.

N

Neville

I sent a bundle to a friend on my placement. She sent back, cannot open file. Just that. I thought I'd chosen the wrong audio compression because I'd used the newer Parquet. Those were the same setting in my head, Herman. I'd read the first half of the instructions, then the bus arrived.



Herman

The Parquet version and audio compression are separate choices. Your reader may not support a file feature even when the recording bytes are fine. That's why we test the intended reader, not just the writer.

N

Neville

Oh, that's much better, actually. I didn't fix the reader. I played the recordings and sent her a voice message of the playback. It included me saying which door each one was. Uncle Hilbert keeps everything in boxes, labelled and dated, with boxes inside boxes, so I thought saying the labels aloud was the important part.



Corn

You delivered the metadata personally.

N

Neville

I did. I also added a column called sounded final. Nobody requested it. Sometimes a door closes and you know it's properly finished. My friend wanted to know whether that was a machine label or mine, so now I've written that it's mine. I'm still working on the little explanation at the top. It keeps getting longer than the recordings.



Herman

That explanation is useful, Neville. A named column still needs a meaning somebody else can reproduce.



Corn

The pressure on Parquet will be whether it can add capabilities without making that handoff harder. Formats such as Nimble and Lance's newer storage designs are pushing at machine learning workloads, but I'd still ship short audio clips in conservative Parquet shards when queryable metadata and a self-contained snapshot are the priorities. Full-length recordings would get a different package.



Herman

Thanks to our producer, Hilbert Flumingtop, and to Neville for covering the desk.



Corn

If you want more of this, try episode three, Safetensors or something else; episode twenty-four sixty-five, JSON-L vs Parquet; and episode seven, Building Custom ASR Tools. This has been My Weird Prompts, the human-AI collaboration podcast. You can find the episodes at my weird prompts dot com.



Herman

I'm Herman Poppleberry. See you tomorrow.



Corn

Until then.

SOURCES

What the research for this episode read before the script was written. Primary sources first.

1. **Apache, last push 2026-10-27; release `apache-parquet-format-2.14.0` (2026-09-11) [primary]**
https://github.com/apache/parquet-format

2. **Parquet docs, Concepts [primary]**
https://parquet.apache.org/docs/concepts/

3. **Parquet docs, Encodings [primary]**
https://parquet.apache.org/docs/file-format/data-pages/encodings/

4. **Parquet docs, Compression [primary]**
https://parquet.apache.org/docs/file-format/data-pages/compression/

5. **Apache Arrow v26.0.0 [primary]**
https://arrow.apache.org/docs/python/generated/pyarrow.parquet.write_table.html

6. **Apache Arrow v26.0.0 [primary]**
https://arrow.apache.org/docs/python/generated/pyarrow.Codec.html

7. **Apache, release 60.1.0 (2026-10-28) [primary]**
https://github.com/apache/arrow-rs

8. **Hugging Face Hub docs, Audio Dataset [primary]**
https://huggingface.co/docs/hub/main/datasets-audio

9. **Hugging Face Datasets, Load audio data [primary]**
https://huggingface.co/docs/datasets/en/audio_load

10. **HF `Audio` feature source [primary]**
https://github.com/huggingface/datasets/blob/main/src/datasets/features/audio.py

11. **Embed image/audio data in dl_and_prepare parquet, merged 2022-09-16 [primary]**
https://github.com/huggingface/datasets/pull/4987

12. **Wikipedia, last edited 2026-09-09**
https://en.wikipedia.org/wiki/Apache_Parquet

13. **dataset card, v0.1**
https://huggingface.co/datasets/arda-argmax/simchoir-parquet

14. **dataset card**
https://huggingface.co/datasets/yuekai/speechio_test

15. **dataset card (viewer error)**
https://huggingface.co/datasets/open-index/hacker-news

16. **2025-08-21**
https://www.jeronimo.dev/the-two-versions-of-parquet/

17. **HN discussion of #16, 2025-08-21**
https://news.ycombinator.com/item?id=44970769

18. **HN, Hacker News archive as Parquet, 2026-03-14**
https://news.ycombinator.com/item?id=47378781

19. **Show HN, 2025-11-17**
https://github.com/kaushiksrini/parqeye

20. **Apache Arrow repo, 17,262 stars**
    https://github.com/apache/arrow