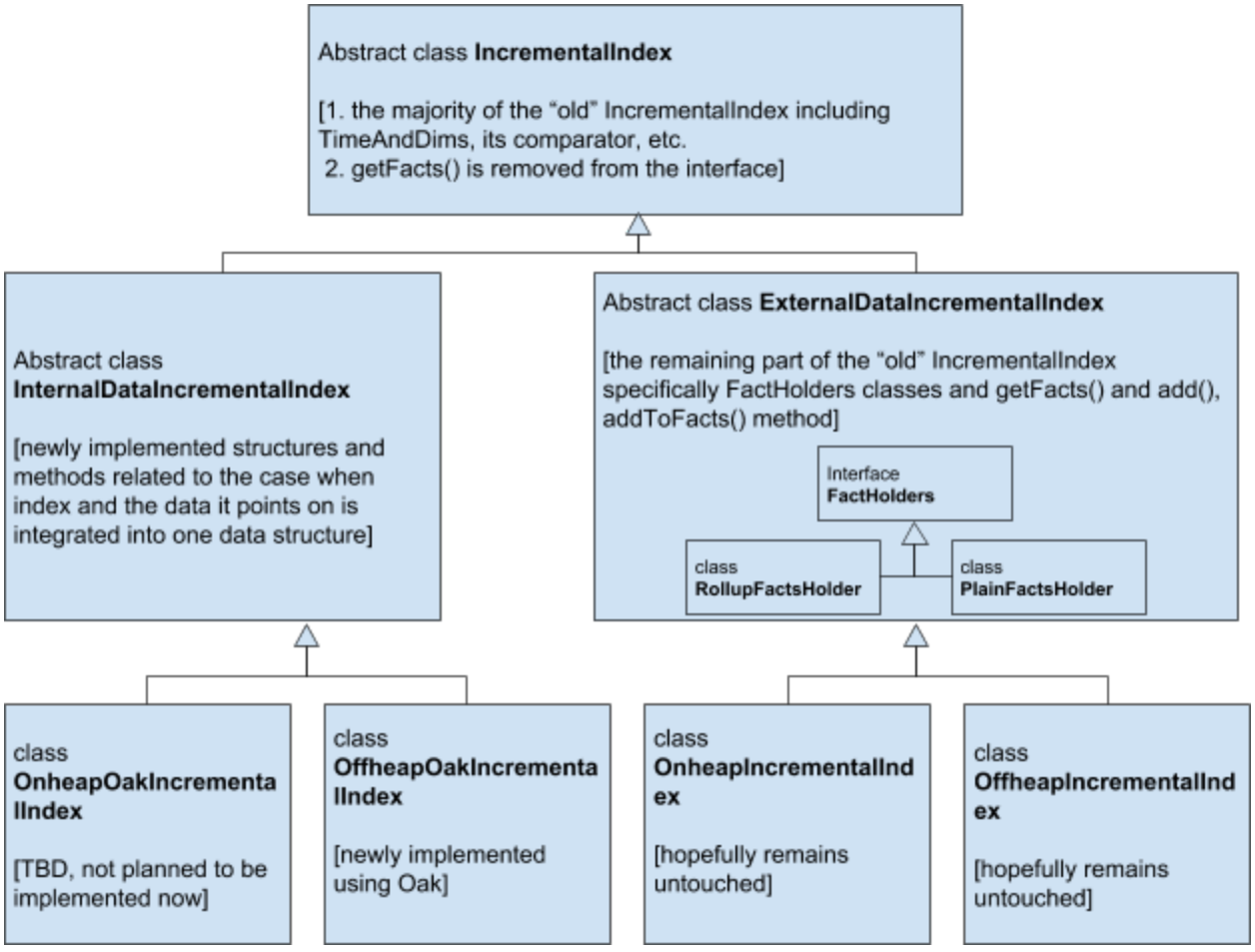
Implementing Oak-based IncrementalIndex: Design Document

In this document we are going to refer to the current design of IncrementalIndex as OldIncrementalIndex, while we suggest to renew the IncrementalIndex structure. Hereby, we gradually present the changes we suggest.

When designing the Oak integration into the Druid code, we have encountered that OldIncrementalIndex assumes the keys are kept in FactHolders while the keys' data is assumed to locate in some separate storage. As Oak assumes the keys and the data/values are tightly coupled, we suggest to create two types of IncrementalIndex: one where data is external to the keys (as it is now) and another one where the data is internal/integrated together with the keys into the key-value map (as it is in Oak). The class names are accordingly: InternalDataIncrementalIndex (from which Oak variants can inherit) and ExnternalDataIncrementalIndex (from which current indexes can inherit).

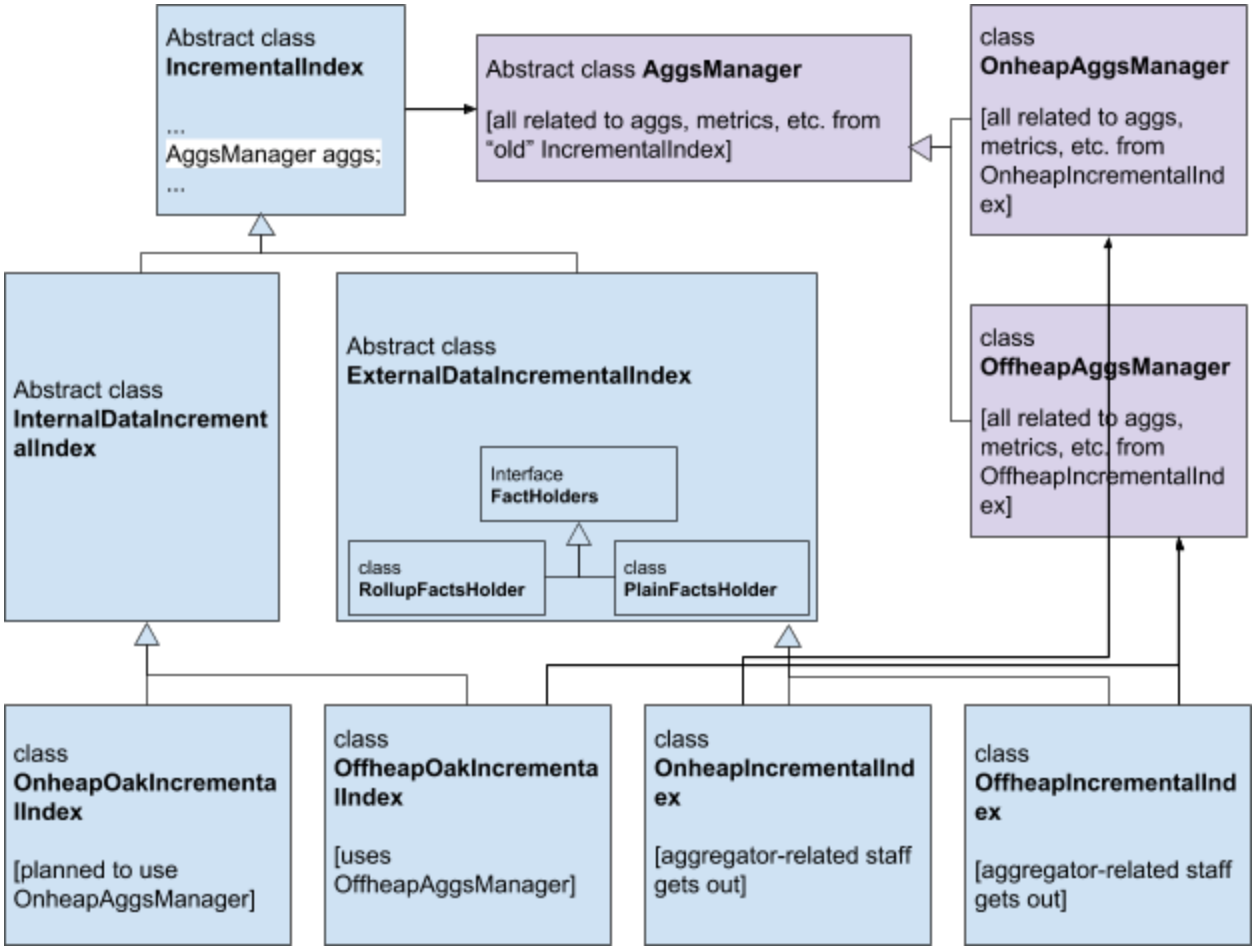
Hereby, a high-level (new) class diagram:



The OldIncrementalIndex is actually separated into two classes the remainder IncrementalIndex and new ExternalDataIncrementalIndex. The following table presents the new location of all classes and methods originated from OldIncrementalIndex that are not going to remain in the IncrementalIndex.

Method/Class	The New Owner
public abstract FactsHolder getFacts();	ExternalDataIncrementalIndex Removed from IncrementalIndex interface!
public int add(InputRow row, boolean skipMaxRowsInMemoryCheck) throws IndexSizeExceededException;	Turns abstract in IncrementalIndex and implemented in InternalDataIncrementalIndex. Current implementation moves to ExternalDataIncrementalIndex
protected abstract Integer addToFacts(AggregatorFactory[] metrics, **boolean** deserializeComplexMetrics, **boolean** reportParseExceptions, InputRow row, AtomicInteger numEntries, TimeAndDims key, ThreadLocal<InputRow> rowContainer, Supplier<InputRow> rowSupplier, **boolean** skipMaxRowsInMemoryCheck) **throws** IndexSizeExceededException;	ExternalDataIncrementalIndex
interface FactsHolder	ExternalDataIncrementalIndex
static class RollupFactsHolder implements FactsHolder	ExternalDataIncrementalIndex
static class PlainFactsHolder implements FactsHolder	ExternalDataIncrementalIndex

The next change we suggest is related to the aggregators managing. In the initial design the aggregators (and related to them metrics) are abstractly defined in OldIncrementalIndex and then on-heap (off-heap) aggregators are implemented in OnheapIncrementalIndex (OffheapIncrementalIndex) respectfully. The same on-heap/off-heap aggregators and metrics managing implementations can be useful in OnheapOakIncrementalIndex and OffheapOakIncrementalIndex. In order to avoid code duplication, we suggest to take aggregators, metrics and all related to be a separate class included via composition pattern in all indexes. The new class is going to inherit to on-heap and off-heap variants. This change is presented in the following class diagram:



The following table presents the new location of every field/class/method after it is moved out (or remains in) OldIncrementalIndex.

Method/Class/Field	**The New Owner**	
private volatile DateTime **maxIngestedEventTime**;	IncrementalIndex	
public static ColumnSelectorFactory makeColumnSelectorFactory()	IncrementalIndex	
private final long minTimestamp; **private final** Granularity **gran**; **private final boolean rollup**; **private final** List<Function<InputRow, InputRow>> **rowTransformers**; **private final** VirtualColumns **virtualColumns**; **private final** Metadata **metadata**; **private final** Map<String, DimensionDesc> **dimensionDescs**; **private final** List<DimensionDesc> **dimensionDescsList**; **private final** Map<String, ColumnCapabilitiesImpl> **columnCapabilities**; **private final** AtomicInteger **numEntries** = **new** AtomicInteger(); *// This is modified on add() in a critical section.*	IncrementalIndex	

`private final ThreadLocal<InputRow> in = new ThreadLocal<>();` `private final Supplier<InputRow> rowSupplier = in::get;`		

`private final AggregatorFactory[] metrics;`
`private final AggregatorType[] aggs;`
`private final boolean deserializeComplexMetrics;`
`private final boolean reportParseExceptions;`
`private final Map<String, MetricDesc> metricDescs;`
	AggsManager	

`protected IncrementalIndex(`
`  final IncrementalIndexSchema incrementalIndexSchema,`
`  final boolean deserializeComplexMetrics,`
`  final boolean reportParseExceptions,`
`  final boolean concurrentEventAdd`
`)`
	IncrementalIndex	

`public static class Builder`	**IncrementalIndex**	
`public boolean isRollup() {return rollup;}`	IncrementalIndex	
`public abstract FactsHolder getFacts();`	ExternalDataIncrementalIndex **Removed from IncrementalIndex interface!**	
`public abstract boolean canAppendRow();`	IncrementalIndex	
`public abstract String getOutOfRowsReason();`	IncrementalIndex	

`protected abstract AggregatorType[] initAggs(`
`  AggregatorFactory[] metrics, Supplier<InputRow> rowSupplier,`
`  boolean deserializeComplexMetrics, boolean concurrentEventAdd);`
	AggsManager	

`protected abstract Integer addToFacts(`
`  AggregatorFactory[] metrics,`
`  boolean deserializeComplexMetrics,`
`  boolean reportParseExceptions,`
`  InputRow row,`
`  AtomicInteger numEntries,`
`  TimeAndDims key,`
`  ThreadLocal<InputRow> rowContainer,`
`  Supplier<InputRow> rowSupplier,`
`  boolean skipMaxRowsInMemoryCheck`
`) throws IndexSizeExceededException;`
	ExternalDataIncrementalIndex	

`public abstract int getLastRowIndex();`	IncrementalIndex	
`protected abstract AggregatorType[] getAggsForRow(int rowOffset);`	AggsManager	
`protected abstract Object getAggVal(AggregatorType agg, int rowOffset, int aggPosition);`	AggsManager	
`protected abstract float getMetricFloatValue(int rowOffset, int aggOffset);`	AggsManager	

`protected abstract long getMetricLongValue(int rowOffset, int aggOffset);`	AggsManager	
`protected abstract Object getMetricObjectValue(int rowOffset, int aggOffset);`	AggsManager	
`protected abstract double getMetricDoubleValue(int rowOffset, int aggOffset);`	AggsManager	
`public void close();`	IncrementalIndex	
`public InputRow formatRow(InputRow row);`	IncrementalIndex	
`public Map<String, ColumnCapabilitiesImpl> getColumnCapabilities();`	IncrementalIndex	

`/**`
`* Adds a new row. The row might correspond with another row that already exists, in which case this will`
`* update that row instead of inserting a new one.`
`* <p>`
`* <p>`
`* Calls to add() are thread safe.`
`* <p>`
`*`
`* @param row the row of data to add`
`*`
`* @return the number of rows in the data set after adding the InputRow`
`*/`
`public int add(InputRow row) throws IndexSizeExceededException`
`{`
`return add(row, false);`
`}`
	IncrementalIndex	

`public int add(InputRow row, boolean skipMaxRowsInMemoryCheck) throws IndexSizeExceededException;`	Turns abstract in IncrementalIndex and implemented in InternalDataIncrementalIndex. Current implementation moves to ExternalDataIncrementalIndex	
`TimeAndDims toTimeAndDims(InputRow row);`	IncrementalIndex	

`private synchronized void updateMaxIngestedTime(DateTime eventTime);`
`public boolean isEmpty();`
`public int size(); private long getMinTimeMillis();`
`private long getMaxTimeMillis(); public List<String> getDimensionNames();`
`public List<DimensionDesc> getDimensions();`
	IncrementalIndex	

`public AggregatorType[] getAggs();`
`public AggregatorFactory[] getMetricAggs();`
`public String getMetricType(String metric);`
`public List<String> getMetricNames();`
`private static AggregatorFactory[]`
	AggsManager	

getCombiningAggregators(AggregatorFactory[] aggregators);		
public DimensionDesc getDimension(String dimension); public ColumnValueSelector<?> makeMetricColumnValueSelector(String metric, TimeAndDimsHolder currEntry); public Interval getInterval(); public DateTime getMinTime(); public DateTime getMaxTime(); public Integer getDimensionIndex(String dimension); public List<String> getDimensionOrder(); private ColumnCapabilitiesImpl makeCapabilitesFromValueType(ValueType type);	IncrementalIndex	
public void loadDimensionIterable(Iterable<String> oldDimensionOrder, Map<String, ColumnCapabilitiesImpl> oldColumnCapabilities); private DimensionDesc addNewDimension(String dim, ColumnCapabilitiesImpl capabilities, DimensionHandler handler); public List<String> getColumnNames(); public StorageAdapter toStorageAdapter(); public ColumnCapabilities getCapabilities(String column); public Metadata getMetadata(); public Map<String, DimensionHandler> getDimensionHandlers(); public Iterator<Row> iterator(); DateTime getMaxIngestedEventTime()	IncrementalIndex	
public Iterable<Row> iterableWithPostAggregations(final List<PostAggregator> postAggs, final boolean descending);	Turns abstract in IncrementalIndex and implemented in InternalDataIncrementalIndex. Current implementation moves to ExternalDataIncrementalIndex	
public static final class DimensionDesc	**IncrementalIndex**	
public static final class MetricDesc	**AggsManager**	
public static final class TimeAndDims	**IncrementalIndex**	
protected ColumnSelectorFactory makeColumnSelectorFactory(final AggregatorFactory agg,final Supplier<InputRow> in,final boolean deserializeComplexMetrics); **protected final Comparator<TimeAndDims> dimsComparator(); private static boolean allNull(Object[] dims, int startPosition);**	IncrementalIndex	
static final class TimeAndDimsComp implements Comparator<TimeAndDims>	**IncrementalIndex**	
interface FactsHolder	ExternalDataIncrementalIndex	
static class RollupFactsHolder implements FactsHolder	ExternalDataIncrementalIndex	

static class PlainFactsHolder implements FactsHolder	ExternalDataIncrementalIndex	
private class LongMetricColumnSelector implements LongColumnSelector	**IncrementalIndex**	
private class FloatMetricColumnSelector implements FloatColumnSelector	**IncrementalIndex**	
private class DoubleMetricColumnSelector implements DoubleColumnSelector	**IncrementalIndex**	

The interface that remains to implement by **InternalDataIncrementalIndex** class and its derived classes:

1. **To implement add() and iterator() need to implement** serializer **(from Time&Dims to ByteBuffer) and** comparator **(for two Time&Dims in the ByteBuffer)**
2. **public int add(InputRow row, boolean skipMaxRowsInMemoryCheck) throws IndexSizeExceededException; (putIfAbsentComputeIfPresent)**
3. **public abstract boolean** canAppendRow();
4. **public abstract** String getOutOfRowsReason();
5. **public abstract int** getLastRowIndex();
6. **public Iterable<Row> iterableWithPostAggregations(final List<PostAggregator> postAggs, final boolean descending);**
7. **protected long** getMinTimeMillis()
8. **protected long** getMaxTimeMillis()
9. **public abstract** Iterable<TimeAndDims> timeRangeIterable(**boolean** descending, **long** timeStart, **long** timeEnd);
10. **public abstract** Iterable<io.druid.segment.incremental.IncrementalIndex.TimeAndDims> keySet();