```
namespace paql
{
    namespace container
    {
        class ElementContainer :
        public std::list< boost::reference_wrapper<paql::element::Element> >,
        public virtual boost::fusion::map
        <
            boost::fusion::pair
            <
                MetaKey<boost::fusion::vector<T00, ..., T0n>, 0>,
                std::map
                <
                    boost::fusion::vector<T00, ..., T0n>,
                    boost::reference_wrapper<paql::element::Element>,
                    paql::FusionVectorComparator
                    <
                        boost::fusion::vector<T00, ..., T0n>
                    >
                >
            >,
            .
            .
            .
            // One boost::fusion::map entry per key
        >
        {
            public:
            template<int keyIndex> paql::element::Element& get
            (T00 a0, ..., T0n an)
            {
                boost::fusion::vector<int>key(a0, ..., an);
                return
                    (boost::unwrap_reference<paql::element::Student>::type&)
                    (
                        (
                            (
                                boost::fusion::at_key
                                <
                                    MetaKey<boost::fusion::vector<int>, 0>
                                >(*this)
                            ).find(key)
                        )->second
                    );
            }
            .
            .
            .
            // one get method per key
            bool insert(paql::element::Element& elem)
            {
                boost::reference_wrapper<paql::element::Element> reference =
                    boost::ref<paql::element::Element>(elem);
                {
                    boost::fusion::vector<int> key(elem.var00, ..., elem.var0n)
;
                    if
                    (
                        (
                            boost::fusion::at_key
                            <
                                MetaKey
                                <
                                    boost::fusion::vector<T00, ..., T0n>, 0>
                                >(*this)
                        ).count(key)
                    ) return false;
                    (
                        boost::fusion::at_key
                        <
                            MetaKey<boost::fusion::vector<T00, ..., T0n>, 0>
                        >(*this)
                    ).insert
                    (
                        std::map
                        <
                            boost::fusion::vector<T00, ..., T0n>,
                            boost::reference_wrapper<paql::element::Element>,
```

```
                        FusionVectorComparator
                        <
                            boost::fusion::vector<T00, ..., T0n>
                        >
                    >::value_type(key, elementToInsert)
                );
            }
            .
            .
            .
            // one insertion scope per key
            this->push_back(boost::ref(element));
            return true;
        }
        void remove(paql::element::Student& element)
        {
            {
                boost::fusion::vector<int> key(element.code);
                (
                    boost::fusion::at_key
                    <
                        MetaKey<boost::fusion::vector<int>, 0>
                    >(*this)
                ).erase(key);
            }
            .
            .
            .
            // one remove scope per key
            this->remove_if
            (
                ContentComparator<paql::element::Student>
                (
                    boost::ref(element)
                )
            );
        }
    };
  }
}
```