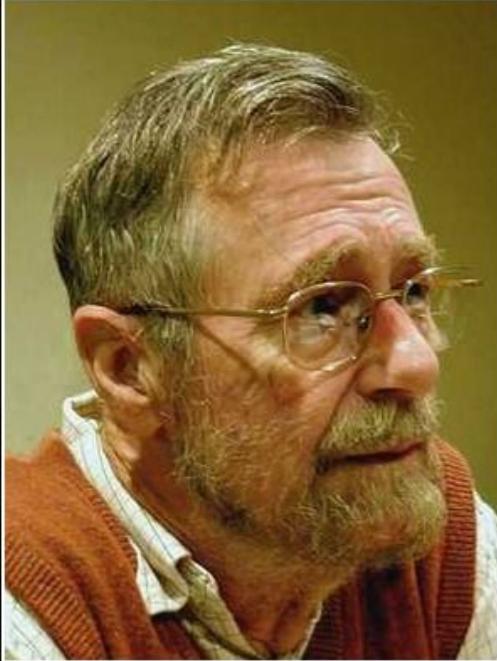


GlideRecord

Interacting with the ServiceNow Database

The Paradox of Programming



If debugging is the process of removing software bugs, then programming must be the process of putting them in.

— *Edsger Dijkstra* —

I don't see why the hell not

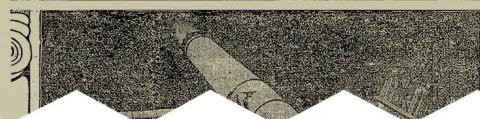
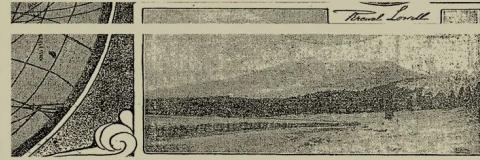
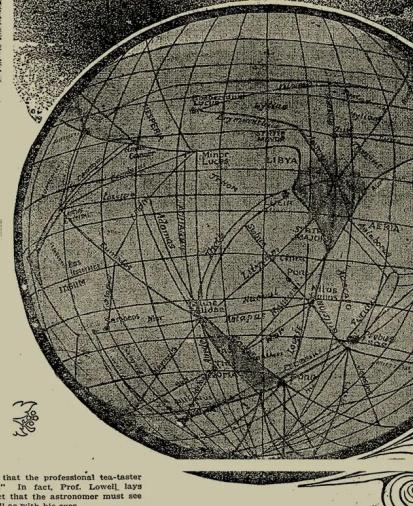
THERE IS LIFE ON THE PLANET MARS

*Prof. Percival Lowell
recognised as the
greatest authority
on the subject, de-
clares there can be
no doubt that living
beings inhabit our
neighbor world.*

ByLilian Whiting:

LHE legions of canals on Mars, forming a colossal network which planned geologists believe to be man-made, the vast deserts which make up the surface of this planet, are an unanswerable argument for the existence of conscious, intelligent life. A thing must be able to live to do such a thing. This theory, of course, was Paley's favorite assertion, but it is none the worse for that. Schliemann discovered 104 canals from Percival Lowell and his staff at the Lowell Observatory at Flagstaff, Arizona, have discovered over 350, and they regard this number as no limit. These canals, moreover, are, in the large asteroids discovered first; but in each opposition of the planet the trained sight will tell the geologist who has not gazed at the celestial expert

It is safe to assume that the professional astronomer can be discredited. In fact, Prof. Lowell lays emphasis on the fact that the astronomer sees with his mind as well as with his eyes.
In consultation with Prof. W. H. Pickering and other eminent astronomers the site of the new observatory for the Harvard mission was to be "a city of Mars" in Arizona, says Prof. Lowell.



Moving Beyond Raw Information

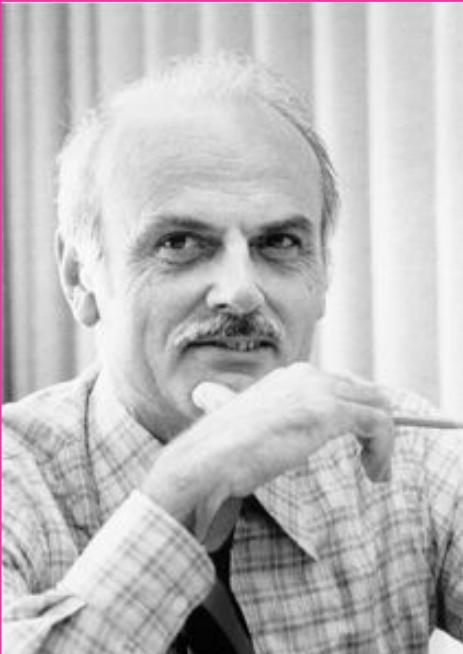
“It is a capital mistake to theorize before one has data.”

Sherlock Holmes



IN THE WORLD OF
DATA
WHAT MATTERS MOST
IS NOT JUST THE
DATA ITSELF
BUT WHAT BUT
DATA ITSELF
WHAT YOU DO
WITH IT.
-DENNIS MCKENNA.
DENNIS MCKENNA

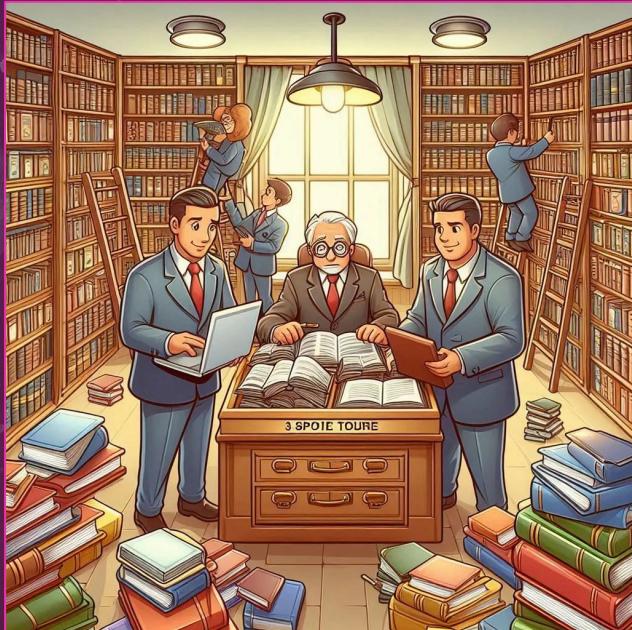
More Than Just Storing Data



"The relational model is not just a way to store data but a way to think about data."

- **Edgar Frank "Ted" Codd** was an English computer scientist who, while working for IBM, invented the relational model for database management, the theoretical basis for relational databases and relational database management systems.

Where Data Lives



- A database is an organized collection of data that is stored, managed, and accessed electronically.
- Databases are designed to efficiently handle large amounts of information and allow users to retrieve, update, or manipulate data in a structured manner.
- The entire database is like the library.
- Just as a library houses a collection of books, a database contains tables

Your Favorite Book Collection

- A table in a database is a collection of related data entries organized into rows and columns.
- Each row represents a single record or entity, while each column represents a specific attribute or field related to that entity.
- Tables in a database are like bookshelves in the library.
- Each table stores specific information, much like each bookshelf holds books of a particular genre or category.
- Different sections, such as Fiction, Non-Fiction, Reference, etc., correspond to different tables within the database.



Where Every Row is a Story!



- Rows on a table are like books on a shelf.
- Each row represents a single record or entry, similar to how each book represents a specific item within the category.
- Columns in a table are like the specific details of each book.
- Each column represents a particular attribute or information about the records, similar to how each piece of book information (title, author, etc.) describes the book.

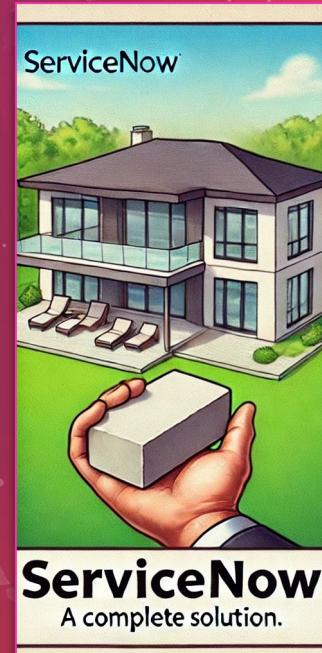
The magical language of databases

- SQL was one of the first commercial languages to use Edgar F. Codd's relational model.
- SQL (Structured Query Language) is a programming language specifically designed for managing and manipulating relational databases.
- It provides a standard way to perform tasks such as querying, updating, inserting, and deleting data within a database.



Where SQL Gets a Day Off!

- ServiceNow instances are hosted on infrastructure that uses MariaDB databases to store all platform data.
- The database holds tables, records, configurations, and platform metadata.
- Users interact with a high-level interface (e.g., forms, lists, dashboards) or APIs (like GlideRecord) that abstract database complexity.
- ServiceNow shields users from writing SQL queries, providing a low-code/no-code environment for data handling.



What's Your Instance Running?

servicenow All

Favorites History Workspaces Admin ServiceNow ★ Domain scope: global Application scope: Service Graph Connector for Microsoft S... Q Search

information

FAVORITES

Database - Information

ALL RESULTS

- > Customer Central
- > Enterprise Architecture
- > Policy and Compliance
- > CSDM
- > Risk
- ✓ System Diagnostics
 - Database
 - Information

★

Information

Type	mysql
Version	5.8.20-MariaDB-log
Driver	MariaDB connector/J
JDBC	2.3.0

Connections

0 free	glide.awa.assigner.77 (Background transaction):SELECT RELEASE_LOCK('dev273018_1.awa-claim-lock') /* dev273018002, gs:SYSTEM, tx:a7bd1c7383561610a7586260ceaad311, hash:1341245160 */
1 free	glide.awa.assigner.77 (Background transaction):SELECT RELEASE_LOCK('dev273018_1.awa-timeout-job-processing-lock') /* dev273018002, gs:SYSTEM, tx:885894b3831610a7586260ceaad311, hash:1341245160 */
2 in use (1734014837385 ms)	"unknown"
3 in use (1734014837385 ms)	"unknown"
4 in use (1734014837385 ms)	"unknown"
5 in use (1734014837385 ms)	"unknown"
6 in use (1734014837386 ms)	"unknown"
7 in use (1734014837386 ms)	"unknown"
8 in use (1734014837386 ms)	"unknown"
9 in use (1734014837386 ms)	"unknown"
10 in use (1734014837386 ms)	"unknown"
11 in use (1734014837386 ms)	"unknown"
12 free	glide.scheduler (Background transaction):SELECT sys_scheduler_assignment0.`assigned_to` FROM sys_scheduler_assignment sys_scheduler_assignment0 /* dev273018002, gs:SYSTEM, tx:a3bdddcb383561610a7586260ceaad311, hash:1341245160 */
13 free	glide.scheduler (Background transaction):SELECT sys_trigger0.`sys_id`, sys_trigger0.`job_classification`, sys_trigger0.`priority`, sys_trigger0.`system_id`, sys_trigger0.`next_action`, sys_trigger0.`priority`, sys_trigger0.`next_action`, sys_trigger0.`sys_id`, sys_trigger0.`job_classification`, sys_trigger0.`priority`, sys_trigger0.`system_id`, sys_trigger0.`next_action` /* dev273018002, gs:SYSTEM, tx:a3bdddcb383561610a7586260ceaad311, hash:1341245160 */
14 free	glide.scheduler (Background transaction):SELECT sys_trigger0.`sys_id`, sys_trigger0.`job_classification`, sys_trigger0.`priority`, sys_trigger0.`system_id`, sys_trigger0.`next_action` /* dev273018002, gs:SYSTEM, tx:a3bdddcb383561610a7586260ceaad311, hash:1758913628 */
15 free	glide.scheduler (Background transaction):SELECT sys_trigger0.`claimed_by` AS `claimed_by`, count(*) AS count0 FROM sys_trigger sys_trigger0 WHERE sys_trigger0.`claimed_by` = 'app129195.ais191.service-now.com:dev273018002' /* dev273018002, gs:SYSTEM, tx:a3bdddcb383561610a7586260ceaad366, hash:1819674779 */
16 free	glide.scheduler.worker.2 (Flow Engine Interactive Event Handler):UPDATE sys_event_processor SET `sys_mod_count` = 488350, `sys_updated_on` = '2024-12-12 14:47:17', `trigger_ids` = 'b56febff5347330acb8ddeef7b12f1,11e5dbe5e27a46698daaf5cbd3718ecc,f2aa16d3833d9610a7586260ceaad366,e2db040783' WHERE sys_event_processor.`trigger_id` = 'app129195.ais191.service-now.com:dev273018002' /* dev273018002, gs:SYSTEM, tx:a3bdddcb383561610a7586260ceaad366, hash:1819674779 */
17 free	glide.scheduler.worker.2 (Flow Engine Interactive Event Handler):UPDATE sys_scheduler_assignment SET `sys_mod_count` = 488350, `sys_updated_on` = '2024-12-12 14:47:17', `trigger_ids` = 'app129195.ais191.service-now.com:dev273018002' /* dev273018002, gs:SYSTEM, tx:a3bdddcb383561610a7586260ceaad366, hash:1819674779 */
18 free	glide.scheduler.worker.2 (Flow Engine Interactive Event Handler):UPDATE sys_scheduler_assignment SET `sys_mod_count` = 488350, `sys_updated_on` = '2024-12-12 14:47:17', `trigger_ids` = 'app129195.ais191.service-now.com:dev273018002' /* dev273018002, gs:SYSTEM, tx:a3bdddcb383561610a7586260ceaad366, hash:1819674779 */
19 free	glide.scheduler.worker.2 (Flow Engine Interactive Event Handler):UPDATE sys_scheduler_assignment SET `sys_mod_count` = 488350, `sys_updated_on` = '2024-12-12 14:47:17', `trigger_ids` = 'app129195.ais191.service-now.com:dev273018002' /* dev273018002, gs:SYSTEM, tx:a3bdddcb383561610a7586260ceaad366, hash:1819674779 */
20 free	glide.scheduler.worker.2 (Flow Engine Interactive Event Handler):UPDATE sys_scheduler_assignment SET `sys_mod_count` = 488350, `sys_updated_on` = '2024-12-12 14:47:17', `trigger_ids` = 'app129195.ais191.service-now.com:dev273018002' /* dev273018002, gs:SYSTEM, tx:a3bdddcb383561610a7586260ceaad366, hash:1819674779 */
21 free	glide.scheduler.worker.2 (Flow Engine Interactive Event Handler):UPDATE sys_scheduler_assignment SET `sys_mod_count` = 488350, `sys_updated_on` = '2024-12-12 14:47:17', `trigger_ids` = 'app129195.ais191.service-now.com:dev273018002' /* dev273018002, gs:SYSTEM, tx:a3bdddcb383561610a7586260ceaad366, hash:1819674779 */
22 free	glide.scheduler.worker.2 (Flow Engine Interactive Event Handler):UPDATE sys_scheduler_assignment SET `sys_mod_count` = 488350, `sys_updated_on` = '2024-12-12 14:47:17', `trigger_ids` = 'app129195.ais191.service-now.com:dev273018002' /* dev273018002, gs:SYSTEM, tx:a3bdddcb383561610a7586260ceaad366, hash:1819674779 */
23 free	glide.scheduler.worker.2 (Flow Engine Interactive Event Handler):UPDATE sys_scheduler_assignment SET `sys_mod_count` = 488350, `sys_updated_on` = '2024-12-12 14:47:17', `trigger_ids` = 'app129195.ais191.service-now.com:dev273018002' /* dev273018002, gs:SYSTEM, tx:a3bdddcb383561610a7586260ceaad366, hash:1819674779 */
24 free	glide.scheduler.worker.2 (Flow Engine Interactive Event Handler):UPDATE sys_scheduler_assignment SET `sys_mod_count` = 488350, `sys_updated_on` = '2024-12-12 14:47:17', `trigger_ids` = 'app129195.ais191.service-now.com:dev273018002' /* dev273018002, gs:SYSTEM, tx:a3bdddcb383561610a7586260ceaad366, hash:1819674779 */

Default-thread-38 (/system_diagnostic_database_page.do):SELECT CURRENT_TIMESTAMP() /* dev273018002, gs:8421EEA7835E9210A7586260CEAAD36B, tx:2fbb03b */

Default-thread-38 (/system_diagnostic_database_page.do):SELECT sys_cache_flush0.sequence, sys_cache_flush0.sys_id, sys_cache_flush0.system_id, sys_cache_flush0.cache_hash:1819674779 /

Default-thread-38 (/system_diagnostic_database_page.do):SELECT sys_user_token0.expires, sys_user_token0.last_token, sys_user_token0.login_method, sys_user_token0.sys_id, sys_user_token0.token:1815999126 /

glide.amb.cluster.synchronizer:SELECT sys_amb_message00060.to_user, sys_amb_message00060.channel, sys_amb_message00060.sys_mod_count, sys_amb_message00060.message00060 /* dev273018002, gs:8421EEA7835E9210A7586260CEAAD36B, tx:2fbb03b */

glide.scheduler.worker.1 (VA Event Processor 0):SELECT sys_trigger0.parent, sys_trigger0.log, sys_trigger0.entered_time, sys_trigger0.business_calendar, sys_trigger0.duration, sys_trigger0.offset, sys_trigger0.system_id, sys_trigger0.sys_mod_count, sys_trigger0.run_month, sys_trigger0.priority, sys_trigger0.time_zone, sys_trigger0.job_context /* dev273018002, gs:8421EEA7835E9210A7586260CEAAD36B, tx:2fbb03b */

glide.scheduler.worker.6 (Process Automation Event Handler):SELECT sys_trigger0.parent, sys_trigger0.log, sys_trigger0.entered_time, sys_trigger0.business_calendar, sys_trigger0.duration, sys_trigger0.offset, sys_trigger0.system_id, sys_trigger0.priority, sys_trigger0.run_month, sys_trigger0.time_zone, sys_trigger0.job_context /* dev273018002, gs:8421EEA7835E9210A7586260CEAAD36B, tx:2fbb03b */

o PDI dev273018

What's Your Instance Running?

The screenshot shows a ServiceNow instance with the following details:

- Page Title:** stats
- Header:** Favorites, History, Workspaces, Admin, Servlet statistics (with a star icon), and a search bar.
- Left Sidebar (FAVORITES):**
 - Stats (selected)
 - All Results
 - Assessments
 - DevTools
 - System Diagnostics
 - Memory Stats
 - Expression Cache Stats
 - Stats
 - Stats (selected)
 - Scheduled Job History By Node
 - Scheduled Job Assignments
 - Scheduled Job History
 - Running Scheduled Jobs
 - Pending Scheduled Jobs
 - System Events and Jobs Dash...
 - Slow Events
 - Slow Interactions
 - Slow Mutex Locks
- Content Area:**

Database Connection Pool(s)
Primary Prefix: glide

Prefix: glide
DB Name: dev273018_1
Status: Online
Busy: 0
Closed: 0
Available: 32
Shared: 0
SharedBusy: 0
Total: 32
Max: 32
High-Watermark: 32
DB Throttler-adjusted Max: 32
DB Resource Usage (%): 0.0
Lifetime opened: 33
Lifetime closed: 0
Average open time: 1
Time in retries: 18:47:32.950
Valve Open Count: 0
Features: []
Open connections:
#0: utilization=1.65% last-used-ago=20 state=free age=3:52:20.357
#1: utilization=1.68% last-used-ago=19 state=free age=3:52:20.354
#2: utilization=1.76% last-used-ago=19 state=free age=3:52:20.353
#3: utilization=1.59% last-used-ago=18 state=free age=3:52:20.352
#4: utilization=1.77% last-used-ago=103 state=free age=3:52:20.350
#5: utilization=1.82% last-used-ago=96 state=free age=3:52:20.349
#6: utilization=1.87% last-used-ago=98 state=free age=3:52:20.348
#7: utilization=1.57% last-used-ago=98 state=free age=3:52:20.346
#8: utilization=1.86% last-used-ago=102 state=free age=3:52:20.345
#9: utilization=1.92% last-used-ago=96 state=free age=3:52:20.344
#10: utilization=1.76% last-used-ago=92 state=free age=3:52:20.343
#11: utilization=1.66% last-used-ago=95 state=free age=3:52:20.342
#12: utilization=1.60% last-used-ago=93 state=free age=3:52:20.341
#13: utilization=1.73% last-used-ago=92 state=free age=3:52:20.340
#14: utilization=1.66% last-used-ago=85 state=free age=3:52:20.339
#15: utilization=1.62% last-used-ago=84 state=free age=3:52:20.338
#16: utilization=1.60% last-used-ago=83 state=free age=3:52:20.336
#17: utilization=1.60% last-used-ago=83 state=free age=3:52:20.335
#18: utilization=1.67% last-used-ago=81 state=free age=3:52:20.334
#19: utilization=1.86% last-used-ago=80 state=free age=3:52:20.333
#20: utilization=1.76% last-used-ago=80 state=free age=3:52:20.332
#21: utilization=1.75% last-used-ago=47 state=free age=3:52:20.331
#22: utilization=1.74% last-used-ago=28 state=free age=3:52:20.330
#23: utilization=1.70% last-used-ago=25 state=free age=3:52:20.329
#24: utilization=2.29% last-used-ago=25 state=free age=3:52:20.328
#25: utilization=2.07% last-used-ago=25 state=free age=3:52:20.327

What's Your Instance Running?

servicenow All

debug sql

FAVORITES

Session Debug - Debug SQL

ALL RESULTS

System Diagnostics

Session Debug

Debug SQL

Debug SQL (Detailed)

Debug SQL (Slow)

Debug SQL (Large Tables)

Incident - INC0014707

Follow Save Update Resolve Delete List Find Sys ID What Runs

Number: INC0014707
Opened: 12/05/2015 21:37:36
Opened by: Kyle Lindauer
* Company:
* Caller: Joey Sedore
Location: 136 North Grand Avenue, West Cov
Category: Datacenter Automation
Subcategory: Reclaim Storage
Configuration item: Precision T5500 Workstation
Impact: 3 - Low
Urgency: 3 - Low
Priority: 4 - Low

Script Debugger

Session Log

Transactions

7917 /api/now/uxmetrics/interactions 8
7918 /list_history.do 16

7918 /list_history.do Page: list_history Query Count: 4 Complete: false Debug: false
06:55:35.200 Time: 0:00:00.000 for: dev273018_1[glide.16] [188074461]
SELECT *** FROM task0 ignore index(sys_updated_on) WHERE task0.sys_class_name = 'incident' AND task0.sys_id = 'dabb21d2d7600200f215a5f75e61034e' ORDER BY task0.sys_updated_on DESC limit 0,30 *** +
06:55:35.205 Time: 0:00:00.003 for: dev273018_1[glide.17] [922809647]
SELECT *** FROM sys_email sys_email WHERE sys_email.'instance' = 'dabb21d2d7600200f215a5f75e61034e' AND sys_email.'sys_created_on' >= 2015-06-10 21:37:36' AND (sys_email.'headers' NOT LIKE '%emailDigest%' OR sys_email.'headers' IS NULL) AND sys_email.'type' IN ('sent', 'received') ORDER BY sys_email.'sys_created_on' DESC ,sys_email.'sys_id' limit 0,251 *** +
06:55:35.209 Time: 0:00:00.001 for: dev273018_1[glide.18] [1810986317]
SELECT *** FROM task task0 WHERE task0.sys_class_name = 'incident' AND task0.sys_id = 'dabb21d2d7600200f215a5f75e61034e' *** +
task0.parent_task0.a.ref_3 AS 'cause_by', '273018_1[glide.19] [2068858184]
task0.watch_list,task0.upon_reject,task0.on_approval,task0.sys_audit_relation0 WHERE task0.sys_updated_on ,task0.a.str_5 AS 'origin_table',task0.approval_history,task0.state,task0.number,task0.sns ,task0.knowledge,task0.sys_created_by ,task0.order ,task0.delivery_plan,task0.cmdb_ci ,dev273018_1[glide.20] [236788045]
task0.cmdb_ci_business_app,task0.impact ,task0.history.set0 WHERE task0.contract ,task0.active ,task0.set0 [02020f215a5f75e61034e] AND task0.work_notes_list ,task0.sn_hr_le_activity ,task0.priorty ,task0.sys_domain_path ,task0.sys_history_set0.language = task0.rejection_goto ,task0.group_list ,task0.caLoc ,task0.business_duration ,task0.a.str_13 AS 'src_vendor_point_of_contact' ,task0.credit_cards ,task0.approval_set ,dev273018_1[glide.22] [951812595]

What's Your Instance Running?

The screenshot shows the ServiceNow Tables & Columns interface. The top navigation bar includes 'Favorites', 'History', 'Workspaces', 'Admin', 'Tables & Columns' with a star icon, and a search bar. The left sidebar has 'FAVORITES' and a list of 'ALL RESULTS' sections: 'System Definition', 'Tables & Columns' (which is expanded), 'System Security', 'Identity and Access Audit', and 'Configure Tables & Fields'. The main content area displays a list of tables under 'Table Names'. One table, 'Incident [incident]', is selected and highlighted with a blue border. To the right of the table list is a 'Column Names' and 'Column Attributes' table. The 'Column Attributes' table lists various fields for the 'incident' table, such as 'Accounting period', 'Active', 'Activity', 'Activity due', 'Actual end', 'Actual start', 'Additional', 'assignee list', 'Comments', 'Approval', 'Approval history', 'Approval set', 'Assigned to', 'Assignment group', 'Business duration', 'Business impact', and 'Business resolve'. At the bottom of the page are buttons for 'Edit Table', 'Schema map', and 'Delete all records'. A small blue footer bar at the bottom left contains the text 'PDI dev273018'.

Click a button to create a new table or application, or browse all applications.

Create Table | Create Application | Browse Applications

Or, select a table to browse its columns and indices.

Table Names

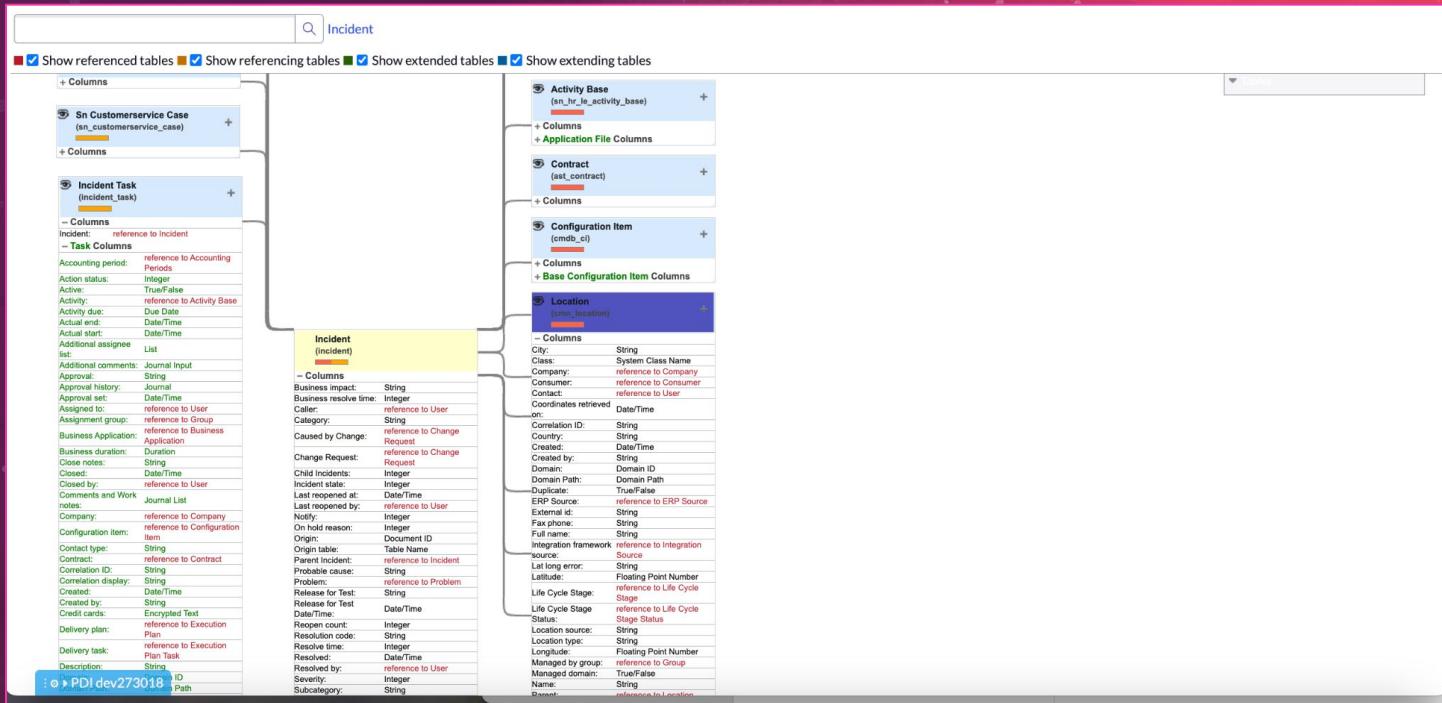
Column Names	Column Attributes
Fields (incident)	
Accounting period	
Action status	
Active	
Activity	
Activity due	
Actual end	
Actual start	
Additional	
assignee list	
Comments	
Approval	
Approval history	
Approval set	
Assigned to	
Assignment group	
Business duration	
Business impact	
Business resolve	

Inbound Queue Record [ih_sync_inbound_queue_record]
Inbound Record (Legacy) [ih_sync_inbound_record]
Inbox Layout [awa_inbox_layout]
Incident [incident]
Incident Fact Table [incident_fact_table]
Incident KCS Article [kb_template_incident_kcs_article]
Incident Metric [incident_metric]
Incident SLA [incident_sla]
Incident spotlight [incident_spotlight]
Incident Task [incident_task]
Incident Time Worked [incident_time_worked]
Incident-KCS article - HTML [kb_template_incident_kcs_article.html]
Incidents Related To Business Apps [sn_dpm_incident_related_to_business_app]
Inclusion Endpoint [cmdb_ci_endpoint_inclusion]
Incomplete IP Identified Device [cmdb_ci_incomplete_ip]
Incremental Offline Deny List [sys_sg_incremental_offline_exclusion_list]
Incremental offline result [sys_sg_incremental_result]
Index Explanation [sys_index_explain]
Index Hint Rewrite [sys_query_index_hint]
Index History [sn_airesearch_global_job_ingest_history]
Index Suggestion [sys_index_suggestion]
Index Suggestion Executions [sys_index_suggestion_exec]
Index Suggestion Rule [sys_suggestion_rule]
Indexes [sys_index]
Indicator [sn_grc_indicator]

Edit Table | Schema map | Delete all records

PDI dev273018

What's Your Instance Running?



The Story of a Sea Lion

Is MariaDB
the better MySQL?



vs.



- MariaDB is a community-developed, commercially supported fork of the MySQL relational database management system (RDBMS), intended to remain free and open-source software under the GNU General Public License.
- Development is led by some of the original developers of MySQL, who forked it due to concerns over its acquisition by Oracle Corporation in 2009.
- MariaDB is intended to maintain high compatibility with MySQL, with exact matching with MySQL APIs and commands, allowing it in many cases to function as a drop-in replacement for MySQL.

The Story of a Sea Lion

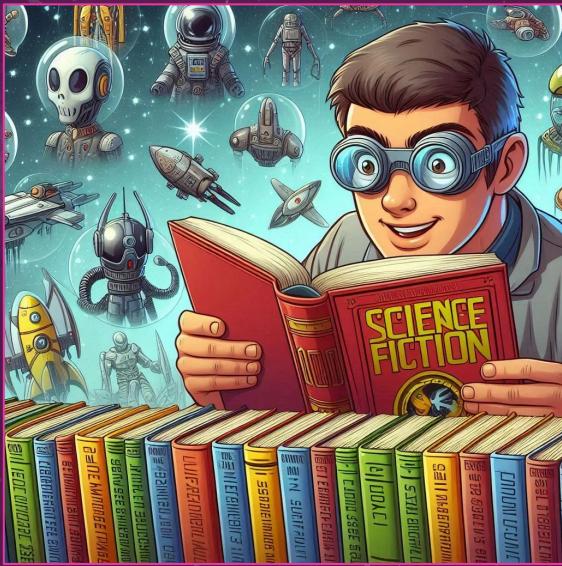


- **Ulf Michael Widenius** (born 3 March 1962), also known as **Monty**, is a Finnish software programmer.
- He is the main author of the original version of the open source MySQL database, a founding member of the MySQL AB company and CTO of the MariaDB Corporation AB.

""It happened when Monty and his older daughter My were snorkeling on one of the islands in the Galapagos. Something big, brown and fast suddenly appeared at an arm's distance, laughing in their faces. Fond memories of this fast and funny creature, scaring the tourists, popped into Monty's mind when asked picking a logo for MariaDB. He wanted to adhere to the tradition of animals as symbols of Open Source projects."

- Widenius has three children – **My**, **Max**, and **Maria** – who inspired the names for MySQL, MaxDB and the MySQL-Max distribution, and MariaDB.

SELECT Query: Finding Books



`SELECT title, author FROM Books WHERE genre = 'Science Fiction';`

- You want to find all science fiction books and learn their titles and authors.
- `SELECT`: Tell the librarian what information you want (titles and authors).
- `FROM`: Specifies the section you're looking in (Books shelf).
- `WHERE`: A filter is added to find only Science Fiction books.

GlideRecord

- The GlideRecord class is the way to interact with the ServiceNow database from a script.
- GlideRecord interactions start with a database query.
- The generalized strategy is:
 - a. Create a GlideRecord object for the table of interest.
 - b. Build the query condition(s).
 - c. Execute the query.
 - d. Apply script logic to the records returned in the GlideRecord object.

GlideRecord(String tableName)

- Creates an instance of the GlideRecord class for the specified table

```
var incGr = new GlideRecord('incident');
```

query()

- Runs a query against the table based on the filters specified by query methods such as addQuery() and addEncodedQuery().
- This method fails if there is a field in the table called "query".
- If that is the case, use the method _query().
- To run queries in a domain-separated instance, use the method queryNoDomain().

```
var incGr = new GlideRecord('incident');  
incGr.query();
```

next()

- Moves to the next record in the GlideRecord.
- Use this method to iterate through the records returned by a GlideRecord query.
- This method fails if there is a field in the table called "next". If that is the case, use the method `_next()`.
- There are several strategies for iterating through returned records.
- The `next()` method and a `while` loop iterates through all returned records to process script logic:

```
var incGr = new GlideRecord('incident');

incGr.query();

while (incGr.next()) {

}
```

next()

- The next() method and an if processes only the first record returned.

```
var incGr = new GlideRecord('incident');

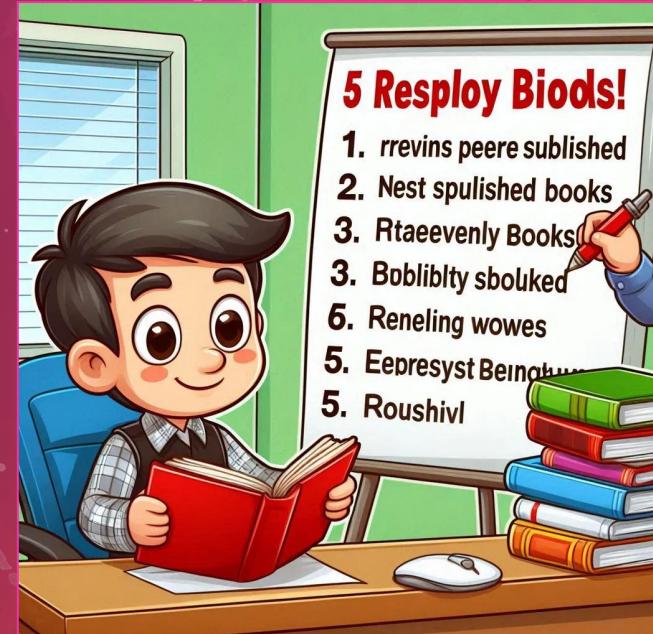
incGr.query();

if (incGr.next()) {
}
```

LIMIT: Getting a Subset of Books

```
SELECT title, author FROM Books ORDER BY year  
DESC LIMIT 5;
```

- You want to list the 5 most recently published books.
- SELECT title, author: Tells the librarian you want titles and authors. FROM Books: Specifies the section.
- ORDER BY year DESC: Orders the books by publication year in descending order (newest first).
- LIMIT 5: Only shows the top 5 results.



setLimit(Number limit)

- Sets the maximum number of records to return in the GlideRecord from a query.

```
var incGr = new GlideRecord('incident');

incGr.setLimit(3);

incGr.query();

while (incGr.next()) {

}
```

`addExtraField(String dotWalkedField)`

- Queries one or more dot-walked fields from a form or script in a single request.
- The `addExtraField()` method allows you to query dot-walked fields in a single database request, rather than perform multiple queries per dot-walked element in a form or script (which requires multiple round trips to the database).
- Levels of field names are separated by dots (periods).
- The format of dotWalkedField follows the left-to-right order of fields in a dotwalked form or script.
- The `addExtraField()` method optimizes the querying of the dot-walked fields.

addExtraField(String dotWalkedField)

```
var incGr = new GlideRecord('incident');

incGr.addEncodedQuery("active=true");

incGr.addExtraField('caller_id.location.name')

incGr.query();

if (incGr.next()) {

    gs.info(incGr.caller_id.location.name);

}
```

Retrieve values from records

- In most cases, don't use dot-walking to get values from a record.
- Dot-walking retrieves the entire object instead of the field value.
- Retrieving the object uses more storage and might cause undesirable results when used in arrays or in Service Portal.
- Instead of retrieving the entire object, you can use one of the following methods to copy the field values:
 - `getValue()`
 - `getDisplayValue()`

Retrieve values from records

- If dot-walking through a GlideElement object is necessary, use the `toString()` method to retrieve values.
- For example, you might need the current caller's manager sys_id to set another reference field.

```
var incGr = new GlideRecord('incident');

incGr.query();

if (incGr.next()) {
    var caller=incGr.caller_id.toString();

    gs.info(typeof caller);

}
```

getValue(String fieldName)

- Retrieves the string value of a specified field or the string value of an attribute in a dynamic attribute store.
- Returns null if the field is empty or the field doesn't exist.
- Boolean values return as "0" and "1" string values instead of false and true.

```
var incGr = new GlideRecord('incident');

incGr.setLimit(3);

incGr.query();

while (incGr.next()) {

    gs.info(incGr.getValue("priority"));

}
```

getDisplayValue(String name)

- Retrieves the display value for the current record or the display value of an attribute in a dynamic attribute store.

```
var incGr = new GlideRecord('incident');

incGr.setLimit(3);

incGr.query();

while (incGr.next()) {

    gs.info(incGr.getDisplayValue("priority"));

}
```

getUniqueValue()

- Gets the primary key of the record, which is usually the sys_id unless otherwise specified.

```
var incGr = new GlideRecord('incident');

incGr.setLimit(3);

incGr.query();

while (incGr.next()) {

    gs.info(incGr.getUniqueValue());

}
```

getRecordClassName()

- Retrieves the class (table) name for the current record.

```
var incGr = new GlideRecord('incident');

incGr.query();

if (incGr.next()) {
    gs.info(incGr.getRecordClassName());
}
```

`addQuery(String name, Object operator, Object value)`

- Provides the ability to build a request, which when executed, returns the rows from the specified table that match the request.
- If you're familiar with SQL, this method is similar to the "where" clause.
- You can make one or more `addQuery()` calls in a single query.
- For this method the queries are AND'ed.
- `addQuery()` is typically called with three parameters; table field, operator, and comparison value.
- It can be called with only two parameters, table field and comparison value, The operator in this case is assumed to be "equal to".

`addQuery(String name, Object operator, Object value)`

```
var incGr = new GlideRecord('incident');

incGr.addQuery("active", "=", true);

incGr.query();

if (incGr.next()) {

gs.info(incGr.getDisplayValue('number'));

}

var incGr = new GlideRecord('incident');

incGr.addQuery("active", "=", true);

incGr.query();

if (incGr.next()) {

gs.info(incGr.getDisplayValue('number'));

}
```

addQuery(String name, Object operator, Object value)

- The addQuery operators are:
 - Numbers: =, !=, >, >=, <, <=
 - Strings: =, !=, STARTSWITH, ENDSWITH, CONTAINS, DOES NOT CONTAIN, IN, NOT IN, INSTANCEOF
- When there are multiple queries, each additional clause is treated as an AND.
- Queries with no query conditions return all records from a table.
- If a malformed query executes in runtime, all records from the table are returned.
- An incorrectly constructed encoded query, such as including an invalid field name, produces an invalid query.
- When the invalid query is run, the invalid part of the query condition is dropped, and the results are based on the valid part of the query, which may return all records from the table.
- You can set the `glide.invalid_query.returns_no_rows` system property to 'true' to have queries with invalid encoded queries return no records.

addActiveQuery()

- Adds a filter to return active records.

```
var incGr = new GlideRecord('incident');

incGr.addActiveQuery();

incGr.query();

if (incGr.next()) {
    gs.info(incGr.getDisplayValue('number'));
}
```

`addInactiveQuery()`

- Adds a filter to return inactive records.

```
var incGr = new GlideRecord('incident');

incGr.addInactiveQuery();

incGr.query();

if (incGr.next()) {
    gs.info(incGr.getDisplayValue('number'));
}
```

`addNotNullQuery(String fieldName)`

- Adds a filter to return records where the specified field is not `null`.

```
var incGr = new GlideRecord('incident');

incGr.addNotNullQuery("short_description");

incGr.query();

if (incGr.next()) {

    gs.info(incGr.getDisplayValue('number'));

}
```

`addNullQuery(String fieldName)`

- Adds a filter to return records where the specified field is `null`.

```
var incGr = new GlideRecord('incident');

incGr.addNullQuery("assigned_to");

incGr.query();

if (incGr.next()) {
    gs.info(incGr.getDisplayValue('number'));
}
```

UNION: Combining Results from Two Queries

```
SELECT title FROM Books WHERE genre = 'Science  
Fiction' UNION SELECT title FROM Books WHERE  
year > 2000;
```

- You want to list books that are either Science Fiction or were published after the year 2000.
- SELECT title FROM Books WHERE genre = 'Science Fiction': Finds titles of Science Fiction books.
- UNION: Combines results from both queries.
- SELECT title FROM Books WHERE year > 2000: Finds titles of books published after 2000.



Subqueries: Finding Books by Popular Authors



```
SELECT title FROM Books WHERE author_id IN  
(SELECT id FROM Authors WHERE popularity > 8);
```

- You want to find books written by very popular authors ($\text{popularity} > 8$).
- `SELECT title FROM Books WHERE author_id IN:` Tells the librarian to find books written by certain authors.
- `(SELECT id FROM Authors WHERE popularity > 8):` Finds authors with a popularity rating greater than 8.

`addEncodedQuery(String query, Boolean enforceFieldACL)`

- Adds an encoded query to other queries that may have been set.
- If there are multiple conditions in query, the conditions are ANDed.
- To use ORs or create technically complex queries, use encoded queries.

```
var incGr = new GlideRecord('incident');

incGr.addEncodedQuery("active=true");

incGr.query();

if (incGr.next()) {

    gs.info(incGr.getDisplayValue('cmdb_ci'));

}
```

getEncodedQuery()

- Retrieves the query condition of the current result set as an encoded query string.

```
var incGr = new GlideRecord('incident');

incGr.addActiveQuery();

incGr.query();

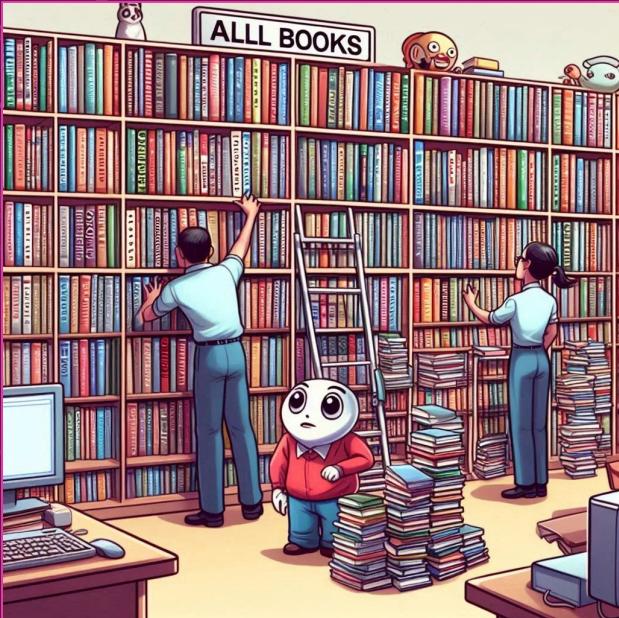
gs.info(incGr.getEncodedQuery());

if (incGr.next()) {

    gs.info(incGr.caller_id.location.name);

}
```

ORDER BY: Sorting Books



SELECT title, author FROM Books ORDER BY title ASC;

- You want to list all books sorted by title in ascending order.
- SELECT title, author: Tells the librarian you want the titles and authors.
- FROM Books: Specifies the section.
- ORDER BY title ASC: Orders the books by title in ascending (A-Z) order.

orderBy(String fieldName)

- Specifies a field name, or path to an attribute within a dynamic attribute store, to use to order the query set.
- To order by multiple fields, call this method multiple times with different field values.

```
var incGr = new GlideRecord('incident');

incGr.orderBy('number');

incGr.setLimit(5);

incGr.query();

while (incGr.next()) {
    gs.info(incGr.getDisplayValue());
}
```

orderByDesc(String fieldName)

- Specifies the field, or an attribute in a dynamic attribute store, to use to order the query set in descending order.

```
var incGr = new GlideRecord('incident');
incGr.orderByDesc('number');
incGr.setLimit(5);
incGr.query();
while (incGr.next()) {
    gs.info(incGr.getDisplayValue());
}
```

getLink(Boolean noStack)

- Retrieves the link for the current record.

```
var incGr = new GlideRecord('incident');

incGr.orderBy('number');

incGr.setLimit(1);

incGr.query();

while (incGr.next()) {

    gs.info(incGr.getLink(true));

}
```

get(Object name, Object value)

- Returns the specified record in the current GlideRecord object.
- This method accepts either one or two parameters.
- If only a single parameter is passed in, the method assumes that it is the sys_id of the desired record.
 - If not found, it then tries to match the value against the display value.
- If two parameters are passed in, the first is the name of the column within the GlideRecord to search.
 - The second is the value to search for.
- If multiple records are found, use next() to access the additional records.

get(Object name, Object value)

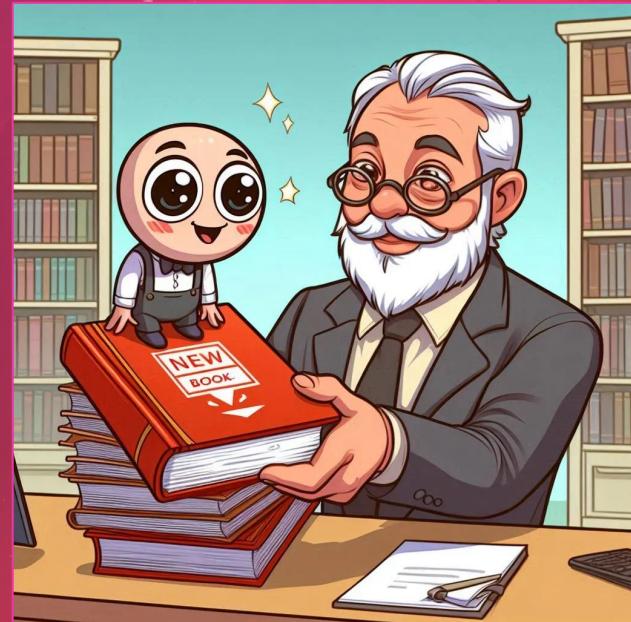
```
var incGr = new GlideRecord('incident');  
incGr.get('1682fd87378c1300023d57d543990e2e');  
gs.info(incGr.getDisplayValue('number'));
```

```
var incGr = new GlideRecord('incident');  
incGr.get('number', "Demo INC0010033");  
gs.info(incGr.getDisplayValue('short description'));
```

INSERT Query: Adding a New Book

```
INSERT INTO Books (title, author, genre, year)  
VALUES ('Dune', 'Frank Herbert', 'Science  
Fiction', 1965);
```

- You want to add a new book to the library.
- **INSERT INTO:** Instructs the librarian to add a new book to the Books section.
- **VALUES:** Provides the details of the new book (Dune, Frank Herbert, Science Fiction, 1965).



UPDATE Query: Updating Book Information



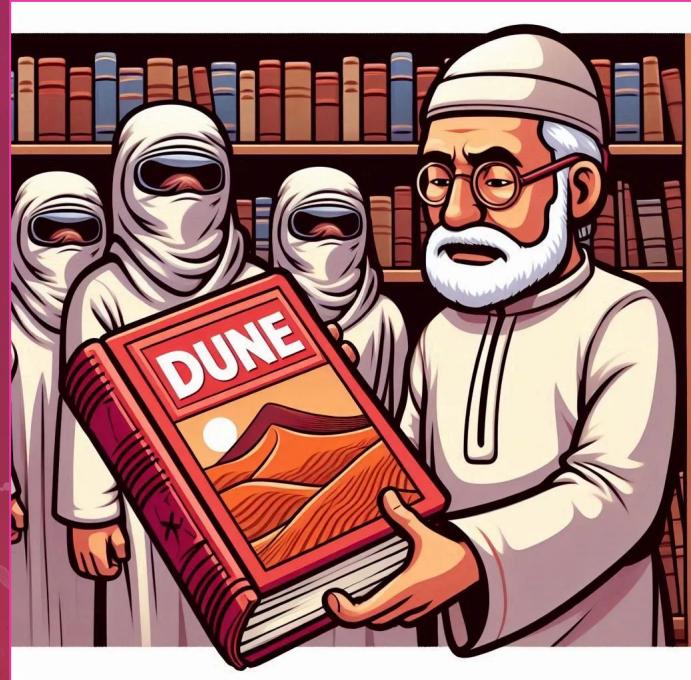
UPDATE Books SET year = 1966 WHERE title = 'Dune';

- You realize the publication year of "Dune" is incorrect and want to update it.
- UPDATE: Tells the librarian to change some details of a book.
- SET: Specifies the new information (year 1966).
- WHERE: Identifies which book to update (title 'Dune').

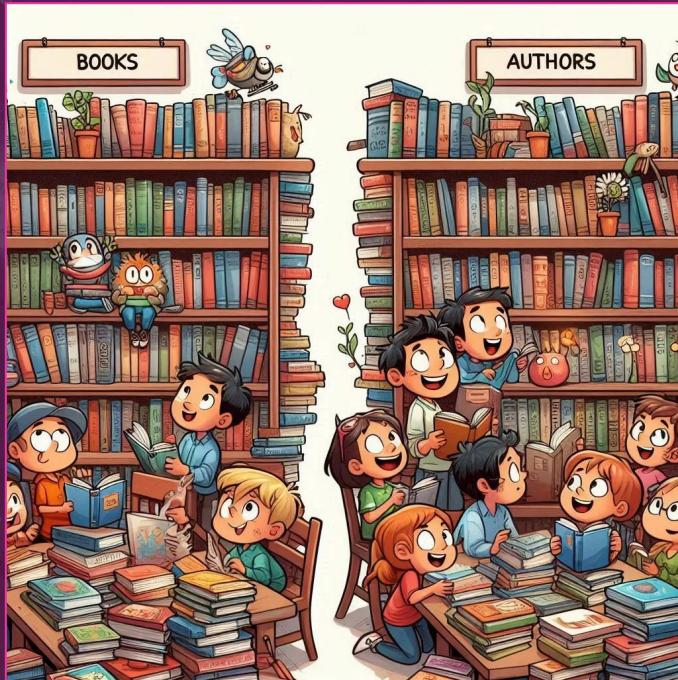
DELETE Query: Removing a Book

DELETE FROM Books WHERE title = 'Dune';

- You want to remove the book "Dune" from the library.
- DELETE FROM: Instructs the librarian to remove a book from the Books section.
- WHERE: Specifies which book to remove (title 'Dune').



JOIN Query: Combining Information from Different Shelves



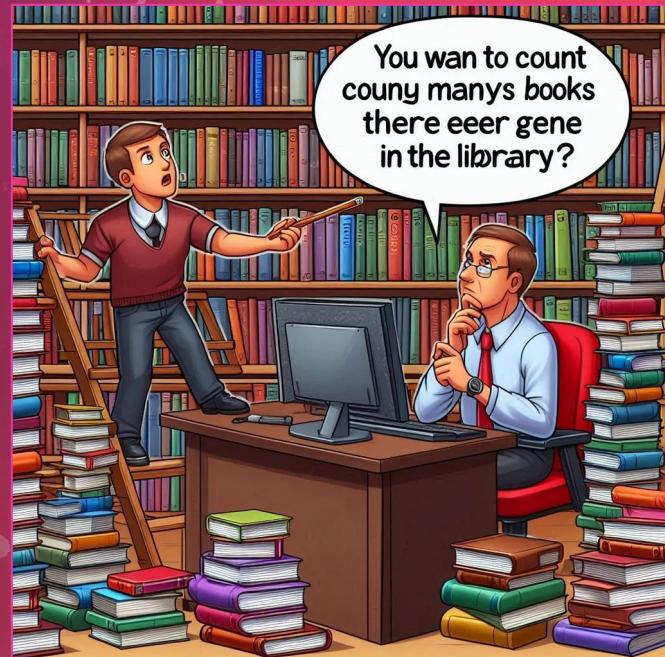
```
SELECT Books.title, Authors.name FROM Books  
JOIN Authors ON Books.author_id = Authors.id;
```

- You want to list all books along with their authors' names by looking at both the Books and Authors shelves.
- JOIN: Asks the librarian to combine information from two sections (Books and Authors).
- ON: Specifies how the information should be matched (Books.author_id = Authors.id).

SELECT with Aggregation: Counting Books

SELECT genre, COUNT (*) FROM Books GROUP BY genre.

- You want to count how many books there are in each genre in the library.
- SELECT genre, COUNT(*) : Tell the librarian you want the genre and the number of books in each genre.
- FROM Books: Specifies you're looking in the Books section.
- GROUP BY genre: Group the books by genre to count them.



EQ

- A complex query is like a story told in SQL; If it's not well-written, it's hard to follow and understand.
- A good SQL query can solve many problems, but a bad one can cause endless headaches.

References

- Mastering SQL
- Server-side Scripting
- GlideRecord
- GlideRecord - Global
- GlideElement
- GlideQueryCondition