Problem

It is hard to manage both, files and SQL data

Why?
It is hard to keep files and records in sync;

- Files get renamed or deleted
- New files appear
- No transactional semantics
- Different permission mechanisms
- Unix vs. SQL users and permissions
- Different content access mechanisms
- SQL connection vs. file access vs. web

Solution

Manipulating files with SQL

- Accessing contents of files
- ACID semantics
- Dealing with URLs
- File management (move, etc)

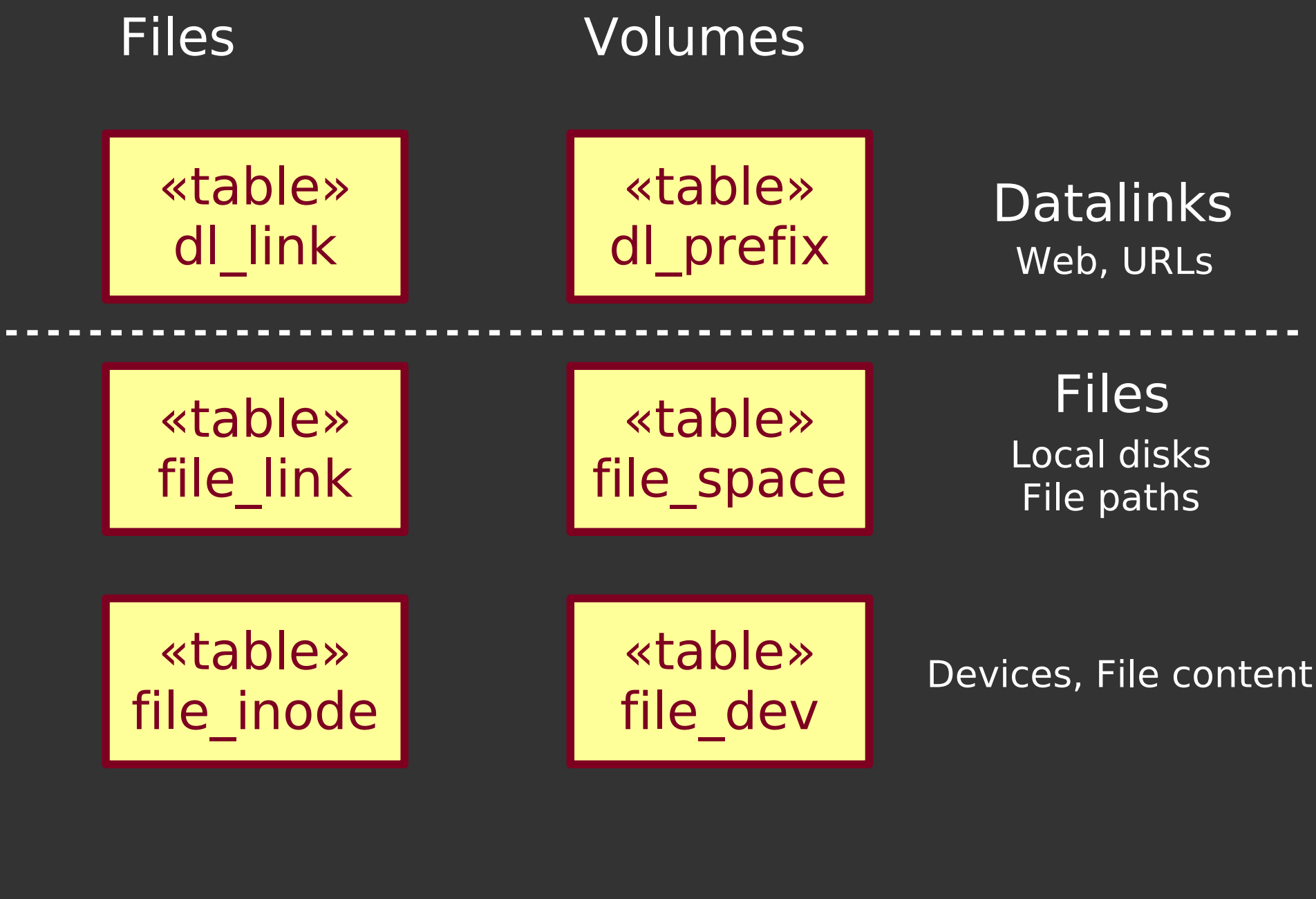
SQL/MED datalinks
according to SQL standard

- DATALINK datatype
  Datatype for storing URLs
- Support functions
  Using values of datalink type
- Special semantics
  - Transactions
  - Referential integrity
  - Access mediation
  - Point-in-time recovery

Other interesting libraries

- Built-in file functions – pg_file_read...
- Contrib adminpack – pg_file_write, pg_file_rename, pg_file_unlink...
- PgFoundry uri – this could be replaced by url_part and url_part_set
- CURL – it is best at grokking URLs. Talks to complicated and secure web servers. It could be used for actual web access.

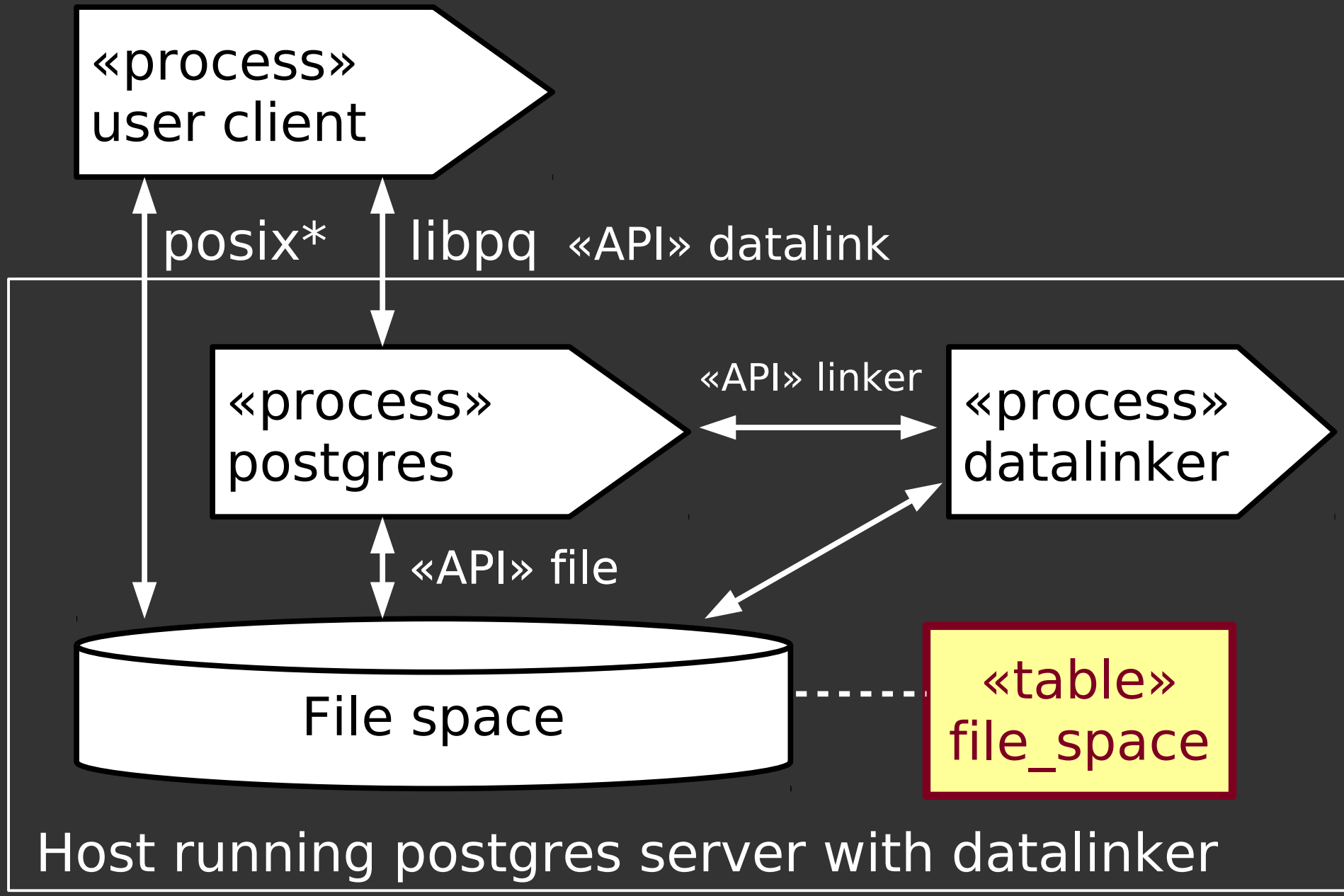
Object hierarchies



APIs

«API» datalink	SQL/MED functions
«API» admin	Server administration functions
«API» URL	URL utility functions
«API» file	Transactional file functions
«API» linker	Datalinker functions

Datalinker process



* only clients on same host

Datalinker responsibilities

- Backup a file
- Delete a file
- Link to SQL (protect) for reading
- Link to SQL (protect) for writing
- Checkpoint
- Rename a file (?)

Datalinker is accessed with «API» linker.
Not intended for end users!

«API» linker

- Internal datalink usage only
- Not intended for end users
- Stores instructions for datalinker in «table» file_xact

«API» file – whole files

- Use `file_put()` to write whole files

```
SELECT file_put('/var/www/index.txt','Hello, world!');
```

- Use `file_get()` to read whole files

```
SELECT file_get('/var/www/index.txt');

Hello, world!
```

When writing files, new file is always created!
This is used to implement transactions.
Old files can be purged separately.

«API» file - incremental

- Use `file_begin()`, `file_write()` and `file_end()` to write files incrementally

  This is useful for functions in procedural languages

```
BEGIN
    fh := file_begin('/var/www/index.html');
    FOR row IN
        SELECT label FROM items
    LOOP
        file_write(fh,row.label||E'\n')
    END LOOP;
    file_end(fh);
END
```