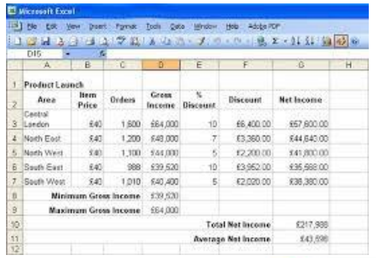
Intro to Databases

What is a database?

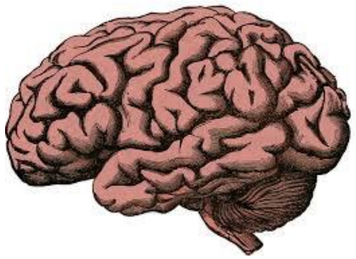
Things that hold info

Accessible

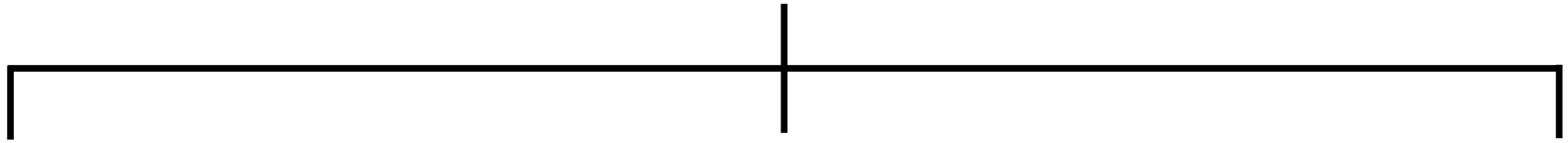


	A	B	C	D	E	F	G	H
1	Product Launch							
2	Area	Item Price	Orders	Gross Income	% Discount	Discount	Net Income	
3	Central London	£40	1,600	£64,000	10	£6,400.00	£57,600.00	
4	North East	£40	1,200	£48,000	7	£3,360.00	£44,640.00	
5	North West	£40	1,100	£44,000	5	£2,200.00	£41,800.00	
6	South East	£40	988	£39,520	10	£3,952.00	£35,568.00	
7	South West	£40	1,010	£40,400	5	£2,020.00	£38,380.00	
8			Minimum Gross Income	£39,520				
9			Maximum Gross Income	£64,000				
10						Total Net Income	£217,988	
11						Average Net Income	£43,598	
12								

Persistent



A database persists information and is accessible via code



organized queryable manageable

Organized: Standard Storage Formatting

- DBs are a collection of Tables (or ***relations***)
- Tables have Columns (***attributes*** / ***fields***) that describe Rows (***instances*** / ***tuples***)
- Duplicate rows are not allowed
- Rows often have a primary key (unique identifier)

Table / Relation

Column / Attribute / Field | Column / Attribute / Field | Column / Attribute / Field

ID	Name	Type
1	Pikachu	lightning
2	Squirtle	water
3	Charmander	fire
4	Bulbasaur	grass

Row / Tuple / Instance

Row / Tuple / Instance

Row / Tuple / Instance

Row / Tuple / Instance

Queryable: via a Standard Language

- A simple, structured query language: SQL
- Declarative (vs. imperative)
- No more hand-rolled algorithms / data structures
- DBMS picks an efficient execution strategy based on indexes, data, workload etc.

SQL

```
-- Pikachu, I choose you!
SELECT id, name
  FROM pokemon
  WHERE type = 'lightning'
  LIMIT 1
```

Manageable: Easy, Safe, Performant

- Offloads work and requisite understanding of programming
- Knowledge is portable
- Abstraction
- Transfer data between systems
- DBMS can make certain guarantees
  - prevent unsafe operations
  - built-in redundancies
  - handle multiple users, threads

ACID Guarantees

- Atomicity
- Consistency
- Isolation
- Durability

What happens if the database crashes?

- Suppose we have a banking system with client accounts
- Every debit must have a matching credit.
- Imagine a crash results in only one table being updated.
  - Database inconsistency → unexpected data and software crashes/bugs
  - Financial risk for clients

Atomic Transactions

- atomic transaction: A set of database operations that must occur together
  - i.e. A debit to one bank account, and a credit to another
- A transaction must either succeed or fail; it cannot partially complete.
- Every database query is represented by a transaction

Consistency

- Specify rules that columns need to follow
  - Gender column can only contain M, F, or U.
  - Savings account must start with S or checking with C
  - Column cannot be null

- Protect the database from inconsistencies and simplify software logic
  - Allows software to make assumptions about underlying data

Resource Management

- Processes can be readers and writers
- Files can have many readers
- If a process has a writer, no other process can read from it, and no other process can write to it

Proposed File Scheme

- Suppose that we have decided not to use a database and instead store our data in a series of files.

- How might our setup fail to serve queries from multiple users?

Deadlock

Databases give us concurrency (Isolation)

- Multiple clients can make queries to read and update without the risk of deadlock or starvation.

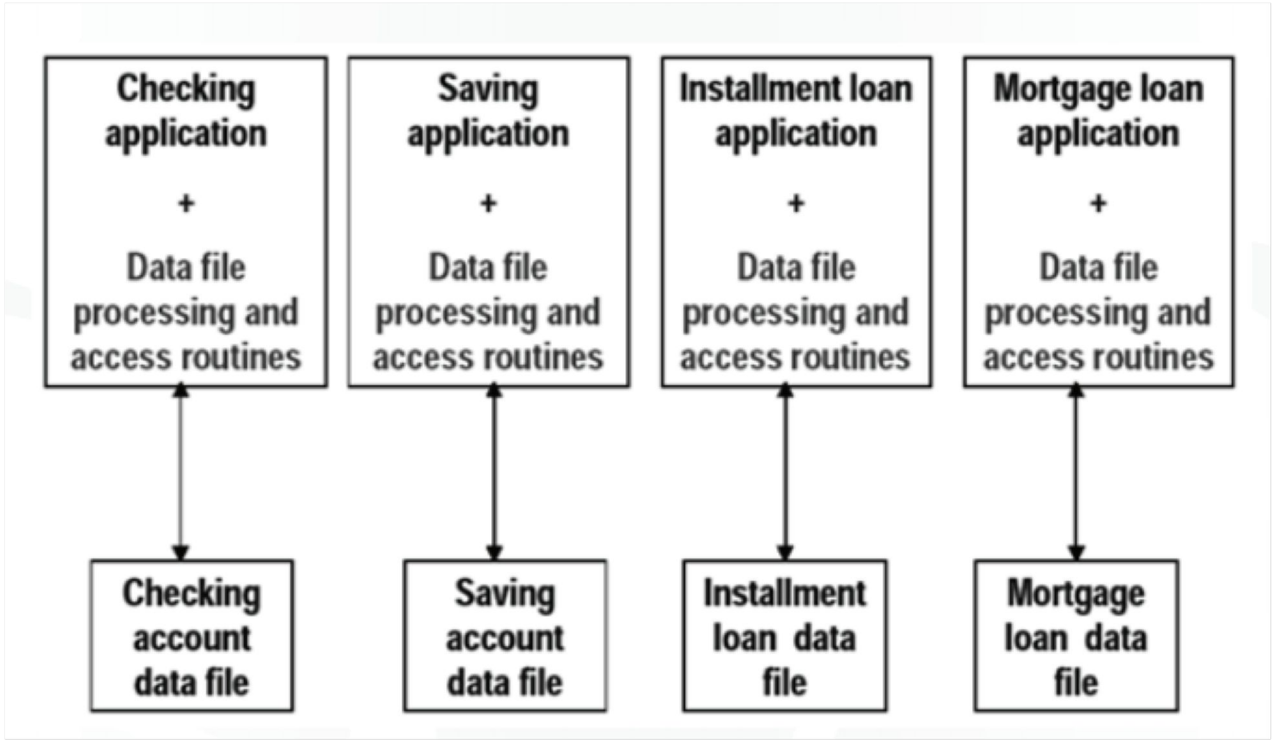
Persistence/Durability

- Databases are also persistent (store information without power)

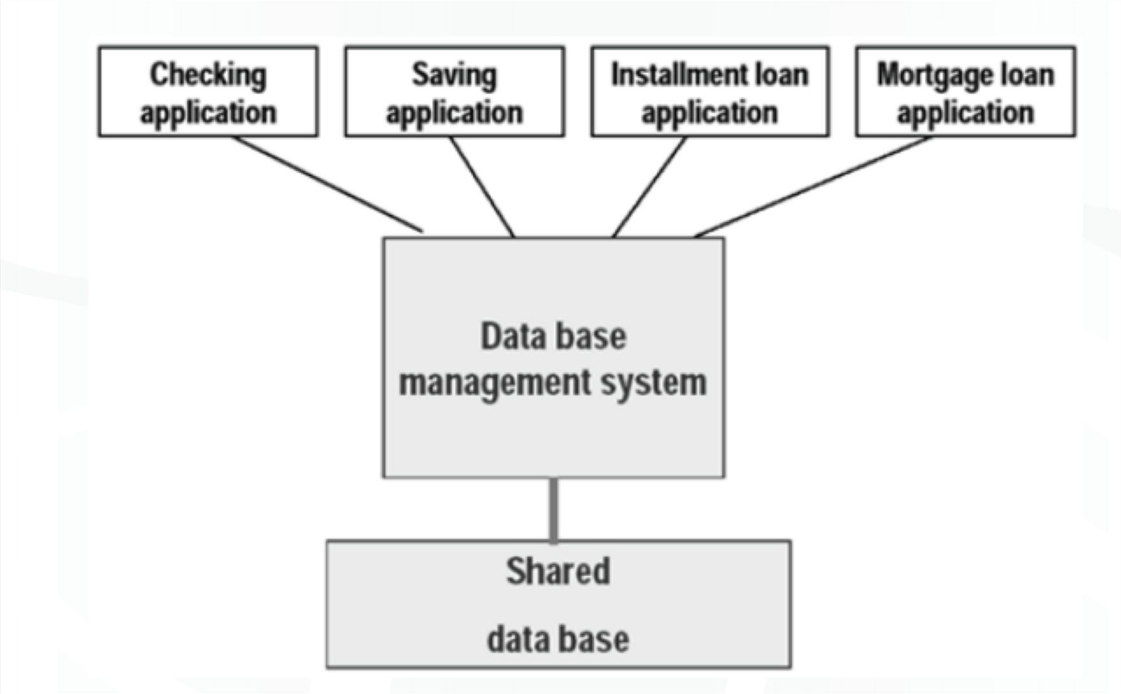
How Did We End Up Here?

Before Relational DBs (ca. < 1970s)

- Data stored in custom “data files”
- Queried via application-specific code
- Advantages
  - Middle layer not needed
  - Solutions customized for each application
- Disadvantages
  - Hard to change the system
  - Knowledge not compounding
  - Data-transfer is difficult



Database Management Systems (DBMS)

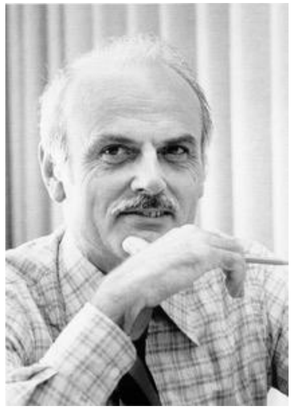


- One layer and language to store and access data
- Sold as a way for "non-technical people" to manage data

"Future users of large data banks must be protected from having to know how the data is organized in the machine (the internal representation)."

– E. F. CODD,
A RELATIONAL MODEL OF DATA FOR
LARGE SHARED DATA BANKS

Relational Databases & Logic



- 1969: Edgar Frank "Ted" Codd outlines ***relational model*** of data
- Wrote Alpha (never implemented) as a ***query language***
- IBM slow to adopt his ideas
  - Competitors started to do so
  - IBM team formed without Codd, created **S**tructured **E**nglish **Qu**ery **L**ang
- SEQUEL way better than what came before
  - 1979: copied by Larry Ellison (from pre-launch papers / talks!) as "SQL"
- SQL became the standard (ANSI 1986, ISO 1987)
  - Codd continued to fault SQL compared to his theoretical model
  - The Third Manifesto: solve the *object-relational impedance mismatch*

Appreciating Databases

- Ubiquitous
- Standardized
- Complex / deep
- Powerful: database admins are
  - Feared by developers
  - …but also taken for granted until things break
  - Befriended by business people
  - Contacted by the government for secret data (e.g. NSA)

Progression of Databases

- Navigational (< 1970s)
  - More common during tape era; entries had references to next entries.
- Relational (> 1970s)
  - Based on relational (table-based) logic, see E.F. Codd.
- NoSQL (> 2000s)
  - "Not only SQL" — document storage, for example.

RDBMS vs NoSQL

- A DBMS doesn't ***have*** to be relational
  - Remember, DBMS is just an application that intelligently stores data and can answer requests to manage that data
- Lately, many "NoSQL" or non-relational DBMSs have been gaining popularity
  - Graph databases (e.g. Neo4J)
  - Document databases (e.g. MongoDB)
  - Hybrids (e.g. PostgreSQL)
- RDBMSs still remain the #1 DB option for now

Some well-known rDBMSs

PostgreSQL

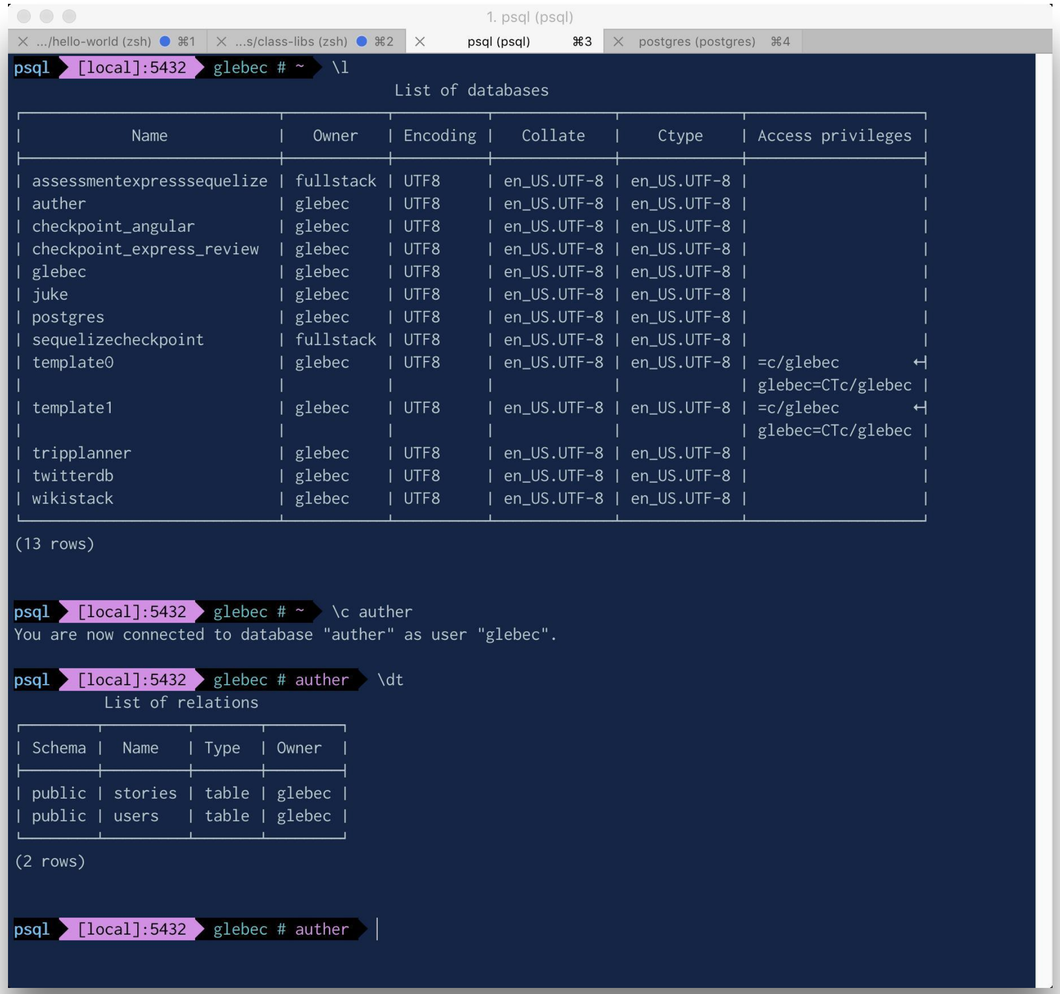
Why PostgreSQL?

- Advanced, powerful, and popular
- Rapid open source development
- Highly extensible (stored procedures)
- Deep SQL standards compliance
- NoSQL ("Not Only SQL"), objective support
- Excellent transactions / ACID reliability; focus on integrity
- Multi-user management / administration

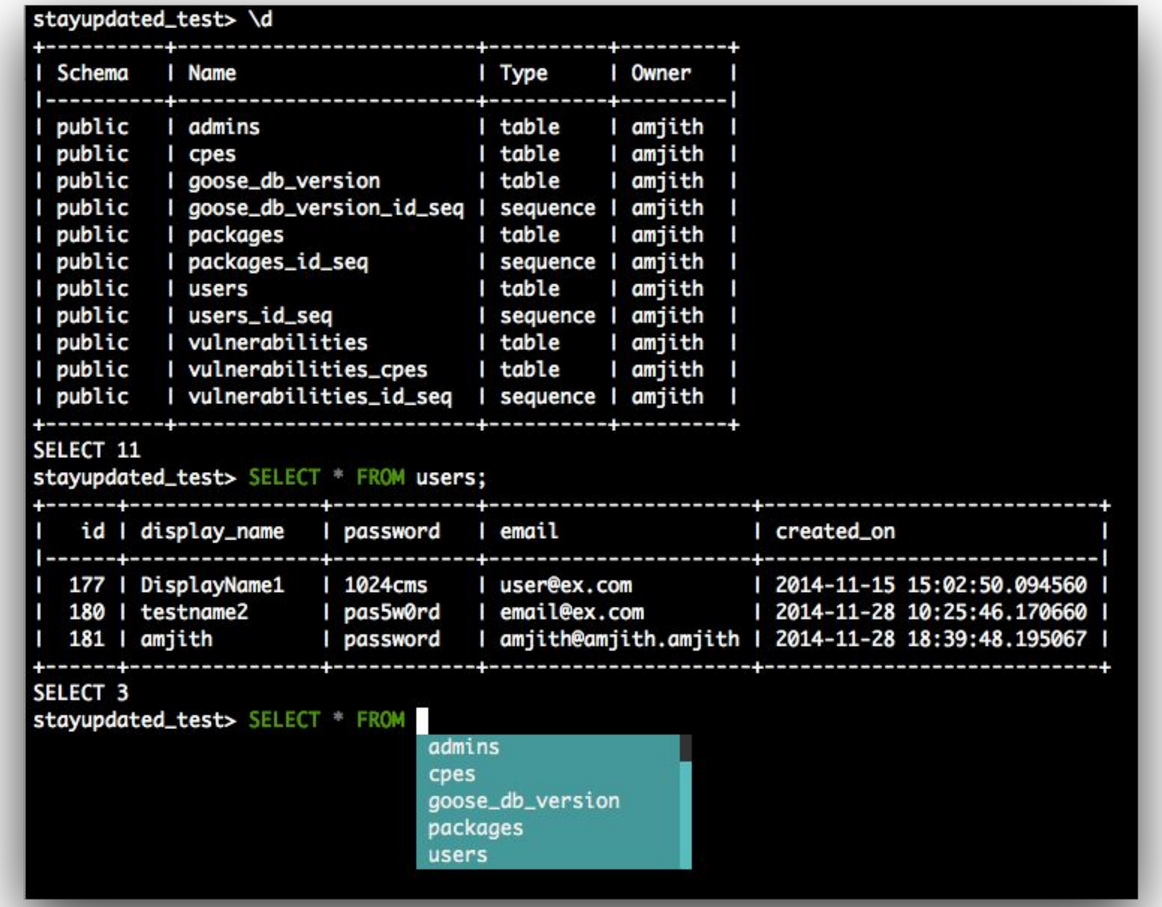
History of PostgreSQL

- 1970s at UC Berkeley:
  INteractive **G**raphics **RE**trieval **S**ystem (INGRES)
- 1980s: POSTGRES ("Post-Ingres")
- 1995: POSTQUEL and Postgres95.
  - monitor -> psql
- 1996: Adopted by the open source community
  - Ongoing: stability, testing, documentation, new features
  - PostgreSQL

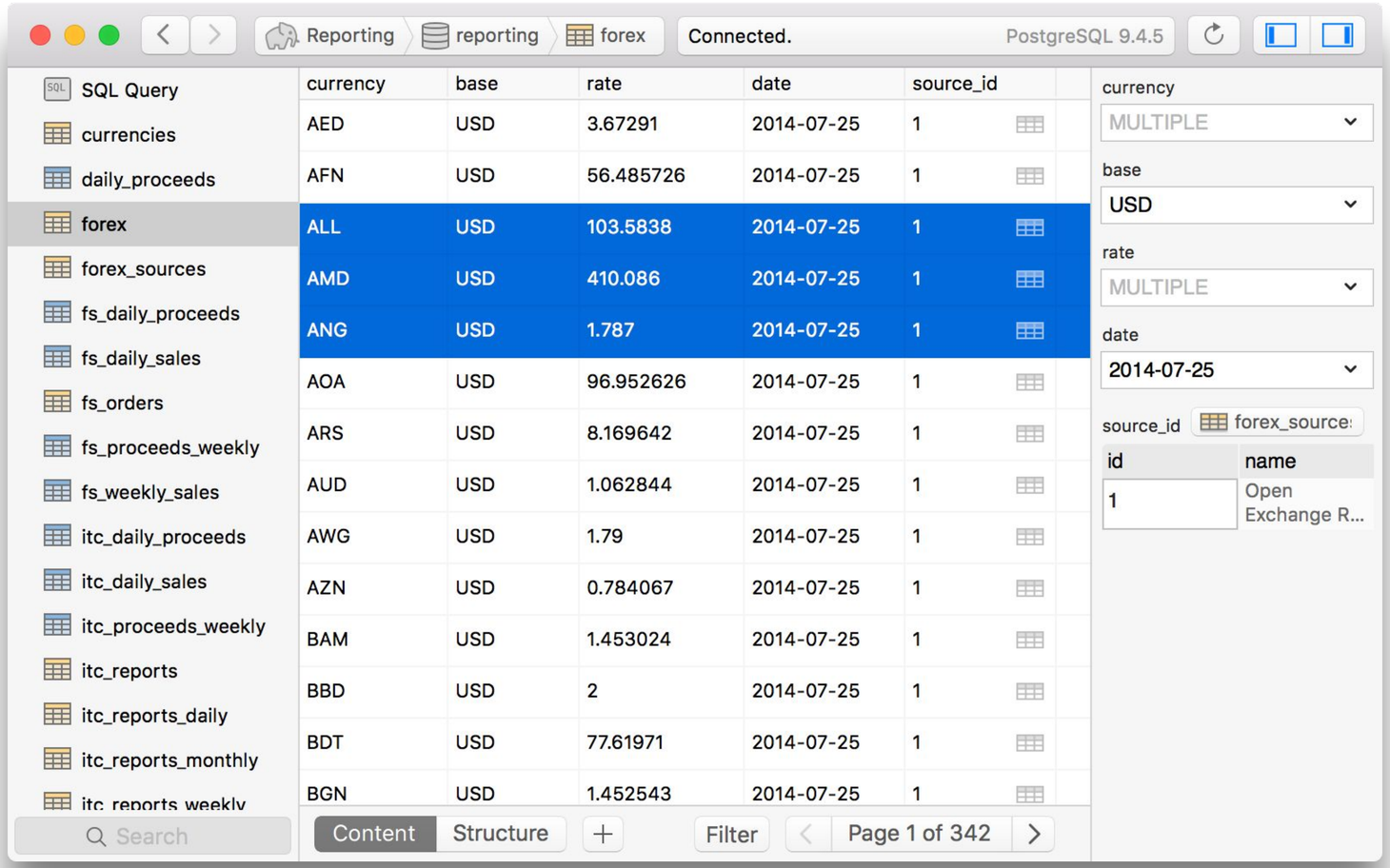
psql


1. psql (psql)
.../hello-world (zsh) ⌘1
...s/class-libs (zsh) ⌘2
psql (psql) ⌘3
postgres (postgres) ⌘4
psql [local]:5432 glebec # ~ \l
List of databases
Name | Owner | Encoding | Collate | Ctype | Access privileges
assessmentexpresssequelize | fullstack | UTF8 | en_US.UTF-8 | en_US.UTF-8 |
auther | glebec | UTF8 | en_US.UTF-8 | en_US.UTF-8 |
checkpoint_angular | glebec | UTF8 | en_US.UTF-8 | en_US.UTF-8 |
checkpoint_express_review | glebec | UTF8 | en_US.UTF-8 | en_US.UTF-8 |
glebec | glebec | UTF8 | en_US.UTF-8 | en_US.UTF-8 |
juke | glebec | UTF8 | en_US.UTF-8 | en_US.UTF-8 |
postgres | glebec | UTF8 | en_US.UTF-8 | en_US.UTF-8 |
sequelizecheckpoint | fullstack | UTF8 | en_US.UTF-8 | en_US.UTF-8 |
template0 | glebec | UTF8 | en_US.UTF-8 | en_US.UTF-8 | =c/glebec glebec=CTc/glebec
template1 | glebec | UTF8 | en_US.UTF-8 | en_US.UTF-8 | =c/glebec glebec=CTc/glebec
tripplanner | glebec | UTF8 | en_US.UTF-8 | en_US.UTF-8 |
twitterdb | glebec | UTF8 | en_US.UTF-8 | en_US.UTF-8 |
wikistack | glebec | UTF8 | en_US.UTF-8 | en_US.UTF-8 |
(13 rows)
psql [local]:5432 glebec # ~ \c auther
You are now connected to database "auther" as user "glebec".
psql [local]:5432 glebec # auther \dt
List of relations
Schema | Name | Type | Owner
public | stories | table | glebec
public | users | table | glebec
(2 rows)
psql [local]:5432 glebec # auther

pgcli


stayupdated_test> \d
Schema	Name	Type	Owner
public	admins	table	amjith
public	cpes	table	amjith
public	goose_db_version	table	amjith
public	goose_db_version_id_seq	sequence	amjith
public	packages	table	amjith
public	packages_id_seq	sequence	amjith
public	users	table	amjith
public	users_id_seq	sequence	amjith
public	vulnerabilities	table	amjith
public	vulnerabilities_cpes	table	amjith
public	vulnerabilities_id_seq	sequence	amjith
SELECT 11
stayupdated_test> SELECT * FROM users;
id	display_name	password	email	created_on
177	DisplayName1	1024cms	user@ex.com	2014-11-15 15:02:50.094560
180	testname2	pas5w0rd	email@ex.com	2014-11-28 10:25:46.170660
181	amjith	password	amjith@amjith.amjith	2014-11-28 18:39:48.195067
SELECT 3
stayupdated_test> SELECT * FROM
admins
cpes
goose_db_version
packages
users

Postico


Reporting
reporting
forex
Connected.
PostgreSQL 9.4.5
SQL Query
currencies
daily_proceeds
forex
forex_sources
fs_daily_proceeds
fs_daily_sales
fs_orders
fs_proceeds_weekly
fs_weekly_sales
itc_daily_proceeds
itc_daily_sales
itc_proceeds_weekly
itc_reports
itc_reports_daily
itc_reports_monthly
itc_reports_weekly
Search
currency base rate date source_id
AED USD 3.67291 2014-07-25 1
AFN USD 56.485726 2014-07-25 1
ALL USD 103.5838 2014-07-25 1
AMD USD 410.086 2014-07-25 1
ANG USD 1.787 2014-07-25 1
AOA USD 96.952626 2014-07-25 1
ARS USD 8.169642 2014-07-25 1
AUD USD 1.062844 2014-07-25 1
AWG USD 1.79 2014-07-25 1
AZN USD 0.784067 2014-07-25 1
BAM USD 1.453024 2014-07-25 1
BBD USD 2 2014-07-25 1
BDT USD 77.61971 2014-07-25 1
BGN USD 1.452543 2014-07-25 1
Content
Structure
Filter
Page 1 of 342
currency
MULTIPLE
base
USD
rate
MULTIPLE
date
2014-07-25
source_id
forex_source
id
name
1
Open Exchange R...

LAB