|  |
| --- |
| AHK SHORTCUT: ?**trial** |
| //---------------TRIAL\_TABLE & KINDA\_TABLE DIAGRAMS: START---------------------+  //TRIAL TABLE: |  //kind: A string that identifies what type of [trial/test] it is. |  //token\_id: Who owns these [tests/trial]? If you own token, you own test. |  //status: tells us if the test is: created,started,submitted etc. |  //began\_on: universal time in milliseconds that [trial/test] was started. |  //ended\_on: universal time in milliseconds that [trial/test] was submitted. |  // |  //KINDA\_TABLE: |  //kind: Same as in trial\_table. |  //token\_id: Who owns this record? If you own the token, you own the record. |  //challenge\_id: Points to data representing the guts of the trial. |  //has\_challenge: True if challenge\_id is valid. False otherwise. |  //effort\_id: Represents the "effort" the ninja put forth in solving the |  // specific challenge of the trial. |  //has\_effort: True if effort\_id is valid. |  // Else we assume no effort has been made. |  // |  //TRIAL TABLE: |  //+------+----------+--------+----------+----------+----------+ |  //| **kind** | **token\_id** | **status** | **began\_on** | **ended\_on** | **allotted** | |  //+------+-----x----+--------+----------+----------+----------+ |  // | |  //KINDA\_TABLE: | |  //+------+-----x----+--------------+---------------+----------| |  //| **kind** | **token\_id** | **challenge\_id** | **has\_challenge** | **effort\_id** /.. |  //+------+----------+--------------+---------------+----------| |  // |-----------+-----------+-----------+ |  // ../ **has\_effort** | **is\_graded** | **grade\_10k** | |  // |-----------+-----------+-----------+ |  // |   |  |  | | --- | --- | | ////WHAT I DIDN'T DO, AND WHY:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX|  //Thought I could have trial\_table use "kind\_id" rather than "kind" |  //So that it could identify the kinda\_table record that way. |  //And then token\_id would only have to appear in trial\_table, not in |  //kinda\_table. And then, in the event of two candidates getting the |  //exact same test and answering the exact same way, they could reference |  //the EXACT SAME KIND\_ID from the kinda\_table. |  // |  //Problem with that: |  //1. They have to start with separate kinda\_table stubs. |  // So you would be doing a merge on the data. |  //2. If we want to add a re-take test in the future, you would have to |  // UN-MERGE those results. Seems like a pain to me. |  ////XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX|  // |  ////WHAT I DID DO, AND WHY: ===================================================|  // |  //KIND: kind is present in BOTH trial\_table and kinda\_table: |  // It is used for redundancy. The kind value should match in both records.|  // Also, if we want to destroy kinda\_table record and create a new test |  // with same token, we could, since trial\_table is also keeping track |  // of the kind. |  // |  //TOKEN\_ID: One trial\_table record per token\_id, one kinda\_table record per |  // token\_id. Thus, we can link/associate the information together |  // using the token\_id. This probably also means we could just make |  // it one big HUGE table. But that would make the entities larger, |  // the transaction utilities larger, the test servlets larger... etc. |  // This seems like a logical division the way I have partitioned the |  // tables. |  // |  //MORE INFO: |  //More on token\_id: |  //Question: Why not tie to ninja\_id or admin\_id? |  // Because Ninjas + Admins are two completely separate user types. |  // They are not even derived from each other. Ninjas cannot own an |  // account and login/password information. Ninjas can however, |  // own tokens. |  // |  // Tokens are the common currency. Whenever we have the choice of |  // using different foreign keys in our design, the arbitration |  // should be solved by using token\_id, if that is one of the |  // choices. | | //More on status: |  // There are other options between created and started. |  // But I can't remember exactly what they were called. But basically, |  // there is another step where the user confirms the token, but has |  // NOT started the test yet. |  //Defining "guts of challenge: |  // Guts == all the questions on test. No meta data. Just questions. |  //More on effort\_id: |  // Could also think of it as |  // a "response". But response is too generic for me. |  // Because response could mean "response to a single question" |  // or "response to a collection of questions. "effort" in this |  // context explicitly means "a collection of responses." |  //More on has\_challenge & has\_effort: |  // Used so we can safely have null references when foreign key |  // does not exist. For example no challenge data or effort data will |  // exist when stubs are first created. When trial-questions have been |  // generated, challenge\_id will be a value >=1, and has\_challenge will|  // == true. has\_effort however will remain false until ninja has |  // submitted all of their answers for grading. Then an effort object |  // will be created to reflect the ninja's responses. |  //==================TRIAL\_TABLE & KINDA\_TABLE DIAGRAMS: END====================+  AHK SHORTCUT:  ?trial | |