**Subject:** Redesigning Our Semantic Model to Improve DirectQuery Performance

Hi Team,

I want to update you on a redesign we are making to our semantic model.

**The main purpose of this change is to improve DirectQuery performance by reducing the size of the detail fact table.**

Right now, the fact table is extremely large (over 600 billion rows with 268 attributes). Whenever a report runs in DirectQuery mode, Power BI sends SQL queries to Databricks that must scan this massive table. That’s why we often see slow performance.

### **What we’re changing**

1. **Slimming down the fact table**
   1. The fact table will only contain foreign keys and the report\_value column.
   2. All 268 attributes will be removed from the fact.
2. **Splitting dimensions into two parts**
   1. **Core dimension (Dual mode):** Contains the most frequently used, lower-cardinality attributes. These load into memory for fast response, but can still query the source when necessary.
   2. **Extended dimension (DirectQuery):** Contains less-used or very granular attributes. These stay in DirectQuery so they don’t consume memory unnecessarily, and are only queried when needed.

### **Why split dimensions?**

Not all attributes are equal. Some are used constantly in reports (e.g., common filters and slicers). These belong in the **Core** table so they run quickly from memory. Other attributes are rarely used, or have huge numbers of unique values. These go into the **Extended** table so they don’t slow down memory or bloat the model, but can still be queried when necessary.

This split gives us the best of both worlds:

* Fast performance for everyday attributes.
* Flexibility to query the detailed attributes only when needed.

### **What this means in practice**

* **DirectQuery will be much faster overall** because the fact table is smaller and queries scan fewer columns.
* **DISTINCT queries (like slicers and filters)** will improve because they’ll run against smaller dimension tables instead of the huge fact.
* Aggregations and Import tables are unaffected — they already run well, and will continue to do so.

### **Expected benefits**

* Faster response times for DirectQuery queries.
* Faster slicers and drilldowns.
* More reports answered from aggregations instead of falling back to detail.
* Better scalability when multiple users are running queries at once.

This redesign is focused on **DirectQuery performance** — slimming down the fact and moving attributes into smarter dimension structures. We’ll monitor performance closely after rollout and adjust which attributes belong in Core vs Extended based on actual usage.

Thanks,  
 Julio

**Subject:** Proposal: Using Surrogate Keys to Optimize Our Power BI Model

Hi [Manager’s Name],

I’d like to propose an optimization for our Power BI / Databricks model: introducing **surrogate keys** in place of business keys for joins between fact and dimension tables. While our current star schema is valid with business keys, relying on them introduces inefficiencies that affect both performance and scalability.

### **Current Limitations with Business Keys**

* **Redundant Records**  
   Dimensions contain multiple copies of the same logical record, only differentiated by business IDs.
* **Larger Model Size**  
   The redundancy inflates both dimensions and fact tables, leading to higher memory consumption.
* **DirectQuery Performance Impact**  
   Larger tables mean more data scanned in Databricks, slowing query response times.
* **Slower Refresh Times**  
   A bigger data volume extends refresh duration, making our processes less efficient.

### **Benefits of Using Surrogate Keys**

1. **Reduced Data Volume**  
    Surrogate keys collapse redundant records into a single unique record, shrinking both dimensions and facts.
2. **Improved Query Performance**  
    Integer-based surrogate keys are lightweight, making joins faster and more efficient than composite business keys.
3. **Faster Refreshes**  
    Smaller model size means less data processed during refresh, reducing end-to-end refresh time.
4. **Better Scalability**  
    By freeing up memory, we can load more attributes in Import mode and keep the model responsive as it grows.
5. **Industry Best Practice**  
    Surrogate keys are a **common BI and Power BI modeling standard**, documented in Microsoft’s star schema guidance ([learn.microsoft.com](https://learn.microsoft.com/en-us/power-bi/guidance/star-schema?utm_source=chatgpt.com)) and dimensional modeling frameworks such as Kimball ([kimballgroup.com](https://www.kimballgroup.com/1998/05/surrogate-keys/?utm_source=chatgpt.com)).

### **Mitigating Schema Change Concerns**

To minimize disruption when columns are added or removed:

* **Mapping Table**: Maintain a mapping of business keys to surrogate keys so schema changes don’t force full reloads.
* **Incremental Refresh**: Refresh only the affected partitions instead of the whole model.
* **Adaptive Key Generation**: Design surrogate key logic to evolve with schema changes, ensuring stable joins over time.
* **Versioned Dimensions**: Keep historical surrogate-keyed records intact while adding new schema versions as needed.

### **Summary**

While business keys technically work in our current star schema, they introduce redundancy and inflate the model, which increases memory usage, slows down DirectQuery performance, and lengthens refresh times. Surrogate keys eliminate these inefficiencies, aligning us with BI best practices and ensuring our model is leaner, faster, and more scalable.

Best regards,  
 [Your Name]