MOBILE APPLICATION DEVELOPMENT

ANDROID (2017)

LECTURE 21: FIREBASE DATABASE RULES

FIREBASE REALTIME DATABASE STRUCTURE

- Firebase's Realtime Database is structured as a single JSON object, which in turns contains nested objects.
- By default, there are almost no security or data format restrictions on what goes into the database, and any authenticated user may write arbitrary data to any part of the database, even overwriting other users' data.
- Realtime Database supports a number of rules, also written in a JSON-like format, which govern how data in the database should be structured and who has permission to read or write that data.

REALTIME DATABASE RULE TYPES

- Realtime Database rules may take on a number of standard forms:
  - Read rules: boolean conditions which govern whether or not data at or below a given location in the JSON tree may be read by the current user.
  - Write rules: boolean conditions which govern whether or not data at or below a given location in the JSON tree may be written to by the current user.
  - Validation rules: boolean conditions which govern how data at a given location should be structured, if new non-null data is being written to that location.
  - Indexing rules: named child keys for a given location, by which children of the current location should be indexed.

READ/WRITE RULES

- Read and write rules cascade downward, meaning that if any read or write rule succeeds or fails at the level of a parent object, all of that object's child objects will have the same read/write availability as the parent.
- Read and write rules are checked before data validation rules are examined, and unlike data validation rules the read and write rules may operate on null data that is being written (for example, to prevent null data from being written).
- Use read rules to govern who has access to a given section of the database.
- Use write rules to create read-only data, to prevent unwanted data deletion, and to validate data that validate rules cannot (such as null data).

VALIDATE RULES

- Validate rules are designed to define and enforce data structure requirements for non-null data that is being written into a database.
- These rules cannot operate on null data, so they cannot be used to prevent data deletion (data at and below a child node is deleted if null is written to that node).
- Can examine either the current data at a location, the new data being written, or a combination of the two to decide whether a write should be validated.
- Validate rules do not cascade, so validating a parent object does not validate its child objects.

INDEXING RULES

- Indexing rules allow data to be indexed more efficiently than the default behavior that Realtime Database provides.
- By default, keys for nodes in a database are indexed by their primary key, but if that node contains many other nodes, there is no efficient default indexing rule for which of the child nodes the parent should be indexed on.
- Indexing rules specify child nodes of a given node, by key, that the parent node should be indexed on. If the choice of child node for indexing is made wisely, it can greatly improve indexing performance.

PREDEFINED RULE VARIABLES

- Realtime Database provides a number of predefined variables which may be referenced in read/write/validate rules to help get information about the data.
- While not exhaustive, some of the useful predefined variables are below:
  - The `now` variable gives the time in milliseconds since the Linux epoch.
  - The `root` variable gives a representation of the root of the database before the current write operation took place.
  - The `data` variable gives a representation of the current node in the database as it existed before the current operation took place.
  - The `newData` variable gives a representation of what the current node in the database would look like if the current operation took place (but the operation hasn't really happened yet).